OPSWAT.



MetaDefender™ Core

MetaDefender Core

v5.24.1

Contents

Installation 21

- Recommended System Configuration 21
- Installation 30
  - Standalone mode 31
  - Shared Database Mode 40
  - Deployment automation support 49
    - Enabling SSL Connections for PostgreSQL via ignition file 57
    - Ignition file - special characters 59
  - CIS Level 1 - Linux Guidelines 61
  - CIS Level 2 - Linux Guidelines 64
  - CIS Level 1 - Windows Guidelines 66
  - CIS Level 2 - Windows Guidelines 66
- Wizard Setup 67
- License Activation 75
  - Scripted license management 80
- File Processing 83
- Product Upgrading 83
- Using external Load Balancer 88
- Performance and Load Estimation 93
  - Standalone DB - Windows 94
  - Standalone DB - Linux 101
  - Shared DB - Windows 109
  - Shared DB - Linux 114
- MetaDefender Core Documentation 120

Configuration 121

- Management Console - Login 121
  - Password Recovery 121
- MetaDefender Configuration 128
  - What happens when you change data_directory? 144
- Policy Configuration 144
  - Workflow Configuration Template 145
    - Blocklist by filetype configuration 164
    - Allowlist by filetype configuration 171

- Security zone configuration 176
- Workflow rule configuration 176
- Quarantine 183

MetaDefender Core Performance Tuning Guide 185

User management 204
- Roles 208
- User directories 213
  - Integrate Active Directory with BeyondTrust Password Safe 220
- Active Directory/LDAP attributes 225
- Change user password 230
- Single Sign-On (SSO) 231
- User directory priority at login 232

User role mapping 233

Multiple Login URLs 239

Amazon Cognito 241
- Configure OIDC SSO 241

ForgeRock 257
- Configure SAML SSO 257

Keycloak 266
- Configure SAML SSO 266
- Configure OIDC SSO 282

Microsoft Active Directory Federation Services (ADFS) 295
- Configure SAML SSO 295
- Configure OIDC SSO 313

Microsoft Entra ID 333
- Configure SAML SSO 333
- Configure OIDC SSO 347

Okta 357
- Configure SAML SSO 357
- Configure OIDC SSO 371

OneLogin 380
- Configure SAML SSO 380
- Configure OIDC SSO 390

PingFederate 397
- Configure SAML SSO 398
- Configure OIDC SSO 417

PingOne 449
- Configure SAML SSO 449
- Configure OIDC SSO 462

Best Practices for Configuration & Security Hardening Guide 473

System Configuration 482

Data Retention 488

Database Management 489
Encryption for Quarantined Files 491
Email Configuration 492
Email Notification 499
Email Template 502
Enabling HTTPS 505
Engines Update Configuration 508
External Scanners and Post Actions 511
Hardening External Scanner and Post Action feature 524
Health Check API Configuration 528
Hot Swap Backup Configuration 531
Logging 533
Advanced Log Configurations 534
Log Message Format 535
Syslog Message Format 537
Error Message Description Table 541
Syslog facility 541
Log Rotation and Retention 544
My OPSWAT Central Management 547
Enrollment and MetaDefender Core Configuration on My OPSWAT Central Management 547
Settings for My OPSWAT Central Management 550
NGINX Configuration 551
API Rate Limiting 553
SSL Configurations 555
Origin Client Source Identification 557
Web Server Processing Statistics Query 558
Password Policy 559
Proxy Configuration 561
Request Rate Limit 563
Scan-from-link Settings 565
Schedule Report 567
Session Policy 569
SSL Connection for bundled PostgreSQL 570
Usage Report 573

Operating 575

Process files with MetaDefender Core 575
Dashboard 576
Average processing time 586
Understanding Block Reason Statistics 587
Processing history 589
Processing history - search by user 591

Processing history - search by extension 592
Advanced Export Result Templates 593
Access history 595
Download quarantine files 597
Engines 598
Library Management 603
Regular Maintenance 615
Database Defragmentation and Optimization 615
Database Maintenance | MetaDefender Core 4.19.0 or above 615
Database Maintenance | MetaDefender Core 4.18.0 617
Download single scan report as PDF 619
Import/Export configuration 622
Archive Forensic 636
Auto Troubleshooting 638

Integration 641

Scan result codes and explanations 641
Request rate limiting 656
Multipart upload guideline 661
Split archive upload guideline 666
Integrate MetaDefender Core syslog into Splunk Enterprise 673

Cloud Deployment 684

Overview 684
Recommended Architectures 684
EKS Cluster 685
Single EC2 689
Autoscaling EC2 692
AWS Database Services 696
Licensing Automation on AWS 699
How to deploy using AWS services 701
EKS Cluster 706
Single EC2 711
Autoscaling EC2 718
Monitoring in AWS 719
Troubleshooting and Maintenance 720
Customize OPSWAT MetaDefender AMI from Marketplace 721
Recommended Architectures 722
AKS Cluster 722
Single Azure VM 724
VM Scale Set 726
Licensing Automation on Azure 728

Deploy using Azure services 732
AKS Cluster 736
Single Azure VM 741
VM Scale Set 747
Recommended Architectures 750
GKE Cluster 750
GCP Deployments 754
GKE Cluster 758
Provisioning OpenShift on your CSP 762
Deploy your Own OpenShift Cluster 763
MD Core installation 764

Container Deployment 768

Overview 768
Kubernetes Components 768
Deploying on Kubernetes 773
Installing MetaDefender Core in K8S 774
Resources Requirements 781
Licensing in K8S 782
Storage Configuration on K8S 784
Upgrade/Deployment Strategy on K8S 785
High Availability on Kubernetes 791
Scaling K8S Cluster 793
Troubleshooting 796
Unable to deploy MetaDefender Core in Kubernetes with containerd engine 2.2.x 801
Overview 802
Docker image published on OPSWAT Docker Hub 803
Using build tool kit with your own docker image 821

Cluster Deployment 824

MetaDefender Cluster Deployment 824

Metascan Engines 825

Anti-malware capabilities 825
Supported file type 827
MetaDefender MetaScan Engine Packages 828
Government & Defense | Anti-malware capabilities 829
AV engine return code 830
Configure ClamAV to use a custom database 831
Set up Windows Defender as a custom engine 832
AV scoring with post action 835

Deep CDR 837

- Supported File Types 837
  - Supported versions 851
  - Max File Size 854
- Configurations 856
  - Retain password protection on sanitized files 883
  - Allow Macros 884
  - Allow Hyperlinks 886
  - Hyperlink redirection 887
  - Image limit configurations 890
  - QR code sanitization 891
  - ASCII-based QR code 891
  - Base64-encoded data in JSON 892
  - Remove invisible characters in TXT 897
  - Mark Sanitization Result as Skipped 898
  - Process Hyperlink Behavior in PDF 899
  - Preserve signed PDFs without sanitization 902
  - Validate XML files against an XSD schema 904
  - Skip Sanitizing Already-sanitized Files 905
- Analysis mode 907
  - Block file based on objects 911
- Deep CDR Performance (Throughput) 917
- Archive Sanitization 923
  - Tombstone file 926
- Setup output file name 928
- Deep CDR Details 932
  - Reason for Action 935
- Troubleshooting sanitization failures 936
- Supported engines-metadata JSON 937

Proactive DLP 939

- Overview 939
  - Supported File Types 940
- Detect sensitive information 951
  - Detection Policy 958
  - Detection Policy - Supported Entities 960
  - Secrets Detection 963
  - Optical Character Recognition (OCR) 967
  - Leverage metadata info added by data classification systems 969
  - Process hidden parts in a cropped image 969
  - Regular expressions 970

Importing Regular Expressions via a TSV File 975
Remove alternative images in PDF 977
Supported engines-metadata JSON 980
Supported metadata type in PDLP 980
Redact/Substitution/Hash sensitive information 983
Watermark files 989
Remove metadata 991
DICOM anonymization 993
Document Identification 998
Data tagging and classification 1001
Detect intentional data leakage 1002
Performance (Throughput) 1005

Adaptive Sandbox 1008

Overview 1008
Supported File Types 1010
System requirements 1015
Upgrade to Java 17 1017
Installation 1019
Licensing 1020
Configuration 1022
Scan Profiles 1030
Load Balancing / Failover 1031
Release notes 1032
Knowledge Base 1032
Internet Connection 1033
Enhancing Threat Detection with Deep CDR & MetaDefender Aether for Core (Adaptive Sandbox) 1033
How do I obtain my OPSWAT MetaDefender Cloud Reputation API Key? 1037
What is displayed on the process result tile? 1038
Enhancing Threat Detection with YARA & MetaDefender Aether for Core (Adaptive Sandbox) 1040
Best practices for API keys in MetaDefender Core Sandbox integrations 1041
How to configure Offline Reputation Service allowlisting in MetaDefender Core Adaptive Sandbox. 1044
Sandbox-Core High Availability Setup 1047

OPSWAT AI Content Inspector Engine 1049

Overview 1049
Supported file types 1050
Detection 1054
AI-Generated Text Detection 1055
AI-Generated Image Detection 1056

Reputation Engine 1059

Overview 1059
Licensing 1061
Reporting False Detection 1062
System requirements 1062
Installation 1062
Configuration 1063
Release notes 1066
Knowledge Center 1066

Software Bill of Materials 1067

Software Bill of Materials (SBOM) 1067
Source code 1067
Recursive check 1069
Container image 1071
License risk detection 1072
SBOM JSON response 1074
Cryptographic Bill of Materials (CBOM) 1079
CBOM JSON response 1085
AI Bill of Materials (AIBOM) 1091
AIBOM JSON response 1094
Analyze CycloneDX/SPDX report file 1099

Country of Origin 1102

Country of Origin 1102
Digital Certificate Verification 1103

Threat Intelligence Engine 1111

Overview 1111
Licensing 1115
System Requirements 1115
Installation 1115
Configuration 1116
Knowledge Center 1120
Release Notes 1120

Predictive Alin AI 1122

What is Predictive Alin AI? 1122
AI-Powered Threat Detection 1123
Continuous Threat Intelligence Pipeline 1124
Pre-execution File Analysis 1125
File Types 1126
Supported File Types 1127

Maximum File Size 1128
Engine Response Structure 1129
Supported Operating Systems 1130

File Structure Validation 1132

What is File Structure Validation engine? 1132
Features 1133
Extracting objects and recursive processing 1134
Blocking files based on rules 1135
Polyglot detection 1137
XML schema validation 1138
JSON schema validation 1139
Understanding the Validation Report 1143

Release Notes 1146

Release notes 1146
Archived release notes 1153
Metascan release notes 1359
Deep CDR release notes 1398
Proactive DLP release notes 1436
File Type Module release notes 1452
File Structure Validation release notes 1468
Archive Module release notes 1469
SBOM release notes 1485
YARA engine release notes 1491
Adaptive Sandbox release notes 1493
Embedded Engine 1493
Remote Engine 1520
Reputation Engine release notes 1531
Country of Origin release notes 1533
Threat Intelligence engine release notes 1535
Predictive Alin AI release notes 1535
OPSWAT AI Content Inspector release notes 1543

Utilities engines 1550

File Type Verification Engine 1550
Supported file types 1550
Configuration 1551
Supported engines-metadata header 1553
Block files based on file info details 1555
Text detection with AI 1558
Custom detection 1560

Detect specific file types only 1574
Archive Extraction Engine 1577
Supported file types 1577
Supported engines-metadata header 1587
Decrypt GPG files 1587
Configuration 1589
Differences between OPSWAT Archive engine and 7-zip 1591
Integrate with external command line tools 1592
Decrypt Microsoft Purview Information Protection (MPIP) files 1593
Archive Compression Engine 1601
Configuration 1601
YARA Engine 1604
Configuration 1606
Upgrading from YARA to YARA-X 1606
Attack Vector Detection Coverage 1608

Knowledge Base 1615

Why are Japanese TXT files detected as "application/octet-stream" instead of "text/plain"? 1615
How to Connect a MetaDefender Core Container to the same instance in Shared Database Mode? 1616
Can local AVs interrupt ongoing scans? 1616
Can I create new workflows? 1618
How to enable crash dumps for MetaDefender Core processes? 1619
Can I control access to the RAM disk in MetaDefender Core v5? 1621
How do I prevent Symantec Endpoint Protection from interfering with the optimal functioning of MetaDefender Core? 1622
What is the retry mechanism for MetaDefender Core connecting to the remote Postgres Database? 1625
How to use MetaDefender Core V5 Blocklist/Allowlist feature? 1625
How can I configure the maximum queue size in MetaDefender Core v5 ? 1626
How do I remove an engine from my MetaDefender instance? 1628
How can the Temp folder be changed? 1629
How can I increase parallel count and Max queue size? 1630
How can I change MetaDefender Core Deep CDR's timeout settings? 1632
My scans keep failing due to an exceeded archive file number, how do I determine the number of files in an archive and then configure my process settings accordingly? 1634
Why are password protected archives blocked and how do I unblock them? 1636
How to disable WEB UI file scan without user authentication? 1639
How To Allow Only Certain Files to be Scanned with MetaDefender Core? 1641
How to generate an API key on the Core deployment? 1642
How to Modify the Hostname of Your MetaDefender Core Server and Potential Impact? 1643
What are the permissions on the shared folder for the temp directory? 1644
How to implement a numerical escape function for HTML? 1646
How to reset password for users? 1649

How to Block/Allow Custom Extensions using RegEx? 1651
How to migrate MetaDefender Core from PostgreSQL local to PostgreSQL remote? 1652
How to change Core Web Management Interface Address Port? 1654
Does MetaDefender Core store original files? 1657
How to install pre-built libgdiplus on RedHat 9? 1658
Does Deep CDR support sanitizing split archives? 1659
How to exclude OPSWAT folder from Anti-Virus Windows Defender? 1659
Where are MetaDefender Core Install Directories and Registry? 1662
Under what circumstances does the scan result not show the username? 1663
How to send MetaDefender Core logs to NAS (Network Attached Storage) Path? 1665
Why is a file blocked by Deep CDR or Proactive DLP but I can still download the sanitized version? 1675
Does Deep CDR support Adobe Illustrator (AI) files in PostScript format? 1676
How to add the source IP in scan details? 1677
How do I integrate Jenkins with MetaDefender Core? 1679
Can I apply a new license key before the current one expires? 1683
How to automate backup configuration settings for MD Core? (API) 1683
Why I can't push configuration due to password storage? 1686
How do I install MetaDefender Core in Rocky Linux 9 1686
How to properly configure Syslog Server for MetaDefender Core? 1688
Why MetaDefender Core's nginx ssl.conf changed to ssl.mdcore.conf? 1689
How to configure SMTP server with authentication under MetaDefender Core to send notification to M365? 1690
How to manage Disk Space Usage in pg_data\base directory in MetaDefender Core for Windows? 1692
What storage type should I use on MetaDefender Core in K8S? 1694
What permissions are needed for the user of the storage of Google Cloud? 1695
How do I setup Hot Swap Backup Configuration between MetaDefender Core instances? 1696
Where are MetaDefender Core Install Directories and Advanced Configurations? 1700
How to retrieve the CRL and OCSP URLs for the Server and ICA Certificates? 1703
Is Import Password Mandatory When Using Ignition File for MetaDefender Core installation? 1705
How does the recursion level work in MetaDefender Core? 1710
How to Enable Debug Mode in a MetaDefender Core Kubernetes Deployment? 1713
How to Select a Specific MetaDefender Core Instance in a PostgreSQL Database with Multiple Instances? 1715
Can you import configuration settings from the latest version to the previous version of Core? 1715
How to Local whitelist a file using AllowList? 1716
What Is exiftool_win.exe and Why Is It Falsely Detected? 1719
How to relocate installation folder in MetaDefender Core on Windows? 1720
How to Set Up MetaDefender Core to Connect to Syslog via TLS? 1724
How to Work with Role Restrictions in MetaDefender Core Workflows? 1725
How Do I Setup Syslog connect to Splunk Cloud by using Universal Forwarder? 1726
How do I customize the bundled PostgreSQL server configuration in MetaDefender Core? 1732
How do I detect PUP/PUA as infected? 1733
What is the difference between "Engine Current Status" and "Engine Advanced Configuration" export options? 1735

Should I Use Subject Alternative Name (SAN) Certificates for Core and Kiosk GUIs? 1740
How to customize Nginx logs to identify mTLS client connections? 1741
How to configure Role Mapping per groups in SSO Integration with Entra ID 1745
Does MetaDefender Core Detect EOS/EOL Status for Open-Source Software? 1752
How Does MetaDefender Core Rotate PostgreSQL Logs? 1753
Why Must Each MetaDefender Core Installation Have a Unique Instance Name? 1754
Why is my Excel document not showing hidden sheets after Deep CDR sanitization? 1755
How to Limit the Number of CPU Cores Used for Scanning in MetaDefender Core 1757
What Protocols Does MetaDefender Core Use for Active Directory Integration? 1760
Why Is the Time Range on MetaDefender Core's Executive Report Different from the Time Range Selected Before Exporting? 1760
What are the AI-Powered Components in MetaDefender Core? 1761
How to remediate significantly increased scanning times during peak hours due to engine updates failing? 1763
What More to Know About Encoding Preference for Entry Name? 1765
Is MetaDefender Core Fully SELinux Compliant? 1766
Why Does MetaDefender Core Require Removing the SSL Private Key Passphrase for PostgreSQL? 1766
When to set an alarm and how? 1767
Why are batch operations slow in MetaDefender Core and how are they resolved in version 5.14.0? 1772
Why Are Japanese Filenames Garbled When Submitting Files to MetaDefender Core via API? 1773
Does MD Core support using a different account type for starting application services 1774
How to export API Keys of MetaDefender Core users 1774
Should Filetype Mismatch Be Blocked or Allowlisted? 1776
Tracking Client Identity Information in MetaDefender Core via API 1777
Why Does My MD Core Deployment on Kubernetes Lose Data After a Restart? 1783
What is the pg_dump database restore file used by MetaDefender Core? 1786
Optimizing PostgreSQL for High-Volume Deployments 1788
How should I correctly restart the Shared Database server? 1792
How can I identify which MetaDefender Core instance sent an alert email? 1794
What Is the Recommended Skip Hash List Size in MetaDefender Core Without Impacting Performance? 1796
Does MetaDefender Core support sanitizing GIT-BUNDLE files? 1798
Can I configure multiple REST ports in MetaDefender Core? 1799
How do I identify and renew the certificate for Active Directory integration in MetaDefender Core? 1799
Retrieving Username Using an API Key 1800
How to Configure MetaDefender Core Syslog to an AWS Ubuntu Instance 1801
How do I reset all Skip by Hash Entries in MetaDefender Core? 1805
Can I export scheduled scan reports via API calls or email? 1807
How to integrate OpenSSL with MetaDefender Core for decrypting files 1808
How does the MD Core Handle Archive Files? 1812
Enabling NGINX Debug Logging in MetaDefender Core 1813
How Is Quarantined File Encryption Handled in MetaDefender Core? 1815
How to exclude .txt files from the File Type Engine? 1816
How to Correctly Update a Certificate in MetaDefender Core? 1818

Information regarding how files are processed and DB usage by Core 1819
How to resolve "There is a problem with this Windows Installer Package" error during the installation of MetaDefender Core? 1821
Can an old SSL certificate be safely removed from the Windows Local Machine certificate store after updating the HTTPS certificate for MetaDefender Core? 1823
Can MetaDefender Core scan GPG encrypted files and how can I configure it to do so? 1824
What can I do if MetaDefender shows a high number of files that failed to scan? 1825
What can I do when MetaDefender Core (Windows) has an insufficient disk space error? 1827
Why is MetaDefender Core unable to connect to the database? 1829
What should I do if an engine is in "failed" or "permanently failed" status? 1833
What are the engine clean-up instructions? 1834
Why is the scan stuck in "processing" state on WebScan UI, when the Core Processing History shows that it is already finished? 1836
Why have the ESET scan engine failed to initialize on the hardened Linux OS? 1837
Why has the Symantec custom engine failed to deploy? 1837
What are common reasons for Deep CDR scan/sanitization failures and how do I fix them? 1838
What do the various error messages mean in MetaDefender Core Rest API? 1841
Why are some modules not appearing in my MetaDefender Core inventory? 1859
Why does my MetaDefender Core install fail? 1860
Why do you sometimes see a discrepancy between the File Type and MIME Type information? 1864
Why files are not sanitized? 1865
Why am I receiving a “400” error for API file submission? 1868
How to Mitigate Scan Failures in MetaDefender Core Triggered by Real-Time Protection Systems? 1868
How to Resolve MetaDefender Core Service Start-up Failures on IPv6-Disabled Linux Systems? 1870
How to Understand OPSWAT's Multiscanning AV Detection Differences Compared to Other Scanners? 1871
How to Troubleshoot Inaccessible Management Console? 1873
How do I fix module update error “Error parsing metadescriptor, message='Could not download metadescriptor'”? 1874
Why does ClamAV show "permanently failed" after enabling the daemon log file? 1877
How to troubleshoot HTTPS failure on MetaDefender Core? 1879
How to Remove Engines from MetaDefender Core using command line? 1881
How to enable debug logging for MetaDefender Core? 1882
Why did the deployment of the Threat Intelligence Module Fail? 1884
Why did the deployment of the Reputation Module Fail? 1885
Why is MetaDefender Core not starting after moving or deleting the certificates from the initial location? 1885
Why doesn’t CDR detect the JavaScript I appended to a PDF file? 1886
Why Deep CDR is not Sanitizing Files Inside a ZIP? 1887
Why does the “deleteafterimport” error appear when importing an MD Core configuration from standalone mode to shared database mode? 1887
What is the meaning of action_ran and action_failed in JSON scan result? 1890
Why can't I sanitize HEIC files? 1891
Why Am I Getting "No Rows to Show" When Using a Proxy Without Authentication in Browser Settings? 1892

How Does TLS 1.0/1.1 Deprecation Affect Compatibility Across OPSWAT Products? 1894
Why can small file changes bypass the Antivirus detection? 1896
How do I verify if MetaDefender products are sending out syslog messages to the syslog server? 1897
How do I fix files being blocked due to unexpected "File type mismatch" verdict? 1899
Why are MetaDefender Core engines permanently failed? 1903
Why Does MetaDefender Core Fail to Connect to the Database After a System Restart? 1904
What do you do if your engine fails to update? 1906
What is causing my Scans to be slow on MetaDefender Core? 1907
What do I do if my modules disappear from the Core UI? 1909
Why Is Core Receiving HTTP 403 or Timeout Errors when downloading engine updates from OCM? 1910
What do I do when licensed engines are not detected despite active licenses? 1911
Why do I receive errors when I upload the Config file in MetaDefender Core? 1913
How to access MetaDefender Core when SSO fails? 1914
Why am I seeing error message "Validation failed" after saving Sandbox module settings? 1916
What Are the Default Credentials for MetaDefender Core on Azure Marketplace VM? 1916
Why does the MetaDefender Core service not start? 1917
Why does Kiosk show "Invalid Core Setup"? 1922
Why is my file rejected even though it appears to be under the configured size limit? 1926
How Can I Solve a Database Indexing Issue? 1928
Why Does MetaDefender Core Show a 'Calculate hash error' When Scanning Malicious Files? 1929
Why MD Core is Not Reachable? 1931
Why am I getting "invalid structure" or "archive corrupted" errors when extracting archives in Core? 1932
Why are my script files being detected as ASCII TXT? 1932
How to Resolve SSL/TLS Errors with MetaDefender Core Webhook Callbacks? 1933
Why does MetaDefender Core show “Blocked” with a scan result of “Not Scanned” when no malware is detected? 1935
How do I remediate MetaDefender Core using too much resource while sanitizing TIFF/TIF image file? 1938
Why are PDF documents being flagged as false positives by VirusBlokAda? 1939
Why did my scanned image fail with "exceeded file size" error? 1940
Why isn’t an attached file inside a PDF being converted to a different format during Deep CDR sanitization? 1942
Why is MetaDefender Core Failing to Connect via Proxy? 1942
Why is unused data removed in Deep CDR resulting in empty emails being received? 1943
Why are MetaDefender Core and PostgreSQL services shutting down unexpectedly? 1945
Why can’t I access the MetaDefender Core user interface after performing an upgrade? 1946
Why is ESET detecting legitimate software as a Potentially Unsafe Application (PUA/PUP)? 1948
Why can’t I configure the SMTP server with authentication set to “None” in MetaDefender Core 5.15.2? 1949
Why are my Engines not initializing/updating over the Internet or OPSWAT Central Management On-Premises 1950
Why am I getting an "Internal server error" when logging into MetaDefender Core via SAML SSO 1952
Why is HTTPS enabled in the UI, but the MetaDefender Core URL doesn’t change to HTTPS? 1953
Why does the MetaDefender Core Deployment ID change after rebooting a Hyper-V VM? 1955
Why are MetaDefender Core Syslog Messages Flagged as Non-Compliant with certain syslog standards? 1957

Why is the "The engine is not longer licensed with the current product key" message displayed for Threat Intelligence after license renewal? 1957
Are Embedded Files Recursively Processed and Reported during CDR? 1958
Why am I getting an error when installing MetaDefender Core on Linux in an air-gap environment? 1959
Why are Encrypted Office Documents with the same extension categorized differently from unencrypted documents? 1961
Why does MetaDefender Core skip quarantining files with the same hash? 1962
How to create a sample self-signed certificate with SAN using OpenSSL 1963
Why is MetaDefender Core Allowing Files with Sensitive Data? (Proactive DLP Threshold Settings) 1964
What to do when my MetaDefender Core instance failing to load with error 'Extension setting up failed: extension: 'ekm', rc: '-610"? 1967
Why am I getting "We couldn't analyze this file because no specific configuration is applied" with an OPSWAT Predictive Alin AI license? 1968
Why am I getting a "File not found, invalid path or access" error when using the 'filepath' header in the RestAPI on Linux? 1969
Can MetaDefender Core process Base64 Content? 1971
MetaDefender Core on K8s: Persisting Quarantine, DLP, and CDR Output Across Pod Restarts 1973
Why do some multipart uploads fail with error code 500? 1976
How to Remove Infected Child Files from Archives in MetaDefender Core 1977
Why does MetaDefender Core fail to trust a self-signed certificate in Kubernetes? 1978
How can I fix a failed or looping MetaDefender Core module update in MyOPSWAT Central Management v10 by clearing the package cache? 1984
Why does custom detection for FileType not work? 1985
Why REST API File Submission with downloadfrom Header Fails When Using RAMDISK for Temp Directory? 1986
Why do I get the "The cert file is not valid" or "the file does not exist" when adding my cert? 1987
Why are MetaDefender Core AppSec Engines Failing to Load After RHEL 8 to RHEL 9 Upgrade? 1988
What is the difference between "fail" and "cancelled" scan result codes in MetaDefender Core? 1990
Why Is a MetaDefender Core Node Stuck in Starting State in a Shared-Database Cluster? 1991
Why is MetaDefender Core slow and returning NGINX 499 errors? 1993
I enabled Secure API in MetaDefender Core but didn't save the API key. How do I reset it? 1995
Why does the MetaDefender Core Management Console display a blank page after an upgrade? 1998
Why am I getting the error “No available rule is present for scanning” in MetaDefender Core? 2001
Fixing 'cellimages.xml does not exist' Error in XLSX Files (WPS Office) 2002
What does the "outdated" field under engines in the health check API indicate? 2005
What is the meaning of the Stall detected messages when downloading engines on MetaDefender Core? 2006
How do I recover MetaDefender Core pods failing with HTTP 403 “Access denied” because of an API key mismatch? 2007
What temporary folder do Custom Engines use ? 2013
What PDF file annotations are Supported/Non-Supported by MetaDefender Core’s Deep CDR? 2013
What does "Potentially Vulnerable File" result mean? 2014
What CAB files are Supported/Non-Supported by MetaDefender’s Deep CDR? 2015
What are the MetaDefender Core Security Policies and how do they work? 2016

Is Metadefender Core compromised while scanning files? 2019
What is the Queue Mechanism on Metadefender Core? 2020
How does the Post Actions feature work? 2021
How does the External Scanners feature work? 2023
How to generate a Core's scan result summary in JSON format 2025
How and why are my image files altered during processing with Deep CDR and Proactive DLP? 2028
Does MetaDefender Core v5 offer real-time antivirus protection on the system where it is installed? 2030
Does MetaDefender Core v5 Detect the NotPetya Ransomware? 2031
Are there any limitations to the MetaDefender Core scan engines? 2033
Can MetaDefender Core block renamed files (such as renamed .exe files)? 2035
How can I export the processing history from the Core console? 2038
How to Utilize YARA Rules with MetaDefender Core? 2040
What is the Design of MetaDefender Core and Its Protective Measures? 2044
What does the items sanitized/removed count in Deep CDR sanitization report mean? 2044
Why image sanitization doesn't report any object in details? 2046
What are the requirements to process XDW/XBD/XCT files with CDR? 2046
How do we utilize API uploads so the correct file type and sanitization occurs every time? 2046
What are the differences between AVs in OPSWAT and the other services? 2048
Does OPSWAT Use Security Checks on Our Code, Libraries, Credentials, etc? 2050
How to access and analyze detailed scan results for an infected file? 2051
Why MetaDefender Core blocks Sandbox Verdicts that are not marked to be blocked? 2053
How to scan multiple files at once with MetaDefender Core 2054
Is MetaDefender Core able to extract, detect and block malicious embedded macros? 2055
Does OPSWAT provide a file dataset that can be used to evaluate MetaDefender Core's capabilities? 2055
How would discontinuing Threat Intelligence affect customers? 2056
How to add custom detection for a file type? 2057
What can I do if MetaDefender Core shows a high number of files that failed to scan? 2062
Why did the required password pop up when scanning an unencrypted file? 2065
How to Test the Syslog Server functionality from MetaDefender Core? 2067
When and how to use “PDF to Image Configuration” Deep CDR feature? 2068
How do I send specific log entries to syslog integration? 2075
How do I send the MetaDefender Core logs in Windows event log? 2077
Does installing an antivirus impact a containerized deployment? 2078
Why does the DeepCDR engine delete the content of the HTML file? 2078
What is the Document Encryption Timeout setting and how does it work? 2080
How do I manually clean up the MD Core engine's local folder? 2082
How can I handle a high CPU load on my Core server 2082
Does MetaDefender Core generate a file download link after all scan sessions? 2084
Can Workflow Settings Be Updated Globally Across Multiple Workflows? 2087
Does MetaDefender Core support Elliptic Curve (EC) Private Key and Certificate for enabling HTTPS? 2090
How to manage false positives in MetaDefender Core? 2091
How to handle split archive files? 2094

How to check the Nginx version used in MetaDefender Core? 2097
What are the List of folders that should be removed after MetaDefender Core uninstallation on Windows? 2097
How to Sanitize Non-Compliant MDZIP Files? 2098
How do I receive MD Core scan results using a static callback URL without specifying the URL in request headers? 2098
What is the difference between the Archive Extraction Engine V.S. Metascan Antivirus engines extraction? 2102
Why does MetaDefender Core Identifys Only the First Part of a Split Archive as an Archive File? 2105
Where can I find sample logs for quarantine errors? 2108
Can Syslog Reliably Detect MetaDefender Core Process Crashes? 2110
Can MetaDefender Core Scan Embedded Payloads or Inline File Content? 2112
How can I export complete archive scan results from MetaDefender Core using the REST API (session_id + extraction APIs)? 2115
Can MetaDefender Core Process Microsoft Purview Information Protection Encrypted Files? 2120
Why does MetaDefender Core not quarantine the same malicious file twice? 2121
When Is the “Database Maintenance Required” Notification Triggered, and How Does the ometascan-vacuum-db Tool Improve Performance? 2122
How to Allow Encrypted Archives in MetaDefender Core? 2124
What are the pros and cons of using MetaDefender Core's archive extraction engine vs. AV engine archive handling? 2125
Will adding a new Ethernet Interface to the MetaDefender Core machine impact its functionality? 2126
Does MetaDefender Core support HTTP2 Protocol? 2127
How to track Settings changes in MetaDefender Core 2128
How Does OPSWAT Determine Which Files Are Acceptable for Content Disarm and Reconstruction (CDR) 2129
Does MetaDefender Core support scanning of .parquet.gpg files? 2130
Using the metadata Header to Track API File Submissions in MetaDefender Core 2131
How is the Main Result Determined in MetaDefender Core Scanning? 2138
What do “Potentially Infected Files” mean in MetaDefender Core? 2139
Does OPSWAT MetaDefender Core Support Scanning TXF Files? 2141
Why Are Office Documents Counted as Multiple Files During Archive Extraction? 2143
How to View SSL/TLS Certificate Details from the Command Line (Windows and Linux) 2144
When should I apply the new license? 2146
How to check whether AV engines are running the latest update from the OPSWAT update server? 2147
How to Use a RAMDISK for the tempdirectory? 2149
Why does ClamAV seem to be slower than other engines? 2153
How does MetaDefender Core determine the runtime nginx.conf location on Windows? 2154
How do I obtain a Trial License for MetaDefender Core? 2157
How long is the support lifecycle for a specific version of MetaDefender Core? 2159
How do I get a False Detection (False Positive or False Negative) corrected? 2162
What is Included in the MetaDefender Core Support Package? 2164
What dependencies does MetaDefender’s Deep CDR for Linux have? 2166
What links, target-services or target host-IPs need to be allowed for Metadefender Core to function accurately? 2169

How do I set up exclusions on my anti-malware software to prevent disruption of the MetaDefender Core scanning process, when my corporate policy does not allow it? 2170
How do I deploy MetaDefender Core to an offline Windows environment? 2171
How do I deploy MetaDefender Core to an offline Linux environment? 2178
Does the Deployment ID change at any time? 2184
How do I activate my offline deployment of MetaDefender Core? 2186
How do I activate/deactivate a MetaDefender Core Deployment via API calls? 2189
How do I update to a new license key on a Metadefender Core server? 2191
What types of licenses are available for MetaDefender Core? 2193
After license renewal does the product activation happen automatically (if the license key is the same)? 2194
How to install .NET 8 Runtime dependencies on Linux Distributions? 2195
How to connect LDAP via SSL on Linux 2197
How to Backup and Restore standalone database for MetaDefender Core? 2198
How to install msttcore-fonts on Linux systems? 2203
How to upgrade Amazon Corretto JRE? 2205
Can I activate the product license before the start date? 2205
How to completely uninstall MetaDefender Core? 2206
How to Trust MetaDefender Core's Self-Signed Certificate When HTTPS Is Enabled? 2208
How are MetaDefender Core AWS AMI images secured and updated? 2209
How to handle high CPU load on MetaDefender Core server? 2209
What will happen to the MD core after the license is expired? 2212
What are the List of folders that should be removed after MetaDefender Core uninstallation on Linux? 2213
How can I fix a failed or looping MetaDefender Core module update by clearing the package cache? 2213
If a New AV Engine Is Released for MetaDefender Core, Is It Automatically Included in My License? 2216
How can performance of MetaDefender Core be increased when deployed on K8S environments? 2216
Why Is MetaDefender Core Consuming More License Slots Than Expected After Activation? 2217
Can MetaDefender Core use Azure SQL Managed Instance instead of PostgreSQL? 2219
Can I delete the ignition file after MetaDefender Core is deployed? 2219
How to migration MD Core postgresql database deployed on dedicated PostgreSQL server? 2220
Is there a virus test I could use to test MetaDefender Core engines? 2221
How do I check if "noexec" flag exists on a Linux OS? 2222
How can I run tests to see the different scan results on MetaDefender Core? 2223
Are scan engine updates automatic? 2225
What operating system patches should be applied to the system hosting MetaDefender Core? 2226
How do I check the update status of MetaDefender Core's licensed AV scan engines? 2227
What is the frequency of signature/definition updates on MetaDefender Core? 2230
Why does the Deep CDR engine keep failing to update? 2232
What Happens After the Discontinuation of Kaspersky Support for Windows and Linux? 2233
How to Troubleshoot an Error related to Connection to Database Failing after an Upgrade to v5.11.0? 2234
Does a CVE affect MetaDefender Core? 2238
End-of-Life Support for CentOS 7 and RedHat 7 in MetaDefender Core and Engines 2241
End of Life Support for Ubuntu 20.04 and Amazon Linux 2 in MetaDefender Core 2243

How will End of Life Support for Windows Server 2016 and Windows 10 versions before 21H2 in MetaDefender Core affect my deployments? 2245
Will MetaDefender Core Experience Downtime During Engine Updates? 2247
How to Update Antivirus Engines in MetaDefender Core Manually? 2248
What Happens After the Discontinuation of VirusBlokAda and NanoAV Support for Windows and Linux? 2253
What Happens After the Discontinuation of RocketCyber Support for Windows and Linux? 2254
Is MetaDefender Core affected by NGINX CVEs? 2256
End of Life Support for Debian 11 in MetaDefender Core 2256
Is MetaDefender Core Affected by PHP2Shell (CVE-2026-109618 and CVE-2026-109619)? 2258
How do I manually upgrade to the latest MetaDefender Core version? 2259
Where can I get old engine versions? 2287
Are older antivirus engine module versions available for downgrade? 2287
[K8S] Upgrade of existing MD Core stack to a newer version by retaining existing shared database 2289
How do I plan and perform a MetaDefender Core version upgrade? 2300
How to recover MetaDefender Core administrator access using an existing admin API key 2307

Troubleshooting 2312

Troubleshooting MetaDefender Core 2312
Which data is contains in a support package 2312
How to Create Support Package With Web Console 2315
How to Create Support Package With Bundle Tools? 2318
Windows Temporary Folder Path Changes for SYSTEM Account 2321
PostgreSQL Connection Loss After Upgrade on CIS-Hardened Ubuntu 2322
Debug logging 2323
How to Read the Metadefender Core Log? 2324
Inaccessible Management Console 2325
Slow or Inaccessible Management Console 2325
Open Connection On PostgreSQL Server 2326
Too Many Sockets or Files Open 2328
Too Many TIME_WAIT Socket 2329
PIN Code Reset Tool 2330
Engine Clean-up Tool 2331
TCP/IP Port Exhaustion 2335
Archive Compression Engine Permanently Failed on MetaDefender Core v4 2337
Enabling metadata Display in Logs 2339
Troubleshooting: Scan history not updating after upgrading MetaDefender Core - Linux 2340
HTTPS Failure on MetaDefender Core 5.2.0 (or newer) 2340
HTTPS Configuration Removal Tool 2342
Customize Internal PostgreSQL User 2342
PostgreSQL User Privilege Requirements 2344
Remote PostgreSQL User Privileges Checking Tool 2349
Database Cleanup Tool in Shared DB Mode 2351

Rectify malformed FileType data in PostgreSQL database 2353
Migrate MetaDefender Core from one machine to another machine 2355
Working with self-signed certificate 2358
Authentication Failure After Upgrading From Core 5.10.1/5.11.0 2358
MetaDefender Core and Out-of-Support Windows OS 2363
SSO Login Failed: Domain or IP Not Allowed 2363
SSL Handshake Failures when fetching Azure Entra URL in SSO configuration 2366
Database dead-tuple alert: what it means and how to tune PostgreSQL autovacuum 2370
MetaDefender Core Deployment Readiness Tool 2373
Upgrade to MetaDefender Core 5.21.0 or later on Windows systems with a non-ASCII locale 2378
PostgreSQL fails to start after upgrade when SSL is enabled 2381
License activation or engine download fails on a clean Windows Server environment 2383

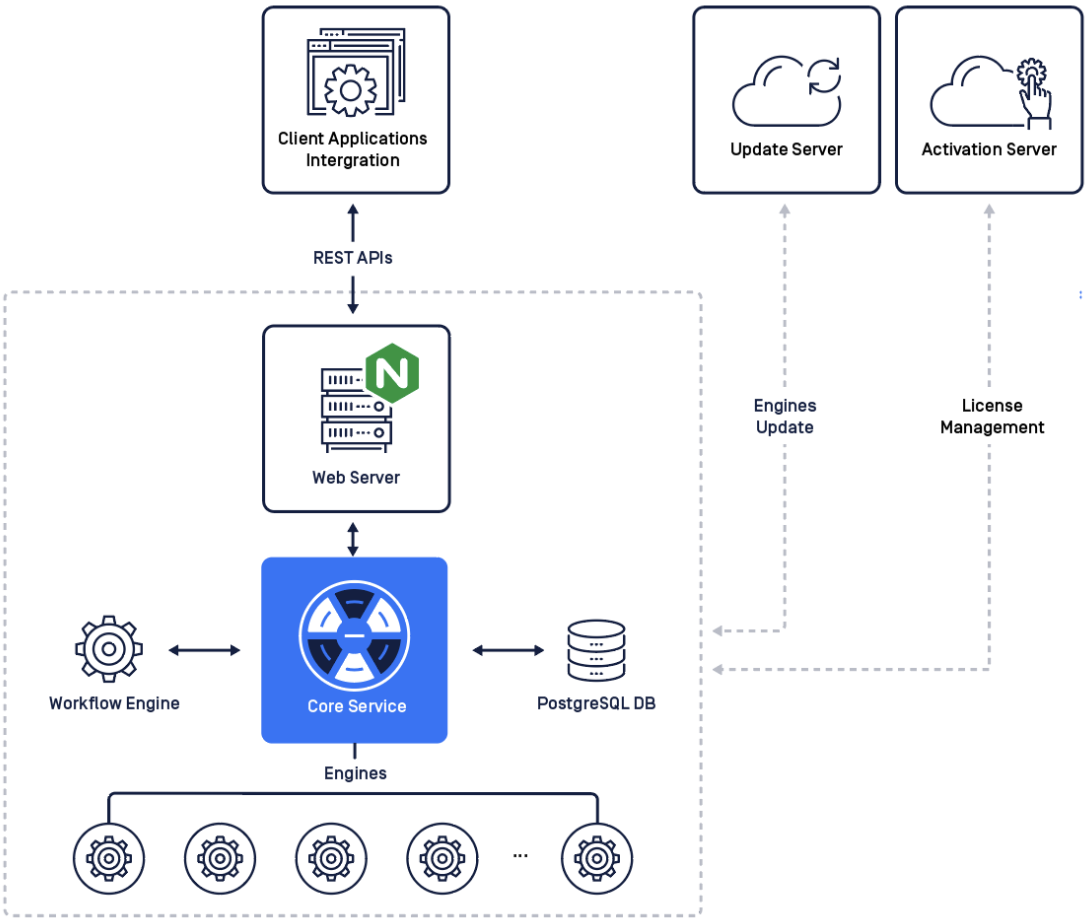
MetaDefender Core CLI (mdcli) **2385**

MDCLI User Guide 2385
Release Notes 2408

API Reference

MetaDefender Core API **2409**

Recommended System Configuration



MetaDefender Core supports a wide range of deployment options, including air-gapped environments, providing the flexibility to choose the environment that best suits customers' requirements and infrastructure.

- Physical server.
- Virtualization platforms like VMware, Hyper-V and XenServer.
- Cloud Deployment.
- Container Deployment.
- MetaDefender Cluster Deployment.

Info

Before installing MetaDefender Core, please refer to the recommended minimum system configuration listed below. The requirements are based on a standardized customer deployment and may require more or less resources to achieve the level of performance desired in your environment. Please note that the server specifications are built to allow a high volume daily processing. Please refer to Performance and Load Estimation for more details.

The recommendations below are for MetaDefender Core, API usage only. For any other use cases, please consult the user guide of the licensed products for accurate recommendations.

For certain use cases these might be adjusted and customized on specific needs and SLAs. We highly recommend to engage our ProServ team to assist in fine tuning MetaDefender and get the maximum performance out of your systems.

Microsoft Windows Deployments

[Windows] Supported Operating Systems

Only support 64-bit platforms.

- Windows Server 2019, 2022, 2025
- Windows 11 only for End-User systems (e.g. Kiosk deployments):
  - 11 23H2
  - 11 IoT (22H2)
  - 11 24H2 and 11 IoT 24H2
  - 11 25H2 and 11 IoT 25H2
  - Based on Microsoft License Terms, you're not allowed to use Windows Desktop as a server, therefore, MetaDefender Core can't be used on these systems for server use cases.

Fast Startup mode is unsupported.

Info

End-customer is responsible of verifying the OS license agreement and choose the right OS based on their planned usage of MetaDefender.

Info

OPSWAT has discontinued support for Windows Server 2016 and Windows 10 versions prior to 21H2 in MetaDefender Core and its associated engines starting October 2025.

More details at End of Life Support for Windows Server 2016 and Windows 10 versions before 21H2 in MetaDefender Core.

[Windows] Recommended System Configuration

The resources listed below (CPU, RAM, disk space) are the minimum recommended for MetaDefender Core:

Package	CPU cores	Free System RAM	Free Disk Space
MetaDefender Core 8	8	8 GB	16 GB (product and

Package	CPU cores	Free System RAM	Free Disk Space
			engines) + 50GB (temp files during processing) + 100GB (when using local PostgreSQL database)
MetaDefender Core 12	16	16 GB	24 GB + 50 GB + 100 GB
MetaDefender Core 16	16	16 GB	32 GB + 50 GB + 100 GB
MetaDefender Core 20	32	16 GB	40 GB + 50 GB + 100 GB
MetaDefender Core MAX	32	32 GB	120 GB + 50 GB + 100 GB

- SSD is recommended for system where MetaDefender Core is installed.
- See more disk space details in Storage Usage Information.
- For virtualization deployments such as VMware, Hyper-V, and XenServer, we recommend static resource allocation for CPU, RAM, and SSD storage to prevent stability issues.
- For any additional AV custom engine, add at least 1 GB of additional RAM and 1 additional CPU cores.
- If Deep CDR and/or Proactive DLP are licensed, add at least 8 GB of additional RAM and 4 additional CPU cores (for a total of at least 12 CPU cores).
- If Embedded Adaptive Sandbox licensed, add at least 8 GB of additional RAM and 4 additional CPU cores (for a total of at least 12 CPU cores).
- SBOM processing requires additional disk space. Ensure at least 15 GB of free disk is available for the engine to operate properly.

[Windows] Third Party Dependencies

- .NET framework 4.5 or above
- Microsoft Visual C++ Redistributable for Visual Studio 2015–2022 (x64), version 14.38 or later.
- PostgreSQL: Only applicable when using a pre-installed PostgreSQL server running remotely.
  - 14.5 - 14.23
  - 15.7 - latest 15.x version
  - 16.4 - latest 16.x version
  - 18.4 - latest 18.x version

Further modules (engines) also have dependencies as described below.

Engine Name	Dependency
Archive engine	Microsoft Visual C++ 2015 - 2022 Redistributable x86 (Required for YZ1 extraction/compression)
Bkav Pro	Microsoft Visual C++ 2015 - 2022 Redistributable x64
Deep CDR	Microsoft Visual C++ 2017 Redistributable Package (Only applicable to engine version 5.8.0 or above)
Embedded Adaptive Sandbox	Amazon Corretto 17Visual C++ Redistributable for Visual Studio is required for versions prior to 2.2.0. For more details: Dependencies and system Requirements
ESET	MetaDefender Core temporary directory should have more than 200MB free disk space
Proactive DLP	CPU must support AVX2 and SSE4.1 instruction set to use OCR feature
Reputation	Visual C++ Redistributable for Visual Studio version 14.38 or higher. CPU must support AVX2 (BMI2) instruction set.
Scrutiny	Microsoft Visual C++ 2015 - 2019 Redistributable x86 and x64
Systweak	.NET runtime 6.0, 7.0 or 8.0

[Windows] Installation Details

MetaDefender Core on Windows uses "C:\Program Files\OPSWAT" folder for storing resources or the installation directory.

MetaDefender will use its resources folder to store temp files as part of the analysis.

It's recommended to exclude this folder from real-time protection monitoring. Knowledge base:

- How to set up exclusions on anti-malware software to prevent disruption
- Mitigating scan failure triggered by real-time protection service
- Can local AVs interrupt ongoing scans?

Linux Deployments

[Linux] Supported Operating Systems

Only support 64-bit platforms.

- Red Hat Enterprise Linux 8, 9
- Rocky Linux 9
- Oracle Linux 9 (version 9.5 or later)
- Debian 12, 13
- Ubuntu 22.04, 24.04
- Amazon Linux 2023 is supported via Amazon Machine Image (AMI)

The product may not function correctly with Red Hat Enterprise Linux or Rocky Linux operating systems running old kernel version such as 372.9, 372.13.

The recommended solution is to ***upgrade to at least kernel version 372.26***. This should help ensure proper functionality.

Plan for legacy OS support

OPSWAT will discontinue support for the following OS in MetaDefender Core and its associated engines:

- Ubuntu 22.04 since **Apr 2027**.

Info

Some AV engines could be failed to initialize and run on MetaDefender Core (e.g. ESET, Kaspersky) due to execution permission required by them on `/var/tmp/ometascan` folder.

Please follow steps at Why have the ESET and Kaspersky scan engines failed to initialize on the hardened Linux OS? for remediation details.

Info

End-customer is responsible of verifying the OS license agreement and choose the right OS based on their planned usage of MetaDefender.

[Linux] Recommended System Configuration

The resources listed below (CPU, RAM, disk space) are the minimum recommended for MetaDefender Core:

Package	CPU cores	Free System RAM	Free Disk Space

Package	CPU cores	Free System RAM	Free Disk Space
MetaDefender Core 5	4	4 GB	10 GB (product and engines) + 50GB (temp files during processing) + 100GB (when using local PostgreSQL database)
MetaDefender Core 10	8	8 GB	20 GB + 50 GB + 100 GB
MetaDefender Core MAX	16	16 GB	40 GB + 50 GB + 100 GB

- SSD is recommended for system where MetaDefender Core resides on.
- See more disk space details in Storage Usage Information.
- For virtualization deployments such as VMware, Hyper-V, and XenServer, we recommend static resource allocation for CPU, RAM, and SSD storage to prevent stability issues.
- For any additional AV custom engine, add at least 1 GB of additional RAM and 1 additional CPU cores.
- If Deep CDR and/or Proactive DLP are licensed, add at least 8 GB of additional RAM and 4 additional CPU cores (for a total of at least 12 CPU cores).
- If the Embedded Adaptive Sandbox is licensed, add at least 8 GB of additional RAM and 4 additional CPU cores (for a total of at least 12 CPU cores).
- SBOM processing requires additional disk space. Ensure at least 15 GB of free disk is available for the engine to operate properly.

[Linux] Third Party Dependencies

Dependencies list:

- openssl
- grep
- lib32stdc++6 (>= 4.5)
- libc6-i386 (>= 2.10)
- procps
- zlib1g
- libcurl3 (>= 7.19.7)
- libcurl4
- tar (Ubuntu/Debian)
- lsb_release (Rocky Linux 9.x/Red Hat 9.x)

- postgresql - only applicable when using a pre-installed PostgreSQL server running remotely
  - 14.5 - 14.23
  - 15.7 - latest 15.x version
  - 16.4 - latest 16.x version
  - 18.4 - latest 18.x version

Not all the above dependencies will need to be installed, it is dependent on different Unix distro and version

Some engines also have dependencies as described below:

Engine name	Dependency
Archive engine	To extract password protected MS Office files, you need to install .NET 8 dependencies Rocky Linux 9.x/Red Hat 9.x: libnsl
Country of Origin	libnsl Amazon Linux 2023: libxcrypt-compat
Deep CDR	• Rocky Linux 9.x: libgomp, ncurses-compat-libs, libnsl • CentOS: libgomp, ncurses-compat-libs • Red Hat 8.x: libgomp, ncurses-compat-libs, libnsl • Red Hat 9.x: libgomp, ncurses-compat-libs, libnsl • Ubuntu/Debian: libgomp1, libncurses5 • .NET 8 dependencies
Embedded Adaptive Sandbox	openjdk-17-jre For more details: Dependencies and system Requirements
FileType engine	libnsl Amazon Linux 2023: libxcrypt-compat
Proactive DLP	mscorefonts .NET 8 dependencies CPU must support AVX2 and SSE4.1 instruction set to use OCR feature

Engine name	Dependency
Reputation	CPU must support AVX2 (BMI2) instruction set
RocketCyber	libgomp

Note: Some dependencies may need to be installed from external repositories, not from the OS default repositories.

[Linux] Installation Details

MetaDefender Core default installation path is using /var folder for storing resources:

- `/var/lib/ometascan` : installation folder with all its resources (database, DLP processed / CDR sanitized / quarantined file storage, engine package and definition updates).
- `/var/log/ometascan` : application logs.
- `/var/run/ometascan` : runtime folder for Core related processes.
- `/var/lib/ometascan` : all successful deployed engines data files.
- `/usr/lib/ometascan` : all shared libraries required by MetaDefender Core, bundled PostgreSQL server related binaries and libraries.
- `/var/tmp/ometascan/resources` : all temporary processing files for MetaDefender Core to process.

Storage Usage Information

Based on the configuration, MetaDefender Core will need additional disk space to store analysis/result data, quarantined files and temporary files:

	Data Storage
Analysis Reports (Processing History)	Approximately 5GB is required for every 1M analysis reports stored. • 100 GB is recommended as a baseline. Database storage requirements are affected by Data Retention settings, engine settings and scanning load.
Quarantined Files	Depends on the customers' dataset and configuration
Sanitized Files	Depends on the customers' dataset and configuration
Temporary processing files	Depends on the customers' dataset and configuration • A 1GB non-archive file requires 1GB free disk space at least.

	Data Storage
	• A 1GB archive file requires (1 * 2)GB free disk space at least. • 50GB is a baseline and should be adjusted based on observation of the solution within the environment, with consideration to peak times and overall file makeup (e.g. complexity, size, archives).
Note	Options are available to control how large and/or complex files are handled to limit the amount of system resources required. See Workflow Configuration Template for more information.

Custom Engines

The recommendations above are specific for MetaDefender pre-packaged bundles.

However for additional Custom Engines, please review the Knowledge base to review additional requirements (if any) for the selected engine.

Browser Requirements for the MetaDefender Core Management Console

One of the following browsers is suggested to view the MetaDefender Core Management Console:

- Microsoft Edge
- Chrome
- Firefox
- Safari

Chrome, Firefox, Safari and Microsoft Edge browsers are tested with the latest available version at the time of release.

Network Bandwidth

In online environment where MetaDefender Core is configured to download engine update packages directly from OPSWAT update server source, then using gigabit network speed (125 MB/sec) is highly recommended to avoid potential network related issue while downloading files, or at least 100 Mbps (12.5 MB/sec) network speed.

If you have installed or if you wish to use the MetaDefender Core in a restricted environment, you will have to allow access to the following hosts:

- https://activation.dl.opswat.com - for license online activation.
- https://update.dl.opswat.com - for engines and databases download.

Note: IP address-based allowlisting on your firewall might fail after some time since OPSWAT uses CDN (AWS Cloudfront) to faster delivery updates over the world, and IP address of edge servers might change over time.

Depending on your location, connections with OPSWAT cloud-based servers be forwarded to specific AWS Cloudfront's edge locations, each is assigned with its own unique IP address ranges. You might also need to allow correct AWS Cloudfront's corresponding IP address ranges by referring to https://ip-ranges.amazonaws.com/ip-ranges.json

Info

Even the OPSWAT update servers host updates for all of the available engines we support, sometimes custom engines might try to connect to their own cloud for updates, but this can be disabled in firewall and they will be updated just from OPSWAT cloud servers instead.

Installation

Starting with MetaDefender Core 4.20.0, the centralized/shared database deployment model is supported (all MetaDefender Core instances share the same database), along with traditional standalone database mode (every MetaDefender Core instance has its own database to manage).

- Standalone mode
- Shared Database Mode

Installation

1. Download the package of your choice from My OPSWAT
2. Install the package on your computer via the Command Line or via the Wizard
3. Open a web browser and point to http://<server name or IP>:<port>
   - The default port is **8008**
   - In case of problem check Inaccessible Management Console page
4. Complete the required steps of the basic configuration wizard
5. You must activate this deployment to use its features

Installation notes

- If the MetaDefender Core package **dependencies are not installed** on your system you may need to have a working Internet connection or you may have to provide the Installation media during the installation. Consult your Operating System documentation on how to use Installation media as a package repository.
- During installation the databases might need to be upgraded. This could take noticeable time depending on database size (eg.: length of scan history).

Standalone mode

Windows

Wizard Installation

The Install Wizard is only for the Windows installer (.msi file).

Since MetaDefender Core 4.19.0, we have switched to using PostgreSQL DBMS which needs a dedicated database server. MetaDefender Core supports two modes to setup PostgreSQL server:

Local PostgreSQL server

This is a bundled PostgreSQL. MetaDefender Core will itself install and setup a new PostgreSQL server locally in the same box with the product.

PostgreSQL user credentials

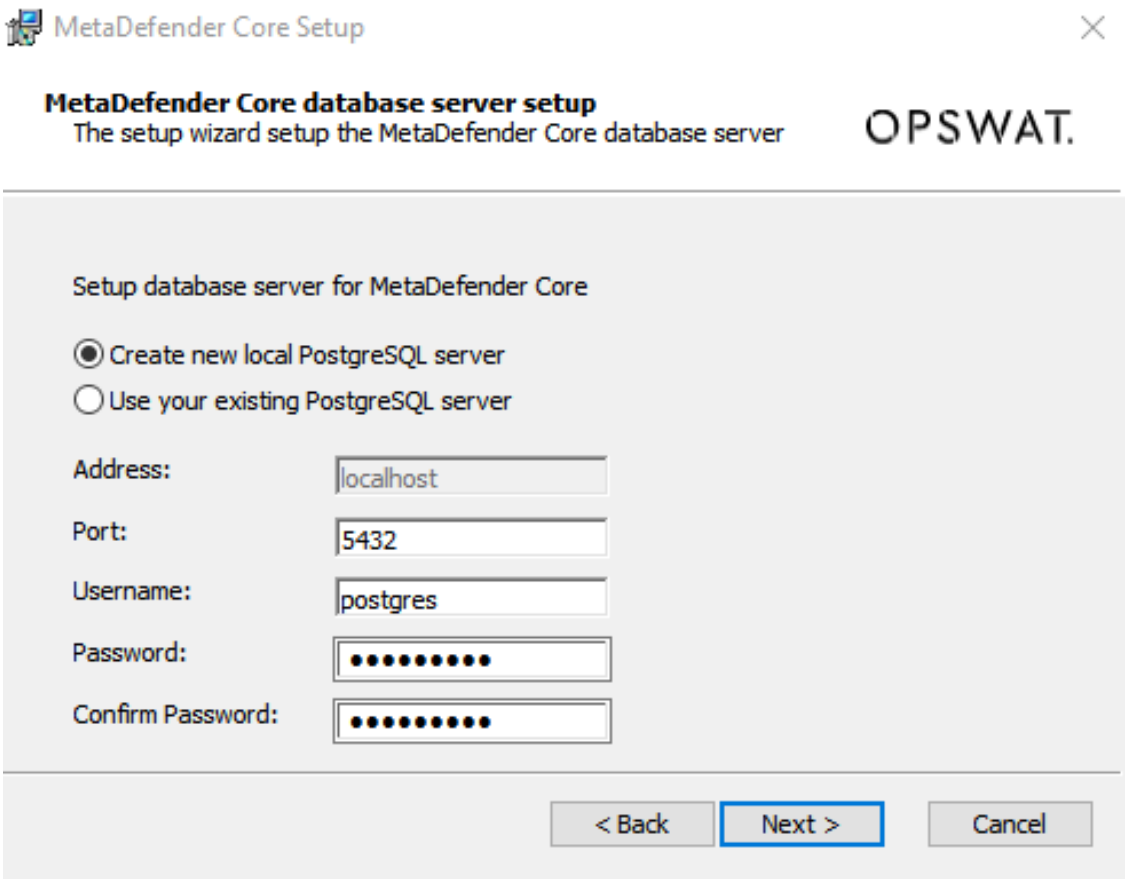
Unicode characters are NOT supported for "Username" and "Password"

"Password" does not accept # (hash) as starting character (e.g. #ABC)

The ignition file

If the ignition file exist in the designated location, then MetaDefender Core will use the details defined in the ignition file, rather than what is passed via the installation wizard.

More info about the ignition file based installation: Command Line Installation



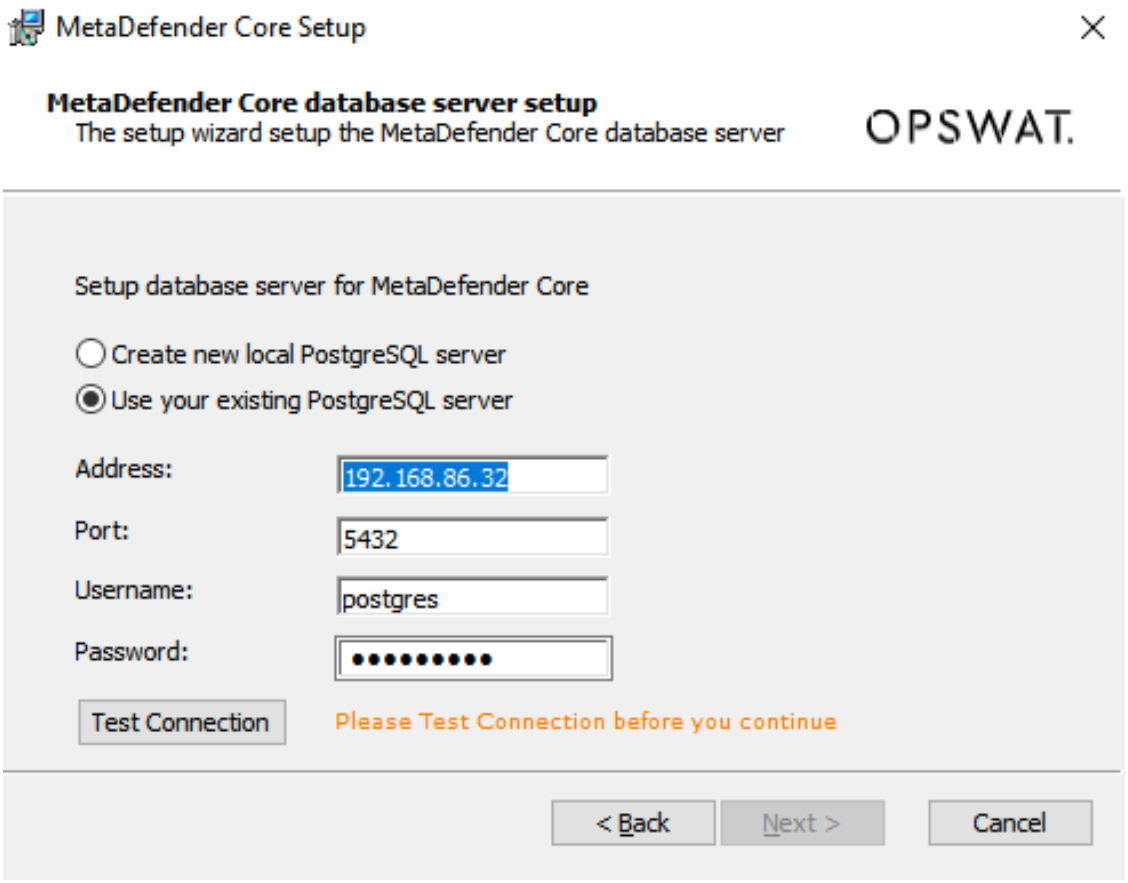
Remote PostgreSQL server

> **Info**
>
> **PostgreSQL version 14/15/16 is required.**

MetaDefender Core will connect to a pre-installed PostgreSQL running remotely.

"Test Connection" button is required, to make sure the PostgreSQL is connected and authenticated successfully. Make sure to check and perform steps at Open Connection On PostgreSQL Server if you are unable to test the connection with remote PostgreSQL server.

> **PostgreSQL user privilege requirements**
>
> MetaDefender Core will need to create and use dedicated PostgreSQL users for database setup, upgrade and its own operations. Those users requires certain privileges configured on PostgreSQL server, otherwise expecting failures to occur while installing and operating MetaDefender Core.
>
> Details: PostgreSQL User Privilege Requirements



Command Line Installation

Ignition File

> **Note**
>
> For command line installation, it is **mandatory** to use MetaDefender Core ignition file to add PostgreSQL server information **BEFORE** installing MetaDefender Core 4.19.0 or newer.
>
> Location: **C:\OPSWAT\ometascan.conf**

MetaDefender Core supports two modes to setup PostgreSQL server:

Local PostgreSQL server

MetaDefender Core will install a new PostgreSQL server locally, next to the product.

A sample iginition file for PostgreSQL server information. **Only non-Unicode characters are supported for "user" and "password"**):

ometascan.conf · bash

```
[dbserver]
type=local
host=localhost
port=5432
```

```
user=postgres
password=non_Unicode_password
```

Remote PostgreSQL server

> **Info**
>
> **PostgreSQL version 14/15/16 is required.**

MetaDefender Core will connect to a pre-installed PostgreSQL server running remotely and will setup its database.

A sample ignition file for PostgreSQL server information:

ometascan.conf · bash

```
[dbserver]
type=remote
host=192.168.86.32
port=5432
user=existingserver_admin_user
password=existing_server_pass
```

How to setup a remote PostgreSQL server

1. Download and install PostgreSQL server version 14.x for Windows x64 at https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
2. Configure PostgreSQL for remote access so that your remote existing PostgreSQL server to accept upcoming connection from MetaDefender Core:

2.1) Modify the **pg_hba.conf** file**:** file resides in the PostgreSQL data directory. Common locations include: **C:\Program Files\PostgreSQL\14\data**(adjust the version number if needed).

pg_hba.conf · bash

```
host    all             all             0.0.0.0/0            scram-sha-256
```

2.2) Modify the **postgresql.conf** file. This file is in the same data directory as **pg_hba.conf**

postgresql.conf · bash

```
listen_addresses = '*'
```

3. Restart **postgres** service.

Install MetaDefender Core

On Windows systems, it is possible to install the product by executing the corresponding .msi file.

From the command line interface, it is also possible to put the following parameters

```bash
msiexec /qn /i <msi file name> <option key>=<option value>
```

where the possible keys and their default values are the following:

Key	Default Value	Description
INSTALLFOLDER	\Program Files\OPSWAT\MetaDefender Core	Customize installation folder for product Example: `INSTALLFOLDER="D:\Products"`
RESTADDRESS	•	REST interface binding IPv4 or IPv6 address ('*' means that service listens on all IPv4 and IPv6 interfaces)
RESTPORT	8008	REST interface binding port
REPORT_ENGINE_ISSUE	true	Enable reporting of engine issue count. (possible values: "true" or "false").

Note

Reporting of engine issue count

If reporting of engine issue count is enabled, MetaDefender Core server will send only the **number** of initialization errors and **number** of unexpected stops for the specific db/engine version. This information is sent over a HTTPS channel when the product downloads the latest package descriptors. This information is used for early detection of any specific 3rd party engine quality issues.

For details on using msiexec, please consult Windows installer documentation.

Linux

Only supports command-line installation.

Ignition File

Note

For command line installation, it is **mandatory** to use MetaDefender Core ignition file to add PostgreSQL server information **BEFORE** installing MetaDefender Core 4.19.0 or newer.

Location: **/etc/opswat/ometascan.conf**

MetaDefender Core supports two modes to setup a PostgreSQL server:

Local PostgreSQL server

MetaDefender Core will install a new PostgreSQL server locally, next to the product.

A sample ignition file for PostgreSQL server information. **Only non-Unicode characters are supported for "user" and "password"**):

ometascan.conf · bash

```
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
```

Remote PostgreSQL server

Info

PostgreSQL version 14/15/16 is required.

MetaDefender Core will connect to a pre-installed PostgreSQL server running remotely and will setup its database.

A sample ignition file for PostgreSQL server information:

ometascan.conf · bash

```
[dbserver]
type=remote
host=192.168.86.32
port=5432
user=existingserver_admin_user
password=existing_server_pass
```

How to setup a remote PostgreSQL server

Debian / Ubuntu

1. Update System Packages

Code · bash

```
sudo apt update
sudo apt upgrade -y
```

2. Add PostgreSQL APT Repository

Code · bash

```
sudo apt install -y wget ca-certificates
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo tee /etc/apt/
trusted.gpg.d/pgdg.asc
echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -c | awk '{print $2}')-pgdg
main" | sudo tee /etc/apt/sources.list.d/pgdg.list
```

3. Install PostgreSQL

Code · bash

```
sudo apt update
sudo apt install postgresql-14
```

4. Once installed, the PostgreSQL service should start automatically. You can verify that it is running:

Code · bash

```
sudo systemctl status postgresql
```

To start or restart the PostgreSQL service manually, you can run:

Code · bash

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
sudo systemctl status postgresql
```

5. By default, the Postgres user is configured without a password. Run the following command to set a password for the Postgres user:

Code · bash

```
sudo -u postgres psql
postgres=# alter user postgres with password 'password';
```

6. Install needed extensions:

Code · bash

```
sudo -u postgres psql
postgres=# CREATE EXTENSION IF NOT EXISTS pg_trgm;
postgres=# CREATE EXTENSION IF NOT EXISTS btree_gin;
```

7. Configure PostgreSQL for remote access

Code · bash

```
sudo nano /etc/postgresql/14/main/postgresql.conf
# Change the following line with your server IP address:
listen_addresses = 'your-server-ip'
sudo nano /etc/postgresql/14/main/pg_hba.conf
# Change the following line
# Accept from anywhere
host all all 0.0.0.0/0 scram-sha-256
# Restart PostgreSQL service
sudo systemctl restart postgresql
```

Red Hat Enterprise Linux 8, 9 / Rocky Linux 9

Note

Due to the shorter support cycle on Red Hat Enterprise Linux, all supported versions of PostgreSQL are not available on this platform. The PostgreSQL Yum Repository will integrate with the normal systems and patch management, and provide automatic updates for all supported versions of PostgreSQL throughout the supported lifetime of PostgreSQL. Reference: https://www.postgresql.org/download/linux/redhat/

1. Install the PostgreSQL repository RPM:

Code · bash

```
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-
redhat-repo-latest.noarch.rpm
sudo dnf -qy module disable postgresql
sudo dnf install -y postgresql14-14.15 postgresql14-server-14.15
```

2. Run these CMDs to initialize the PostgreSQL:

Code · bash

```
sudo /usr/pgsql-14/bin/postgresql-14-setup initdb
```

```
sudo systemctl start postgresql-14
sudo systemctl enable postgresql-14
sudo systemctl status postgresql-14
```

3. By default, the Postgres user is configured without a password. Run the following command to set a password for the Postgres user:

Code · bash

```
sudo -u postgres psql
postgres=# alter user postgres with password 'password';
```

4. Install needed extensions:

Code · bash

```
sudo dnf install -y postgresql14-contrib
sudo -u postgres psql
postgres=# CREATE EXTENSION IF NOT EXISTS pg_trgm;
postgres=# CREATE EXTENSION IF NOT EXISTS btree_gin;
```

5. Configure PostgreSQL for remote access

Code · bash

```
sudo nano /var/lib/pgsql/14/data/postgresql.conf
# Change the following line with your server IP address:
listen_addresses = 'your-server-ip'
sudo nano /var/lib/pgsql/14/data/pg_hba.conf
# Change the following line
# Accept from anywhere
host all all 0.0.0.0/0 scram-sha-256
# Restart PostgreSQL service
sudo systemctl restart postgresql-14
```

A sample ignition file for PostgreSQL server information:

ometascan.conf · bash

```
[dbserver]
type=remote
host=192.168.86.32
port=5432
user=existingserver_admin_user
password=existing_server_pass
```

After installing MetaDefender Core successfully, you may want to remove all credentials info created for the [dbserver] section in the ignition file for security reasons.

> **Warning**
>
> If the MetaDefender Core package dependencies are not installed on your system, you may need to have a working Internet connection, or you may have to provide the Installation media during the installation. Consult your Operating System documentation on how to use Installation media as a package repository.

Install MetaDefender Core

Debian / Ubuntu package (.deb)

Code · bash

```
sudo dpkg -i <filename> || sudo apt-get install -f
```

Red Hat Enterprise Linux 8, 9 package / Rocky Linux 9 package (.rpm)

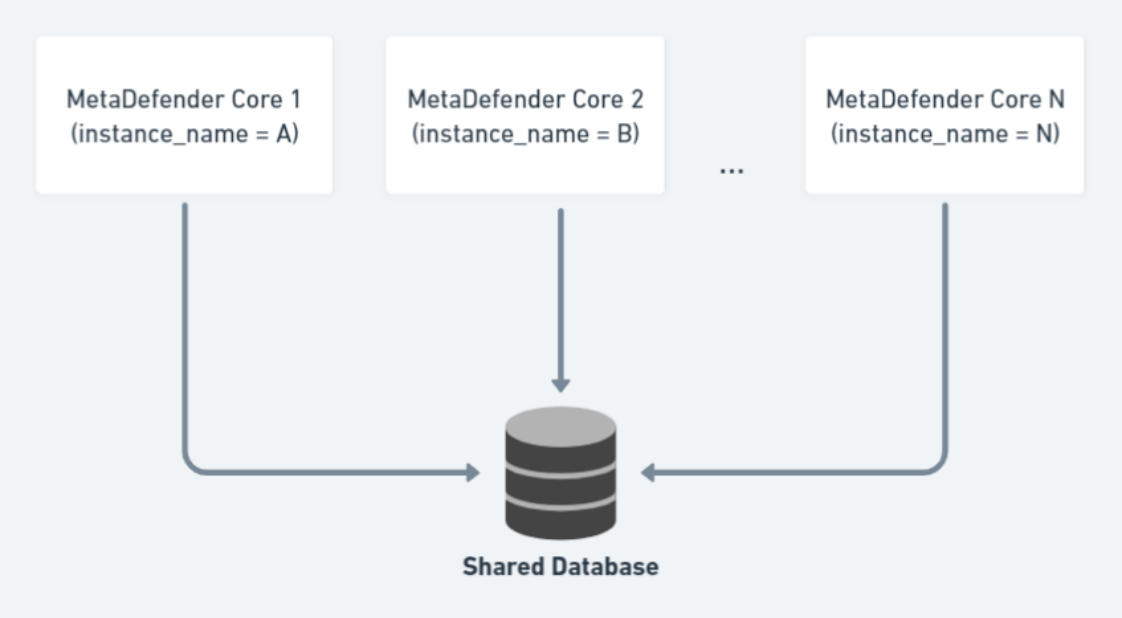
Code · bash

```
sudo dnf install -y <filename>
```

Shared Database Mode

Starting with MetaDefender 4.20.0, you can use a centralized database deployment model, where all MetaDefender Core instances can share the same external database.

That helps auto-sync all product configurations (**except Single-Sign-On** that requires unique settings for each Core), processing history, live/on-going scan result query, audit, statistics data.



This article demonstrates how to install and setup MetaDefender Core 4.20.0 or newer with shared database mode.

Important notes & current limitations:

Currently not supported

- Does not fully support Central Management use cases when using a shared database, compared to standalone database mode. For more details, refer to My OPSWAT Central Management.
- Does not support different versions of MetaDefender Core in shared DB mode. All instances must run the same version when operating in this mode.
- Does not support to switch from shared database mode back to standalone database mode.
- Does not support to change instance_name of MetaDefender Core in shared database deployment model after installation.
- Does not support upgrading older MetaDefender Core from non-persistent mode to using the shared database mode.
- Does not support upgrading older MetaDefender Core from standalone mode to using the shared database mode.
- Does not support to upgrade multiple MetaDefender Core instances at once in shared database mode
  - Stop all Core instances' scanning service before performing the upgrade.
  - Upgrade each Core server one by one instead.
  - For each Core, when the upgrade finished, then you can start using that Core for scanning without waiting for the other Core instances' upgrade to be done.

Limitations

- To synchronize scan results and enable clients to fetch scan results of an in-progress file from any MetaDefender Core instances behind a load balancer, PostgreSQL in shared mode has to receive a large amount of data from

many instances, and it may cause bottlenecks at database service. Memory usage on PostgreSQL may be high due to queuing data meanwhile resource on MetaDefender Core instances may not be effeciently utilized.

- The feature Reuse results by hash may not work smoothly.

Disclaimer

MetaDefender Core does not support mode switching.

Requirements for shared database setup:

- Using a dedicated existing PostgreSQL database server (see detailed requirements for PostgreSQL version at Recommended System Configuration).
- Before setting up, make sure to follow steps at Open Connection On PostgreSQL Server so that all MetaDefender Core instances could connect to the central remote PostgreSQL server.
- MetaDefender Core will need to create and use dedicated PostgreSQL users for database setup, upgrade and its own operations. Those users requires certain privileges configured on PostgreSQL server, otherwise expecting failures to occur while installing and operating MetaDefender Core, details: PostgreSQL User Privilege Requirements
- All MetaDefender Core instances must be installed on the same Operating System (Windows / Linux).
- The first MetaDefender Core instance connects to the shared database must be ready first (license activated, users/groups/roles created) before setting up other MetaDefender Core instances to connect and use that shared database.

Engine update:

- If engine updates is configured to be via Folder source, all MetaDefender Core instances must be able to access to the configured update folder, and the option DELETE FILES AFTER IMPORT will be unchecked and prevented for modification.

Authentication:

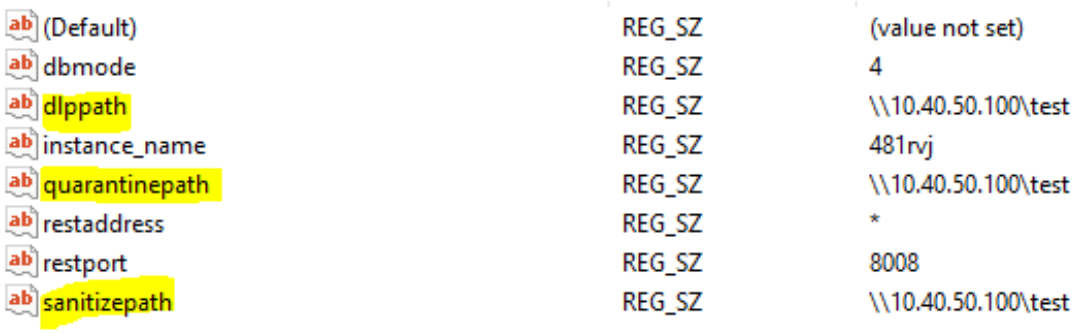
- Single-Sign-On (SSO) settings may be stored in the shared database and therefore visible across Core instances; however, SSO behavior depends on the login/redirect URL used to access Core.
  - If users access each Core instance via a different hostname/FQDN, you must configure SSO per URL (and register each corresponding redirect URI at the Identity Provider), otherwise sign-in may fail due to redirect URI mismatch.
  - If all users access Core only through a single, canonical load-balanced URL, a single SSO configuration that uses that load-balanced URL is supported, provided all nodes are consistently fronted by the same external URL (reverse-proxy headers/canonical URL handling) and authentication sessions are handled appropriately (e.g., sticky sessions if required, or shared authentication sessions where available).
- Version note: MetaDefender Core v5.17.1 introduces support for multiple login/redirect URLs for SSO directories, which can reduce the need for separate per-URL SSO directory configurations in multi-domain deployments.

- If an user is locked for login on a certain MetaDefender Core instance(e.g. exceeded the number of failed login times), that user is not prevented for logging in on any other instances.
- If an user is locked for sign-in on a certain MetaDefender Core instance (e.g. exceeded the number of failed login times), if the admin releases lockout on that user on that MetaDefender Core instance, that user is still not able to login on other MetaDefender Core instances.

Sanitized / DLP / Quarantined location changed after MetaDefender Core install:

Windows:

Shared storage path will be stored in 3 strings under key global in registry: quarantinepath, dlppath, sanitizepath as below



You can set either 1, 2 or 3 different paths for the 3 settings above (means that quarantined, DLP and sanitized files will be stored in same or different storages corresponding). In case shared storage is not set in installation, values of 3 settings above will be empty.

Restart MetaDefender Core service after changed.

Linux:

Open file `/etc/ometascan/ometascan.conf` and set shared storage path for 3 keys quarantinepath, dlppath, sanitizepath under section [global]

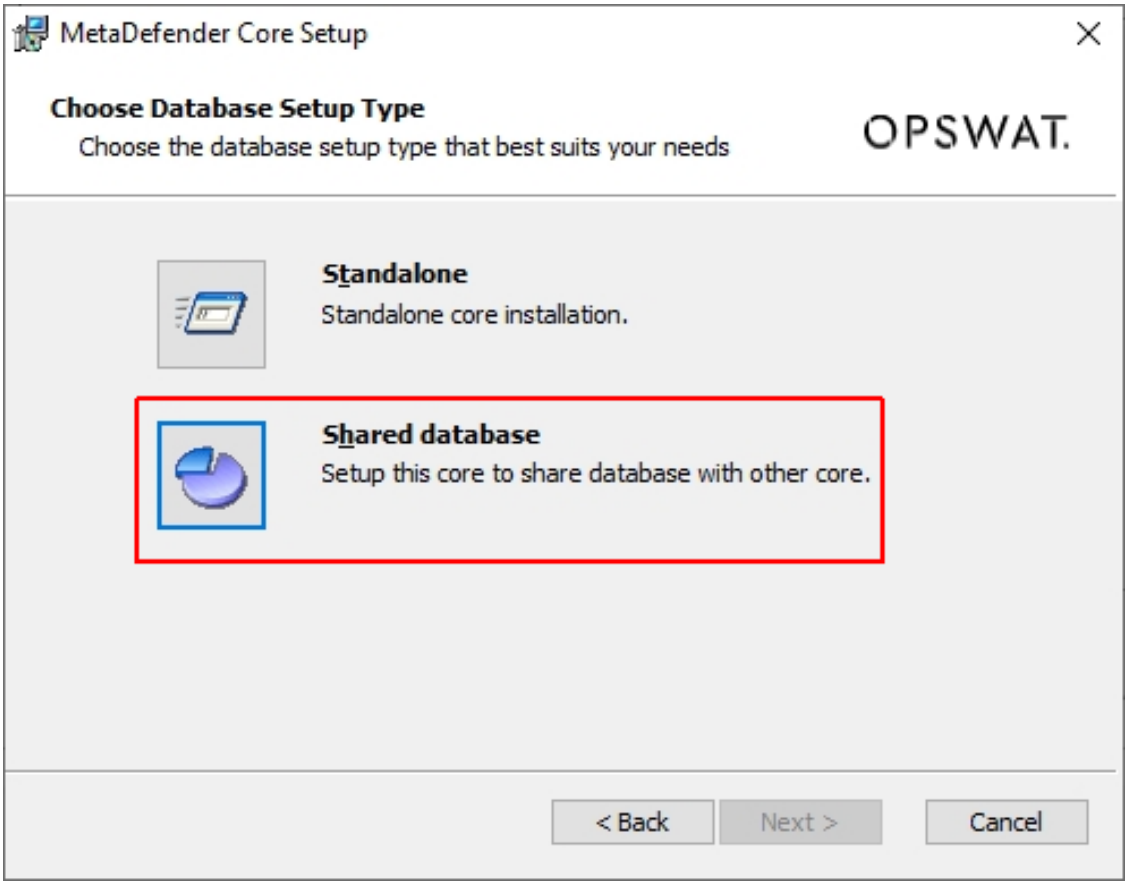
Restart ometascan service after changed.

Wizard Installation (Windows Only)

> **Info**
>
> Only applicable to MetaDefender Core 4.20.0 or newer using wizard on Windows OS.

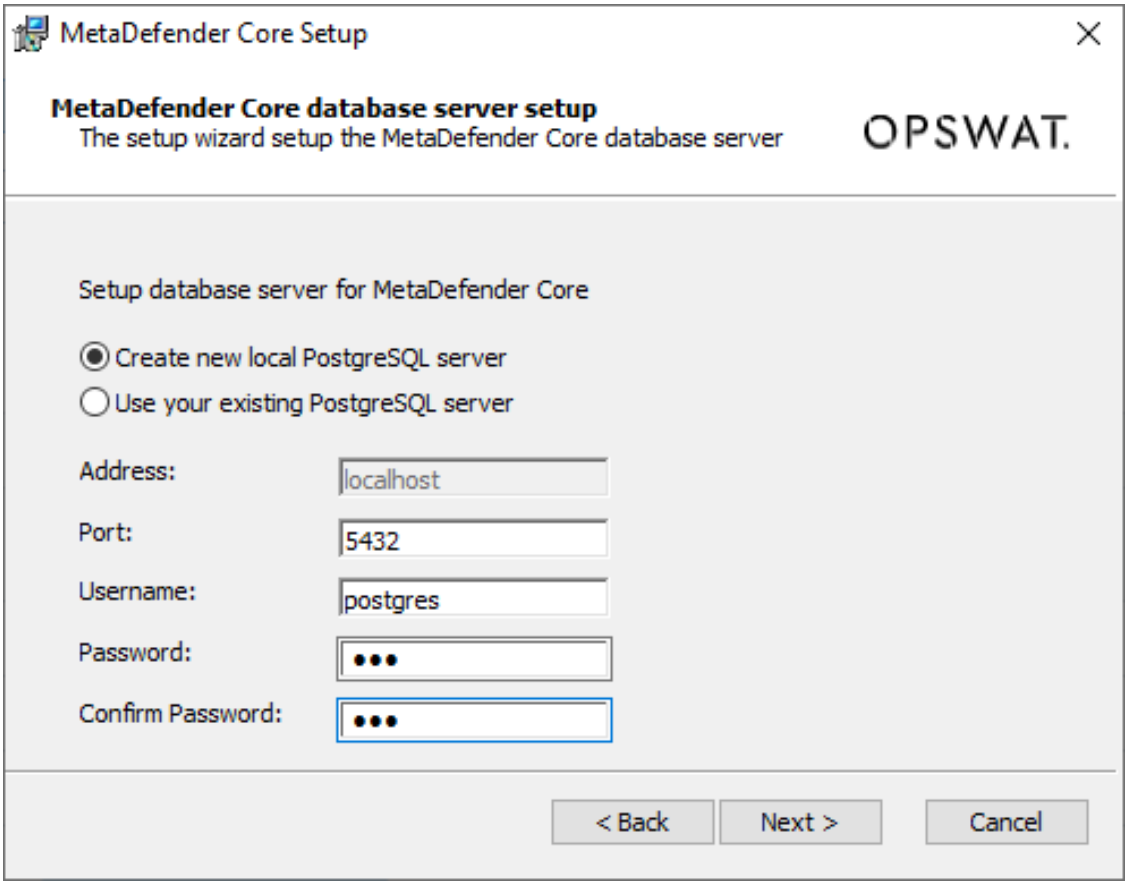
Clean install scenario

1. **Choose Shared database option:**

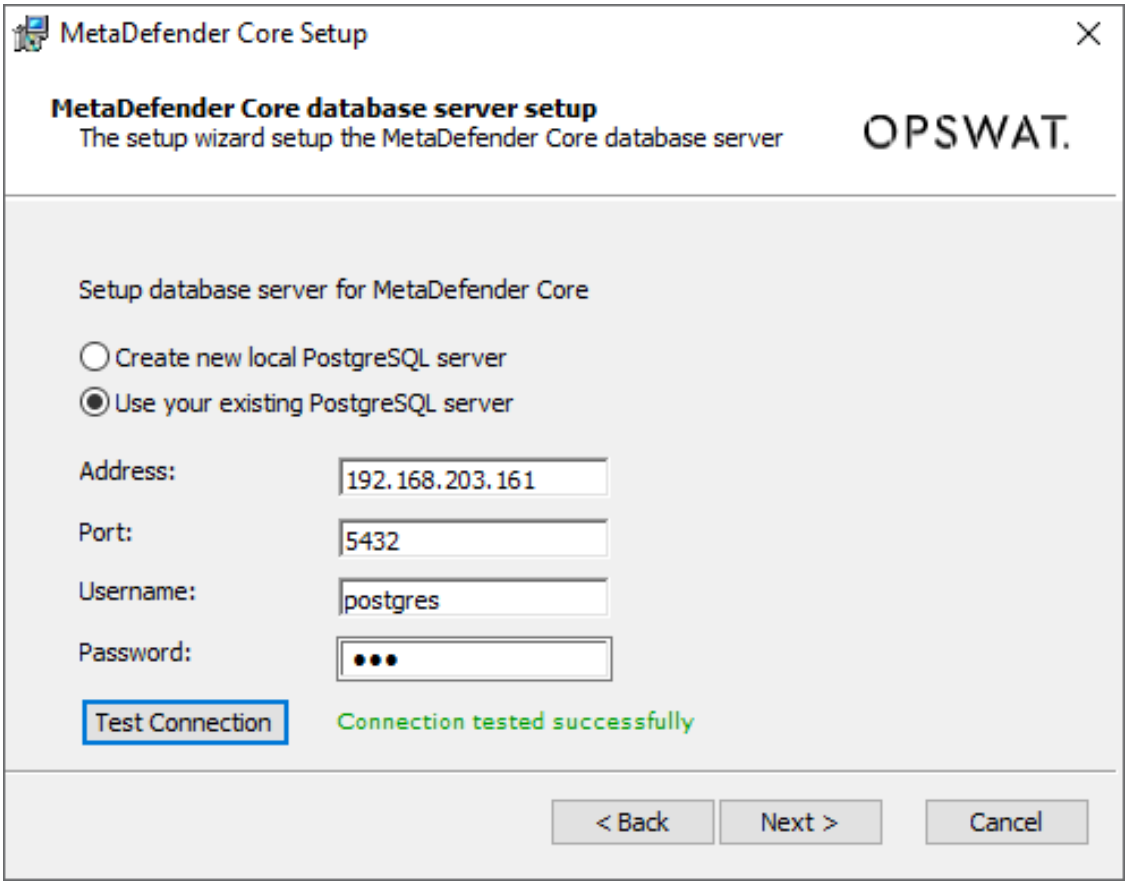


2. **Setup database server for MetaDefender Core:**

You can choose to create a native / built-in PostgreSQL server and setup the shared database on (then other MetaDefender Core instances will want to connect to that PostgreSQL server afterwards)



Or choose to connect to a pre-setup existing PostgreSQL server:



PostgreSQL connection configuration

"Test Connection" button is required, to make sure the PostgreSQL is connected and authenticated successfully. Make sure to check and perform steps at Open Connection On PostgreSQL Server if you are unable to test the connection with remote PostgreSQL server.

PostgreSQL user privilege requirements

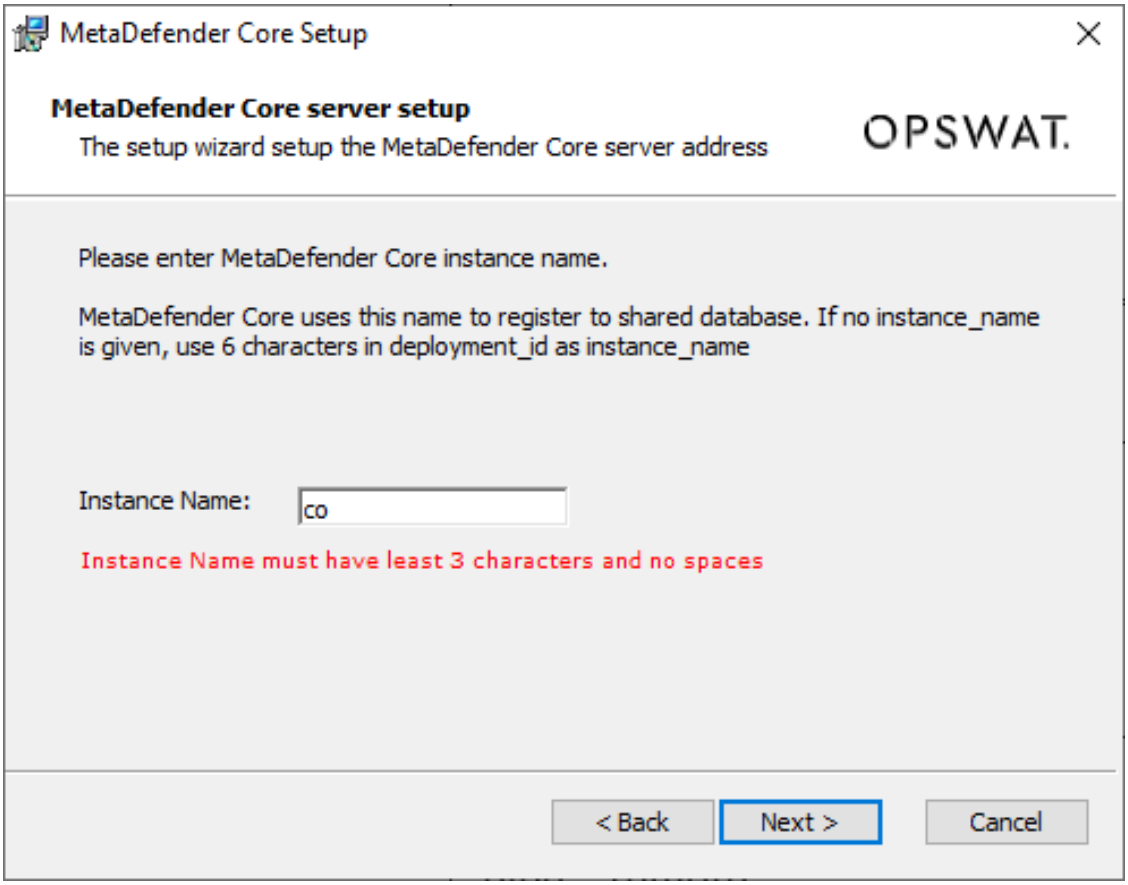
MetaDefender Core will need to create and use dedicated PostgreSQL users for database setup, upgrade and its own operations. Those users requires certain privileges configured on PostgreSQL server, otherwise expecting failures to occur while installing and operating MetaDefender Core.

Details: PostgreSQL User Privilege Requirements

3. **Create an unique instance name for that MetaDefender Core (different from other MetaDefender Core instances):**

Instance name must contain at least 3 characters and no white space.

If you choose to leave it blank, which means it will use the default instance name with 6 characters extracted from that MetaDefender Core deployment ID.

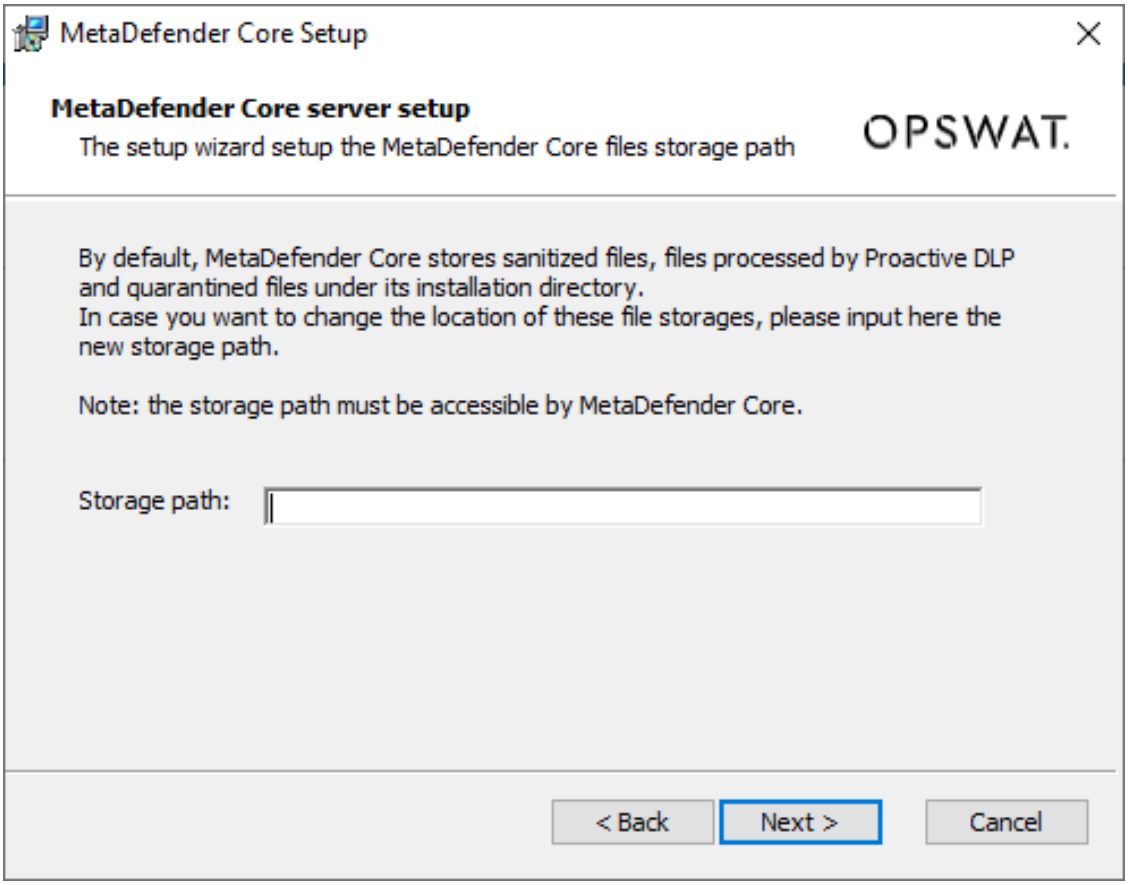


4. **Input the file storage path (sanitized, DLP, quarantined files)**

In the shared database model, all MetaDefender Core instances could use the same folder path to store all their own sanitized, DLP, quarantined files after processed.

The default location is: ***<MetaDefender Core installation folder>\data\ {sanitized|dlp|quarantine}***

If you wish to use that default location, just leave the input box blank and press Next.



Otherwise, you can specify a different location of sanitized/dlp/quarantined files. It could be a local path on the same machine, or a shared folder located in a remote connected machine. Make sure that folder is open access (read and write) to MetaDefender Core.

For example:

- C:\new_folder
- \\192.168.1.1\new_folder

Command Line Installation

> **Note**
>
> For command line installation (on both Linux and Windows), it is **mandatory** to use MetaDefender Core ignition file to add PostgreSQL server information **BEFORE** installing MetaDefender Core 4.19.0 or newer.

Clean install scenario

1. Create ignition file if not existed

- Windows: **C:\OPSWAT\ometascan.conf**

- Linux: ***/etc/opswat/ometascan.conf***

An example of an ignition file:

ometascan.conf · bash

```
eula=true

[global]
dbmode=4
instance_name=MetaDefenderCoreA
storage_path=D:\path\

[user]
name=admin
password=admin
email=admin@local

[dbserver]
type=remote
host=192.168.1.1
port=5432
user=postgres
password=your_own_password
```

Field	Required	Mapping to wizard UI	Possible values
global/dbmode	optional default is 1 (standalone)		1 - Standalone 2 - Central Hub deployment 3 - Non-persistent **4 - Shared**
global/instance_name	optional default is 6 characters of deployment ID		Unicode and number At least 3 characters No whitespace
global/storage_path	optional default is empty		Path to storages (DLP, quarantined, sanitized files) It could be a local path on the same machine, or a shared folder located in a remote connected machine. Make sure that

Field	Required	Mapping to wizard UI	Possible values
			folder is open access (read and write) to MetaDefender Core. For example: • C:\folder • \\192.168.1.1\folder

2. Run installers using the command line to install

Please refer to Command Line Installation for the installation command.

Deployment automation support

Note

An Ignition file could be utilized for either / both of following use-cases:

1. Pre-define PostgreSQL server information for MetaDefender Core to connect and setup database: only required on Linux installation, or Windows with command line installation. Learn more details at:
    1. Standalone Mode > Command Line Installation (applicable on version 4.19.0 or newer).
    2. Shared Database Mode > Command Line Installation (only applicable on version 4.20.0 or newer).
2. Automating to accept EULA, create a default local admin user and import MetaDefender Core configurations upon installed: Those steps could be automated using ignition file described in this page.

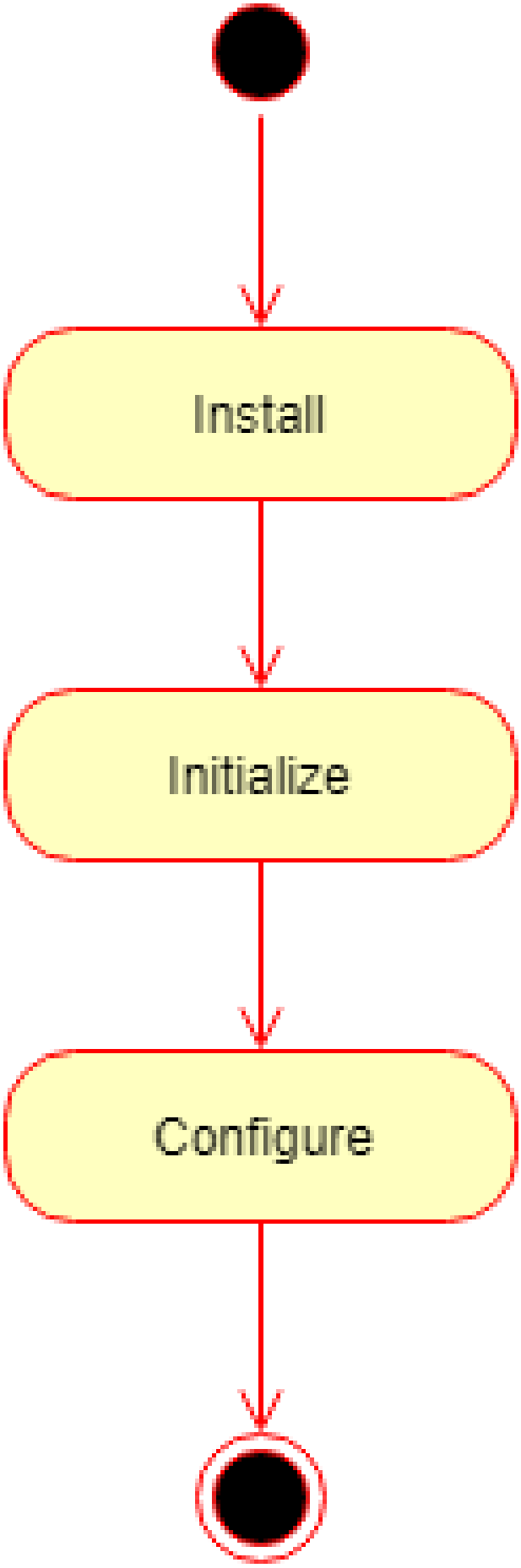
Ignition file location to create:

- **Windows**:`C:\OPSWAT\ometascan.conf`
- **Linux**: `/etc/opswat/ometascan.conf`

The product supports fully automated deployment. It means that it can be installed and configured with no human interaction.

The automated deployment can be split to three steps on a high level:

1. Installation
2. Initialization
3. Configuration


Install
Initialize
Configure

Installation

To automate the installation, install the product from the command line and provide the installation-time options as parameters to the installer. For further details see Command Line Installation

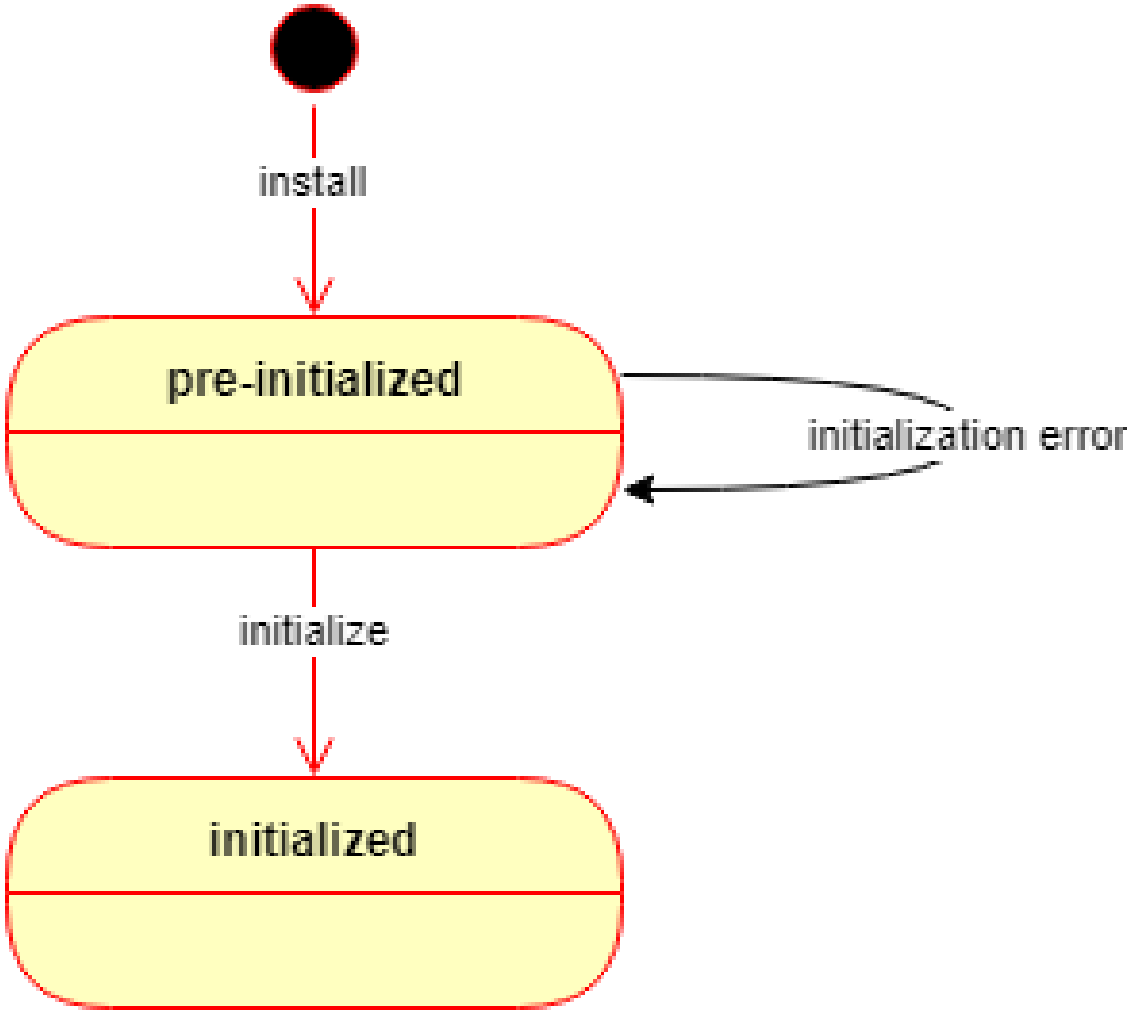
After the installation is complete, the product starts up and waits in a pre-initialized status. The product may be initialized in two ways:

1. Manually using the Wizard Setup or
2. Automatically using an *ignition file* (see below).

Warning

If the automated initialization fails for some reason (e.g. the ignition file is not in place) then the automated initialization may be retried after fixing the problem (e.g. placing the ignition file in its lookup location) and restarting the OPSWAT MetaDefender Core service.

Until the product is in pre-initialized status, it will try the automated initialization every time after a service (re)start.

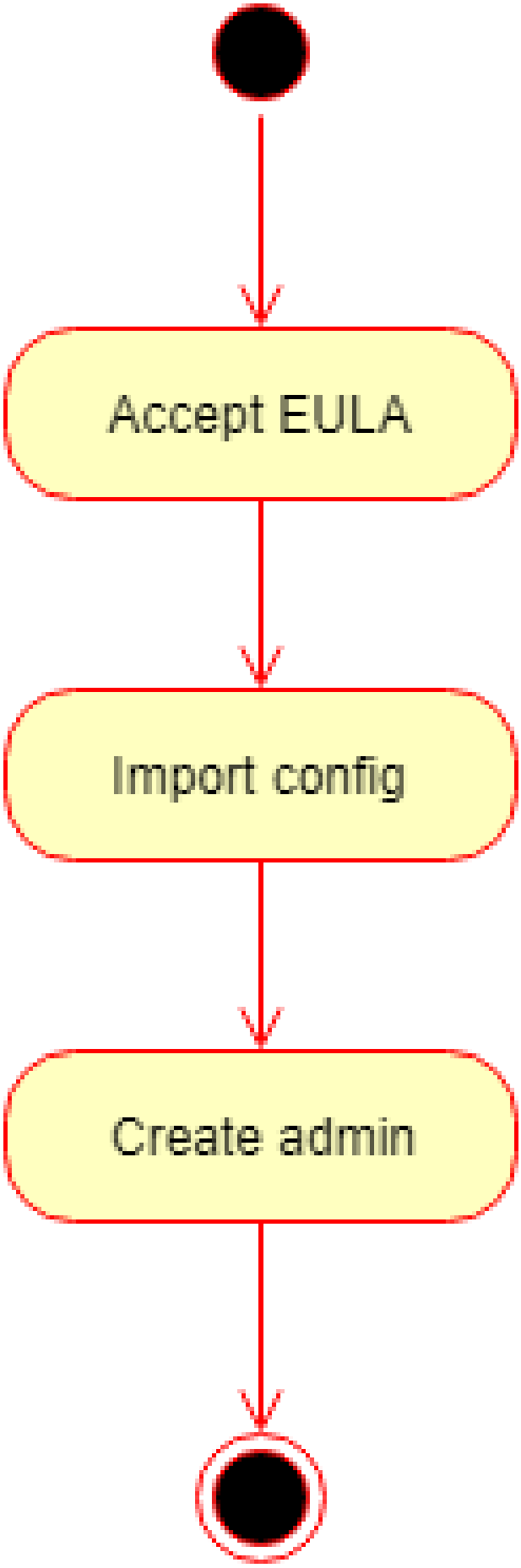


Initialization

Initialization is the process of bringing the product to an operable status.

Basically the initialization consists of the following steps:

1. Accept the End User License Agreement (EULA),
2. Import product configuration and
3. Create the first administrator user account.


Accept EULA
Import config
Create admin

Ignition file

The initialization process can be configured in a file called the *ignition file*.

Please check more details if you have special characters in Ignition file: Ignition file - special characters

> **Warning**
>
> In Linux, the ignition file and the folder /etc/opswat should have the permission 755 and must be in **conf** format.

Ignition file location

- **Windows:** `C:\OPSWAT\ometascan.conf`
- **Linux:** `/etc/opswat/ometascan.conf`

Example:

ometascan.conf · bash

```
eula=true
[user]
name=admin
password=admin
email=admin@local
apikey=44c96cbb4e4e6c0a1d16bcb3c3da24def317
[config]
import=config_package.zip
import_password=123
[security]
session_cookie_samesite=Lax
[internal]
skip_startup_pages=true
skip_upgrade_scan_data=false
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
private_username=my_internal_db_user
private_password=mypassword
[ocm]
ocm_url=https://my.opswat.com
reg_code=XXXXXXXX
[max_body_size]
body_size=body_size_in_byte
[https]
cert_name=my_cert_name
cert_path=C:/OPSWAT//certs/server.crt
```

```
key_path=C:/OPSWAT/certs/server.key
passphrase=my_pass_if_any
tls_version=1.2,1.3
```

Ignition file fields

The ignition file accepts the following fields:

Detailed initialization process

1. After the product has been started, it looks for the ignition file in the designated location.

- **Windows**: `C:\OPSWAT\ometascan.conf`
- **Linux**: `/etc/opswat/ometascan.conf`

2. If an ignition file is found, then

2.1) It gets validated, and if it is valid, then based on the information found in the ignition file:

- The EULA is accepted,
- The configuration is imported,
- The administrator account is created.
- If any of the above steps fails, then the error is logged, and the initialization gets terminated.

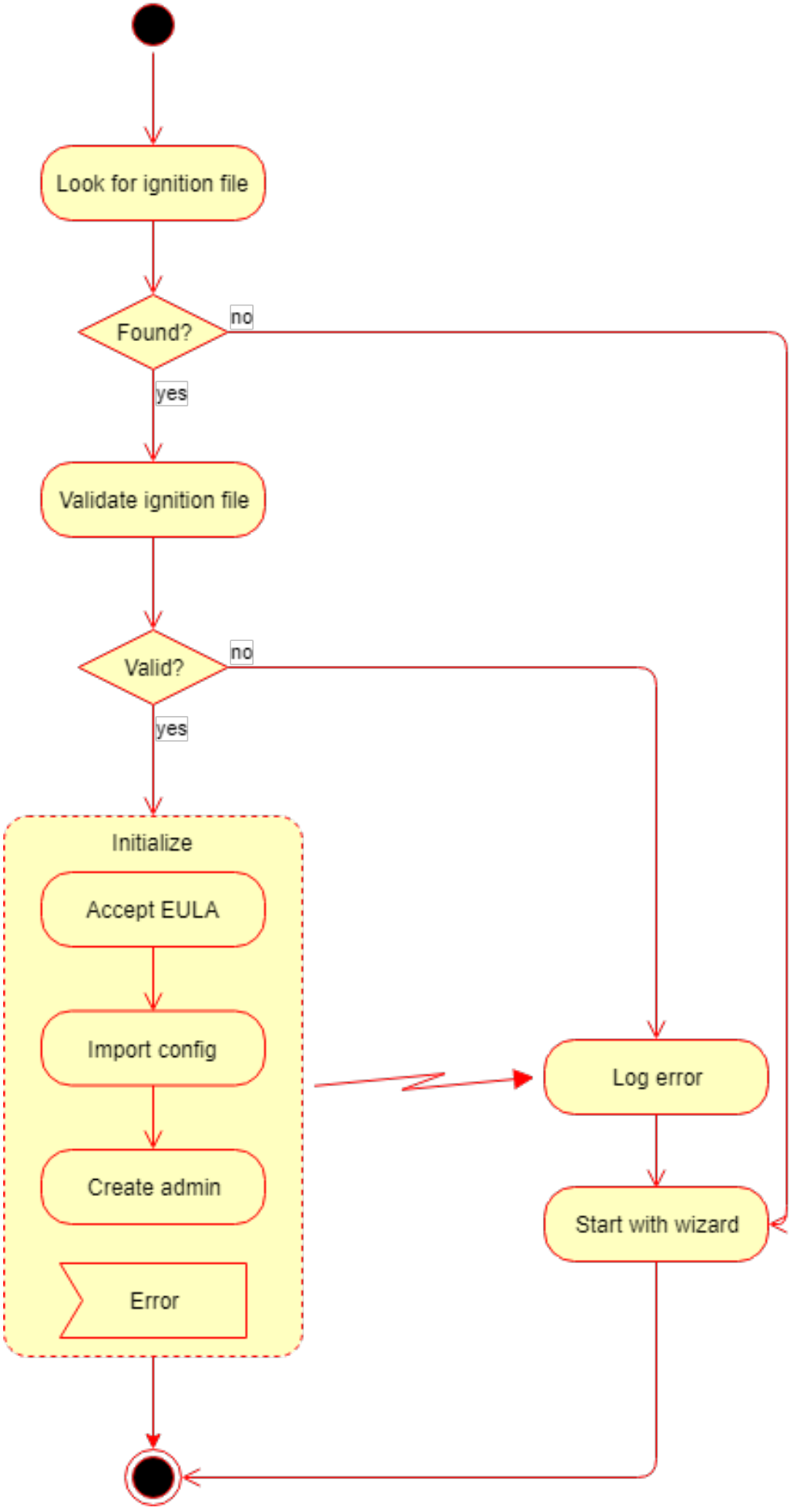
> **Info**
> In this case the product starts normally: if for example the basic configuration wizard has not been completed yet, then it must be completed first.

2.2) If it is not valid, then the error is logged, and the initialization gets terminated.

> **Info**
> In this case the product starts normally: if for example the basic configuration wizard has not been completed yet, then it must be completed first.

3. If there is no ignition file, then no initialization is performed.

> **Info**
> In this case the product starts normally: if for example the basic configuration wizard has not been completed yet, then it must be completed first.


Look for ignition file
Found?
no
yes
Validate ignition file
Valid?
no
yes
Initialize
Accept EULA
Import config
Create admin
Error
Log error
Start with wizard

Warning

If the automated initialization fails for some reason (e.g. the ignition file is not in place) then the automated initialization may be retried fixing the problem (e.g. placing the ignition file to its lookup location) and restarting the OPSWAT MetaDefender Core service.

Until the product is in pre-initialized status, it will try the automated initialization every time after a service (re)start.

Configuration

After initialization is complete, the product is ready with both the default and imported configurations.

This configuration can be later changed by calling the configuration API functions. For further details about the API, see Configuration related APIs

Enabling SSL Connections for PostgreSQL via ignition file

Overview

Starting with MetaDefender Core version 5.16.0, users can optionally enable SSL connections for the bundled PostgreSQL database during installation using the ignition file. This feature enhances security by encrypting data transmissions between the application and the database.

Key Considerations

- This feature applies exclusively to the bundled PostgreSQL database.
- It is available only for automated deployments via the ignition file.
- **Enabling the SSL connection has overhead and might impact the performance in general.**
- It does not apply during upgrades of MetaDefender Core.
- SSL connections can be disabled by customers at any time post-installation.
- This configuration is not supported in Docker environments.

Setting Up SSL Connections

Enabling SSL is an optional step performed during the installation of MetaDefender Core using the ignition file. In addition to preparing the ignition file, you must create a configuration file named `postgres_ssl.conf` in the following fixed location:

- **Windows:** `C:\opswat\postgres_ssl.conf`
- **Linux:** `/etc/opswat/postgres_ssl.conf`

The file must be placed exactly in this path for the installation process to detect and apply the settings.

Contents of the Configuration File

The `postgres_ssl.conf` file should contain the following PostgreSQL SSL parameters. Replace the placeholders with the absolute paths to your certificate, key, and CA files:

Code · bash

```
ssl = on
ssl_groups = 'prime256v1'
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_cert_file = '<absolute_path_to_cert_file>'
ssl_key_file = '<absolute_path_to_key_file>'
ssl_ca_file = '<absolute_path_to_root_ca_cert_file>'
```

For example, in Linux

Code · bash

```
ssl = on
ssl_groups = 'prime256v1'
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_cert_file = '/etc/tmp_cert/server.crt'
ssl_key_file = '/etc/tmp_cert/server.key'
ssl_ca_file = '/etc/tmp_cert/root.crt'
```

FIPS 140-3 Compliant Configuration from MetaDefender Core 5.22.0

From MetaDefender Core 5.22.0, the bundled PostgreSQL ships with a FIPS 140-3 validated OpenSSL module. On these versions, non-approved algorithms such as X25519 are unavailable to the database server regardless of configuration.

Please note that the PostgreSQL default for `ssl_groups` is `X25519:prime256v1`, which is not supported on these deployments. Specify only approved groups in `postgres_ssl.conf` - if the configuration references an algorithm the module does not provide, the PostgreSQL service fails to start. This applies only to standalone deployments using the bundled PostgreSQL.

Users can store the cert/key files in any place, but please make sure that the cert/key files have the right permissions, so MetaDefender Core can copy them to the right location. For example, in Linux, the files need to have **read permission** for other users. Users can use this command in Linux to set the permissions

Code · bash

```
chmod 555 -R /path/to/store/cert_files/
# For example
chmod 555 -R /etc/tmp_cert/
```

Additional Notes

During installation, the specified certificate file (server.crt), key file (server.key), and root CA file (root.crt) will be automatically copied to the PostgreSQL data directory:

- Windows: `<installation folder>/data/pg_data`
- Linux: `/var/lib/ometascan/pg_data`

Once the installation is complete and verified, you may safely delete the original files from their source locations if they are no longer needed.

How to disable bundled PostgreSQL SSL connection

1. Go to `postgres` folder and remove the file `postgres_ssl.conf`
    1. Windows: `<installation folder>/postgres/postgres_ssl.conf`
    2. Linux: `/usr/lib/ometascan/postgres/postgres_ssl.conf`
2. Go to pg_data folder, remove the file `pg_hba.conf` then rename `pg_hba_bk.conf` to `pg_hba.conf`
3. Restart the MetaDefender Core PostgreSQL service.

Ignition file - special characters

Overview

The **ignition file** automates first-time setup of MetaDefender Core (EULA, admin account, database, HTTPS, and more). It uses INI-style `key=value` lines under sections such as `[user]` and `[dbserver]`.

Some characters have special meaning. If a password or passphrase contains them and is not written correctly, MetaDefender Core may read a **truncated or wrong value** without a clear error message.

Default file location

- **Windows:** `c:\opswat\ometascan.conf`
- **Linux:** `/etc/opswat/ometascan.conf`

Be careful — the most common mistake

A **semicolon** (`;`) inside an unquoted password is treated as the start of a comment. Everything after `;` on that line is ignored.

Fields to watch

Check these ignition-file fields carefully when values contain special characters:

- `[user] password`, `[user] apikey`

- `[dbserver] password`, `[dbserver] private_password`
- `[config] import_password`
- `[https] passphrase`

Special characters

Character	Why it matters	What to do
`;`	Comment / ends unquoted values	Wrap the whole value in `"..."`. **Do not** use `\;`.
`=`	Separates key and value	Wrap in quotes if `=` appears inside the value.
`"`	Quote delimiter	Wrap in quotes; escape inner quotes as `\"`.
`\`	Escape prefix	Write a literal backslash as `\\`.
`,`	List separator	Wrap in quotes if comma is part of the value.
`#`	No special meaning	Normal character — no extra escaping needed.

How to write passwords safely

1. If the value contains `;`, `=`, or `,` — wrap it in **double quotes**.
2. Never use `\;` to escape a semicolon — it does **not** work.
3. Escape `\` as `\\` and `"` as `\"`.
4. Quote-doubling (`""`) is **not** supported — use `\"` instead.

Examples

Password with semicolon

Intended password: `Ew6I]7;a:SRD`

Code · bash

```
[user]
```

```
name=admin
password="Ew6I]7;a:SRD"
email=admin@local
```

Password with semicolon and double quote

Intended password: `Say;"hello"`

Code · bash

```
[user]
password="Say;\"hello\""
```

Database password

Code · bash

```
[dbserver]
password="p@ss;w=rd"
```

Quick reference

Value contains…	Do this
`;`	Wrap in `"..."` — not `\;`
`"`	Wrap in quotes; use `\"` for each inner quote
`\`	Write `\\`
`;` and `"`	Quotes around value + `\"` for inner quotes

CIS Level 1 - Linux Guidelines

Red Hat Enterprise Linux 9

Support Red Hat Enterprise Linux 9

For more details about Center for Internet Security (CIS) please refer to this document

Instruction steps

I. Install OpenSCAP

bash · bash

```
yum install openscap-scanner scap-security-guide
```

II. Generate a result file and a HTML report using OpenSCAP scanner tool

Code · bash

```
oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_cis_server_l1 --results
scan_results.xml --report scan_report.html /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

III. Remediation of CIS Level 1 issues

Generate a remediation script based on the `ssg-rhel9-ds.xml` file:

Code · bash

```
oscap xccdf generate fix --profile xccdf_org.ssgproject.content_profile_cis_server_l1 --fix-type
bash --output remediations.sh /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

And execute remediation script.

Code · bash

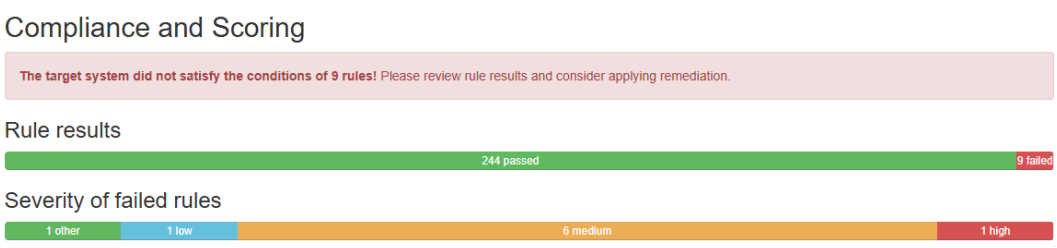
```
./remediations.sh
```

IV. Review the results after remediation

Code · bash

```
oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_cis_server_l1 --results
scan_results.xml --report scan_report.html /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
```

Example:



Ubuntu 22 Pro

Install the UA client

Code · bash

```
sudo apt update
sudo apt install ubuntu-advantage-tools
```

Set up the Ubuntu Security Guide

Code · bash

```
sudo ua enable usg
sudo apt install usg
```

Check SCAP Content Overview (Security Content Automation Protocol)

Code · bash

```
sudo oscap info /usr/share/ubuntu-scap-security-guides/1/benchmarks/ssg-ubuntu2204-ds.xml
```

Auditing an Ubuntu System for DISA-STIG compliance

Code · bash

```
sudo usg audit cis_level1_server
```

The report is generated in `/var/lib/usg/`

Applying the CIS rules to a set of systems

There are 2 ways that apply CIS rules

Method 1: directly using usg command - **recommend**

Code · bash

```
sudo usg fix cis_level1_server
```

Method 2: using usg to generate script and then run the script

Code · bash

```
sudo usg generate-fix cis_level1_server --output fix.sh
```

```
#And the run
./fix.sh
```

A reboot is require to take the effect after apply the fix.

Notes

- CIS Level 1 requires `/tmp` folder to be mounted in a separate partition. Please ensure that that new partition have enough disk space for MetaDefender Core to run.
- CIS Level 1 requires that "Ensure No World-Writable Files Exist".
  - For now, when freshly installing MetaDefender Core, all its binary files meet the requirement.
  - When installing/updating engines, some engines might create additional files for its operation, and it might violate this requirement. In this case, you need to again execute the remediation script in the step III.

CIS Level 2 - Linux Guidelines

Support Ubuntu Pro 22.04, 24.04

For more details about Center for Internet Security (CIS) please refer to this document.

Instruction steps

Install the UA client

bash · bash

```
sudo apt update
sudo apt install ubuntu-advantage-tools
```

Set up the Ubuntu Security Guide

Code · bash

```
sudo ua enable usg
sudo apt install usg
```

Check SCAP Content Overview (Security Content Automation Protocol)

Code · bash

```
sudo oscap info /usr/share/ubuntu-scap-security-guides/1/benchmarks/ssg-ubuntu2204-ds.xml
```

Auditing an Ubuntu System for DISA-STIG Compliance

```bash
sudo usg audit cis_level2_server
```

The report is generated in `/var/lib/usg/`

Applying the CIS rules to a set of systems

There are 2 ways to apply CIS rules.

Method 1: directly using usg command - recommend

```bash
sudo usg fix cis_level2_server
```

Method 2: using usg to generate a script and then run the script

```bash
sudo usg generate-fix cis_level2_server --output fix.sh
#And then run
./fix.sh
```

Notes:

- A **reboot is required** to take effect after applying the fix.
- CIS Level 2 requires `/tmp` folder to be mounted in a separate partition. Please ensure that the new partition has enough disk space for MetaDefender Core to run.

Running fix for Ubuntu 24.04

When generating the fix script in **Ubuntu 24.04**, there will be 2 rules (in the generated script - fix.sh) that change the permission and owner of the folder `/var/log:`

`#240: xccdf_org.ssgproject.content_rule_file_groupownerships_var_log`

`#255: xccdf_org.ssgproject.content_rule_file_ownerships_var_log`

These 2 rules will change the permission and owner of the product log also `var/log/ometascan` .

To prevent this, before running the fix script, the user needs to add `metascan` group, for example, with rule #240.

```bash
group="root"
find -L /var/log/ -type f -regextype posix-extended \
```

```
    ! -group root ! -group adm ! -group metascan  \
    ! -name 'gdm' ! -name 'gdm3' \
    ! -name 'sssd' ! -name 'SSSD' \
    ! -name 'auth.log' \
    ! -name 'messages' \
    ! -name 'syslog' \
    ! -path '/var/log/apt/*' \
    ! -path '/var/log/gdm/*' \
    ! -path '/var/log/gdm3/*' \
    ! -path '/var/log/sssd/*' \
    ! -path '/var/log/[bw]tmp*' \
    ! -path '/var/log/cloud-init.log*' \
    ! -regex '.*\.journal[~]?' \
    ! -regex '.*/lastlog(\.[^\/]+)?$' \
    ! -regex '.*/localmessages(.*)' \
    ! -regex '.*/secure(.*)' \
    ! -regex '.*/waagent.log(.*)' \
    -regex '.*' -exec chgrp $group {} \;
```

Also do the same with rule #255 to prevent update permission and the owner of `var/log/ometascan`

CIS Level 1 - Windows Guidelines

Setup

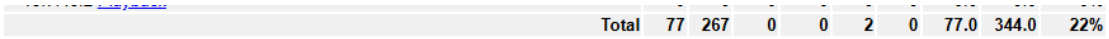
Windows version has been tested: Windows Server 2022 and 2025

We use the image available on AWS: CIS Hardened Image Level 1 on Microsoft Windows Server 2025

This image has been hardened CIS Level 1

When using the image above, we can find the CIS report already available at `C:\CIS_Hardening_Reports\`

`Base_CIS-CAT_Report.html` - Base report, the report before apply CIS - meet 22% CIS points



Total	77	267	0	0	2	0	77.0	344.0	22%

`CIS-CAT_Report.html` - CIS report, the report after apply CIS - meet 97% CIS points



Total	334	10	0	0	2	0	334.0	344.0	97%

CIS Level 2 - Windows Guidelines

Setup

Windows version has been tested: Windows Server 2022

We use the image available on AWS: CIS Hardened Image Level 2 on Microsoft Windows Server 2022

This image has been hardened CIS Level 2

When using the image above, we can find the CIS report already available at `C:\CIS_Hardening_Reports\`

`Base_CIS-CAT_Report.html` - Base report, the report before applying CIS - meet 21% CIS points

Total	82	309	0	0	2	0	82.0	391.0	21%

`CIS-CAT_Report.html` - CIS report, the report after applying CIS - meets 97% CIS points

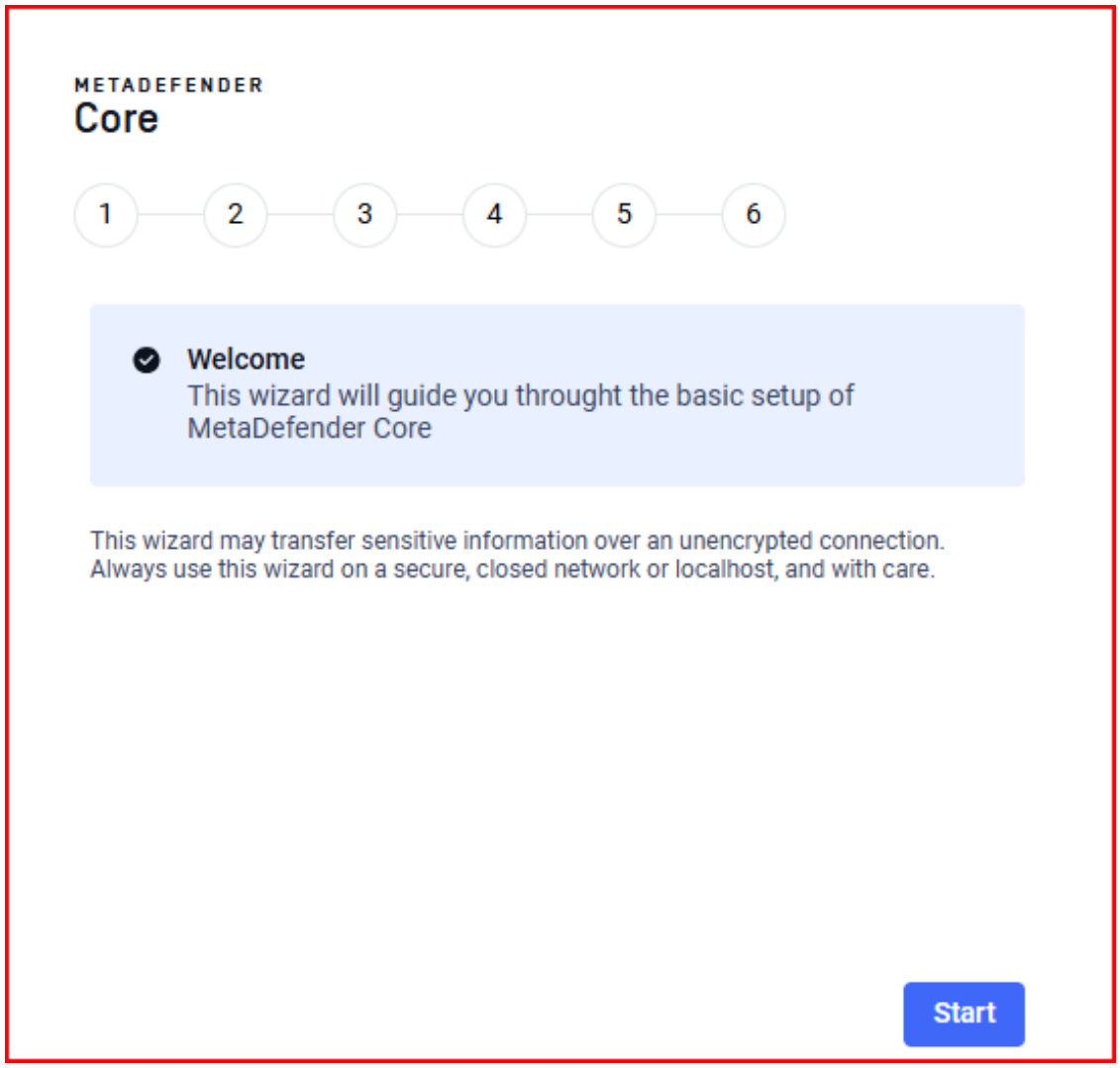
Total	380	11	0	0	2	0	380.0	391.0	97%

Wizard Setup

Introduction

Access the MetaDefender Core management console for the first time (i.e. http://localhost:8008), you are to complete a quick walkthrough for wizard setup in order to be able to use the product.

The administrative management console will be available only upon you have successfully finished this wizard stage.

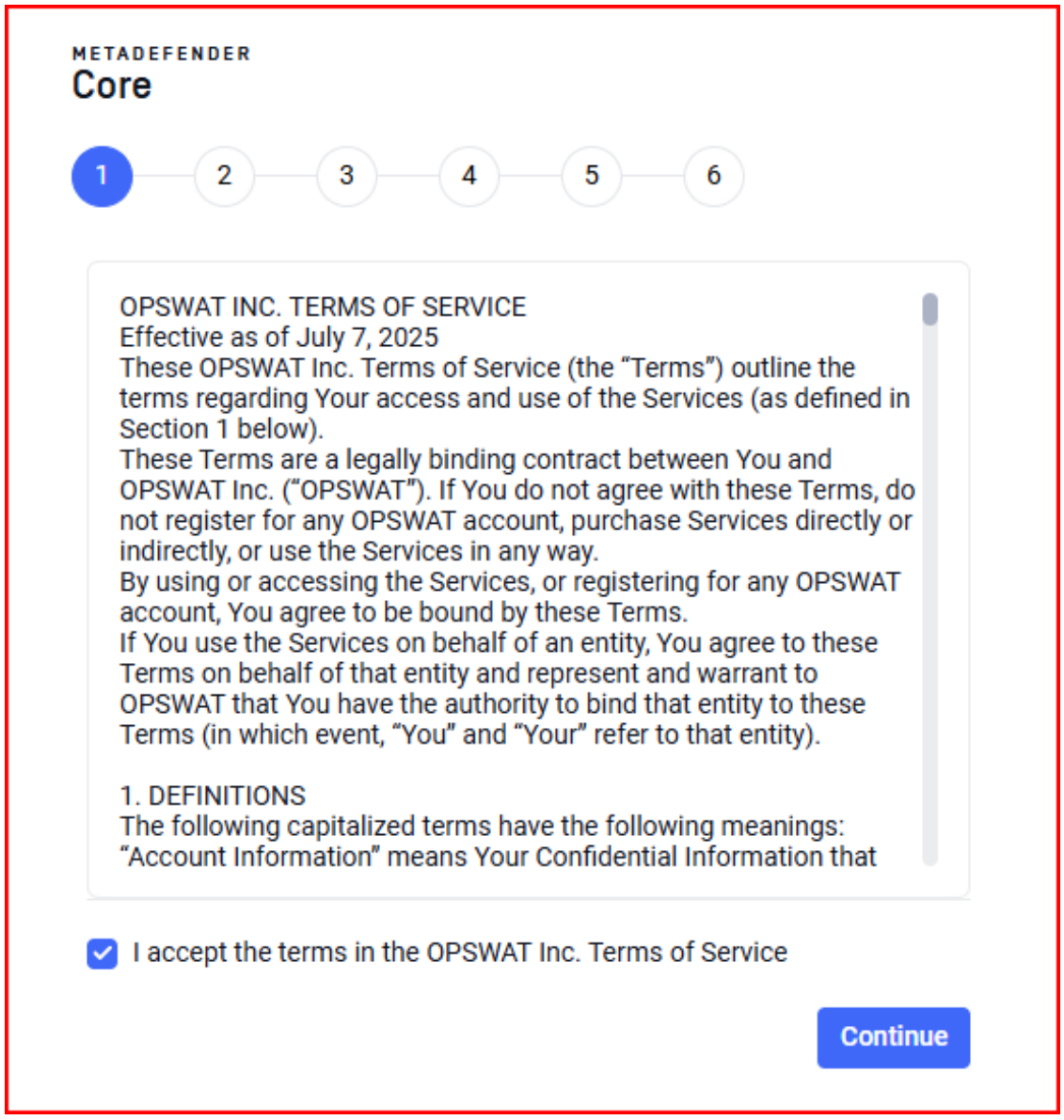


Warning

This wizard may transfer sensitive information over an unencrypted connection. Always use this wizard on a secure, closed network or localhost, and with care.

Basic configuration steps

End-User License Agreement



In the first page you can find the End-User License Agreement. You have to accept the terms before moving on.

Please read through the EULA (https://docs.opswat.com/policies/privacy-policy/portal-privacy-policy) carefully and if you agree with it, check `I accept the terms in the **License Agreement**` and click `Continue` to proceed.

Local Admin User Setup

The next step is to set up a default local administrator account.

This account will be the first one being able to access the administrative management console, and to create accounts for other users. You have to fill out all fields in this page to be able to move forward. When you are done,

click NEXT to continue.

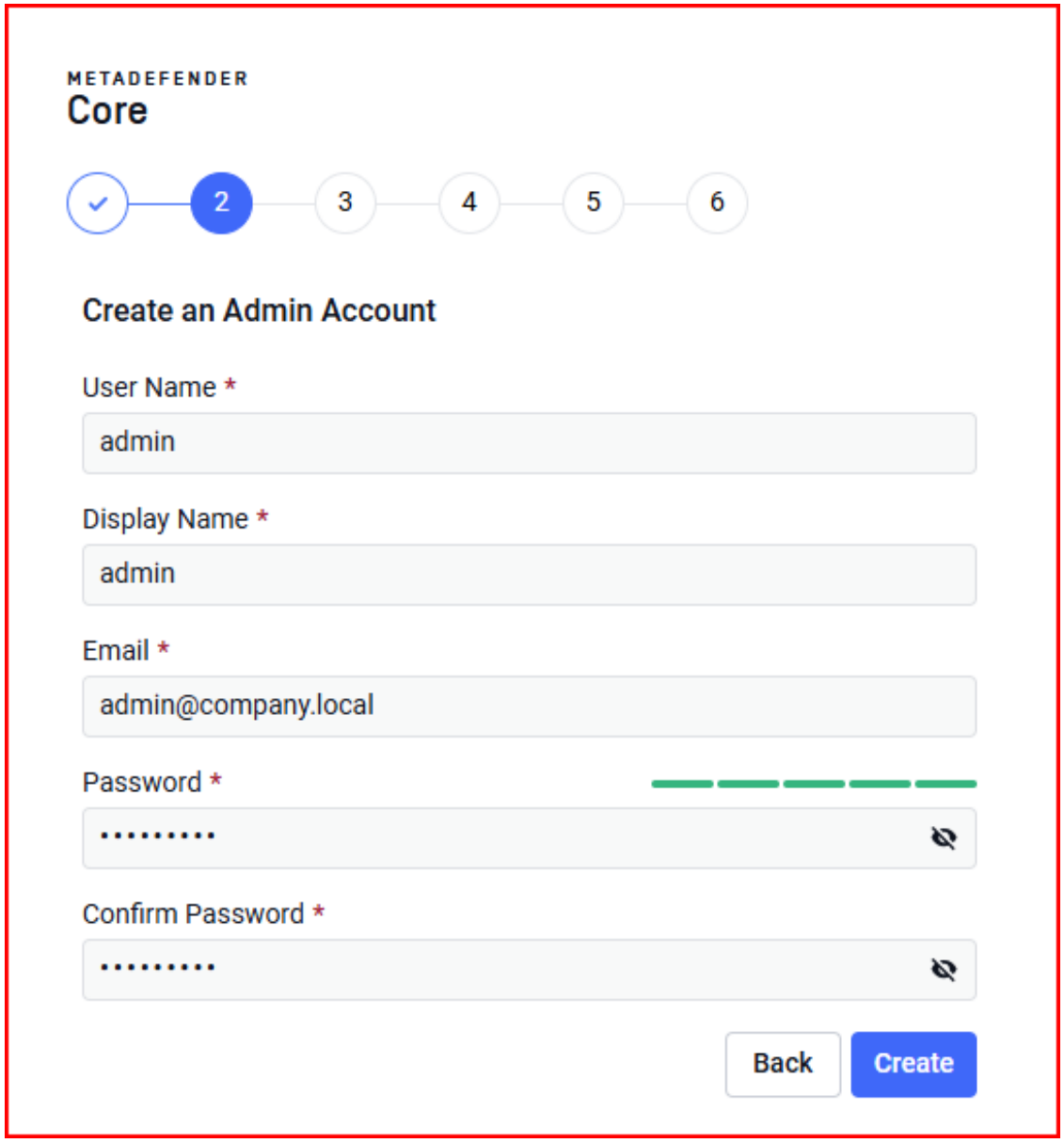
Info - User Directory

The administrator account, that is created via the basic configuration wizard, is always added to the LOCAL user directory as a member.

Learn more about user directory: User directories

The following information is required for the administrator account:

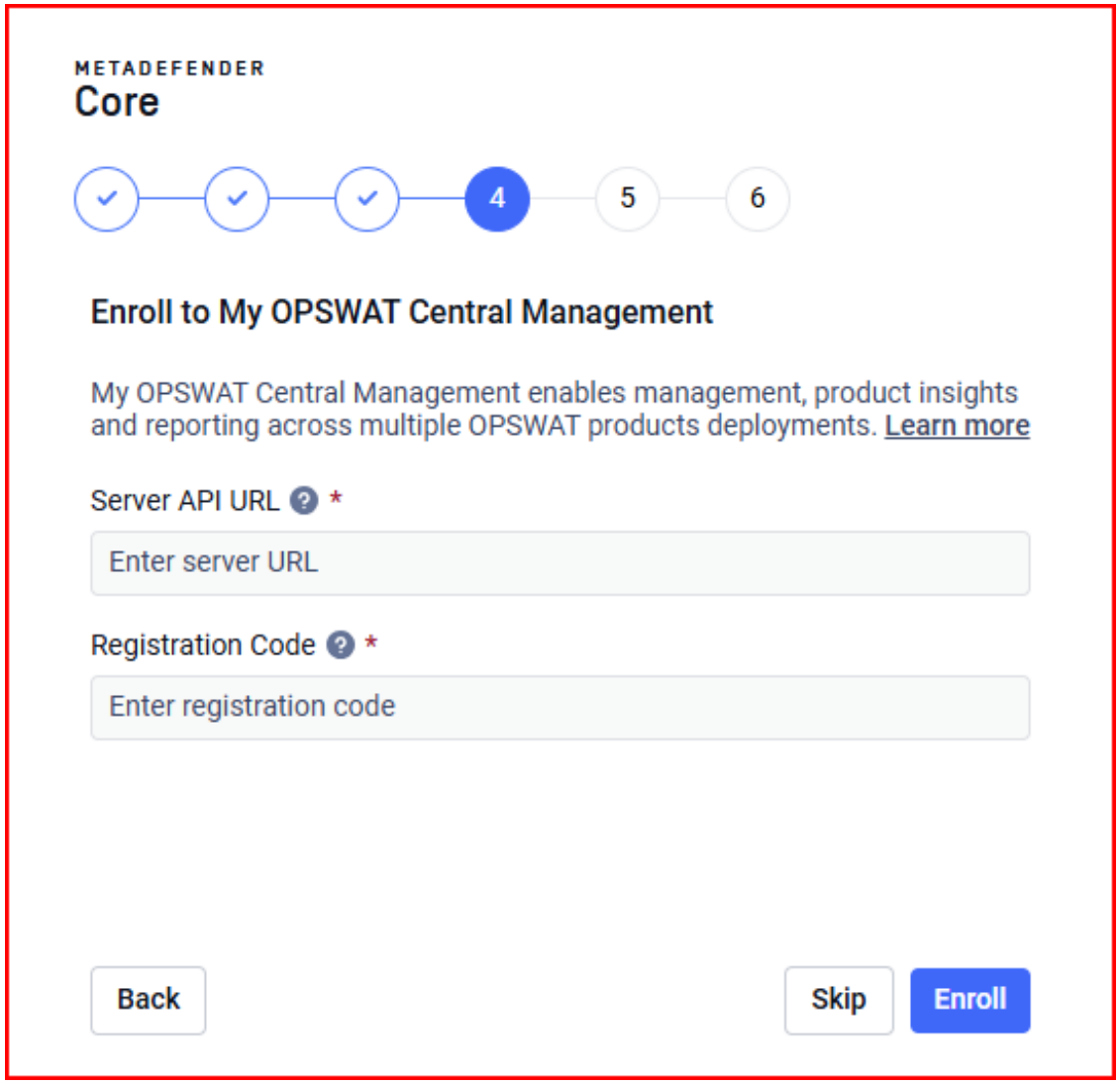
ACCOUNT NAME	**The unique name of the account that is used at the time of login and in log messages for accountability.**
ACCOUNT DISPLAY NAME	Name of the person bound to this account. This name (appended to the name of the account's user directory) is displayed in the top right corner of the administrative management console.
PASSWORD	Password of the user bound to this account that is used at the time of login. Passwords sent clear-text As long as TLS is not configured for the basic configuration wizard, passwords are sent clear-text over the network and may be disclosed to unauthorized parties. As a mitigation action: 1. Either use the wizard on *localhost* or on a direct network connection, or 2. Enabling HTTPS as soon as possible and change the password immediately if it has already been set.
EMAIL	Email address of the person bound to this account.



License activation

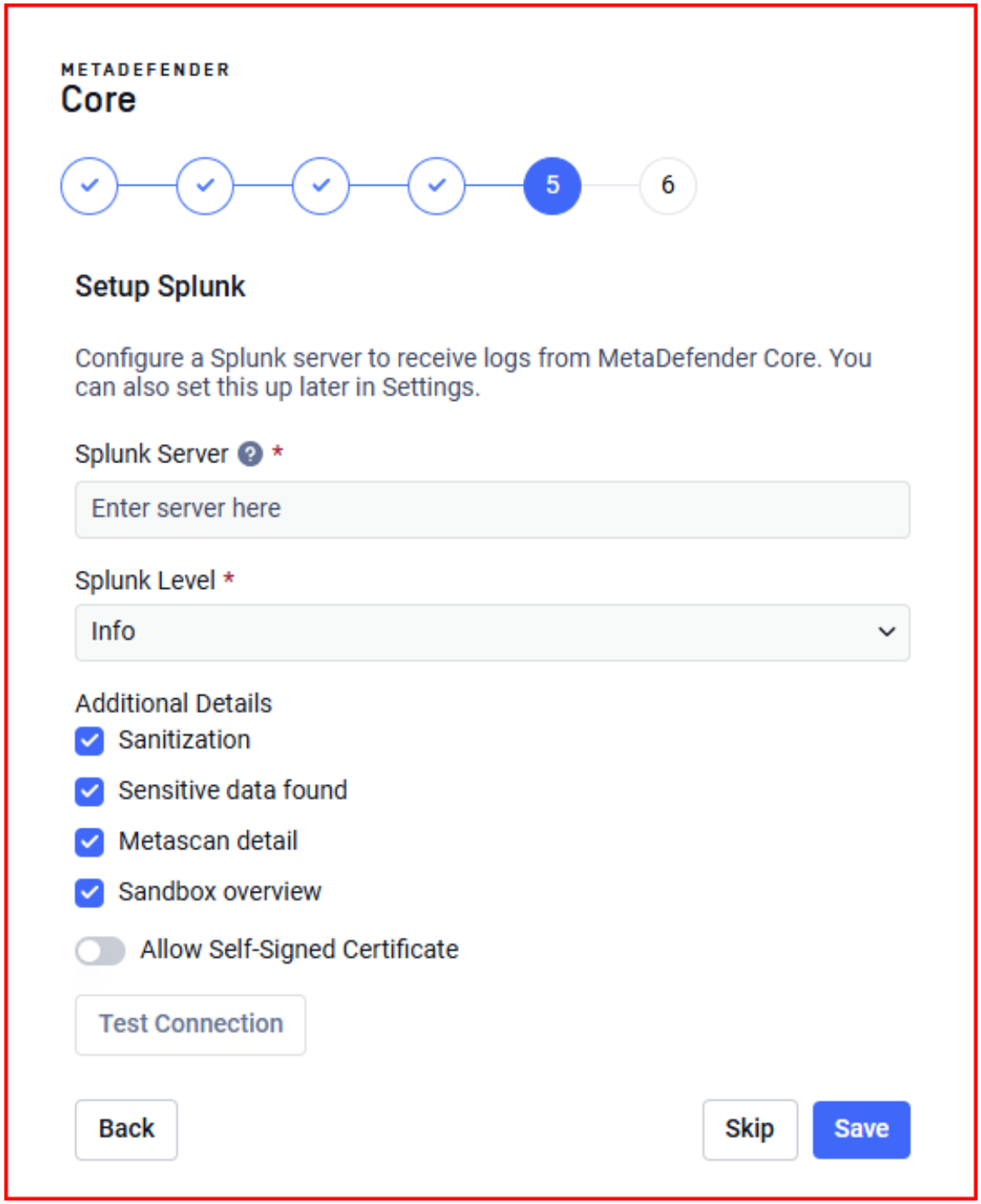
For license activation details see License Activation

Enroll to My OPSWAT Central Management - Optional



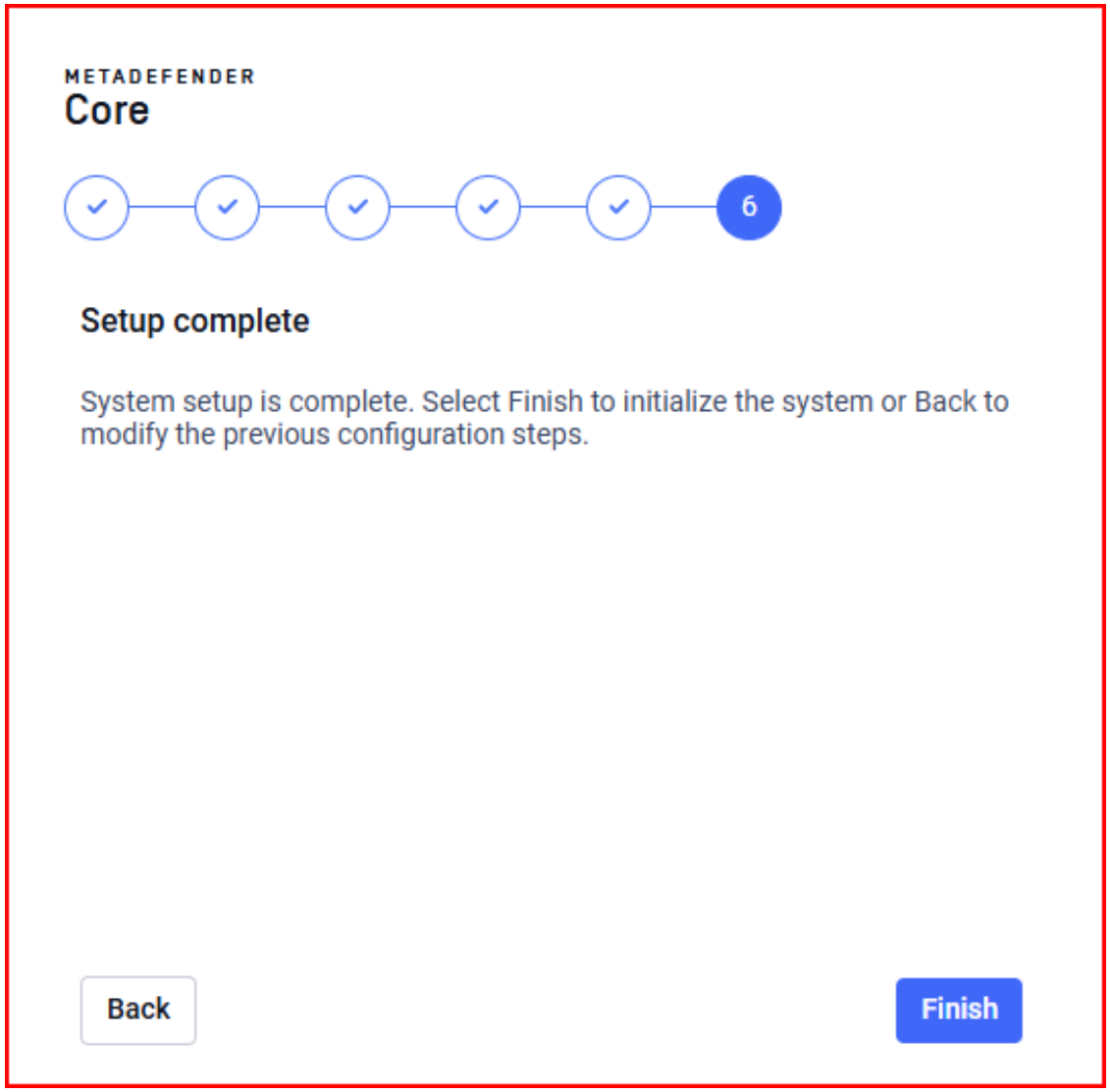
Setup Splunk - Optional

This step configures a Splunk destination so that MetaDefender Core forwards its log messages to your Splunk server. It is optional, you can skip it here and configure Splunk later from **Settings > Logs** page.



Wizard completion

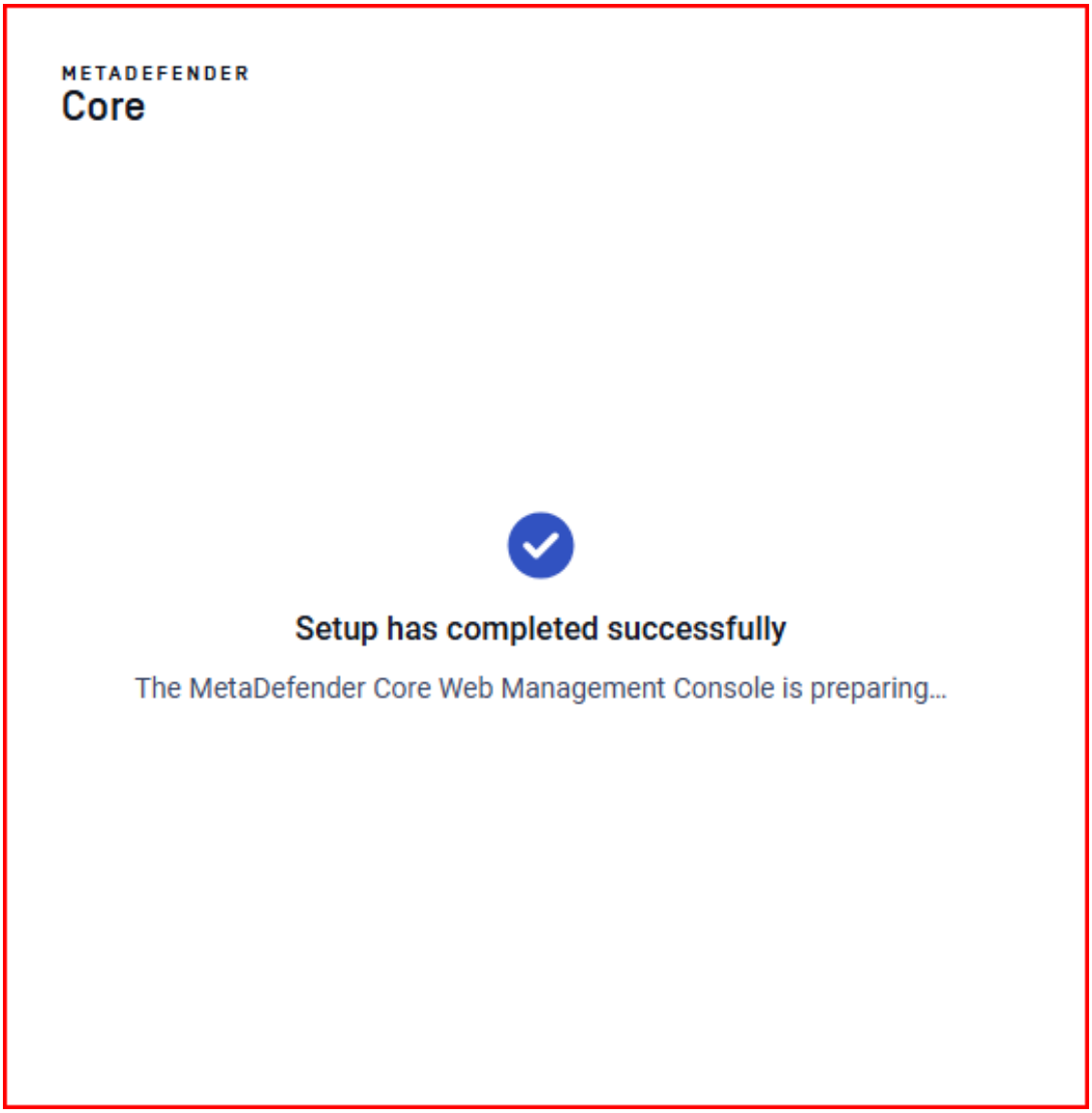
After you have completed every steps you are ready to finish the wizard and start using the product. Click the `Finish` button to complete the wizard.



Warning

The product's service will be restarted and the browser will be redirected to the administrative management console. This could take several seconds.

You can login to the Web Management Console with the administrator user that have just been created in the previous steps.



Setup HTTPS

Transport Layer Security (TLS) is a cryptographic protocol that provides communications security over a computer network. Websites, like the Web Management Console, are able to use TLS to secure all communications between their servers and web browsers.

The TLS protocol aims primarily to provide confidentiality (privacy) and data integrity between two communicating computer applications.

Warning

No TLS for the wizard

By default, TLS is not enabled for the basic configuration wizard. As a consequence sessions between the wizard's backend and the browser may be insecure.

Info

Performing the same steps as for the Web Management Console, it is possible to set up TLS for the basic configuration wizard. Remember completing the TLS setup before launching the wizard.

For instructions to set up TLS see Enabling HTTPS

License Activation

To activate your installation go to the **Settings > License** menu in the Web Management Console. If you have no valid license, you will only see your installation's **deployment ID**. You will also see a warning in the Web Management Console header.

After successful activation, the product will start downloading the latest available scan engines and malware databases. You can follow the status of the scan engine installation on the **Engines** page.

When all the licensed engines are installed, you can start using the MetaDefender Core to process files - File Processing.

Deployment ID

Each MetaDefender Core instance represented by an unique system fingerprint called `Deployment ID`

MetaDefender Core licensing is purely based on deployment ID activated. Therefore, everytime your MetaDefender Core is activated with a different deployment ID, then it will be captured as a different activation record accumulated into your total activation times allowed.

Notice

Deployment ID is auto generated by MetaDefender Core product upon installed, and it might be auto changed when you alter the system where MetaDefender Core relies on (e.g. network card, OS hostname).

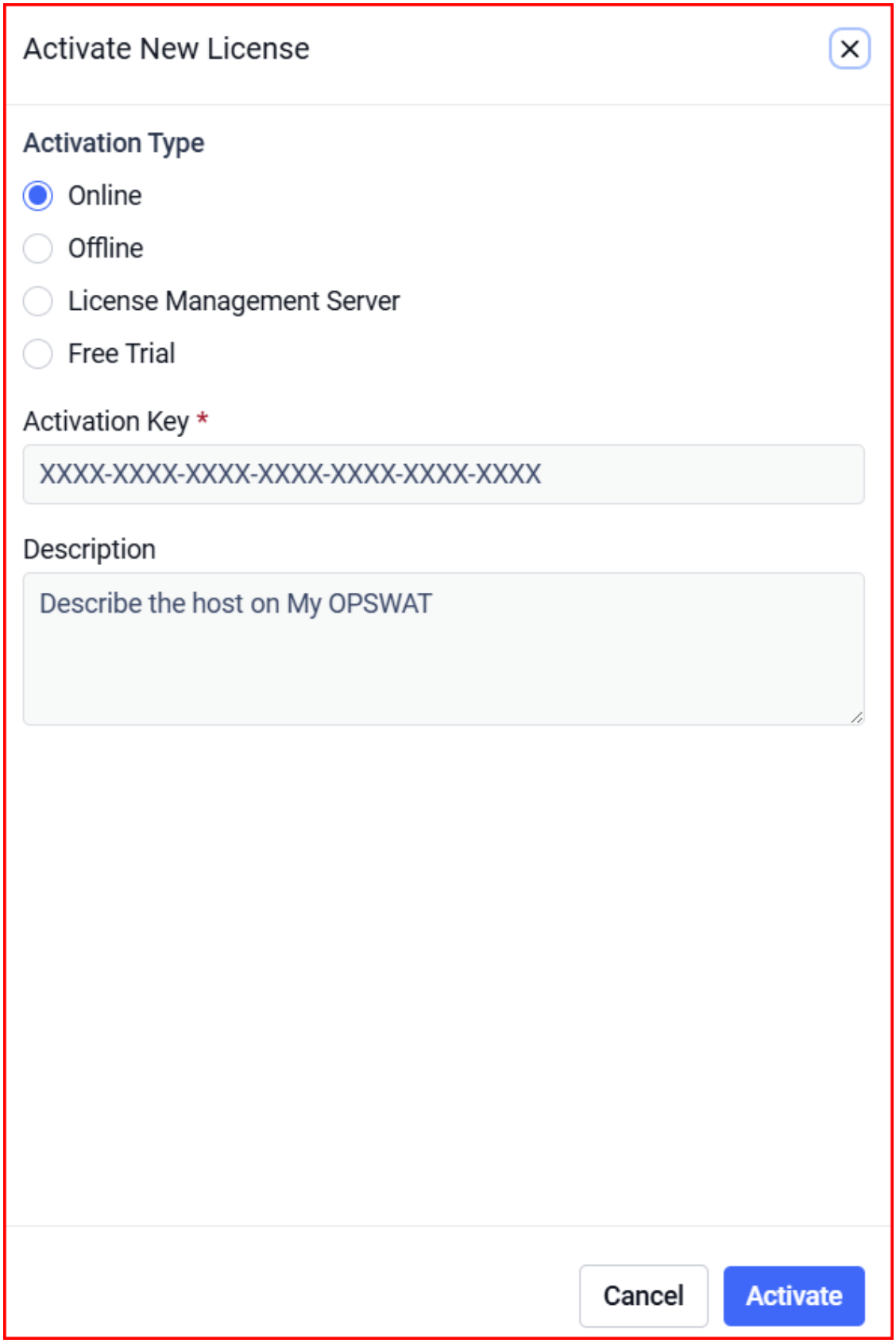
Deployment ID is an UUID version 4 random based string. Learn more at: https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_4_(random)

Deployment ID is auto generated by MetaDefender Core product upon installed, and it might be auto changed when you update the system where MetaDefender Core relies on.

Activation Modes

There are three different options for the product activation:

- Online activation: MetaDefender will connect directly to the OPSWAT licensing server online, and acquire its license based on your Activation key and its Deployment ID.
- Offline activation: Requiring you to upload a license file (.yml). That license file could be retrieved by activating your license via OPSWAT Portal. Follow the displayed instructions for details.
- Request trial key online: For evaluation purpose, you can receive an trial Activation key via email. Follow the displayed instructions for details.



Online activation

MetaDefender will connect directly to the OPSWAT licensing server online, and acquire its license based on your Activation key and its Deployment ID.

Required to input your Activation License key, and optional for description (but we recommend this for troubleshooting and auditing purpose if needed).

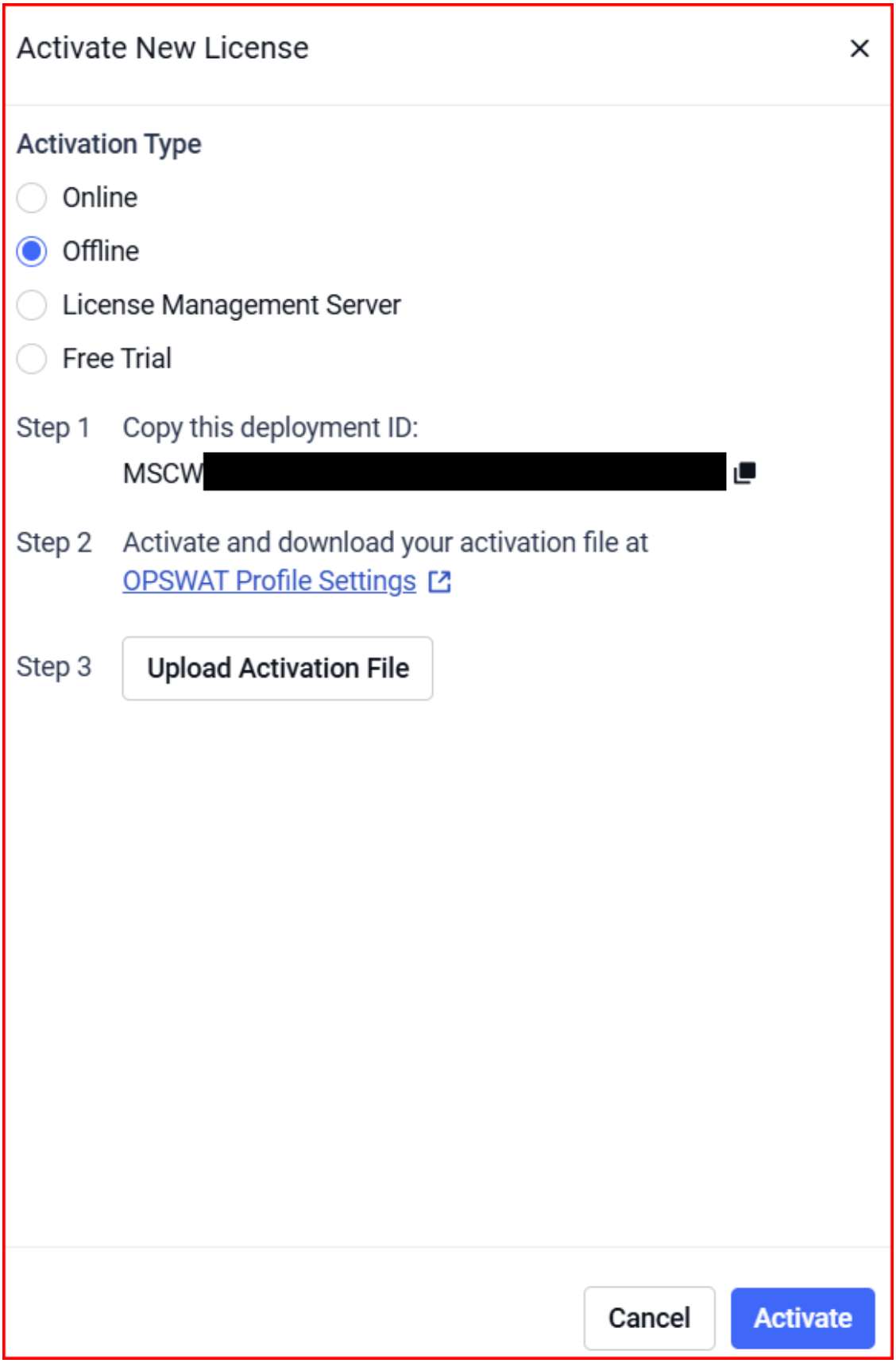
Warning

If you have installed or if you wish to use the MetaDefender Core in a restricted environment, you will have to allow access to the following hosts' for license activation:

- https://activation.dl.opswat.com Note: *IP address-based whitelisting on your firewall might fail after some time since OPSWAT uses CDN (Content Delivery Network) to faster delivery updates over the world, and IP address of edge servers might change over time.*

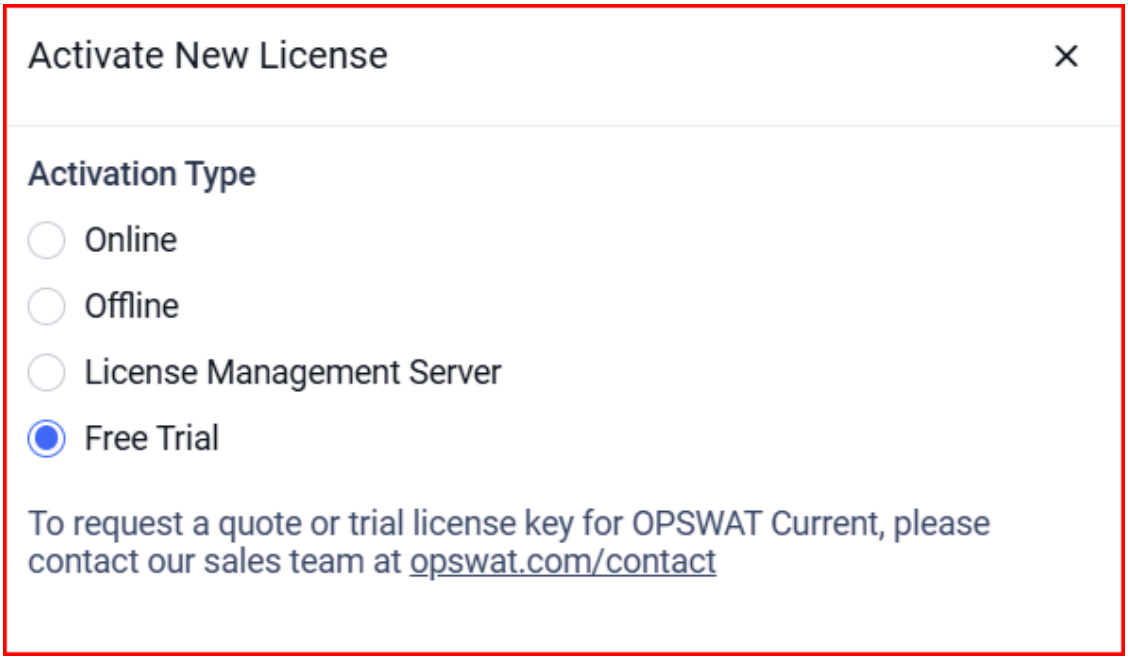
Offline activation

With no internet connection on the server the Metadefender v5 instance may be activated indirectly from a different machine, that has internet connection. The Deployment ID of the Metadefender v5 instance and the the Activation key received at the time of purchasing the product will be required. Follow the steps on the screen to activate the product offline.



Request trial key online

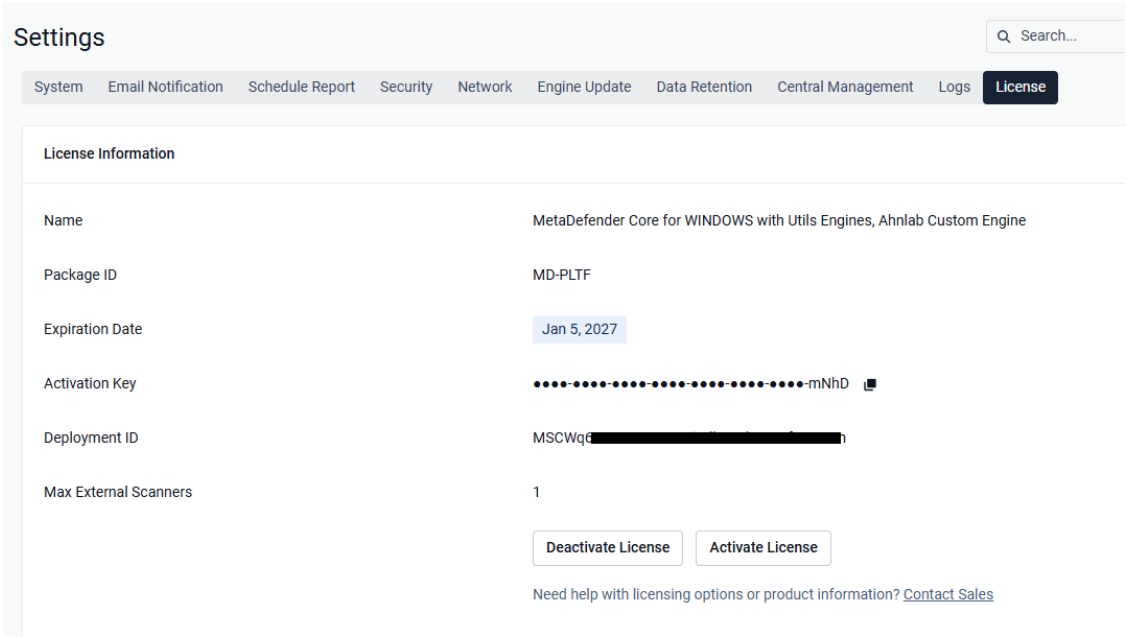
An evaluation license could be acquired by contacting our sales team.



Check Your License Details

For more license details and activating your MetaDefender Core, go to **Settings > License** menu tab on the Web Management Console:

- Name: product name as on your order (e.g. MetaDefender Core for WINDOWS with Utils Engines, Deep CDR Module, Metascan Windows - 4 engines, Threat Intelligence Engine)
- Package ID: product identification as on your order (e.g. MD-PLTF)
- Expiration Date: last day of license validity
- Activation key: key used for activating product
- Deployment ID: identification of this installation



Scripted license management

Using REST API calls, there is an option for scripted activation and deactivation for MetaDefender Core servers with **no Internet connection or if your infrastructure is using automation to create/destroy MetaDefender Core instances**. In this chapter, the steps of these two scenarios are described.

Requirements

- An installed MetaDefender Core instance without an Internet connection
- another computer that has an Internet connection and can run your activation scripts
- a manual or automated way to transfer data between the two computers

Activation steps

1. In order to activate MetaDefender Core, the deployment ID and activation key are needed to generate the license file.
   1. The activation key should be purchased from OPSWAT.
   2. MetaDefender Core deployment ID can be queried by using the REST API (for details, see GET Get Current License Information page)

> **Info**
> Save this deployment ID in your system, you might need this ID when the instance is unavailable at the time of deactivation.

2. Activate the license and get the license file using the following URL, where <quantity> is 1.

activation call · bash

```
curl -X GET https://activation.dl.opswat.com/activation?key=<activation-key>&deployment=<deployment unique-ID>&quantity=<quantity>
```

If the activation is successful, a license file is downloaded. Save this file.

HTTP status codes can be:

HTTP Response	Body	Comment
200 Ok	license file	Activation was successful.
200 Activation failed	error: '<user conform error message>'	Failed activation

HTTP Response	Body	Comment
200 Invalid parameter	error: 'Could not activate your product because the Activation Key you provided is invalid. Check if you typed it correctly or open a support ticket if problem persist.'	Invalid key format
200 Invalid parameter	error: 'Could not activate your product because the Deployment ID you provided is invalid. Check if you typed it correctly or open a support ticket if problem persist.'	Invalid deployment format
200 Invalid parameter	error: 'Could not activate your product because the quantity you provided is invalid. Check if you typed it correctly or open a support ticket if problem persist.'	Invalid deployment format
400 Bad request		Missing key, quantity or deployment
500 Internal server error	error: 'Internal server error (<error ID>). Please contact support'	

Info

In case of any activation issue, contact OPSWAT support for help

3. Upload license file to MetaDefender Core: The license file should be uploaded to the MetaDefender Core v4 to activate the product. For details, see page: POST Upload license key

Deactivation steps

1. For deactivation, the following information is required:
    1. Activation key
    2. MetaDefender Core deployment ID: can be queried by using the REST API (for details, see GET Get Current License Information page)
2. Deactivate license using the following URL:

deactivation call · bash

```
curl -X GET https://activation.dl.opswat.com/deactivation?key=<activation-
key>&deployment=<deployment unique-ID>
```

Response HTTP status codes can be:

HTTP Response	Body	Comment
200 Ok	result: ok	Successful deactivation
200 No active license found	error: 'Could not found any active license with the given parameters'	The license has not been activated yet or it has been deactivated already.
200 Invalid parameter	error: 'Could not deactivate your product because the Activation Key you provided is invalid. Check if you typed it correctly or open a support ticket if problem persist.'	Invalid key format
200 Invalid parameter	error: 'Could not deactivate your product because the Deployment ID you provided is invalid. Check if you typed it correctly or open a support ticket if problem persist.'	Invalid deployment format
400 Bad request		Missing key or deployment
500 Internal server error	error: 'Internal server error (<error ID>). Please contact support'	

Important notes

Warning

Product activation is tied to several hardware, operating system, and software parameters. In the event of one or more major hardware or software parameter changes, the product may become deactivated. Operating system updates and other software updates should not affect the activation status.

If the product is online-activated and has a live Internet connection, then in the event of a deployment ID change, the product reactivates itself. If the product doesn't have an Internet connection, then the administrator is responsible for reactivating the product.

Because of the above, the administrator should consider licensing restrictions of the offline deployments if

using any virtualization and/or containerization technologies.

File Processing

MetaDefender Core provides REST API interface for file processing business.

There are several ways to process files with MetaDefender Core:

- Analyze Files via Web Interface
- Process Files via REST API

Product Upgrading

Info

Simply run installer file, details could be found at Installation.

For command-line-mode upgrading (e.g. on **Linux**), just run the command line to install the new version of the product.

Recommendation for MetaDefender Core upgrade

Our goal is to help you execute the product upgrade continuously, efficiently, and correctly.

PostgreSQL service upgrade

- **Bundled PostgreSQL Service**: If you are using the bundled PostgreSQL server included in the installer, the PostgreSQL service will be automatically upgraded to the newer version (if available). No additional action is required for this process.
- **Remote PostgreSQL Service**: For remote PostgreSQL servers, customers are responsible for managing and upgrading the service independently. Upgrading MetaDefender Core does not update the remote PostgreSQL service.

Before MetaDefender Core upgrade

Prepare a proper disaster recovery plan (DRP) including some common practices:

- Backup, or take a snapshot of your current working MetaDefender Core server.
  - This is to help quickly rollback for any upgrade failure might cause later for any reason.
- Optionally backup your PostgreSQL database, especially when using remote shared database mode.
  - Reference: https://www.postgresql.org/docs/current/backup.html
  - Please do it after stopping all linked MetaDefender Core instances to avoid potential data collision.

- Lastly, if backing up entire server is not feasible, then at least export MetaDefender Core configuration package Import/Export configuration just in case you'll need to start off a new server with current MetaDefender Core configuration to keep the scanning service running without any impact.

On the PostgreSQL server, double check server resource to ensure sufficient RAM memory and disk space to be preserved for the upgrade process.

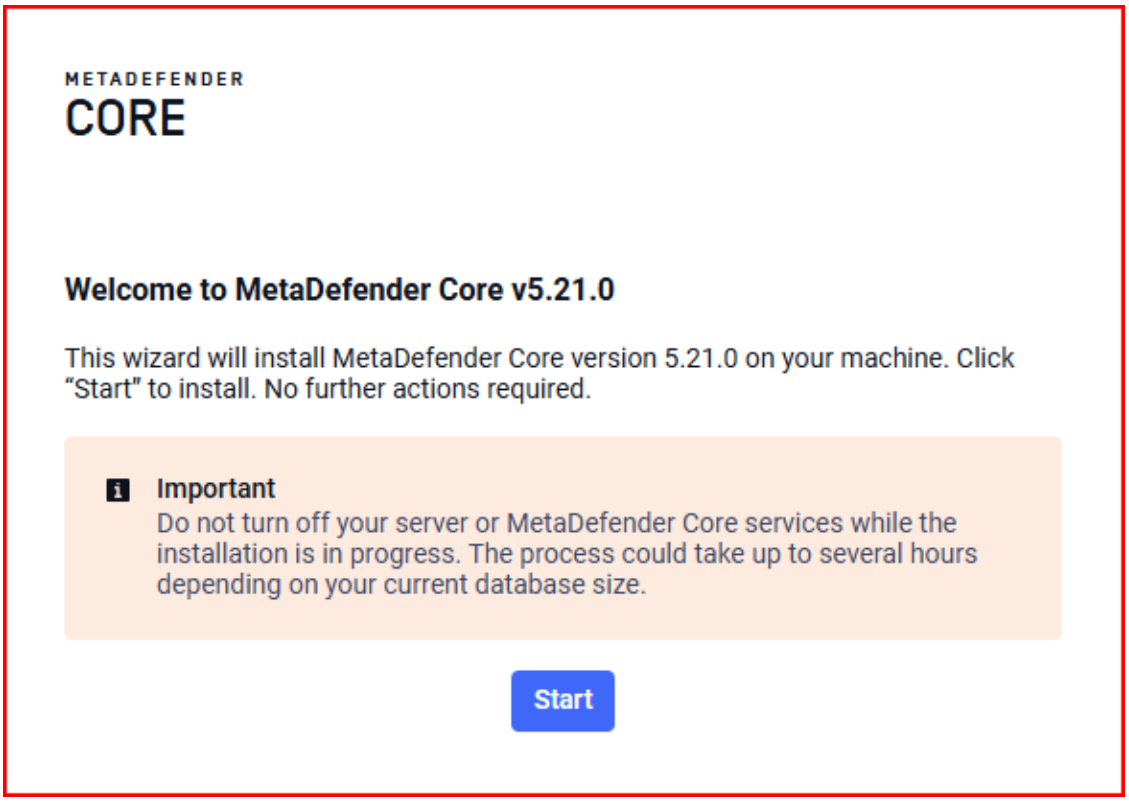
- 8GB RAM available at least during the upgrade.
- Disk availability is greater than MetaDefender Core database size.

While doing the upgrade

Applicable for shared database deployment:

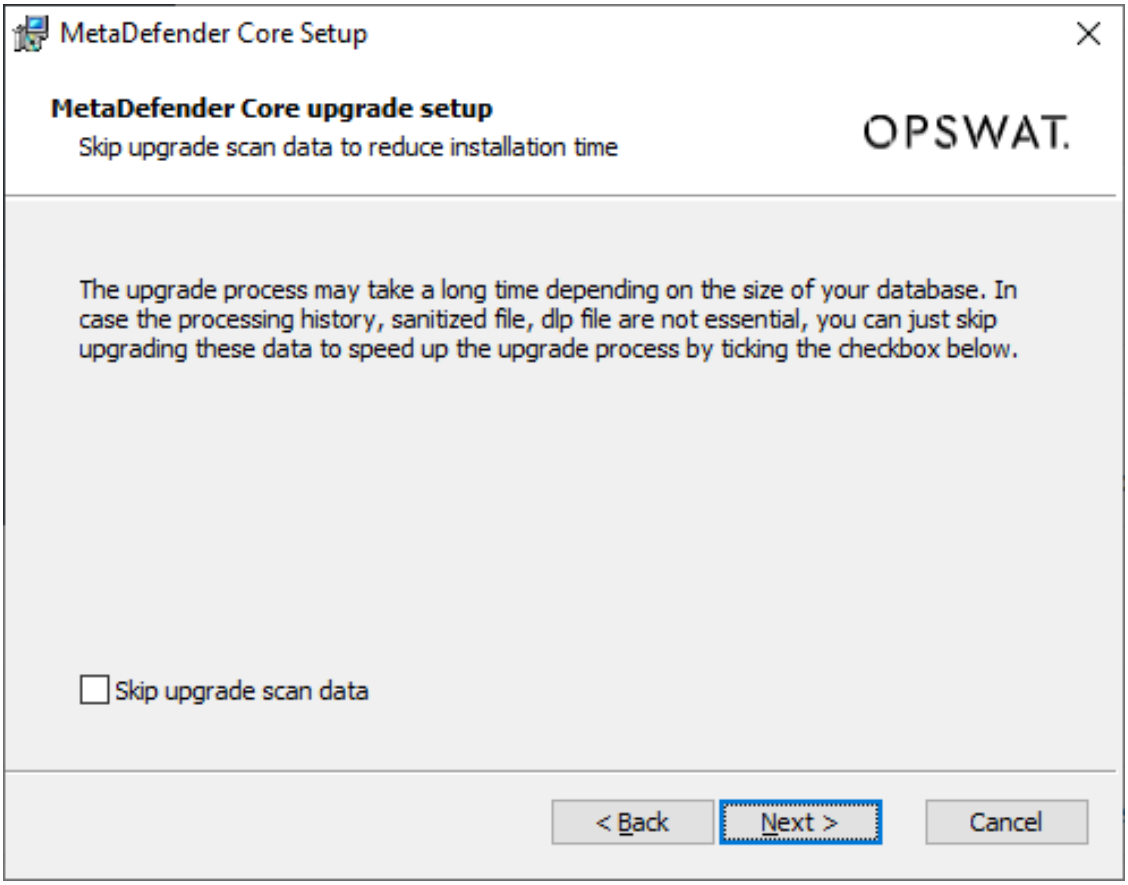
- Stop all MetaDefender Core instances' scanning service before performing the upgrade.
- Upgrade one (1) MetaDefender Core instance at a time.
- Proceed to upgrade MetaDefender Core instance where the bundle local PostgreSQL server is used for shared database setup.
- Otherwise is using remote PostgreSQL then the upgrade order does not matter.

For large size database, mostly for scan history, expecting long time to be spent on the upgrade process.



The product supports an option to skip data migration for scan history to expedite the process drastically and mitigate any failure risk. You can manually backup all data from your PostgreSQL server for offline usage if any.

Check "Skip upgrade scan data" to enable this feature while doing the upgrade.



or add following key in the ignition file:

Code

```
[internal]
skip_upgrade_scan_data = 0
```

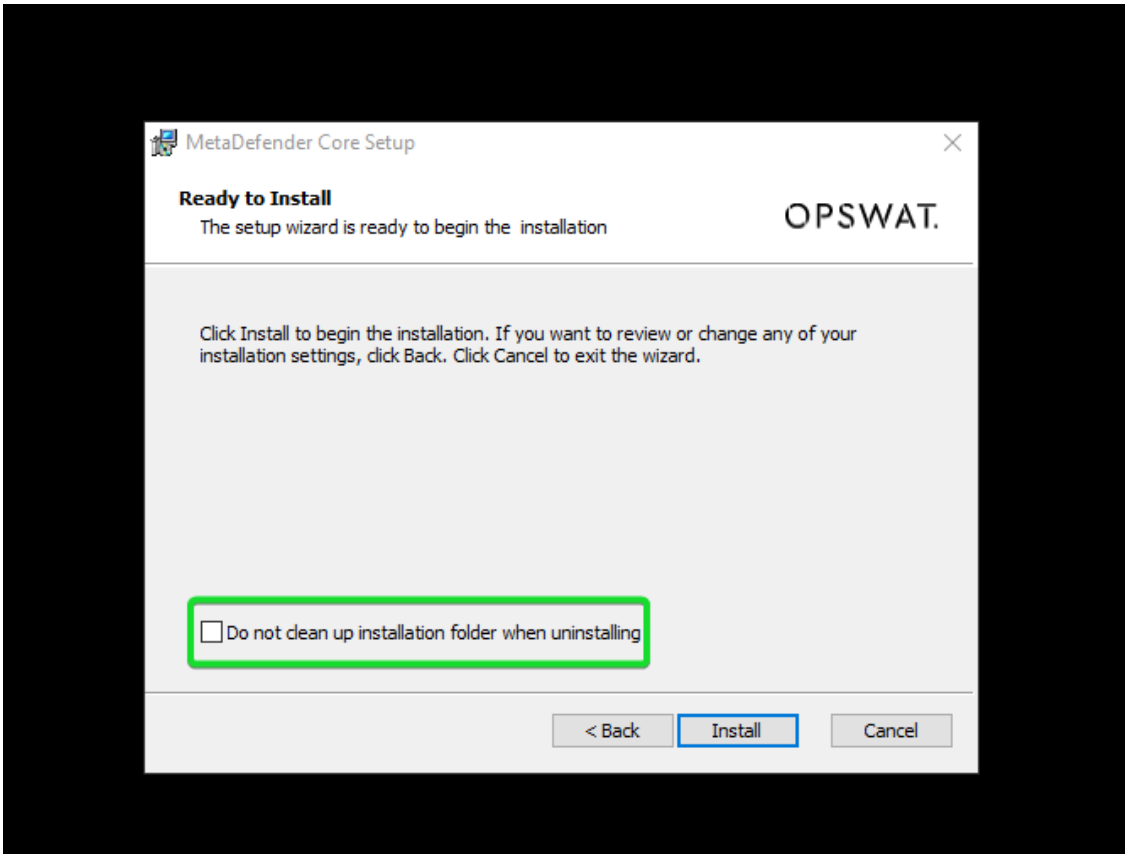
Info

You might notice this option during the upgrade process

"Do not clean up installation folder when uninstalling"

This option is designed for advanced users who need to preserve the installation directory after product removal. It provides control over whether the installer removes the entire installation folder during uninstallation (via msiexec)

- **Standard Users**: **If you are performing a typical uninstallation through the wizard interface, you can safely ignore this option**. The default behavior (cleaning up the installation folder) is appropriate for most use cases.
- **Advanced Users**: Select this option if you:
  - Need to preserve custom files or configurations stored in the installation directory
  - Plan to reinstall the product and want to retain certain files
  - Are performing administrative tasks that require keeping the installation structure intact



Rollback

MetaDefender Core provides automatic rollback protection during the installation phase, but **rollback is not possible after a successful execution of the installer package**. For this reason, **we strongly recommend backing up your server before performing any upgrade**.

Automatic Rollback During Installation

If an issue occurs during the installation process, the installer will automatically rollback changes:

Windows (WIX Installer)

- The WIX installer automatically handles rollback if errors occur during the installation wizard
- Common scenarios that trigger rollback:
  - Insufficient permissions
  - Missing required third-party libraries
  - Installation file corruption

Linux (Package Managers)

- **Debian-based systems (APT/dpkg)**: The package manager automatically rolls back if the `.deb` package installation fails

- **RHEL-based systems (YUM/RPM)**: The package manager automatically rolls back if the `.rpm` package installation fails

In all cases, the installer or package manager handles the rollback process automatically—no user intervention is required.

Post-Installation Rollback Limitations

Once the package has been extracted and installed successfully, **automatic rollback is no longer available**. At this point:

- You cannot use the installer or package manager to revert to the previous version
- The only recovery option is to restore from a backup

> **Best Practices**
>
> To ensure you can recover from upgrade issues:
>
> 1. **Always back up your server** before starting the upgrade process
> 2. **Test the upgrade** in a non-production environment first
> 3. **Verify system requirements** (disk space, permissions, dependencies) before upgrading
> 4. **Plan for downtime** to allow for backup creation and testing after upgrade

Troubleshooting

If the upgrade process ever runs into an error:

- Double check your network and services, applicable to remote PostgreSQL server.
- Optionally configure MetaDefender Core to skip migrating scan history.
  - Windows
    - Open Windows registry, and navigate to HKLM\Software\OPSWAT\Metascan
    - Add a new key `internal/skip_upgrade_scan_data` with value = 1
  - Linux
    - Open MetaDefender Core configuration file `/etc/ometascan/ometascan.conf`
    - Add new key into the file
      - [internal]
      - skip_upgrade_scan_data=1
- Note: This key will be auto removed upon the upgrade finishes.
- Then restart MetaDefender Core service to retry the upgrade.

- Might retry this a few times until the upgrade is done.

Just in case the upgrade process still fails, please contact OPSWAT support with the support package - How to Create Support Package With Bundle Tools? This will help us analyze the root cause and figure out an optimal solution for your system failure.

Using external Load Balancer

Application Load Balancing (Layer 7)

This page provides information about recommended ways to use sessions with a Layer 7 load-balancer.

Most load-balancers have the ability to provide cookies for pinning subsequent traffic from a client to the appropriate server. This method is called sticky session, session persistence or session affinity. Using cookies efficiently requires the client to know when a cookie should be sent, should not be sent or should be deleted.

Sticky session load balancing

Each MetaDefender Core instance has its own databases and application sessions that cannot be seen by other instances. Therefore, in order to get the related data/response to our queries we should ask the appropriate MetaDefender Core v5 server. On the other hand, to keep the advantages of the used load balancing method, cookies should not be sent if it is not necessary.

Single file scanning

Step	Stage	Task	Cookie usage
1.	Sending file	Initiate processing a file on the client side. Send file through the load-balancer. (See POST File Analysis)	Cookie should not be sent
2.	Sending file	Save the cookie and the data_id you got from the load-balancer	Save cookie
3.	Getting result	Request result related to data_id saved in step 2. (See GET File Analysis)	Send cookie saved in step 2
4.	Getting result	If processing is in progress (See GET File Analysis), wait a little while and repeat step 3.	Send cookie saved in step 2

Step	Stage	Task	Cookie usage

Batch scanning

Step	Stage	Task	Cookie usage
1.	Open batch	Initiate processing file(s) in batch. Request a batch ID through the load-balancer. (See POST Initiate Batch)	Cookie should not be sent
2.	Open batch	Save the cookie and the batch_id you got from a Core server through the load-balancer.	Save cookie
3.	Sending files	Send file through the load-balancer. (See POST File Analysis (using batch header))	Send cookie saved in step 2
4.	Sending files	Save the data_id you got from the load-balancer.	•
	Sending files	Status/result of scanning of sent files can be queried: (See GET File Analysis)	Send cookie saved in step 2
5.	Sending files	Repeat step 3-4. with files wanted to be in the same batch.	•
6.	Getting batch status	See GET Batch Status	Send cookie saved in step 2
7.	Close batch	Tell the server that no more files will be sent to this batch. (See POST Close Batch). (This will only be successful if all the files sent to the	Send cookie saved in step 2

Step	Stage	Task	Cookie usage
		batch have been processed already. Repeat this step until batch is closed.)	
8.	Getting results	Request results related to batch ID saved in step 2. (See GET Download Batch Signed Result	Send cookie saved in step 2

If it does not matter which upstream server responds, the queries should be sent without cookie.

Info

It is recommended not to send cookies when it's not necessary to allow load-balancer to use its own method to share the load between MetaDefender Core v5 servers.

Limitations, additional notes

Using load-balancing between MetaDefender Core servers does not support:

- Global scan history
- Core server administration through load-balancer

OPSWAT products that support HTTP load balanced MetaDefender Cores

Product name	Minimum version
MetaDefender Kiosk	4.3.4
MetaDefender Endpoint	Windows: 7.6.247.0 Mac: 10.4.243.0
MetaDefender Email Gateway Security	4.3.0
MetaDefender ICAP Server	4.3.0
MetaDefender Managed File Transfer	1.3.0
MetaDefender Storage Security	2.0.7

DNS Load Balancing

Using this method is logically similar to Layer 7 load-balancing.

Briefly how it works

Client uses a domain name to send a query to a server. Client's DNS server has more "A" records for that name with different IPs. When a client resolves the server's domain name DNS server randomly chooses an IP for that name to send back. When a session is used on the application layer, client should know the IP address of the Core server that handles that specific session. In every other case, client should resolve the domain name with DNS query to let requests to be balanced between MetaDefender Core servers.

Single file scanning

Step	Stage	Task	Addressing
1.	Choose a Core server by using DNS load balancing	Resolve the Core servers' common domain name.	Use domain name
2.	Choose a Core server by using DNS load balancing	Save the IP gotten from the DNS server.	Save the IP
3.	Sending file	Initiate processing a file on the client side. (See POST File Analysis)	Use IP saved in step 2
4.	Sending file	Save the data_id got from the Core server	•
5.	Getting result	Request result related to data_id saved in step 2. (See GET File Analysis)	Use IP saved in step 2
6.	Getting result	If processing is in progress (See GET File Analysis), wait a little while and repeat step 3.	Use IP saved in step 2

Batch scanning

Step	Stage	Task	Addressing
1.	Choose a Core server by	Resolve the Core	Use domain name

Step	Stage	Task	Addressing
	using DNS load balancing	servers' common domain name.	
2.	Choose a Core server by using DNS load balancing	Save the IP gotten from the DNS server.	Save the IP
3.	Open batch	Initiate processing file(s) in batch. Request a batch ID. (See POST Initiate Batch)	Use IP saved in step 2
4.	Sending files	Send file to the specific Core server with the basch ID saved in step 3. (See POST File Analysis (using batch header))	Use IP saved in step 2
5.	Sending files	Save the data_id you got from the Core server.	•
6.	Sending files	Status/result of scanning of sent files can be queried: (See GET File Analysis)	Use IP saved in step 2
7.	Sending files	Repeat step 3-4. with files wanted to be in the same batch.	•
8.	Getting batch status	See GET Batch Status	Use IP saved in step 2
9.	Close batch	Tell the server that no more files will be sent to this batch. (See POST Close Batch). (This will only be successful if all the files sent to the batch have been processed already. Repeat this step until	Use IP saved in step 2

Step	Stage	Task	Addressing
		batch is closed.)	
10.	Getting results	Request results related to batch ID saved in step 2. (See GET Download Batch Signed Result).	Use IP saved in step 2

Limitations, additional notes

Using load-balancing between MetaDefender Core servers does not support:

- Global scan history
- Core server administration via DNS load-balancing

OPSWAT products that support DNS load balanced MetaDefender Cores

Product name	Minimum version
MetaDefender Kiosk	does not support yet
MetaDefender Endpoint	does not support yet
MetaDefender Email Gateway Security	does not support yet
MetaDefender ICAP Server	does not support yet
MetaDefender Managed File Transfer	does not support yet

Performance and Load Estimation

Disclaimer

These results should be viewed as guidelines and not performance guarantees, since there are many variables that affect performance (file set, network configurations, hardware characteristics, etc.). If throughput is important to your implementation, OPSWAT recommends site-specific benchmarking before implementing a production solution.

Some factors that affect performance

- MetaDefender Core version
- MetaDefender Core engine package and configuration
  - set of engines (which and how many)
  - product configuration (e.g., thread pool size)
- System environment
  - server profile (CPU, RAM, hard disk)
  - client application location - remote or local
  - system caching and engine level caching
- Dataset
  - encrypted or decrypted
  - file types
    - different file types (e.g., document, image, executable)
    - archive file or compound document format files
  - file size
  - bad or unknown (assume to be clean)
- Performance tool

How test results are calculated

Performance (mainly scanning speed) is measured by throughput rather than unit speed. For example, if it takes 10 seconds to process 1 file, and it also takes 10 seconds to process 10 files, then performance is quantified as 1 second per file, rather than 10 seconds.

- total time / total number of files processed: 10 seconds / 10 files = 1 second / file.

Standalone DB - Windows

Disclaimer

These results should be viewed as guidelines and not performance guarantees, since there are many variables that affect performance (file set, network configurations, hardware characteristics, etc.). If throughput is important to your implementation, OPSWAT recommends site-specific benchmarking before implementing a production solution.

Environment

Using AWS environment with the specification below:

MetaDefender Core

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type	Benchmark
Windows Server 2022	c5a.8xlarge	32	64	Up to 10	SSD	Amazon EC2 c5a.8xlarge - Geekbench

Client tool

A simple tool written in Python to collect files in a designated folder and submit requests to MetaDefender Core.

This client tool is running on CentOS 7

Code · python

```
files_to_scan = list of files to scan
scan_futures = [asyncio.create_task(self.scan_file(file_path)) for file_path in files_to_scan]

async def scan_file(self, file_path):
    api_url = f'{self.core_url}/file'

    with open(file_path, 'rb') as file:
        file_content = file.read()

    headers = {'Content-Type': 'application/octet-stream', 'filename': file_path}

    max_retries = 20  # Maximum number of retry attempts
    retry_delay = 120  # Delay in seconds before retrying

    for attempt in range(max_retries + 1):
        try:
            response = requests.post(api_url, data=file_content, headers=headers)
            status_code = response.status_code
            if status_code == 200:
                response_json = response.json()
                data_id = response_json.get('data_id')
                return status_code, response_json, data_id

            if status_code == 503:
                print(f"Received status code {status_code}. Retrying in {retry_delay}
seconds...")
                await asyncio.sleep(retry_delay)
            else:
                return None, None, None
```

```
        except requests.RequestException as e:
            return None, None, None

    return None, None, None
```

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
CentOS 7	c5.4xlarge	16	32	Up to 10	SSD

Dataset

Detailed information of dataset below will be used for testing:

File category	File type	Number of files	Total size (MB)	Average file size (MB)
Adobe	PDF	370	385 MB	1.0 MB
Executable	EXE	45	309.5 MB	6.9 MB
	MSI	15	45.75 MB	3.1 MB
Image	BMP	80	515 MB	6.4 MB
	JPG	420	237.5 MB	0.6 MB
	PNG	345	169 MB	0.5 MB
Media	MP3	135	865 MB	6.4 MB
	MP4	50	500 MB	10.0 MB
Office	DOCX	235	190 MB	0.8 MB
	DOC	225	486 MB	2.2 MB
	PPTX	365	860 MB	2.4 MB
	PPT	355	1950 MB	5.5 MB

File category	File type	Number of files	Total size (MB)	Average file size (MB)
	XLSX	340	283.5 MB	0.8 MB
	XLS	335	284.5 MB	0.8 MB
Text	CSV	100	236 MB	2.4 MB
	HTML	1075	76 MB	0.1 MB
	TXT	500	210 MB	0.4 MB
Archive	ZIP	Compressed files: 10 Extracted files: 270	Compressed size: 125.5 MB Extracted size: 156.5 MB	Avg compressed size: 12.6 MB Avg extracted size: 0.6 MB
Summary (compressed)		5000	7728.5 MB	1.55 MB average file size
Summary (extracted)		5260	7759.5 MB	1.48 MB average file size

Product Information

- MetaDefender Core 5.23.0
- Engines:
  - Metascan 8: Ahnlab, Avira, ClamAV, ESET, Bitdefender, K7, Quick Heal, VirIT Explorer
  - Metascan 12: Metascan 8, Varist, Ikarus, Emsisoft, Tachyon
  - Metascan 16: Metascan 12, Bkav Pro, CMC, Xcitium, Zillya!
  - Deep CDR: 8.1.0
  - Proactive DLP: 3.4.2
  - Archive: 8.1.0
  - File type analysis: 8.1.0
  - File-based vulnerability assessment: 4.3.0.0
  - Adaptive Sandbox: 3.3.0

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update
- Max file size: 99999999

Archive extraction settings

- Max recursion level: 99999999
- Max number of extracted files: 99999999
- Max total size of extracted files: 99999999
- Timeout: 10 minutes
- Handle archive extraction task as Failed: true
  - Extracted partially: true

Metascan AV settings

- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Adaptive Sandbox

- Max file size: 18446744073709 MB (max)
- Timeout: 86400 (max)

Performance test results

Summary

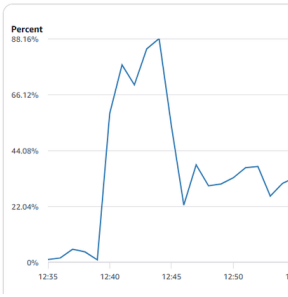
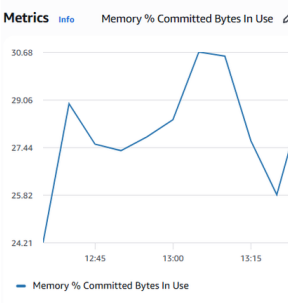
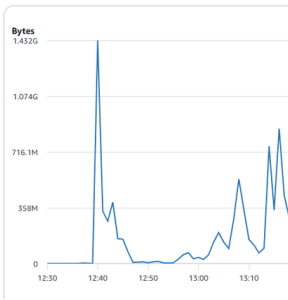
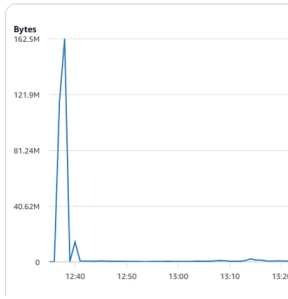
MetaDefender Core with single engine (technology):

MetaDefender Core with common engine packages:

System resource utilization charts

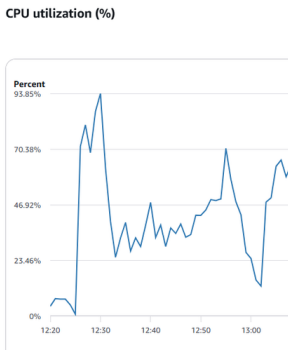
MetaDefender Core with Metascan 8 + Deep CDR + Proactive DLP + Vulnerability + Sandbox:

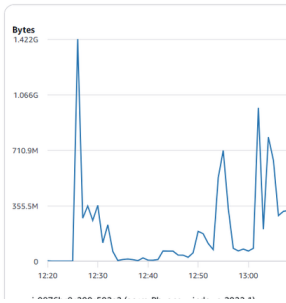
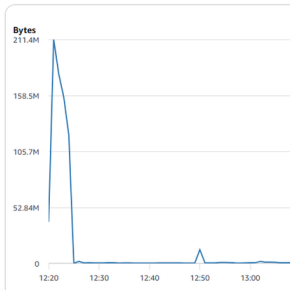
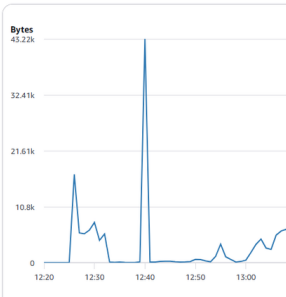
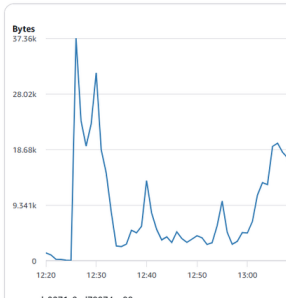
	Utilization charts
CPU usage	

	Utilization charts
	CPU utilization (%) 
Memory usage	
Network usage	Network in (bytes)  Network out (bytes) 
Disk usage	

	Utilization charts
	

MetaDefender Core with Metascan 16 + Deep CDR + Proactive DLP + Vulnerability + Sandbox:

	Utilization charts
CPU usage	
Memory usage	

	Utilization charts
Network usage	Network in (bytes)  Network out (bytes) 
Disk usage	Read throughput (KiB/s)  Write throughput (KiB/s) 

Standalone DB - Linux

Disclaimer

These results should be viewed as guidelines and not performance guarantees, since there are many variables that affect performance (file set, network configurations, hardware characteristics, etc.). If throughput is important to your implementation, OPSWAT recommends site-specific benchmarking before implementing a production solution.

Environment

Using AWS environment with the specification below:

MetaDefender Core

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type	Benchmark
Ubuntu 24.04	c5a.8xlarge	32	64	Up to 10	SSD	Amazon EC2 c5a.8xlarge - Geekbench

Client tool

A simple tool written in Python to collect files in a designated folder and submit requests to MetaDefender Core.

This client tool is running on CentOS 7

Code · python

```
files_to_scan = list of files to scan
scan_futures = [asyncio.create_task(self.scan_file(file_path)) for file_path in files_to_scan]

async def scan_file(self, file_path):
    api_url = f'{self.core_url}/file'

    with open(file_path, 'rb') as file:
        file_content = file.read()

    headers = {'Content-Type': 'application/octet-stream', 'filename': file_path}

    max_retries = 20  # Maximum number of retry attempts
    retry_delay = 120  # Delay in seconds before retrying

    for attempt in range(max_retries + 1):
        try:
            response = requests.post(api_url, data=file_content, headers=headers)
            status_code = response.status_code
```

```
            if status_code == 200:
                response_json = response.json()
                data_id = response_json.get('data_id')
                return status_code, response_json, data_id

            if status_code == 503:
                print(f"Received status code {status_code}. Retrying in {retry_delay}
seconds...")
                await asyncio.sleep(retry_delay)
            else:
                return None, None, None

        except requests.RequestException as e:
            return None, None, None

    return None, None, None
```

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
CentOS 7	c5.4xlarge	16	32	Up to 10	SSD

Dataset

Detailed information of dataset below will be used for testing:

File category	File type	Number of files	Total size (MB)	Average file size (MB)
Adobe	PDF	370	385 MB	1.0 MB
Executable	EXE	45	309.5 MB	6.9 MB
	MSI	15	45.75 MB	3.1 MB
Image	BMP	80	515 MB	6.4 MB
	JPG	420	237.5 MB	0.6 MB
	PNG	345	169 MB	0.5 MB
Media	MP3	135	865 MB	6.4 MB

File category	File type	Number of files	Total size (MB)	Average file size (MB)
	MP4	50	500 MB	10.0 MB
Office	DOCX	235	190 MB	0.8 MB
	DOC	225	486 MB	2.2 MB
	PPTX	365	860 MB	2.4 MB
	PPT	355	1950 MB	5.5 MB
	XLSX	340	283.5 MB	0.8 MB
	XLS	335	284.5 MB	0.8 MB
Text	CSV	100	236 MB	2.4 MB
	HTML	1075	76 MB	0.1 MB
	TXT	500	210 MB	0.4 MB
Archive	ZIP	Compressed files: 10 Extracted files: 270	Compressed size: 125.5 MB Extracted size: 156.5 MB	Avg compressed size: 12.6 MB Avg extracted size: 0.6 MB
Summary (compressed)		5000	7728.5 MB	1.55 MB average file size
Summary (extracted)		5260	7759.5 MB	1.48 MB average file size

Product Information

Product versions:

- MetaDefender Core version 5.23.0
- Engines:
  - Metascan 5: Ahnlab, Bitdefender, ClamAV, ESET, K7
  - Metascan 10: Metascan 5 + Avira, Varist, IKARUS, Quick Heal, TACHYON

- Metascan 20: Metascan 10 + Lionic, CMC, CrowdStrike Falcon ML, Aurora, Trellix, Xcitium, RocketCyber, Sophos, Webroot SMD, Xvirus Anti-Malware
- Deep CDR: 8.1.1
- Proactive DLP: 3.4.2
- Archive: 8.1.1
- File type analysis: 8.1.1
- File-based vulnerability assessment: 4.3.0.0
- Adaptive Sandbox: 3.3.0

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update
- Max file size: 99999999

Archive extraction

- Max recursion level: 99999999
- Max number of extracted files: 99999999
- Max total size of extracted files: 99999999
- Timeout: 10 minutes
- Handle archive extraction task as Failed: true
  - Extracted partially: true

Metascan

- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Adaptive Sandbox

- Max file size: 18446744073709 MB (max)
- Timeout: 86400 (max)

Performance test results

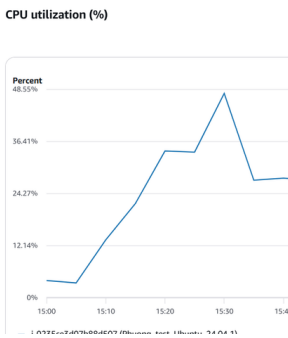
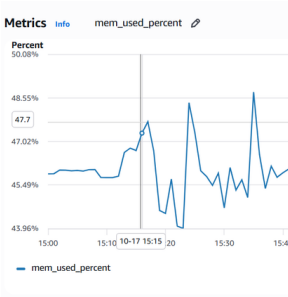
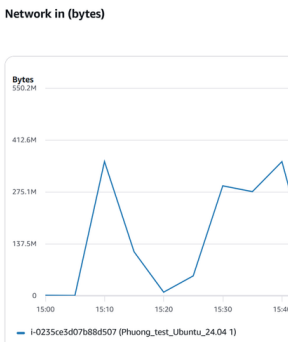
Summary

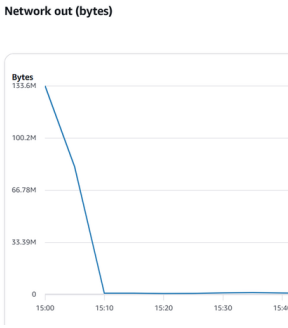
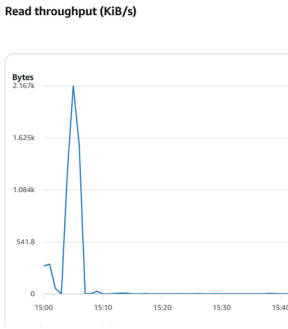
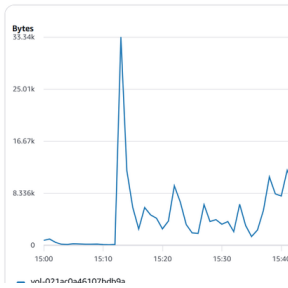
MetaDefender Core with single engine (technology):

MetaDefender Core with common engine packages:

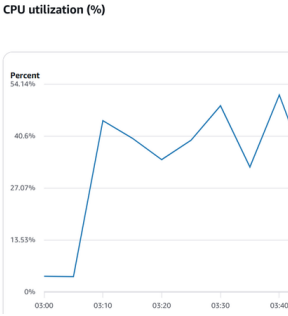
System resource utilization charts

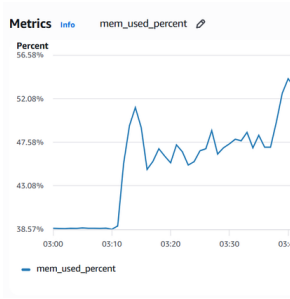
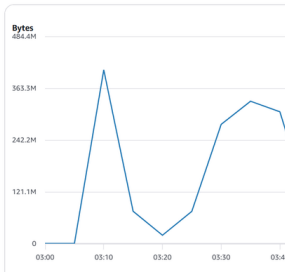
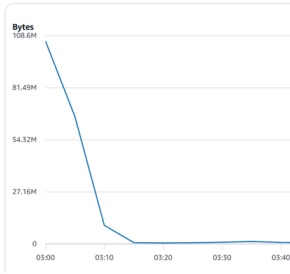
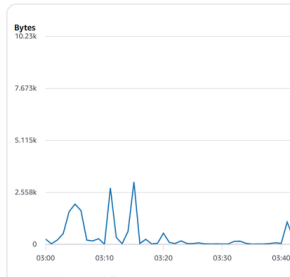
MetaDefender Core with Metascan 10 + Deep CDR + Proactive DLP + Vulnerability + Sandbox:

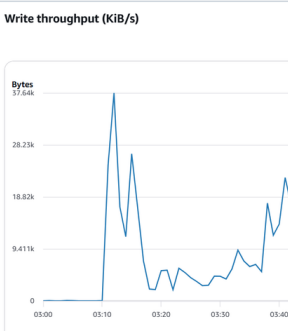
	Utilization charts
CPU usage	
Memory usage	
Network usage	

	Utilization charts
	
Disk usage	 

MetaDefender Core with Metascan 20 + Deep CDR + Proactive DLP + Vulnerability + Sandbox:

	Utilization charts
CPU usage	
Memory usage	

	Utilization charts
	
Network usage	**Network in (bytes)**  **Network out (bytes)** 
Disk usage	**Read throughput (KiB/s)** 

	Utilization charts
	

Shared DB - Windows

Disclaimer

These results should be viewed as guidelines and not performance guarantees, since there are many variables that affect performance (file set, network configurations, hardware characteristics, etc.). If throughput is important to your implementation, OPSWAT recommends site-specific benchmarking before implementing a production solution.

Environment

Using AWS environment with the specification below:

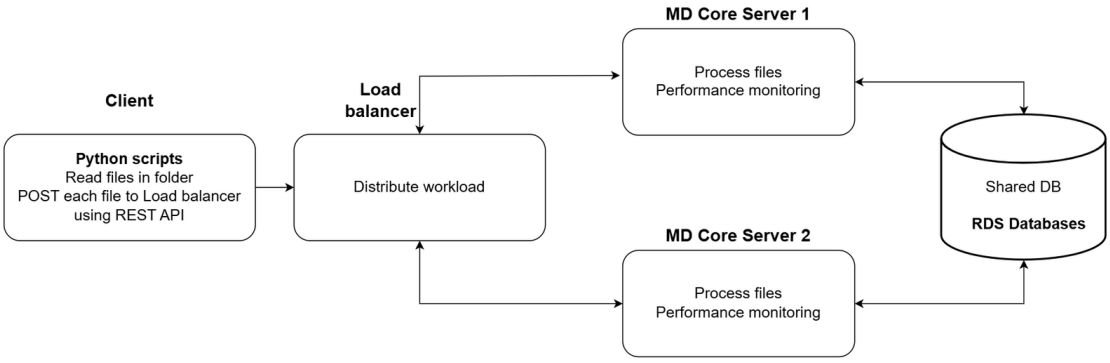
MetaDefender Core

	OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
MetaDefender Core #1	Windows Server 2022	c5a.8xlarge	32	64	Up to 10	SSD
MetaDefender Core #2	Windows Server 2022	c5a.8xlarge	32	64	Up to 10	SSD

RDS

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
Windows Server 2022	db.m7i.4xlarge	16	64	Up to 10	SSD

Deployment Model



Using a AWS Load Balancer to distribute files sent from the client tool to two (2) different MetaDefender Core servers applying Round Robin algorithm. With this algorithm, each MetaDefender Core server is supposed to receive same number of requests.

Client tool

A simple tool written in Python to collect files in a designated folder and submit requests to Load Balancer mentioned above.

Code · python

```
files_to_scan = list of files to scan
scan_futures =[]
for file_path in files_to_scan:
    scan_futures.append(asyncio.create_task(self.scan_file(file_path, self.load_balancer_url))

async def scan_file(self, file_path, core_url):
    api_url = f'{core_url}/file'

    with open(file_path, 'rb') as file:
        file_content = file.read()

    headers = {'Content-Type': 'application/octet-stream', 'filename': file_path}

    max_retries = 20  # Maximum number of retry attempts
    retry_delay = 120  # Delay in seconds before retrying

    for attempt in range(max_retries + 1):
```

```
        try:
            starttime = time.time()
            response = requests.post(api_url, data=file_content, headers=headers)
            endtime = time.time()
            status_code = response.status_code
            if status_code == 200:
                self.post_wait = self.post_wait + endtime - starttime
                self.num_post_req += 1
                response_json = response.json()
                data_id = response_json.get('data_id')
                return status_code, response_json, data_id

            if status_code == 503:
                print(f"Received status code {status_code}. Retrying in {retry_delay}
seconds...")
                await asyncio.sleep(retry_delay)
            else:
                return None, None, None

        except requests.RequestException as e:
            return None, None, None

    return None, None, None
```

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
CentOS 7	c5.4xlarge	16	32	Up to 10	SSD

Dataset

Detailed information of dataset below will be used for testing:

File category	File type	Number of files	Total size (MB)	Average file size (MB)
Adobe	PDF	370	385 MB	1.0 MB
Executable	EXE	45	309.5 MB	6.9 MB
	MSI	15	45.75 MB	3.1 MB
Image	BMP	80	515 MB	6.4 MB

File category	File type	Number of files	Total size (MB)	Average file size (MB)
	JPG	420	237.5 MB	0.6 MB
	PNG	345	169 MB	0.5 MB
Media	MP3	135	865 MB	6.4 MB
	MP4	50	500 MB	10.0 MB
Office	DOCX	235	190 MB	0.8 MB
	DOC	225	486 MB	2.2 MB
	PPTX	365	860 MB	2.4 MB
	PPT	355	1950 MB	5.5 MB
	XLSX	340	283.5 MB	0.8 MB
	XLS	335	284.5 MB	0.8 MB
Text	CSV	100	236 MB	2.4 MB
	HTML	1075	76 MB	0.1 MB
	TXT	500	210 MB	0.4 MB
Archive	ZIP	Compressed files: 10 Extracted files: 270	Compressed size: 125.5 MB Extracted size: 156.5 MB	Avg compressed size: 12.6 MB Avg extracted size: 0.6 MB
Summary (compressed)		5000	7728.5 MB	1.55 MB average file size
Summary (extracted)		5260	7759.5 MB	1.48 MB average file size

Product Information

Product versions:

- MetaDefender Core 5.23.0
- Engines:
  - Metascan 8: Ahnlab, Avira, ClamAV, ESET, Bitdefender, K7, Quick Heal, VirIT Explorer
  - Metascan 12: Metascan 8, Varist, Ikarus, Emsisoft, Tachyon
  - Metascan 16: Metascan 12, Bkav Pro, CMC, Xcitium, Zillya!
  - Deep CDR: 8.1.0
  - Proactive DLP: 3.4.2
  - Archive: 8.1.0
  - File type analysis: 8.1.0
  - File-based vulnerability assessment: 4.3.0.0
  - Adaptive Sandbox: 3.3.0

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update
- Max file size: 99999999

Archive extraction settings

- Max recursion level: 99999999
- Max number of extracted files: 99999999
- Max total size of extracted files: 99999999
- Timeout: 10 minutes
- Handle archive extraction task as Failed: true
  - Extracted partially: true

Metascan AV settings

- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Adaptive Sandbox

- Max file size: 18446744073709 MB (max)
- Timeout: 86400 (max)

Performance test results

MetaDefender Core with single engine (technology)

Summary metrics:

System resource utilization:

MetaDefender Core with common engine packages

Summary metrics:

System resource utilization:

Recommendations

Controlling total processing time of each MD Core server:

In this deployment model, we should organize and send files in the way that it best utilizes the load of each MD Core server. It is not a good practice if one Core server is free while the other one is busy. By optimizing the distribution of files, we can ensure that each Core server is utilized efficiently, thereby improving overall system performance. Furthermore, this approach can help prevent bottlenecks and minimize the chances of system overload.

Adding proper number of MD Core servers to the cluster:

Adding more Core servers to this model will increase more load on the shared database. When adding a new MD Core server, users should monitor performance of database server such as memory/CPU consumption, disk usage, network bandwidth, request response time and so on… to see if it still can handle the load. This is important in order to maintain optimal performance and ensure that the database server can continue to efficiently serve the needs of the system.

Optimizing database server for better performance:

Continuing to add more Core servers to this model may result in increased strain on the shared database. As such, it is crucial to ensure that the database is optimized to handle the additional load effectively. Users can consider adjusting default database settings of PostgresSQL to optimize for more data load if needed. Here is where we can adjust PostgresSQL database settings: <PostgreSQL install location\version>\data\postgresql.conf.

Besides that, MD Core also supports a parameter (db_connection) for users to specify max connections that MD Core can handle, take a look at this guideline: MetaDefender Configuration.

Shared DB - Linux

Disclaimer

These results should be viewed as guidelines and not performance guarantees, since there are many variables that affect performance (file set, network configurations, hardware characteristics, etc.). If throughput is important to your implementation, OPSWAT recommends site-specific benchmarking before implementing a production solution.

Environment

Using AWS environment with the specification below:

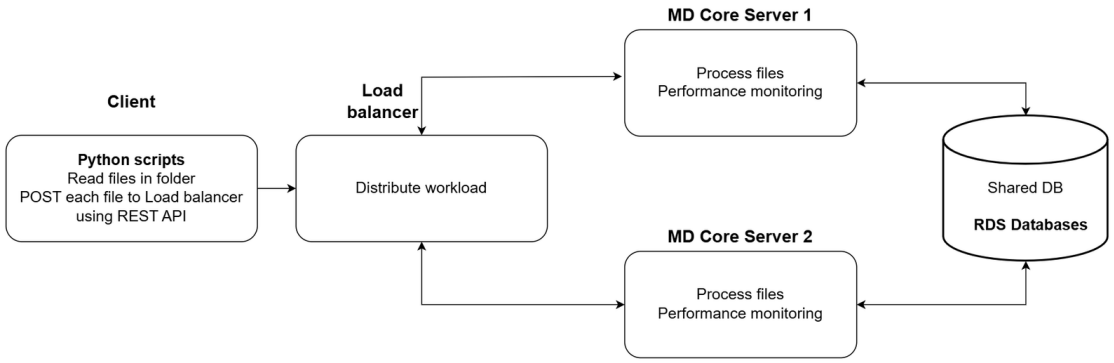
MetaDefender Core

	OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
MetaDefender Core #1	Ubuntu 24.04	c5a.8xlarge	32	64	Up to 10	SSD
MetaDefender Core #2	Ubuntu 24.04	c5a.8xlarge	32	64	Up to 10	SSD

RDS

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
Windows Server 2022	db.m7i.4xlarge	16	64	Up to 10	SSD

Deployment Model



Using a AWS Load Balancer to distribute files sent from the client tool to two (2) different MetaDefender Core servers applying Round Robin algorithm. With this algorithm, each MetaDefender Core server is supposed to receive same number of requests.

Client tool

A simple tool written in Python to collect files in a designated folder and submit requests to Load Balancer mentioned above.

Code · python

```
files_to_scan = list of files to scan
scan_futures =[]
for file_path in files_to_scan:
    scan_futures.append(asyncio.create_task(self.scan_file(file_path, self.load_balancer_url))

async def scan_file(self, file_path, core_url):
    api_url = f'{core_url}/file'

    with open(file_path, 'rb') as file:
        file_content = file.read()

    headers = {'Content-Type': 'application/octet-stream', 'filename': file_path}

    max_retries = 20  # Maximum number of retry attempts
    retry_delay = 120  # Delay in seconds before retrying

    for attempt in range(max_retries + 1):
        try:
            starttime = time.time()
            response = requests.post(api_url, data=file_content, headers=headers)
            endtime = time.time()
            status_code = response.status_code
            if status_code == 200:
                self.post_wait = self.post_wait + endtime - starttime
                self.num_post_req += 1
                response_json = response.json()
                data_id = response_json.get('data_id')
                return status_code, response_json, data_id

            if status_code == 503:
                print(f"Received status code {status_code}. Retrying in {retry_delay}
seconds...")
                await asyncio.sleep(retry_delay)
            else:
                return None, None, None

        except requests.RequestException as e:
            return None, None, None

    return None, None, None
```

OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type
CentOS 7	c5.4xlarge	16	32	Up to 10	SSD

Dataset

Detailed information of dataset below will be used for testing:

File category	File type	Number of files	Total size (MB)	Average file size (MB)
Adobe	PDF	370	385 MB	1.0 MB
Executable	EXE	45	309.5 MB	6.9 MB
	MSI	15	45.75 MB	3.1 MB
Image	BMP	80	515 MB	6.4 MB
	JPG	420	237.5 MB	0.6 MB
	PNG	345	169 MB	0.5 MB
Media	MP3	135	865 MB	6.4 MB
	MP4	50	500 MB	10.0 MB
Office	DOCX	235	190 MB	0.8 MB
	DOC	225	486 MB	2.2 MB
	PPTX	365	860 MB	2.4 MB
	PPT	355	1950 MB	5.5 MB
	XLSX	340	283.5 MB	0.8 MB
	XLS	335	284.5 MB	0.8 MB
Text	CSV	100	236 MB	2.4 MB

File category	File type	Number of files	Total size (MB)	Average file size (MB)
	HTML	1075	76 MB	0.1 MB
	TXT	500	210 MB	0.4 MB
Archive	ZIP	Compressed files: 10 Extracted files: 270	Compressed size: 125.5 MB Extracted size: 156.5 MB	Avg compressed size: 12.6 MB Avg extracted size: 0.6 MB
Summary (compressed)		5000	7728.5 MB	1.55 MB average file size
Summary (extracted)		5260	7759.5 MB	1.48 MB average file size

Product Information

Product versions:

- MetaDefender Core version 5.23.0
- Engines:
  - Metascan 5: Ahnlab, Bitdefender, ClamAV, ESET, K7
  - Metascan 10: Metascan 5 + Avira, Varist, IKARUS, Quick Heal, TACHYON
  - Metascan 20: Metascan 10 + Lionic, CMC, CrowdStrike Falcon ML, Aurora, Trellix, Xcitium, RocketCyber, Sophos, Webroot SMD, Xvirus Anti-Malware
  - Deep CDR: 8.1.1
  - Proactive DLP: 3.4.2
  - Archive: 8.1.1
  - File type analysis: 8.1.1
  - File-based vulnerability assessment: 4.3.0.0
  - Adaptive Sandbox: 3.3.0

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update
- Max file size: 99999999

Archive extraction settings

- Max recursion level: 99999999
- Max number of extracted files: 99999999
- Max total size of extracted files: 99999999
- Timeout: 10 minutes
- Handle archive extraction task as Failed: true
  - Extracted partially: true

Metascan AV settings

- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Adaptive Sandbox

- Max file size: 18446744073709 MB (max)
- Timeout: 86400 (max)

Performance test results

MetaDefender Core with single engine (technology)

Summary metrics:

System resource utilization:

MetaDefender Core with common engine packages

Summary metrics:

System resource utilization:

Recommendations

Controlling total processing time of each MD Core server:

In this deployment model, we should organize and send files in the way that it best utilizes the load of each MD Core server. It is not a good practice if one Core server is free while the other one is busy. By optimizing the distribution of files, we can ensure that each Core server is utilized efficiently, thereby improving overall system performance. Furthermore, this approach can help prevent bottlenecks and minimize the chances of system overload.

Adding proper number of MD Core servers to the cluster:

Adding more Core servers to this model will increase more load on the shared database. When adding a new MD Core server, users should monitor performance of database server such as memory/CPU consumption, disk usage, network bandwidth, request response time and so on… to see if it still can handle the load. This is important in order to maintain optimal performance and ensure that the database server can continue to efficiently serve the needs of the system.

Optimizing database server for better performance:

Continuing to add more Core servers to this model may result in increased strain on the shared database. As such, it is crucial to ensure that the database is optimized to handle the additional load effectively. Users can consider adjusting default database settings of PostgresSQL to optimize for more data load if needed. Here is where we can adjust PostgresSQL database settings: <PostgreSQL install location\version>\data\postgresql.conf.

Besides that, MD Core also supports a parameter (db_connection) for users to specify max connections that MD Core can handle, take a look at this guideline: MetaDefender Configuration.

MetaDefender Core Documentation

The users can consult this web page or, alternatively, they can download the manual in pdf format from the link below:

MetaDefender Core manual

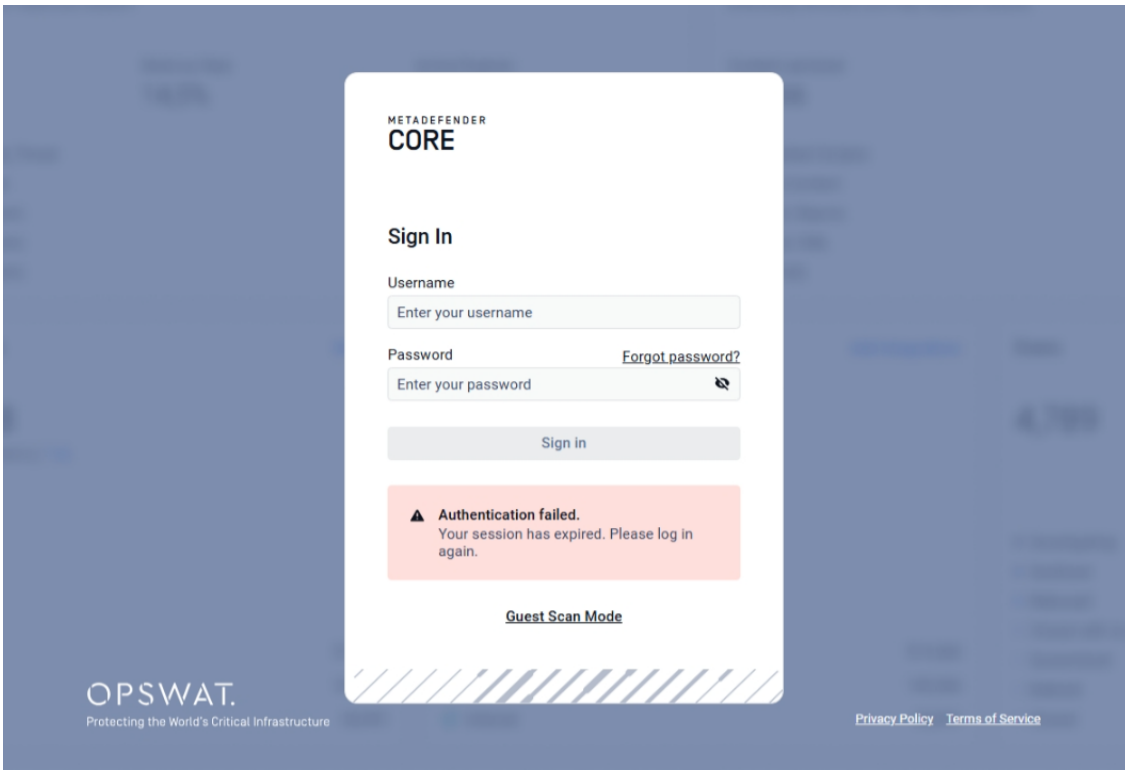
- Download link
- SHA256: F7631B4B6992B1E9C2F774BB4FED5D4D61831FB69159AAC1C2780940BBD62704

OPSWAT MetaDefender AGD Documentation_v1.9

- Download link
- SHA256: EEDDE2A51E95779986204CBB764878B339D2C3E15952D50A15FC8FF60BAD855C

Management Console - Login

The management console is available at: http://<MetaDefender Core Server>:<port> where <MetaDefender Core Server> is the name or IP address of the system where MetaDefender Core is installed. For example: http://localhost:8008



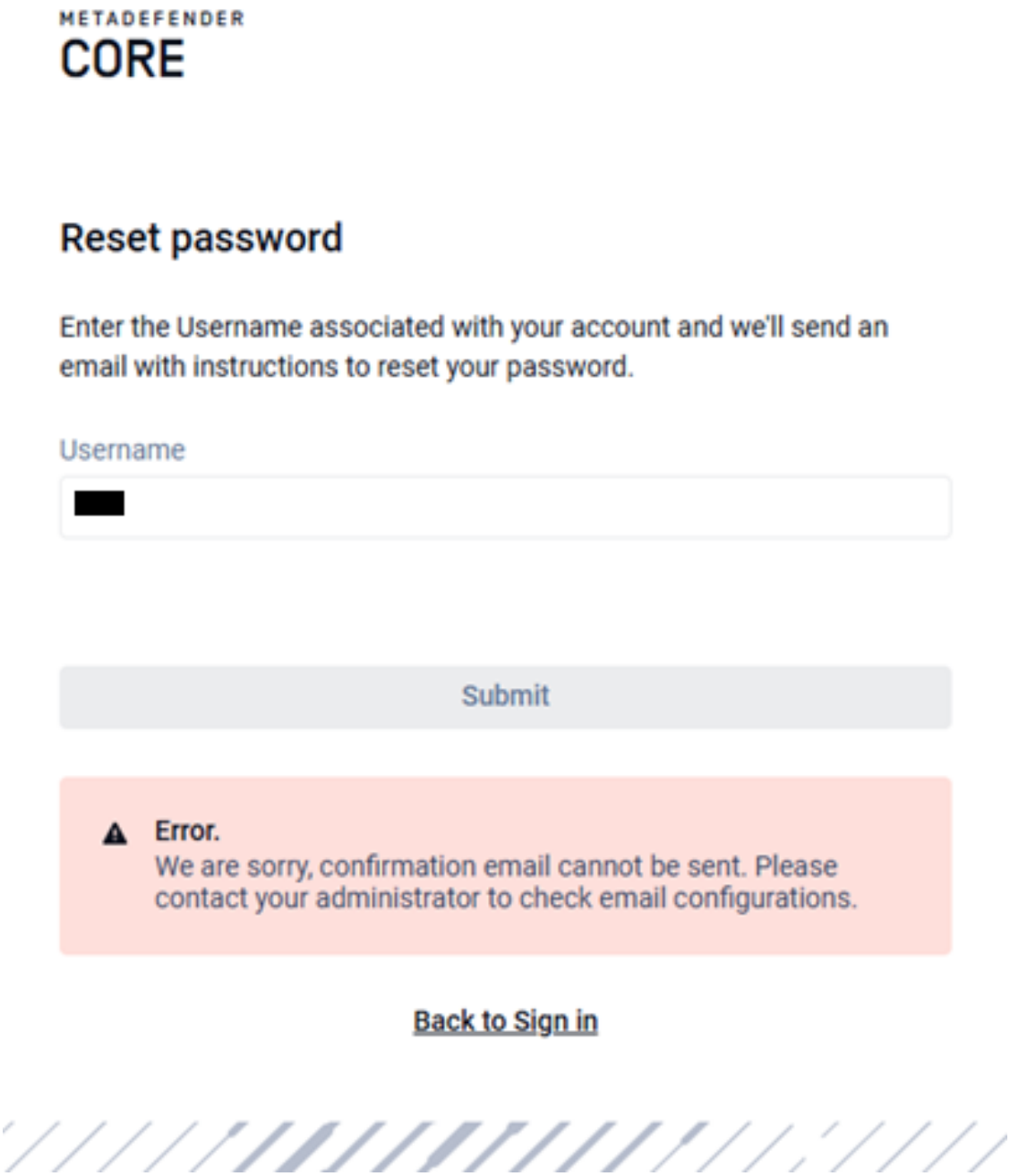
The "Forgot Password" feature requires authentication via email, and force the affected user to change their password at the first login time for security reason. More details at Password Recovery.

Therefore, it is mandatory for MetaDefender Core's administrators to follow steps at Email Configuration and ensure all SMTP configurations set properly beforehand.

Password Recovery

Prerequisites

- Password Recovery **only supports local users**, not applicable to Active Directory / LDAP users where all their credential settings should be managed on Domain controller / LDAP server.
- The feature authenticates local users via email to reset their own password (assuming user was registered with a valid email on MetaDefender Core, if not then user should update their email properly on User Management page), and therefore it is mandatory for MetaDefender Core's administrators to follow steps at Email Configuration and ensure all SMTP configurations set properly beforehand.
  - Otherwise, expecting users to hit following warning message when trying to reset their password: `"We are sorry, confirmation email cannot be sent. Please contact your administrator to check email configurations."`



How This Feature Works

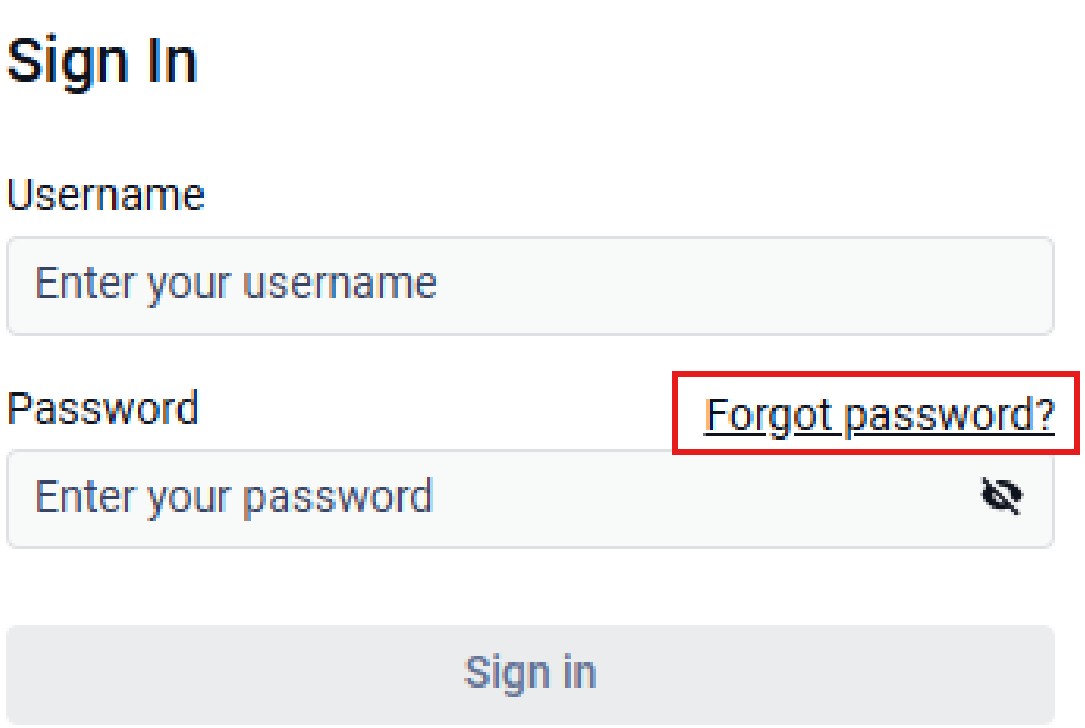
Just in case MetaDefender Core user credentials are lost or forgotten, basically any local user (not AD / LDAP) will be supported to reset their password by either one of two methods:

- **Forgot password (active):** Any local user can choose to reset their own password.
- **Reset password by administrators (passive):** Any local user's password can be reset by administrators.

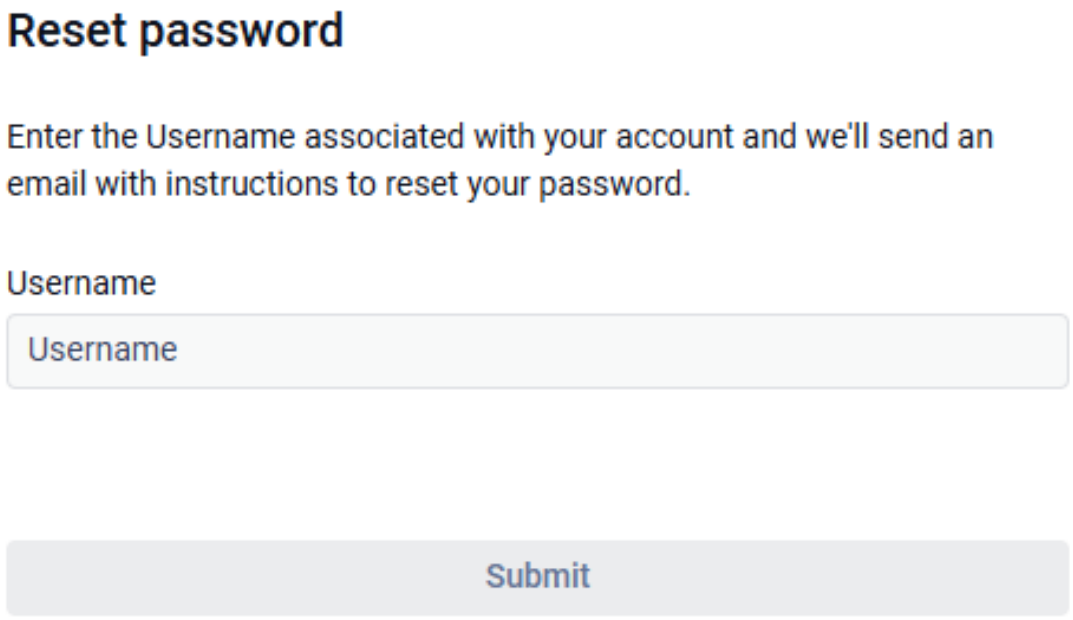
Both methods require authentication via email, and force the affected user to change their password at the first login time for security reason.

Forgot Password

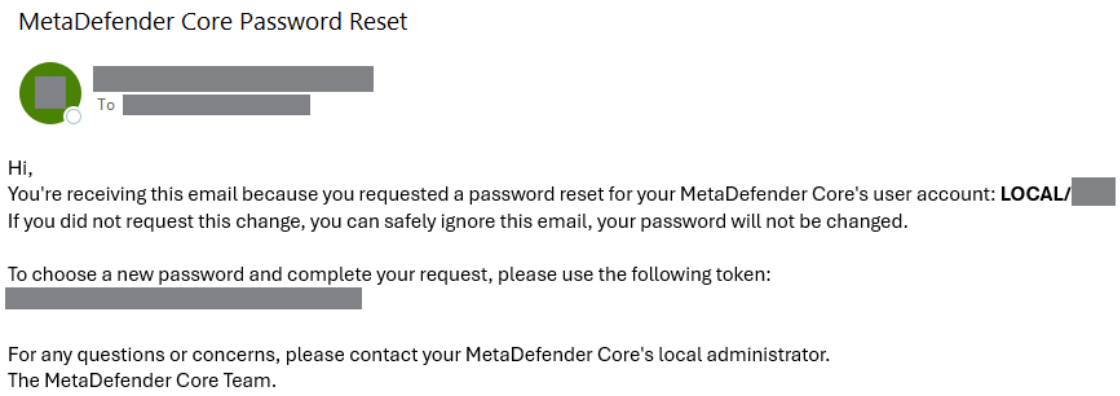
Any local user registered with email on MetaDefender Core should be able to reset their own password by clicking on **Forgot password?** link on login page.



You will be redirected to Recover Password page.



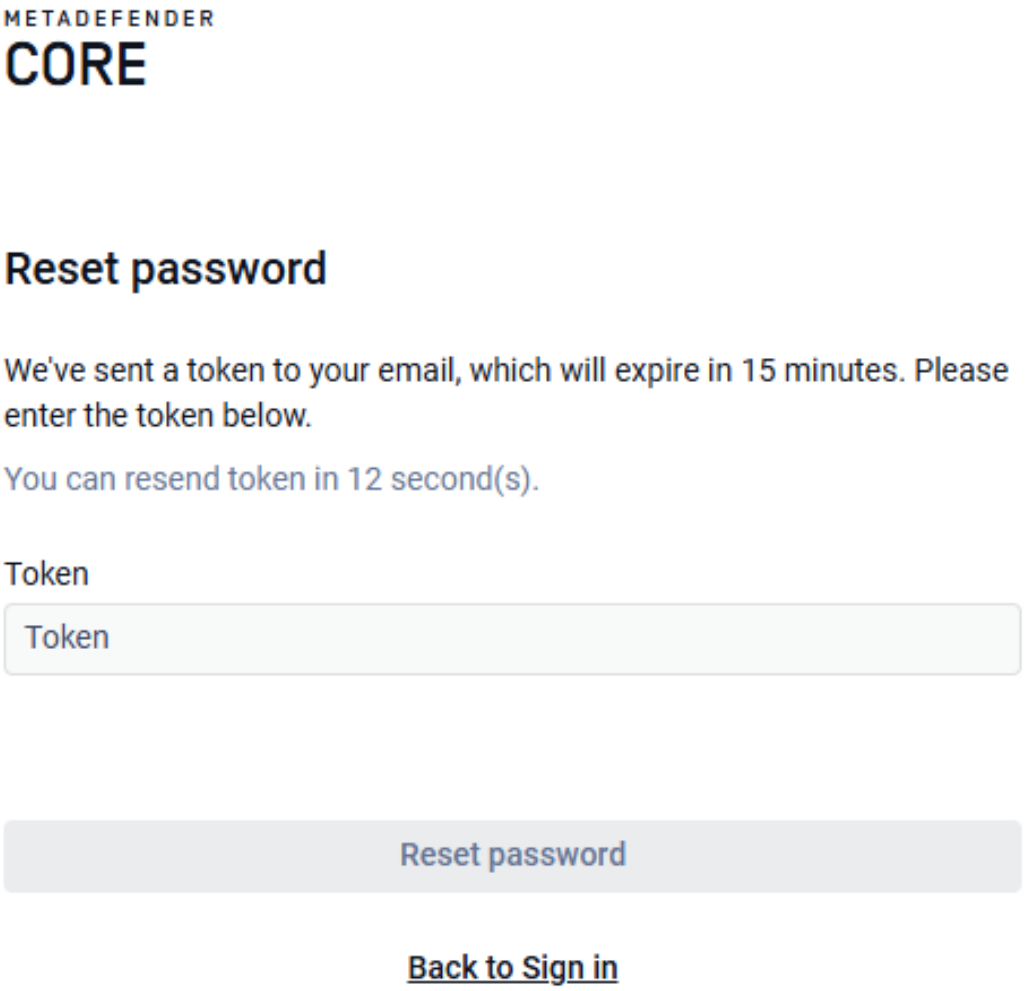
An email with a token will be sent to user's registered email entitled `MetaDefender Core Password Reset`



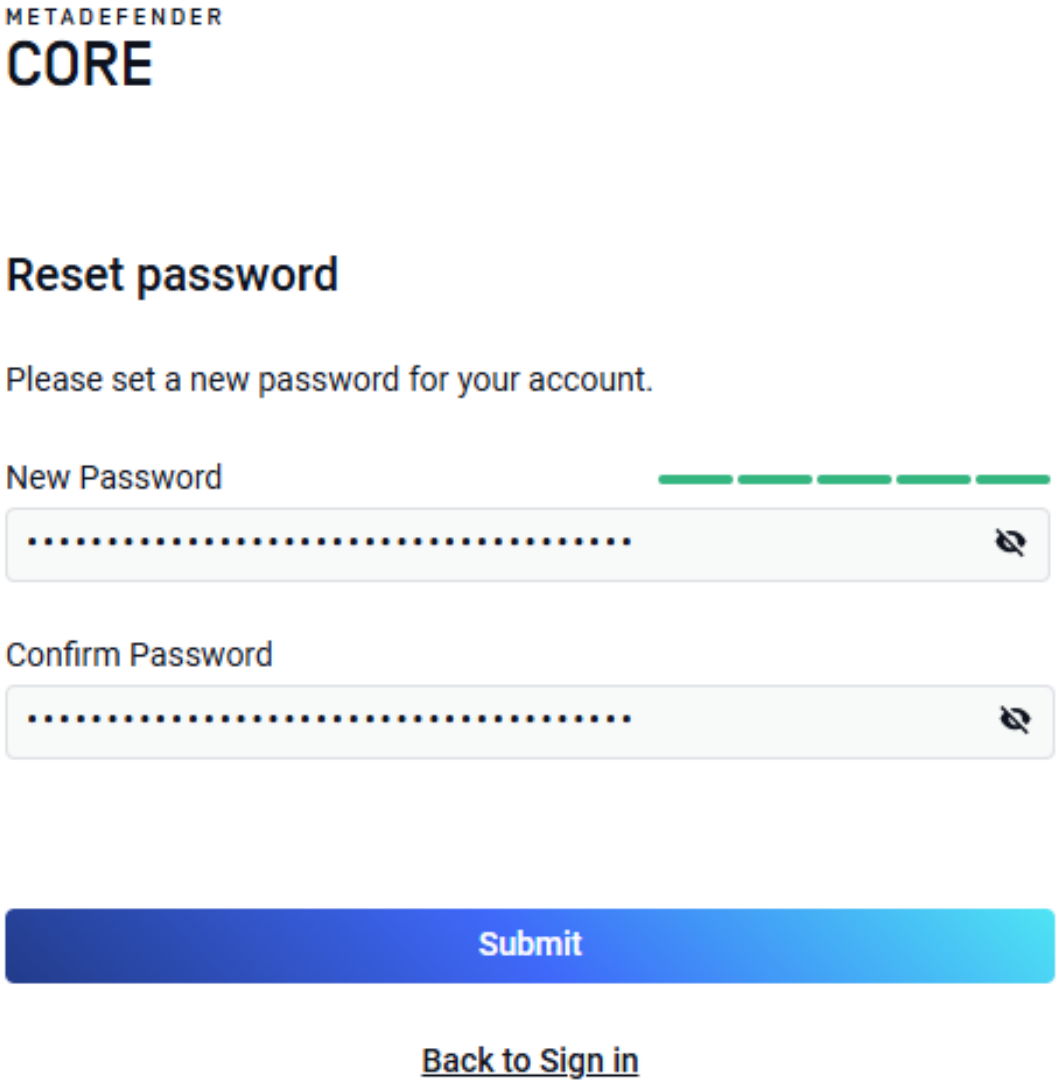
Warning

If that user does not take any action, the token will be expired in 15 minutes

Input the token from the mail and press `Reset password` in MetaDefender Core management console



Following steps in email will be redirected to MetaDefender Core management console where user will be allowed to create a new password:

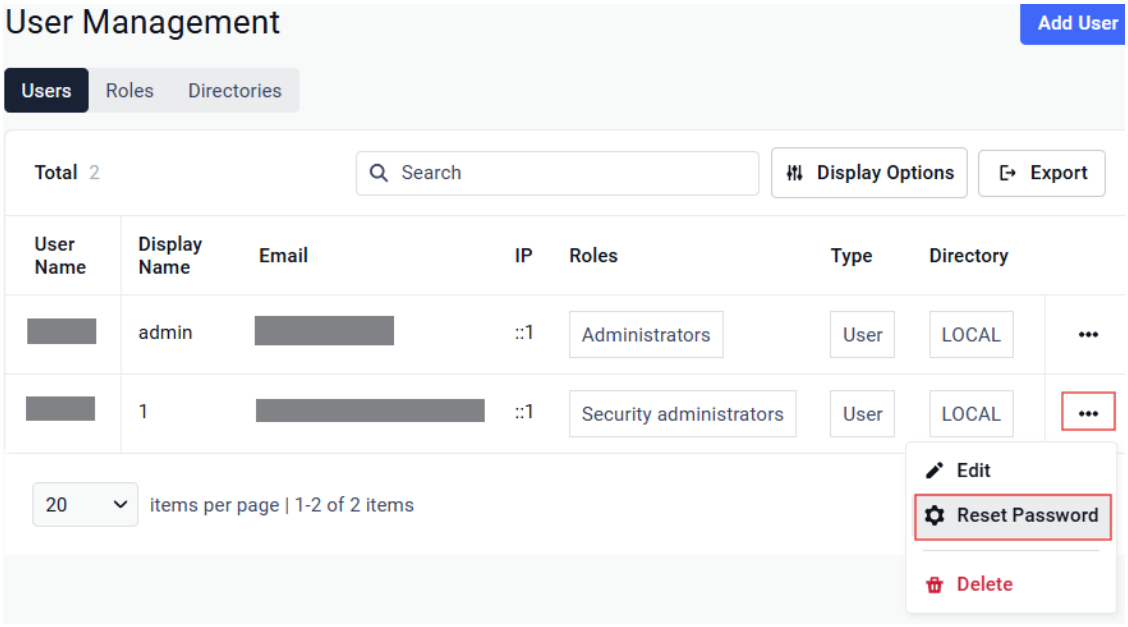


You will be automatically redirected to MetaDefender Core dashboard after resetting password successfully.

Reset Password By Administrators

As an administrator, you are allowed to reset password of any local user on MetaDefender Core, it could be either other local user (admin / non-admin) or even oneself.

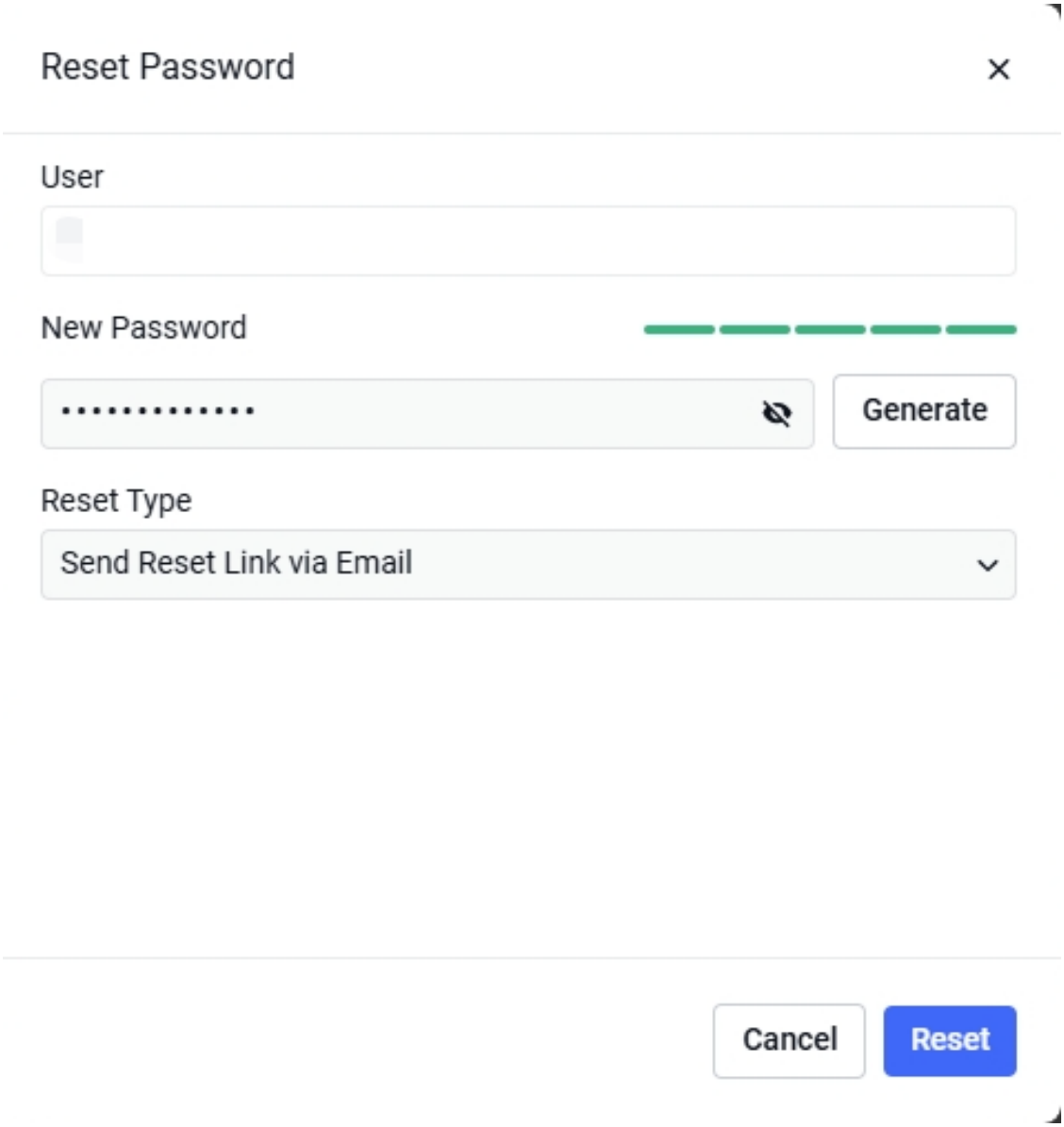
Go to "User Management" under "Users And Groups", mouse over on user to choose "Reset Password" option



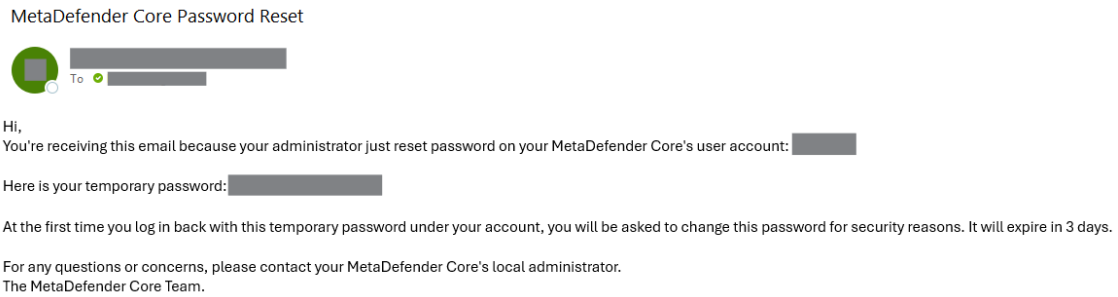
Admin to RESET PASSWORD

Then administrator must create a new password by clicking **Generate** link or typing any text in the textbox.

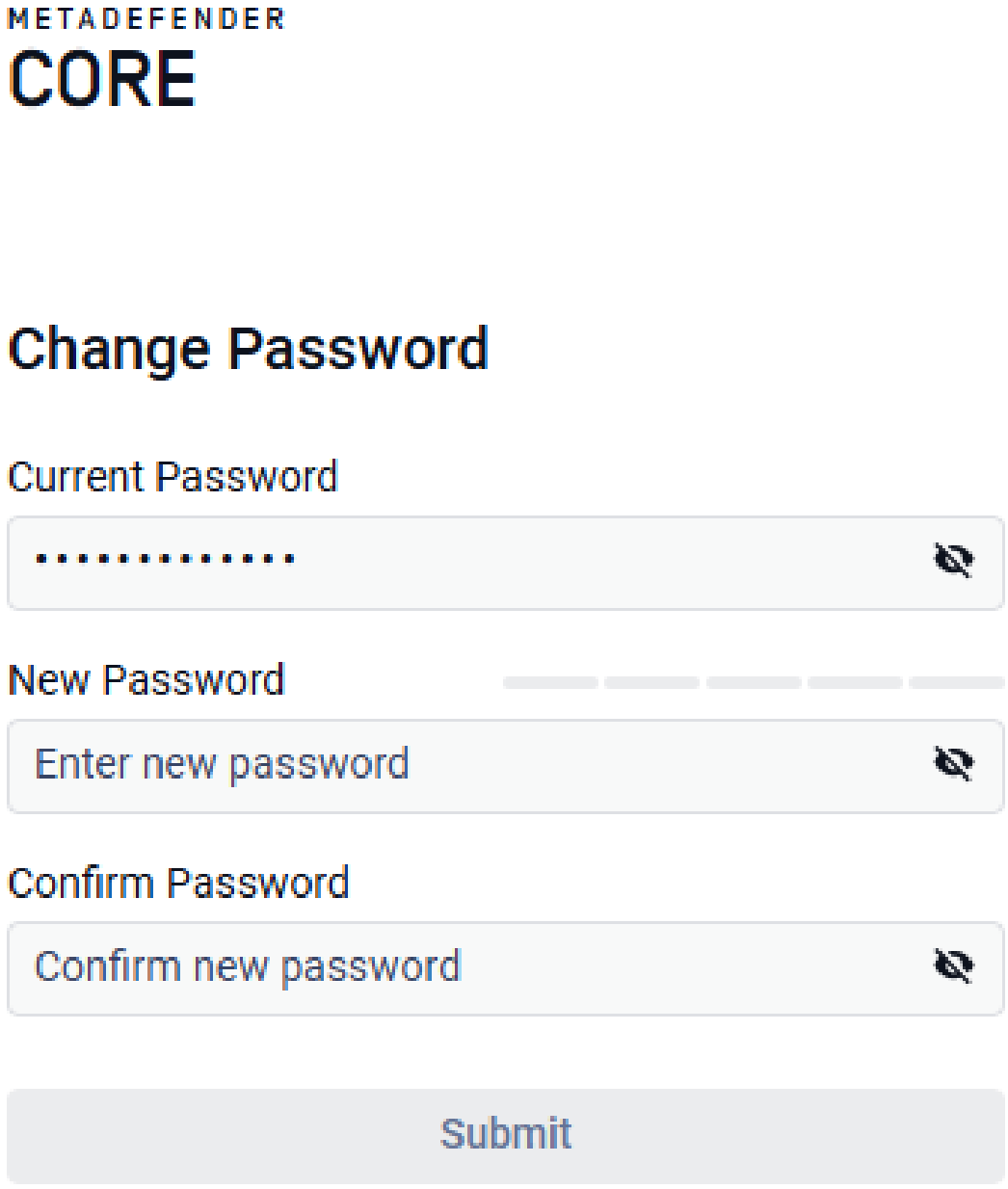
Click RESET PASSWORD button once done.



An email titled "MetaDefender Core Password Reset" will be sent to user's email. Please check the inbox to collect a temporary password.



When the impacted user tries to log into MetaDefender management console afterwards with the provided temporary password in the email at the first time, that user will be redirected to page where they are required to change their password.



Once successfully changed the password, the user will be automatically redirected to the MetaDefender dashboard.

MetaDefender Configuration

Linux

The configuration file for the server is located in `/etc/ometascan/ometascan.conf`

> **Notice**
> After modifying the server configuration file you must restart the Metadefender Core service in order for the

changes to take effect. You should use the distribution-standard way to restart the service.

[global] section

parameter	required	description
`restaddress`	required	IP address of the computer that runs MetaDefender Core to serve REST API and web user interface (* means listening from all interfaces including IP version 4 and 6). Just in case IP version 6 is not enabled on the system, then changing it to `0.0.0.0` to limit to IP version 4 only. Default value: *
`restport`	required	Designated port number for the web management console and REST interface. Default value: 8008
`report_engine_issue`	optional	Enable reporting of engine issue count. (possible values: "true" or "false") Default value: true
`dlppath`	optional	Directory for DLP-processed database and items. Default value: [Data directory]/dlp
`quarantinepath`	optional	Directory for quarantine database and quarantined items. Default value: [Data directory]/quarantine
`sanitizepath`	optional	Directory for sanitized database and sanitized items. Default value: [Data directory]/sanitized

parameter	required	description
`archiveforensicpath`	optional	Directory for storing captured archive forensic files. Default value: [Data directory]/archive_forensic
`dbmode`	optional	Support database mode, possible values: • 0: Not used for now, reserved • 1: Standalone Core (default) • 2: Core instance in Cluster deployment • 3: Non-persistent mode (Core will not write any scan result into database, client must use webhook scanning fashion to retrieve scan result) • 4: Shared database model (all Core instances will share the same database) After changed, a Core service restart is required to take effect. Only available starting MetaDefender Core 4.19.2
`tempdirectory`	optional	Full path of a directory to use for storing temporary files rather than using their default directories: `/var/tmp/` and `/tmp` Users need to prepare this directory in advance. MetaDefender Core creates a subfolder called `ometascan`and `ometascan/resources` in the directory. Default: `/var/tmp/ometascan/resources/`
`tempdirectory_create_timeout`	optional	Maximum time allowed for

parameter	required	description
		MetaDefender Core to access device and create temporary folder on it.
`aigcdpath`	optional	Directory for output images stored by OPSWAT AI Content Inspector engine to display file processing results. Default value: [Data directory]/aigcd

[logger] section

Info

From version 5.15.1, MetaDefender Core (MD Core) allows configuring log file and syslog settings via the MD Core UI or API endpoints (Advanced Log Configurations). Therefore, the configurations here will be overridden if they conflict with the configurations for the log file or syslog set through the MD Core UI or API endpoints.

We highly recommend configure logger via MetaDefender Core UI.

key	required	description
`logfile`	optional	Full path of a logfile to write log messages to. Default value: /var/log/ometascan/ometascan.log
`loglevel`	optional	Level of logging. Supported values are: debug, info, warning, error. Default value: info
`log_rotation`	optional	Should only set this key when **logfile** key is also set accordingly. Supported values: • 0: All logs are not rotated, except for NGINX log. • 1 (default mode), enable to rotate log:

key	required	description
		• Rotation process will be performed every day or when file size reaches 1GB. • Limit rotated log to be stored is 30 files, the oldest log will be deleted if file number reaches the limit. • Rotated log name format: <logname>-<yyyyMMdd>.gz (e.g.: core.log-20200330.gz), all saved in same location with what you set in **logfile.** • All generated log packages included in MetaDefender Core support package.
`syslog`	optional	Switch on logging to a local ('local') or remote ('protocol://<hostname>:<port>') syslog server (Multiple server can be specified separated with comma). More details: HERE
`syslog_level`	optional	Level of logging. Supported values are: debug, info, warning, error
`local_timezone`	optional	Set local timezone for events sending to local syslog server. Default value: false
`override`	optional	Override specific log ids to display them on another level e.g.: "1723:error,663:info" If there is no level set for an id, it will be displayed on every occasion. e.g.: "1723,663:info" means id 1723 dump message will be displayed every time and id 663 warning message is reduced to info level.

key	required	description
`cef`	optional	If true, the log format is Common Event Format. Default value: false
`nginx_logfile`	optional	File name and path to store the NGINX logs. If this value is changed, the /etc/logrotate.d/ometascan should be changed accordingly. Default value: /var/log/ometascan/nginx-ometascan.log

Notice

Setting both `syslog` and `syslog_level`, or none of them.

Notice

Setting both `logfile` and `loglevel`, or none of them.

[internal] section

key	required	description
`db_connection`	optional	Define maximum number of concurrent connections allows MetaDefender Core to open to work with PostgreSQL database server. Only available starting MetaDefender Core 4.19.1. Default value: 10
`data_directory`	optional	Full path for MD Core's data (database, updates etc.) E.g. /var/lib/ometascan/test Default value: /var/lib/ometascan
`skip_upgrade_scan_data`	optional	When enabled (set to 1), upgrading MetaDefender Core will auto skip

key	required	description
		migrating history processing data which is usually big in size (only migrate configurations and audit history). This setting is to save upgrade time when users do not need to migrate entire scan data. Default value: 0
`parallelcount`	optional	Set maximum number of threads (files) sending to engine at the same time, applicable to all engines. Default value: 20 Exception: • Archive engine (extraction): default = -1 (unlimited) • Archive engine (compression): default = 20 • Proactive DLP engine: default = 5 • Sandbox engine: default = 5 • SBOM engine: default = 5 • Threat Intelligence engine: default = 5 • OPSWAT AI Content Inspector engine: default =5
`parallelcount_<enginename>`	optional	`<enginename>` is the first part of engine id which all can be found in <MD Core folder>\data\updates\metadescriptor For example: engine id: `symantec_1_windows` → `<enginename>` = `symantec` Some common use-cases: • **ds (parallelcount_ds)**: Deep CDR engine. By default, `parallelcount_ds` = 20

key	required	description

		• **7z (parallelcount_7z)**: Archive engine, applicable to archive extraction only. By default, `parallelcount_7z` = -1 (unlimited threads)
		  • **7z_extract (parallelcount_7z_extract)**: Archive engine, extraction only. By default, `parallelcount_7z_extract` = -1 (unlimited threads)
		  • **7z_compress (parallelcount_7z_compress)** : Archive engine, compression only for archive sanitization. By default, `parallelcount_7z_compress` = 20

`next_extraction_polling_interval`	optional	Fine-tuning this interval between the range of 100-200ms may help stabilize the performance and processing time when dealing with small archive files or office document files under high load. In case this polling interval is set to out of range (invalid number, < 100, or > 1000), the application cannot start, and an exception will log to system event log. Default value: 1000

[ocm] section

key	required	description
`allow_ocm_urls`	optional	The allowlist of enrollment URLs permitted for enrollment in My OPSWAT Central Management

Windows

The configuration for the server is located in **Windows Registry:** `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\`

> **Notice**
> After modifying the server configuration file you must restart the MetaDefender Core service in order for the changes to take effect.

Global

`HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global`

parameter	type	required	description
`restaddress`	string value	required	IP address of the computer that runs MetaDefender Core to serve REST API and web user interface (* means listening from all interfaces including IP version 4 and 6). Just in case IP version 6 is not enabled on the system, then changing it to `0.0.0.0` to limit to IP version 4 only. Default value: *
`restport`	string value	required	Designated port number for the web and REST interface. Default value: 8008
`report_engine_issue`	string value	optional	Enable reporting of engine issue count. (possible values: "true" or "false"). Default value: true
`dlppath`	string value	optional	Directory for DLP-processed database and items.

parameter	type	required	description
			Default value: [installdir]\data\dlp
`quarantinepath`	string value	optional	Directory for quarantine database and quarantined items. Default value: [installdir]\data\quarantine
`sanitizepath`	string value	optional	Directory for sanitized database and sanitized items. Default value: [installdir]\data\sanitized
`archiveforensicpath`	string value	optional	Directory for storing captured archive forensic files. Default value: [installdir]\archive_forensic
`dbmode`	string value	optional	Support database mode, possible values: • 0: Not used for now, reserved • 1: Standalone Core (default) • 2: Core instance in Cluster deployment • 3: Non-persistent mode (Core will not write any scan result into database, client must use webhook scanning fashion to retrieve scan result) • 4: Shared database model (all Core instances will share the same database)

parameter	type	required	description
			After changed, a Core service restart is required to take effect. Only available starting MetaDefender Core 4.19.2
`tempdirectory`	string value	optional	Full path of a directory to use for storing temporary files. Users need to prepare this directory in advance. MetaDefender Core creates a subfolder called `resources` in this folder. Default: `<installation directory>\data\resources`
`tempdirectory_create_timeout`	string value	optional	Maximum time allowed for MetaDefender Core to access device and create temporary folder on it.
`aigcdpath`	string value	optional	Directory for output images stored by OPSWAT AI Content Inspector engine to display file processing results. Default value: [Data directory]/aigcd

Logger

`HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`

Info

From version 5.15.1, MetaDefender Core (MD Core) allows configuring log file and syslog settings via the MD Core UI or API endpoints (Advanced Log Configurations). Therefore, the configurations here will be overridden if they conflict with the configurations for the log file or syslog set through the MD Core UI or API endpoints.

Event Viewer logging can only be configured in the registry.

We highly recommend configure logger via MetaDefender Core UI.

key	type	required	description
`logfile`	string value	optional	Full path of a l messages to. Default value: Files\OPSWAT Core\data\logs
`loglevel`	string value	optional	Level of loggin values are: deb error. Must set value **logfile** key is al accordingly. Default value: i

`log_rotation`	string value	optional	This setting is on Windows or use built-in OS Should only se **logfile** key is al accordingly. Su
			• 0: Core logs
			• 1 (default m rotate log:
			  • Rotation p performed when file 1GB.
			  • Limit rota stored is 3 oldest log file numbe limit.
			  • Rotated lo <logname <yyyyMM core.log-2 saved in s

key	type	required	description
			with what **logfile.** • All genera included i Core supp
`wineventlog_level`	string value	optional	Level of event Supported valu info, warning, e Default value: i
`syslog`	string value	optional	Value can only 'protocol:// <hostname>:<p server can be s separated with details: HERE
`syslog_level`	string value	optional	Level of loggin values are: deb error. Must set when **syslog** ke accordingly.
`local_timezone`	string value	optional	Set local timez sending to loca Default value: f
`override`	string value	optional	Override specif display them o e.g.: "1723:erro If there is no le it will be displa occasion. e.g.: means id 1723 will be displaye id 663 warning reduced to info
`cef`	string value	optional	If true, the log f

key	type	required	description
			Common Even Default value: f
`nginx_logfile`	string value	optional	File name and NGINX logs. Default value: [installdir]\ngir

`nginx_log_rotation`	string value	optional	This setting is
			on Windows or
			use built-in OS
			Should only se
			nginx_logfile k
			accordingly. Su
			• 0: Nginx logs
			• 1 (default), e
			log:
			  • Rotation p
			performed
			regardless
			  • Limit rotat
			stored is 3
			oldest log
			file numbe
			limit.
			  • Rotated lo
			<logname
			<yyyyMMd
			nginxlog.l
			all saved i
			with what
			nginx_log
			  • All genera
			included i
			Core supp

Notice

Setting both `syslog`and `syslog_level`, or none of them.

Notice

Setting both `logfile` and `loglevel`, or none of them.

Internal

`HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\internal`

key	type	required
`db_connection`	string value	optional
`data_directory`	string value	optional
`skip_upgrade_scan_data`	string value	optional
`parallelcount`	string value	optional

key	type	required
`parallelcount_<enginename>`	string value	optional
`next_extraction_polling_interval`	string value	optional

key	type	required

OCM

key	required	description
`allow_ocm_urls`	optional	The allowlist of enrollment URLs permitted for enrollment in My OPSWAT Central Management

What happens when you change data_directory?

- If you update the `data_directory` registry value to a new location and restart MD Core, the application will create a **new, fresh data folder** at the specified path.
- **A new database and empty folders** for Quarantine and Sanitize, etc will be created in the new location. The UI will show the Welcome screen, and you must go through the initial setup process as if it's a fresh installation.
- **Existing data (credentials, processing history, configuration, etc.) from the old data folder will not be available** in the new location unless you manually migrate the data.

In case, you want to completely move all the installation folder and files to another location (drive), please check this documentation: How to relocate installation folder in MetaDefender Core on Windows?

Policy Configuration

The MetaDefender Core server can be configured to use different scanning profiles for different clients. The selection is based on the client's source IP address.

In case multiple scanning profiles are configured for the given client, the client can choose which one to use. If a client does not have a scanning profile specified, MetaDefender Core uses the first matching profile from the Workflow rules.

All configuration options related to the policies are found under the **Policy** menu.

How Policies Work

A policy is pairing a *user* with a *workflow template* based on a workflow *rule*.

Users can be grouped into *zones* based on their network address.

Workflow templates can be created/modified to change how file scanning is carried out.

Creating a policy means creating a rule, where a source zone will be paired with a workflow template.

How a file scan is processed via the REST API

When MetaDefender Core receives a scan request through the REST API it will match the source address through the zones in the list of rules and apply the first matching rule's workflow. The processing request then will then be processed based on this specific workflow.

If a workflow is provided by the REST request it still should be one which has a matching rule. Otherwise the scan request will fail.

How a file scan is processed on the web UI

When MetaDefender Core receives a scan request through the web UI it will match the source address through the list of rules. The user will be able to select only those workflows with a matching rule. This scan request then will then be processed based on the workflow selected by the user.

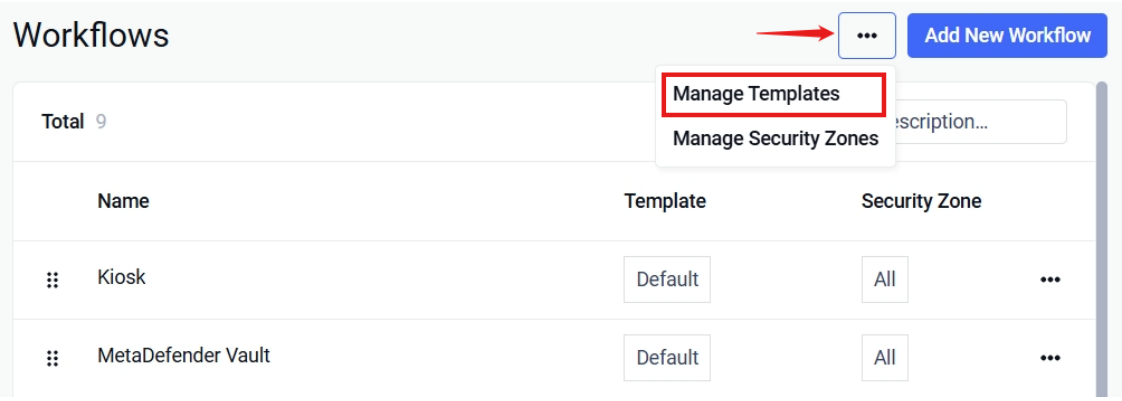
Workflow Configuration Template

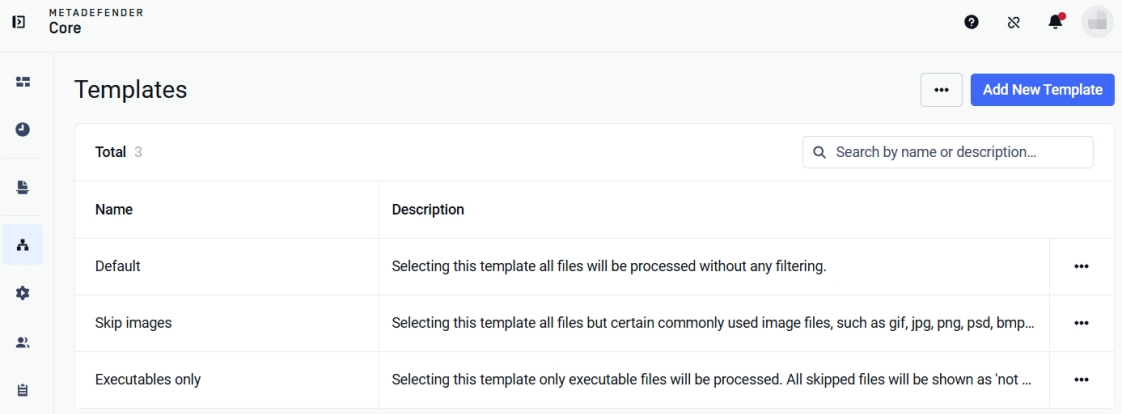
The Workflow templates page is found under **Workflows > Manage Templates** after successful login.

These workflow templates define the processing methods that can be used by the rules.

MetaDefender Core comes with predefined workflow templates that cannot be modified, however they can be copied and the created workflow templates are fully customizable.

> **NOTE: These predefined workflow templates cannot be modified or removed**
>
> It is highly recommended to use less workflow template and rather more rules based on the workflow templates.





Workflow templates

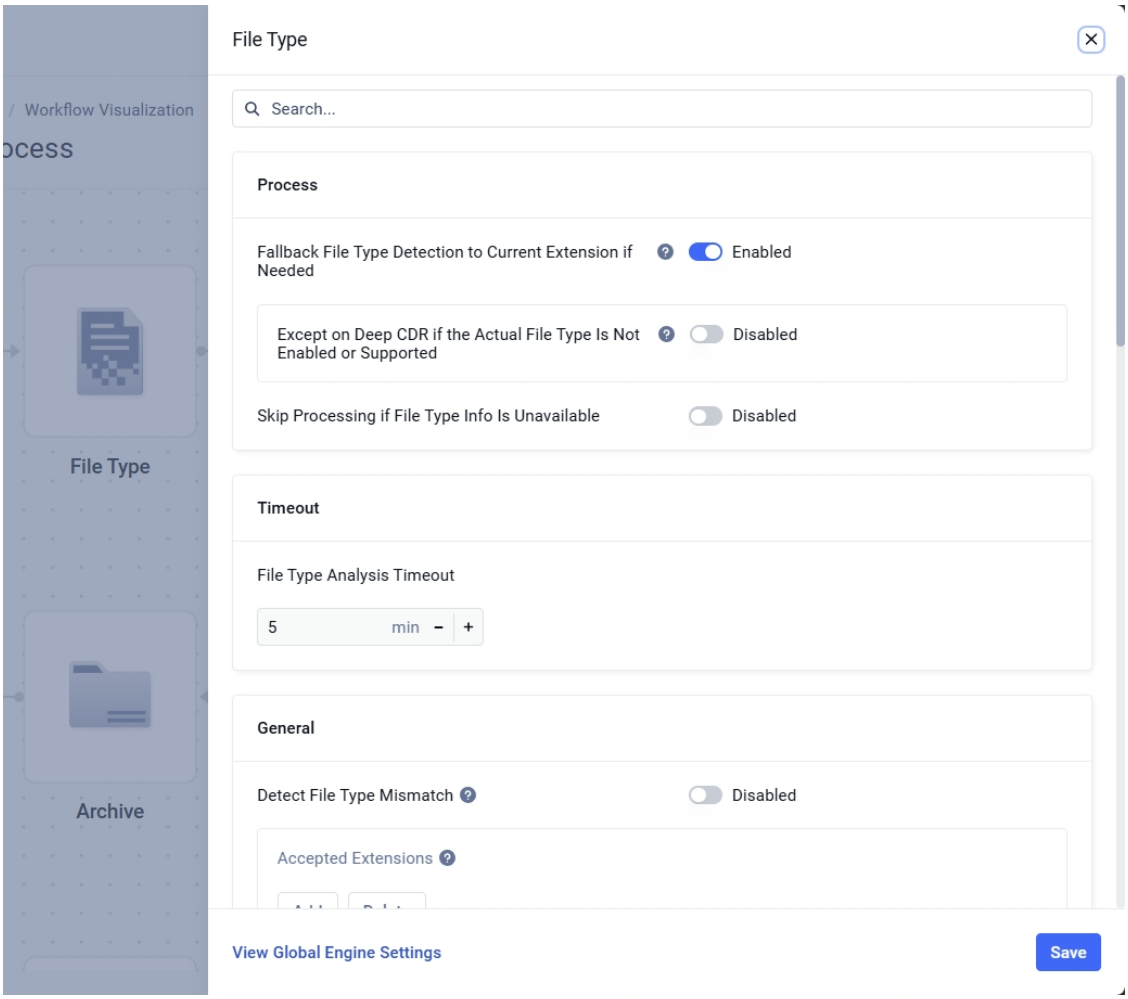
When clicking on a workflow template a window pops up showing different tabs related to the workflow templates different kind of properties.

File Type

For all information about features powered by File Type, please learn more at File Type Verification Engine.

Detect file type mismatch can be enabled to identify files where the file extension differs from the actual file type.

Other configurations of File Type engine can be found at File type configurations.

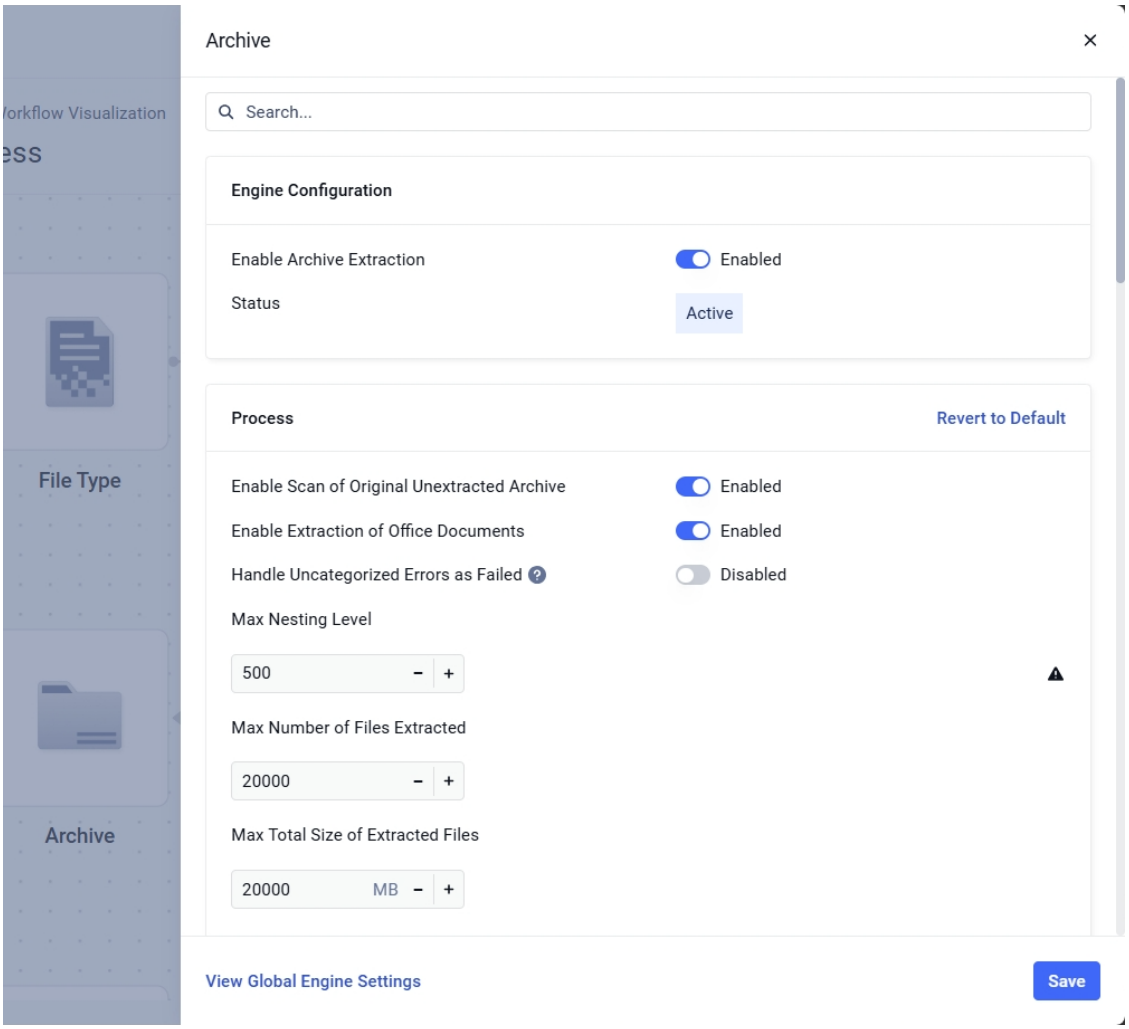


Archive

On the Archive tab the archive handling can be enabled or disabled as well as other parameters can be set.

The max recursion level defines how deep extraction should go into the archive, the number of maximum extracted files also can be set as well as the overall maximum size of these files.

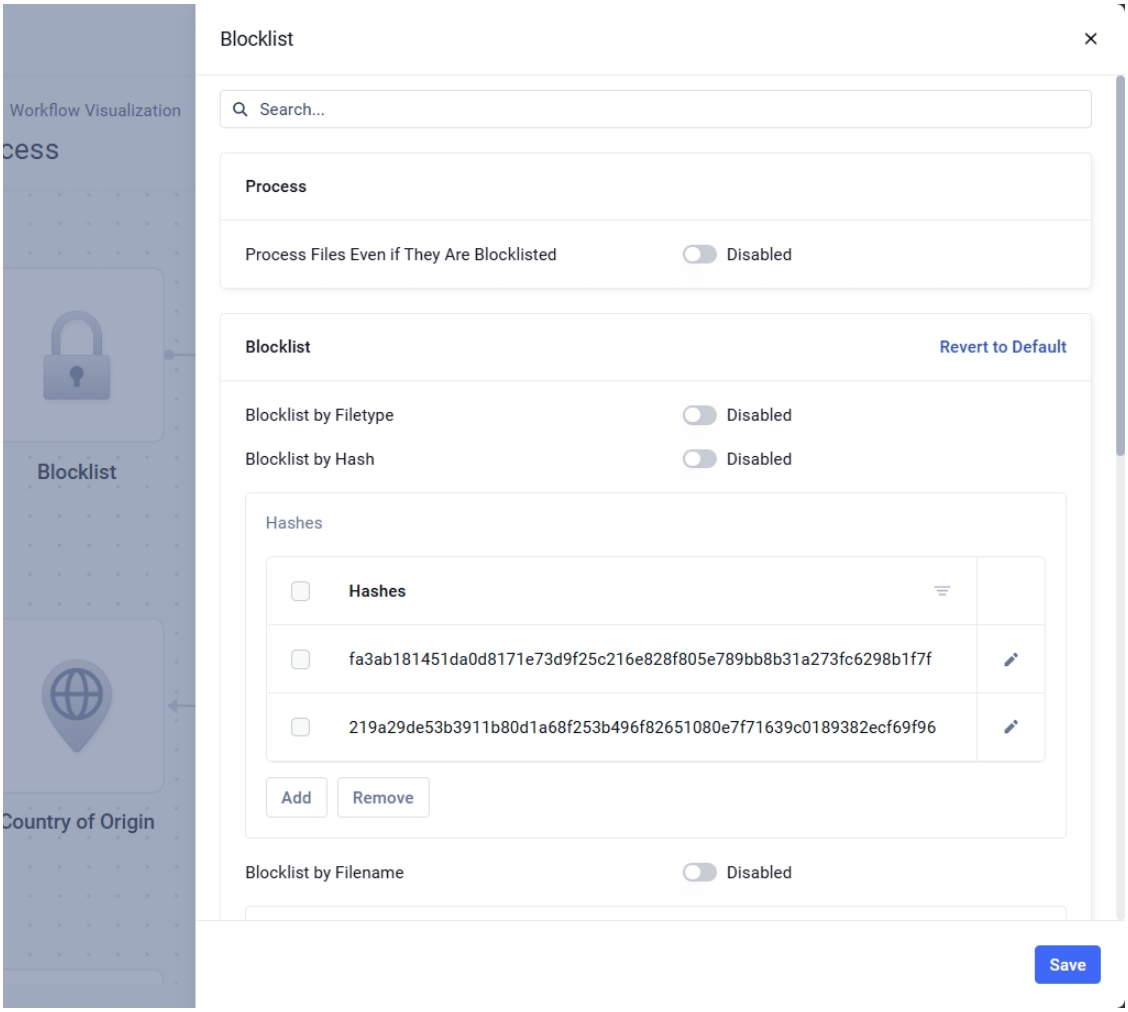
It is also possible to disable scanning the archive itself, and a timeout for the whole process can be set as well.



Blocklist

Info

Since MetaDefender Core 4.19.0, user is allowed to configure to exclude child extracted files from archive file type for being blocklisted.



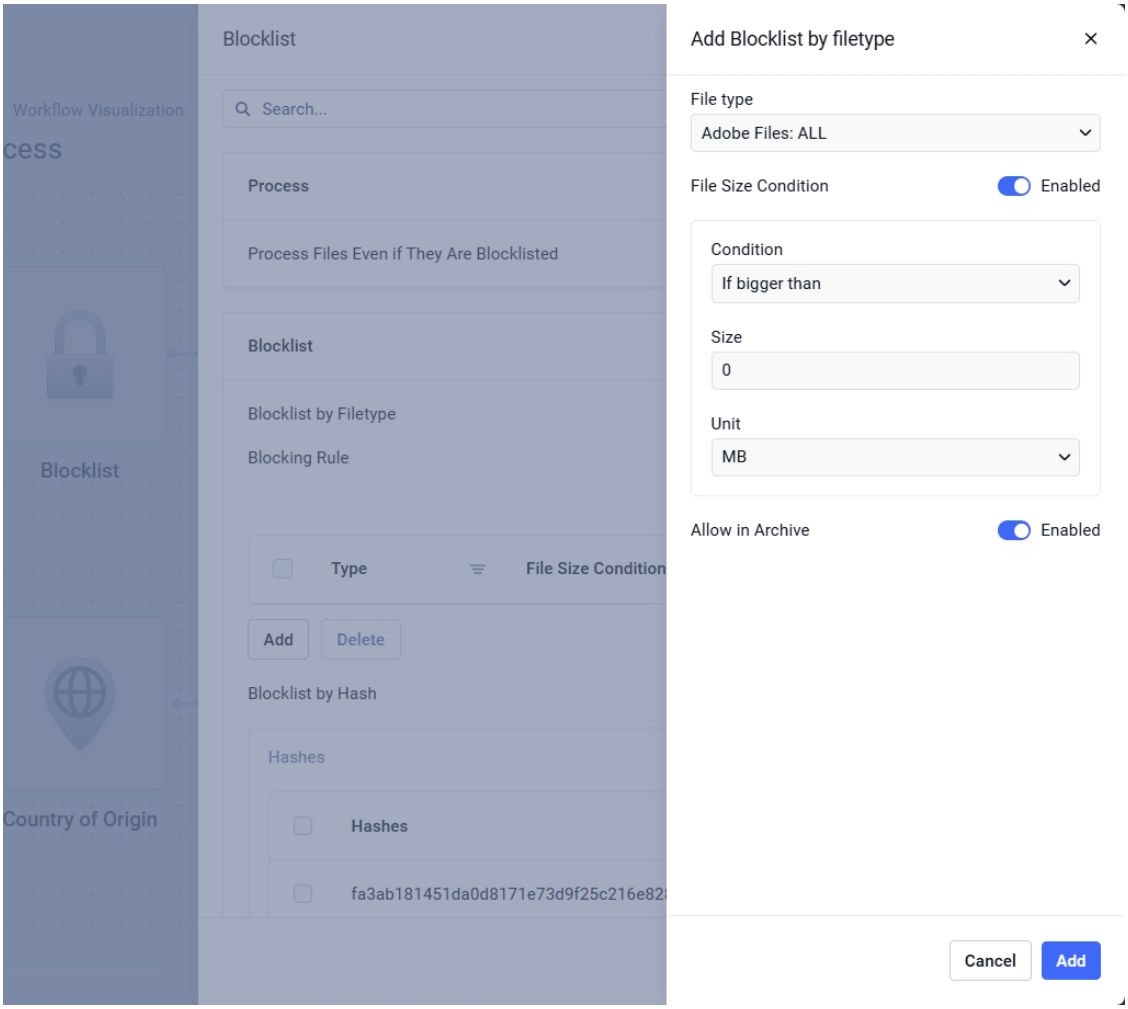
During scan, it is possible to create blocklists where files depending on their file type, checksum (hash), filename, extension, file size or MIME-TYPE. All of these can be stored in the fields on the Blocklist tab. Administrators can select tactic "Block All" or "Block All Except". Also it is available to blocklist all the files coming from the same group, such as Executable Files, Media Files and others.

When filtering by mime-type or filename, the filter is handled as a regular expression.

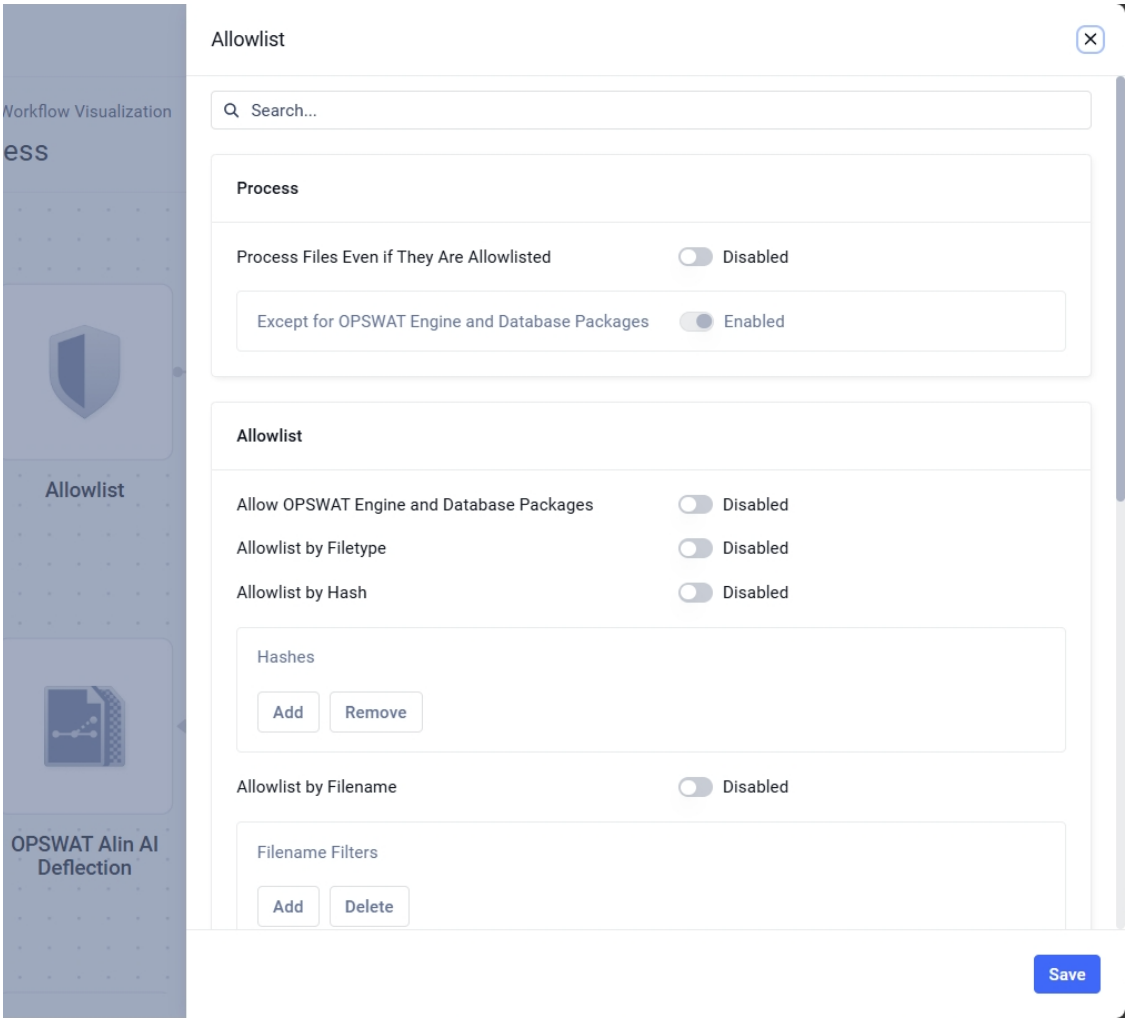
Exceptions can be defined in **Exceptions (by mime-type)** section using regular exceptions. For instance, if all office files have to be blocked except docx files, then **Office documents** group should be chosen and `^application/vnd.openxmlformats-officedocument.wordprocessingml.document$` expression should be given as exception.

Since version 5.18.0, MetaDefender Core has introduced a refreshed and modernized interface for managing Blocklist and Allowlist by file type.

More details: Blocklist by filetype configuration


Blocklist
Workflow Visualization
cess
Blocklist
Country of Origin
Search...
Process
Process Files Even if They Are Blocklisted
Blocklist
Blocklist by Filetype
Blocking Rule
Type
File Size Condition
Add
Delete
Blocklist by Hash
Hashes
Hashes
fa3ab181451da0d8171e73d9f25c216e82
Add Blocklist by filetype
File type
Adobe Files: ALL
File Size Condition
Enabled
Condition
If bigger than
Size
0
Unit
MB
Allow in Archive
Enabled
Cancel
Add

Allowlist



During scan, it is possible to create allowlists where files depending on their file type, checksum (hash), filename, extension, file size or MIME-TYPE. All of these can be stored in the fields on the Allowlist tab. Also it is available to allowlist all the files coming from the same group, such as Text Files, Office Documents and others.

When filtering by mime-type or filename, the filter is handled as a regular expression.

Exceptions can be defined in **Exceptions (by mime-type)** section using regular exceptions. For instance, if all office files have to be allowed except docx files, then **Office documents** group should be chosen and `^application/vnd.openxmlformats-officedocument.wordprocessingml.document$` expression should be given as exception.

Metascan

SKIP ANY FURTHER PROCESSINGS WITH AV SCANNING RESULT[S]: When enabled, this setting will halt further file processing if the aggregated antivirus (AV) scan result matches any value in the selected list. This can help optimize performance by avoiding redundant processing steps when a conclusive AV result has been obtained.

EXCLUDE ENGINES: Anti-malware engines not to be used in this workflow also can be listed here.

SUSPICIOUS DETECTED HANDLED AS: By enabled, you are able to decide if Suspicious result on any particular engine is considered as Infected or No Threat Found result, and it will take consideration into overall process result which also is constraint by threat detected threshold setting.

NUMBER OF ACTIVE ANTI-MALWARE ENGINES: You can specify the number of active anti-malware engines required for performing a processing. When disabled, no active anti-malware engine is needed to be up to start a processing.

REJECT SUBMISSIONS WHEN DEFINITIONS OF A NUMBER OF ENGINES SURPASS A CERTAIN AGE: This setting includes two sub-settings: *Definition Age Threshold* and *Outdated AV Engine Threshold*. When enabled, it rejects file submissions if the definition age of a certain number of antivirus (AV) engines exceeds the configured thresholds. This ensures that scans are performed using up-to-date threat definitions, maintaining the reliability and accuracy of the scanning results.

SCAN FAILURE THRESHOLD: It is possible to enable and set a threshold value for failed engine results. If the number of failed results for the currently scanned object reaches this threshold, the overall result will be marked as failed. This threshold value will not influence the scan result if there are other outcomes, such as "suspicious", except in cases where the result is categorized as "infected".

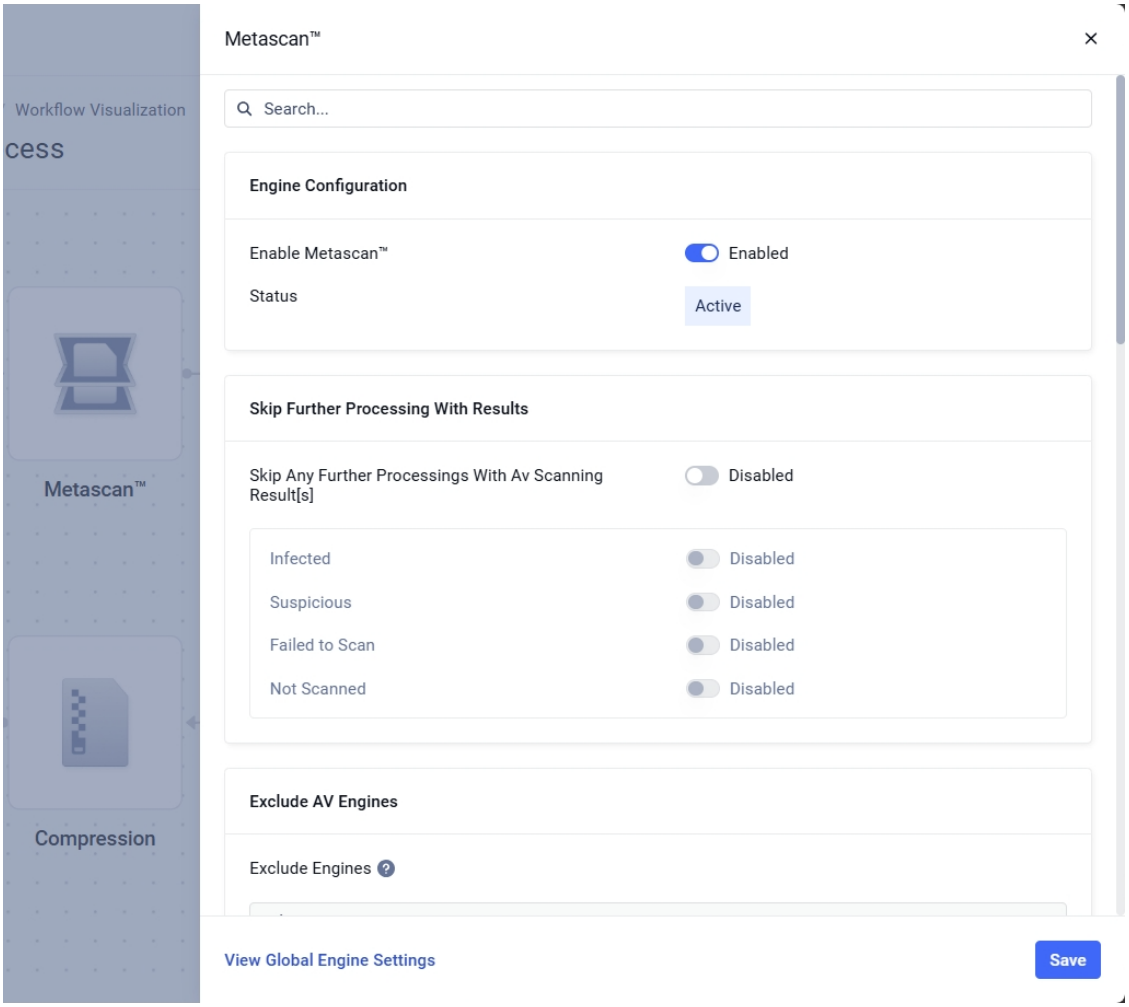
THREAT DETECTED THRESHOLD: When checked, this setting supports two configuration options INFECTED LIMIT and the SUSPICIOUS LIMIT when SUSPICIOUS DETECTED HANDLED AS is disabled, and its handling logic will be described as following:

- If the number of infected engine results is between these values the overall result will be suspicious.
- If the INFECTED LIMIT is reached the overall result will be always infected.
- If none of them is reached the overall result will be the highest priority engine result (infected results are ignored).

Nevertheless, if **SUSPICIOUS DETECTED HANDLED AS** is also enabled, regardless of handling as Infected or No Threat Found, SUSPICIOUS LIMIT setting are no longer taken into account. The following is the handling logic:

- If the INFECTED LIMIT is reached, the overall result will be always infected.
- Otherwise, the overall result will be the highest priority engine result (infected results are ignored).

PER ENGINE TIMEOUT / EXTERNAL SCANNER TIMEOUT / GLOBAL SCAN TIMEOUT: The timeout for the different engines and the whole scanning process also can be set. The maximum allowed size of scanned objects can be set also on this tab as well.

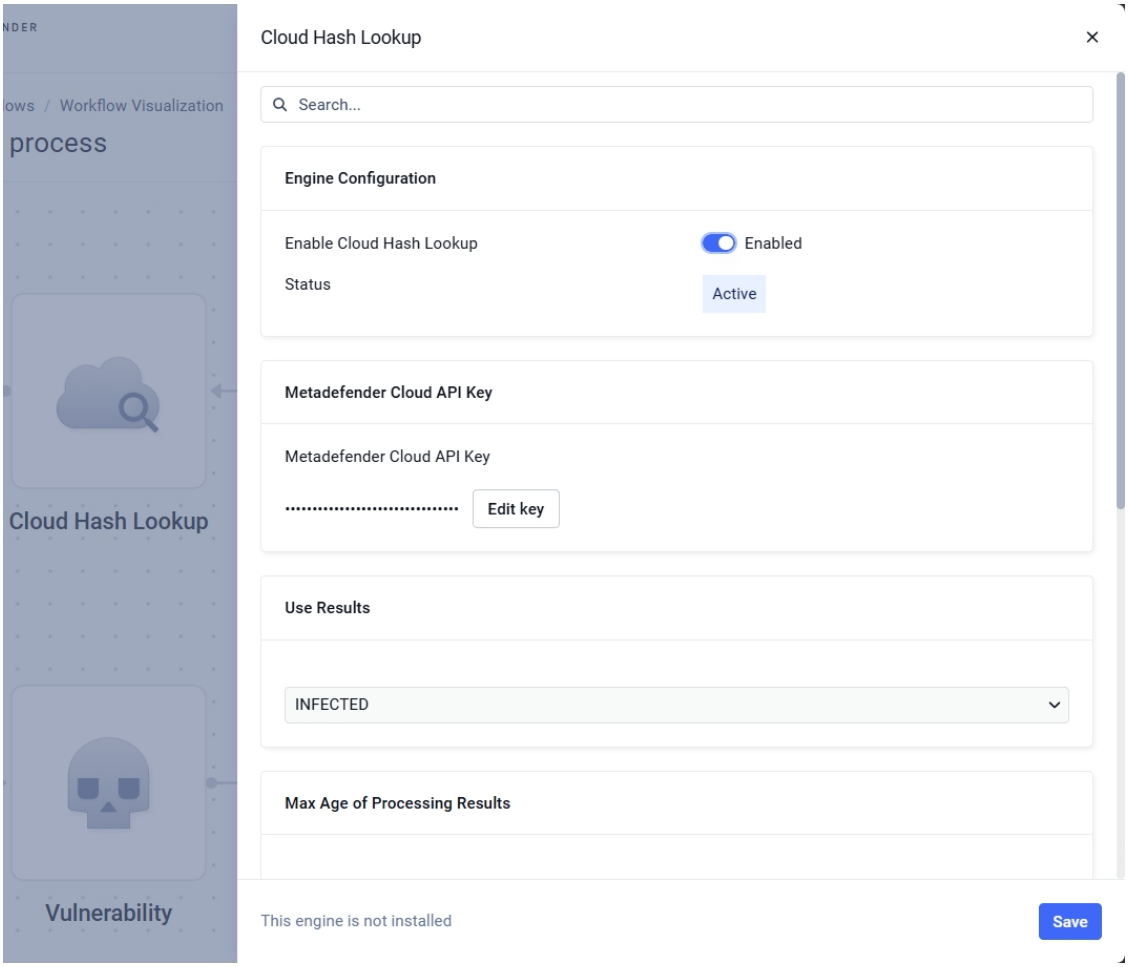


Cloud Hash Lookup

When MetaDefender Cloud workflow element is enabled, online database of MetaDefender Cloud will be used as source for hash lookups.

Available options:

1. **Use results**: INFECTED or ALL RESULTS If INFECTED is chosen, then only that result will be accepted as result, otherwise all type of results will be taken into account.
2. **MetaDefender Cloud API key**: An API key is necessary to have access to the MetaDefender Cloud database. API Key Information can be found on https://metadefender.opswat.com, under **Account Information** page.
3. **Maximum age of scan results**: Only results that are not older than what is set here will be considered as a valid result.
4. **Excluded engines' name**: Name of the engines whose results are not to be taken into account.
5. **Minimum hit count**: To consider a verdict as a valid one, there should be at least as many result for a hash as it has been set here. (If **Use result** is set to INFECTED, then only infected results will be counted in.)
6. **Time out**: The time interval within which the response should be received from MetaDefender Cloud.



Deep CDR

By enabling Deep CDR, one can convert from a set of supported file types into another (or the same). By doing so lot of vulnerabilities can be got rid out of rendering the resulting file be more safe. Both the types to be sanitized and the target file type can be set. To set the file types that you want to sanitize you should tick on corresponding checkboxes. In addition, you can also tick on "**ENABLE FOR ALL FILE TYPES**" to choose all supported file types. File name from sanitized files can be defined by using "OUTPUT FILENAME FORMAT" field. For usage and meanings of variables, please refer to Setup output file name page.

By default, MetaDefender Core allows files, where sanitization fails. This behavior can be overridden enabling "BLOCK FILES IF **SANITIZATION FAILS OR TIMES OUT**".

The maximum allowed time for data sanitization to be made can be configured through the "**DEEP CDR TIMEOUT**" and "**TRY COUNT**" options, where first one means that data sanitization should finish within the configured time frame, otherwise abort the conversion and latter means the number of times product should retry in case of a failed conversion.

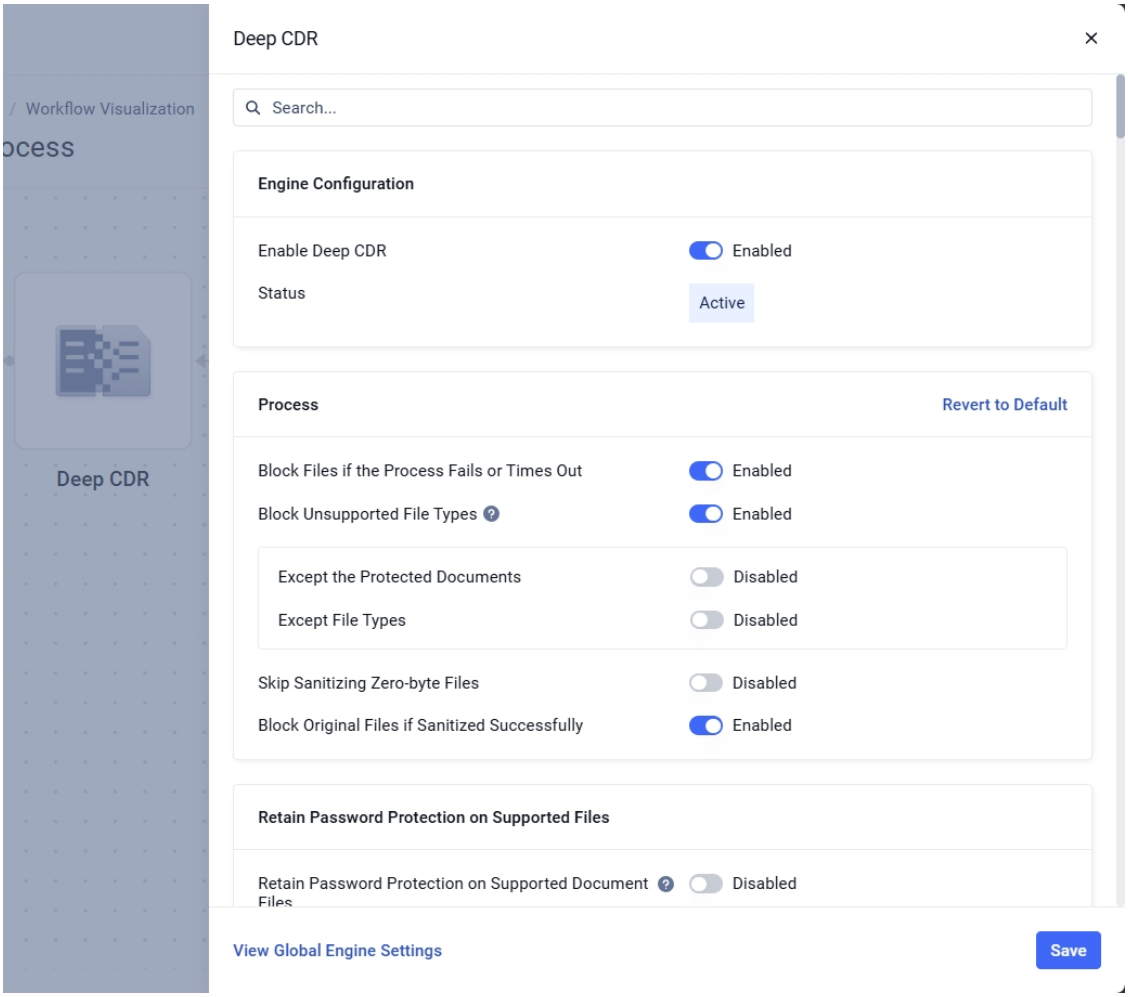
When "**DISTINGUISH PARTIAL ARCHIVE SANITIZATION RESULT**" checked, MetaDefender Core will return "Partial Sanitization" processing result for Deep CDR when only some of child files in original archive files are sanitized successfully.

Beware, however, that possible data loss or change may occur during conversion, thus this feature is disabled by default.

Result of sanitization can be either downloaded on the scan page or retrieved the data ID via REST. See Fetch

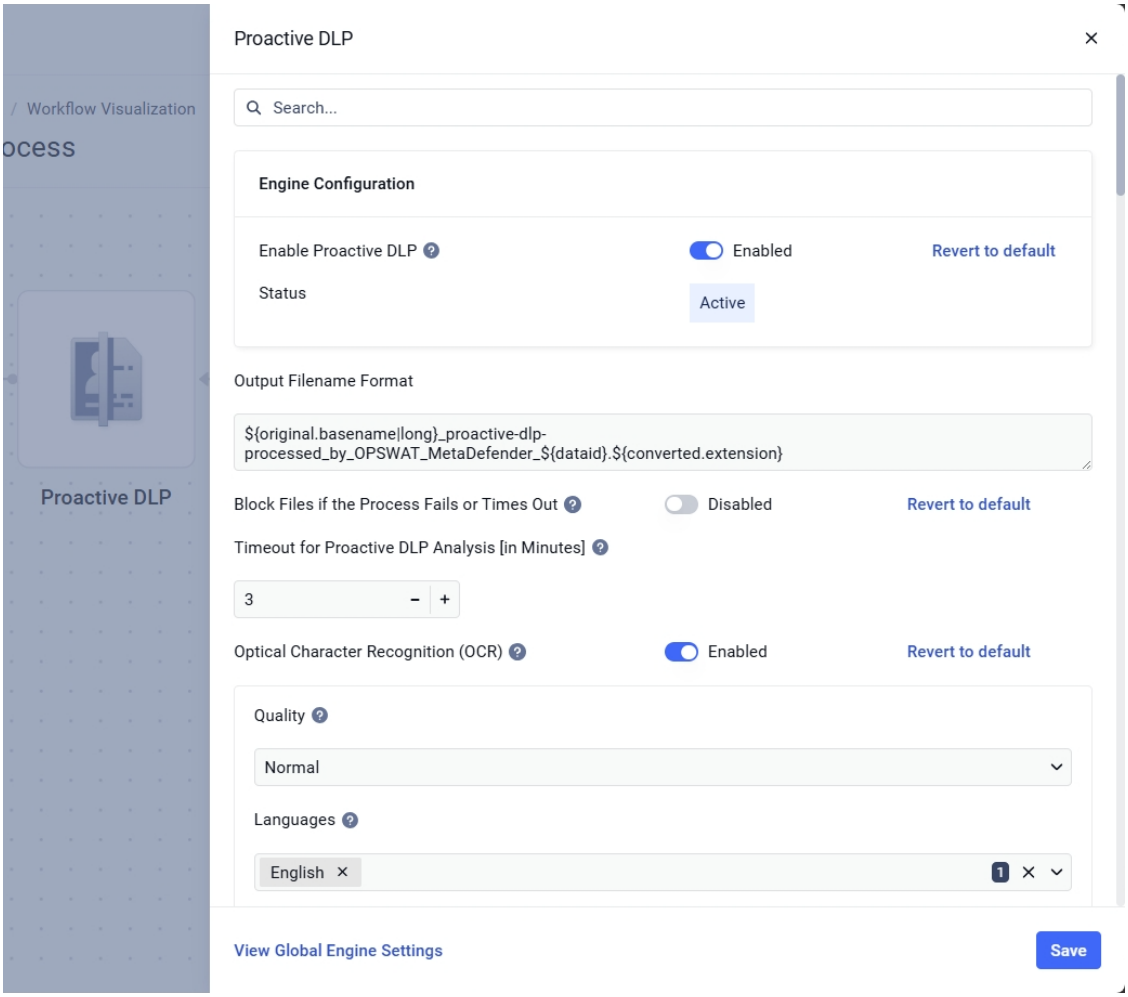
processing result. Note that /hash API does not provide such information.

Length of time the system stores sanitized files can be set in **Settings > Data retention**.



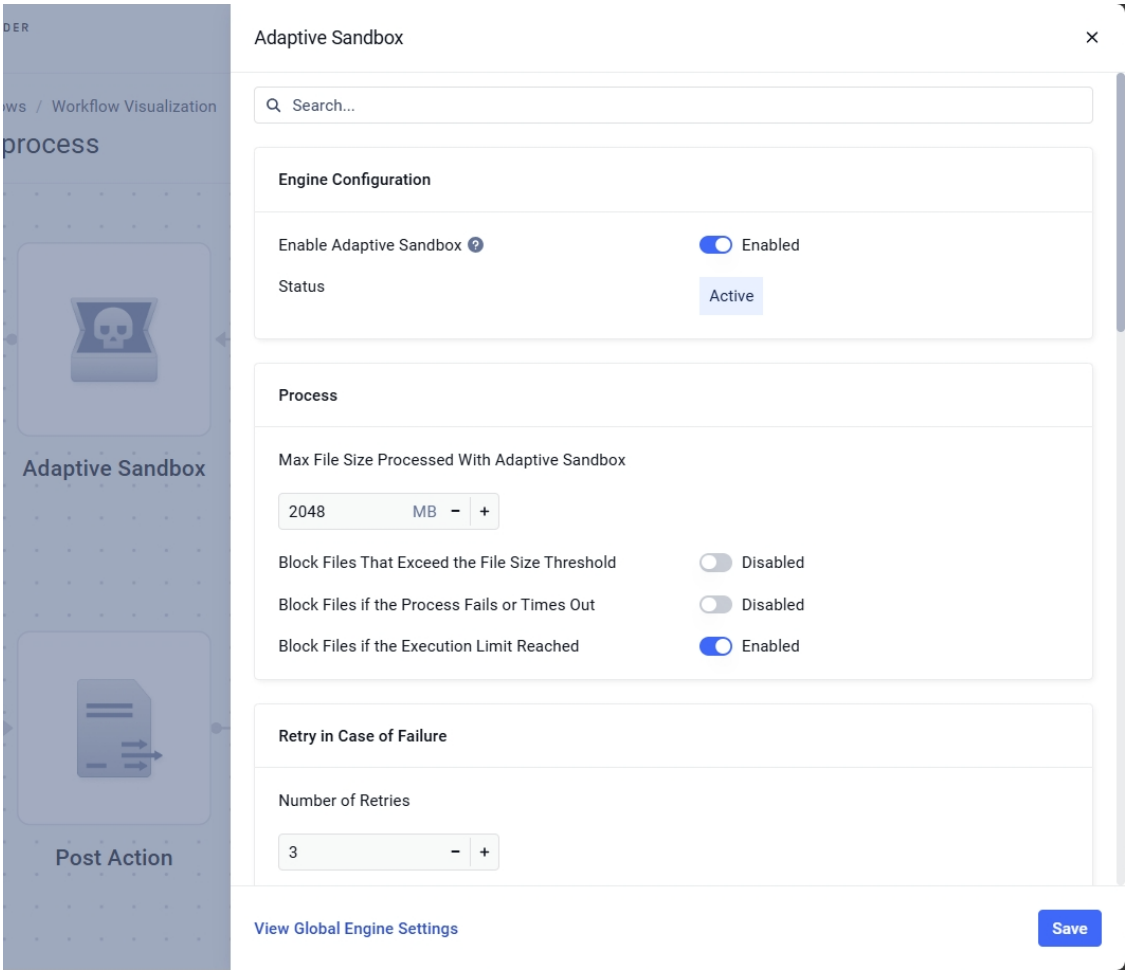
Proactive DLP

For all information about features powered by Proactive DLP, please learn more at Proactive DLP overview.



Adaptive Sandbox

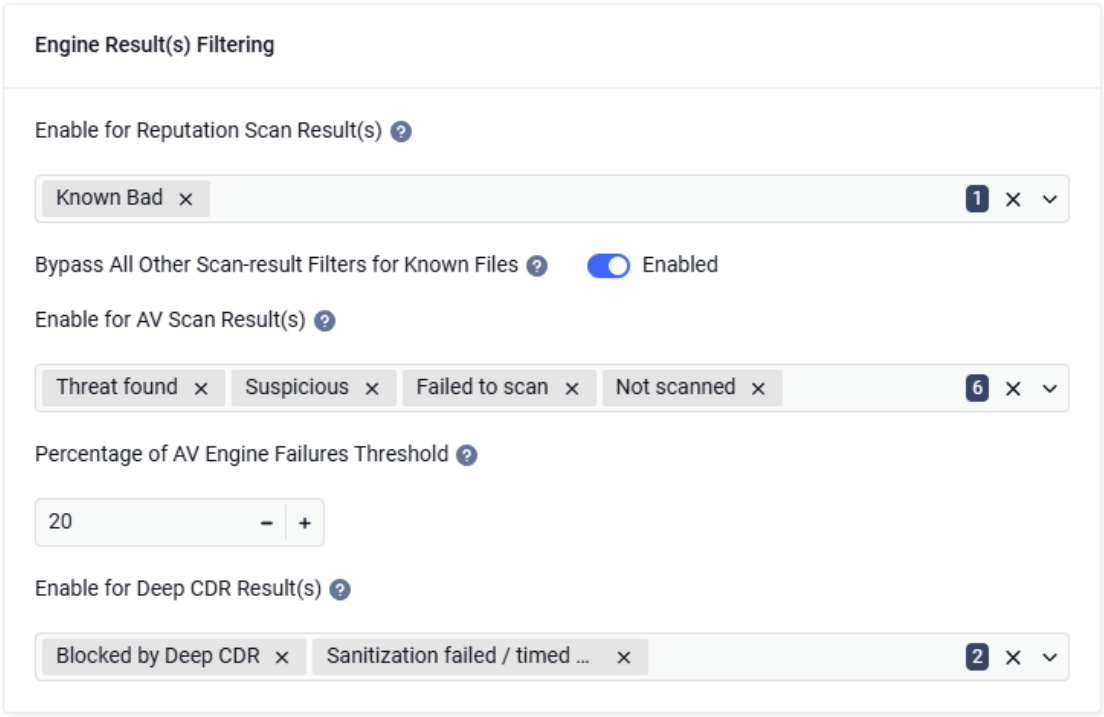
For all information about features powered by Adaptive Sandbox, please learn more at Adaptive Sandbox overview.



With File type Filtering configuration, you can configure Adaptive Sandbox engine to run only for the selected file types. All the supported file types are selected by default.

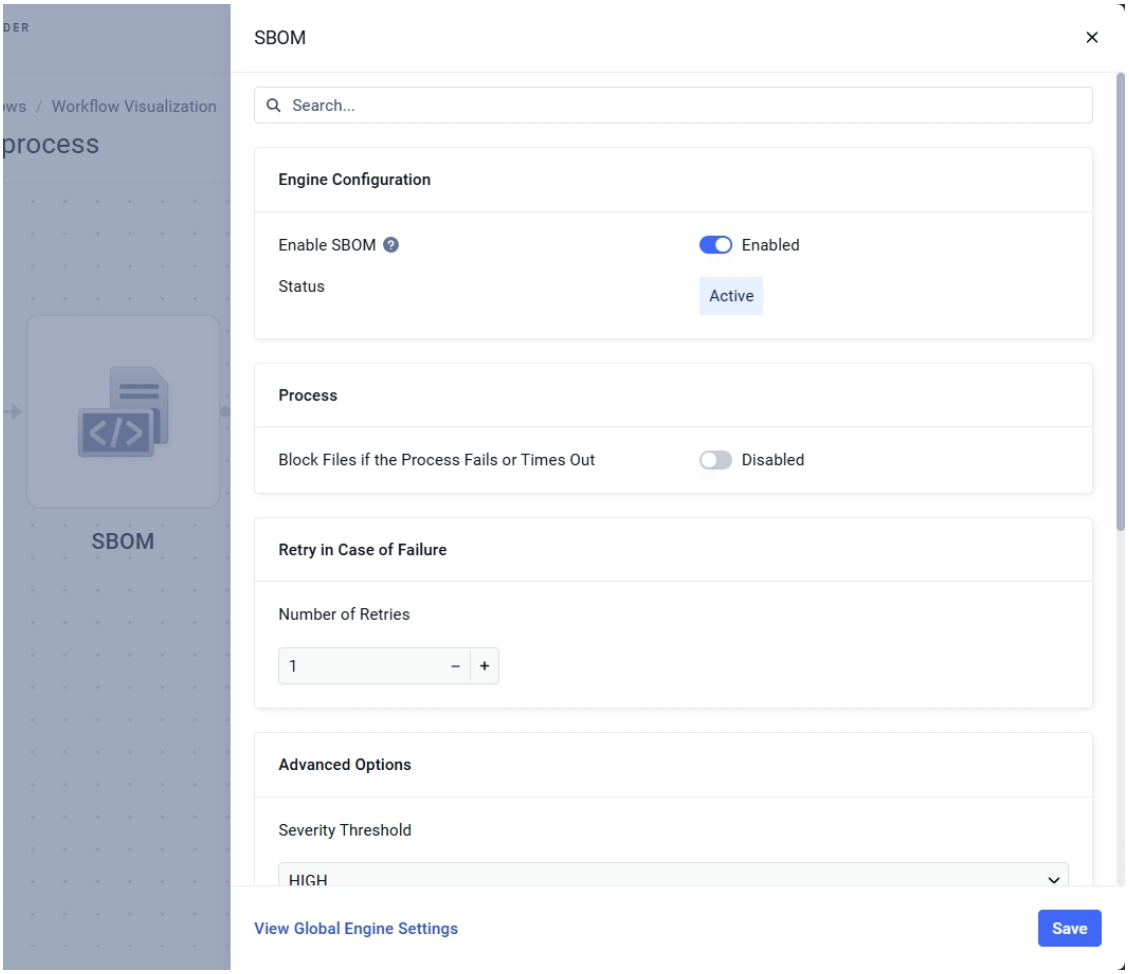
With the Result Filtering configuration, you can configure Adaptive Sandbox engine to process files based on

- All Reputation results, including Known Bad or Known Good.
- All Metascan AV scan results, including No Threat Found or any threats found. By default, No Threat Found is not selected.
- All Deep CDR results, regardless of successful sanitization or failures. By default, only sanitization failures are selected.

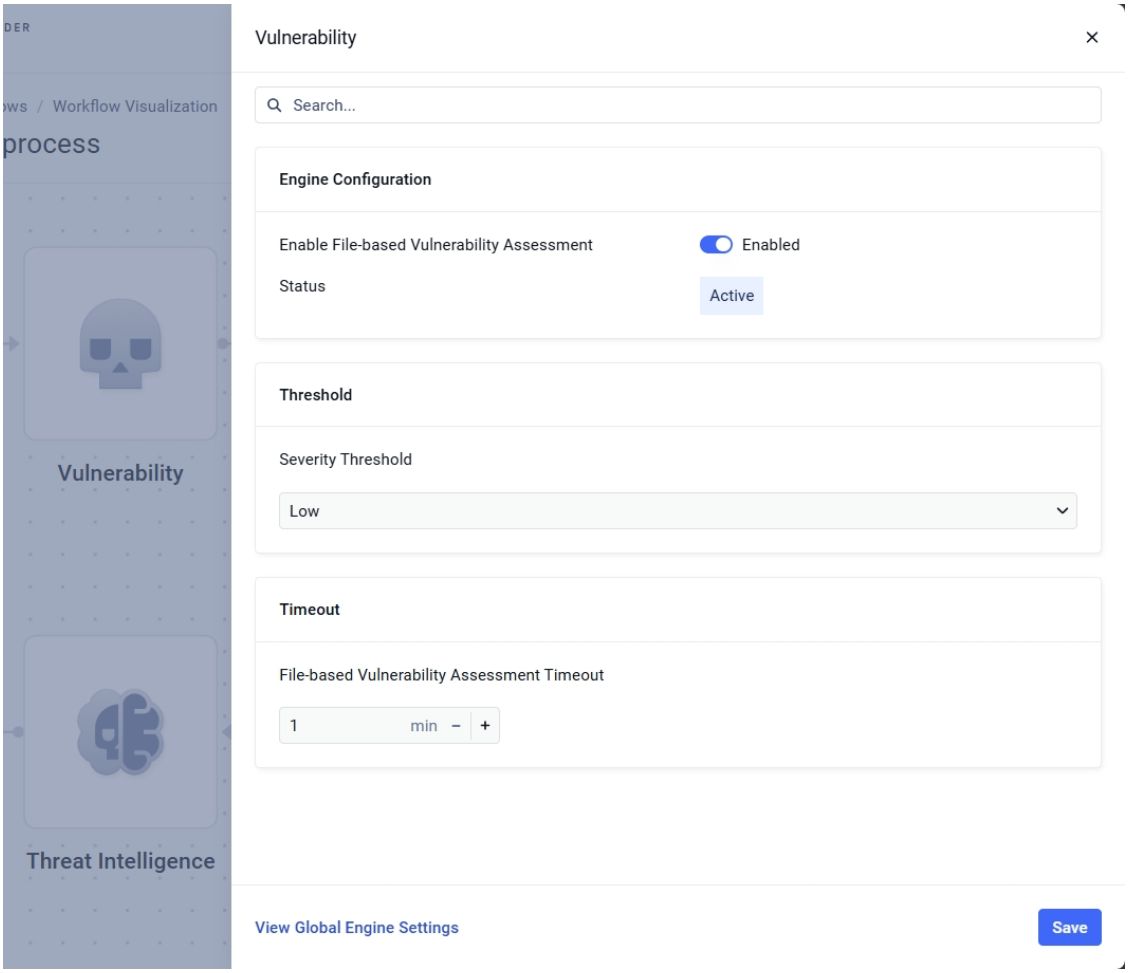


SBOM

For all information about features powered by SBOM, please learn more at Software Bill of Materials (SBOM).

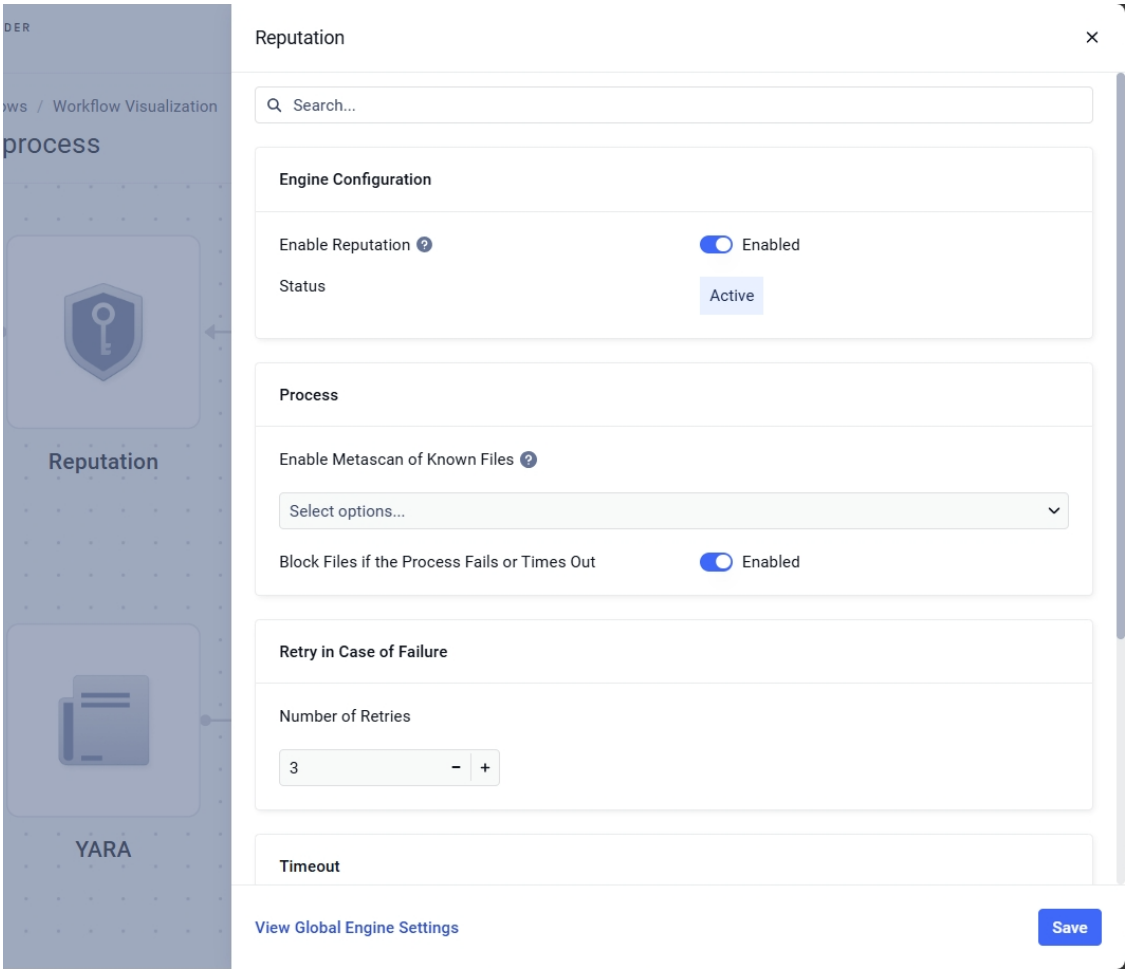


Vulnerability



File-based Vulnerability Assessment scans and analyzes binaries and installers to detect known application vulnerabilities. Administrators has an ability to configure severity threshold and timeout settings.

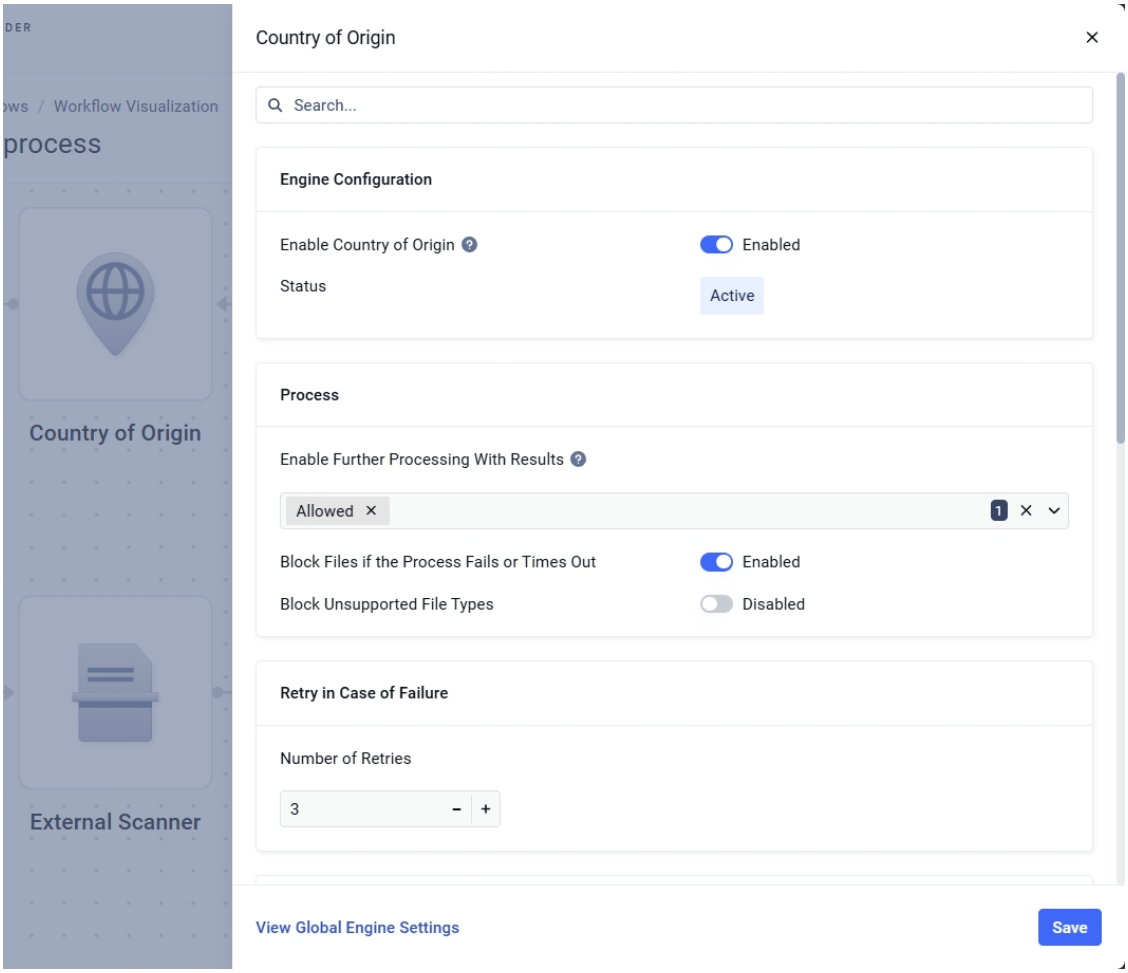
Reputation



For all information about features powered by Adaptive Sandbox, please learn more at Reputation overview.

Country of Origin

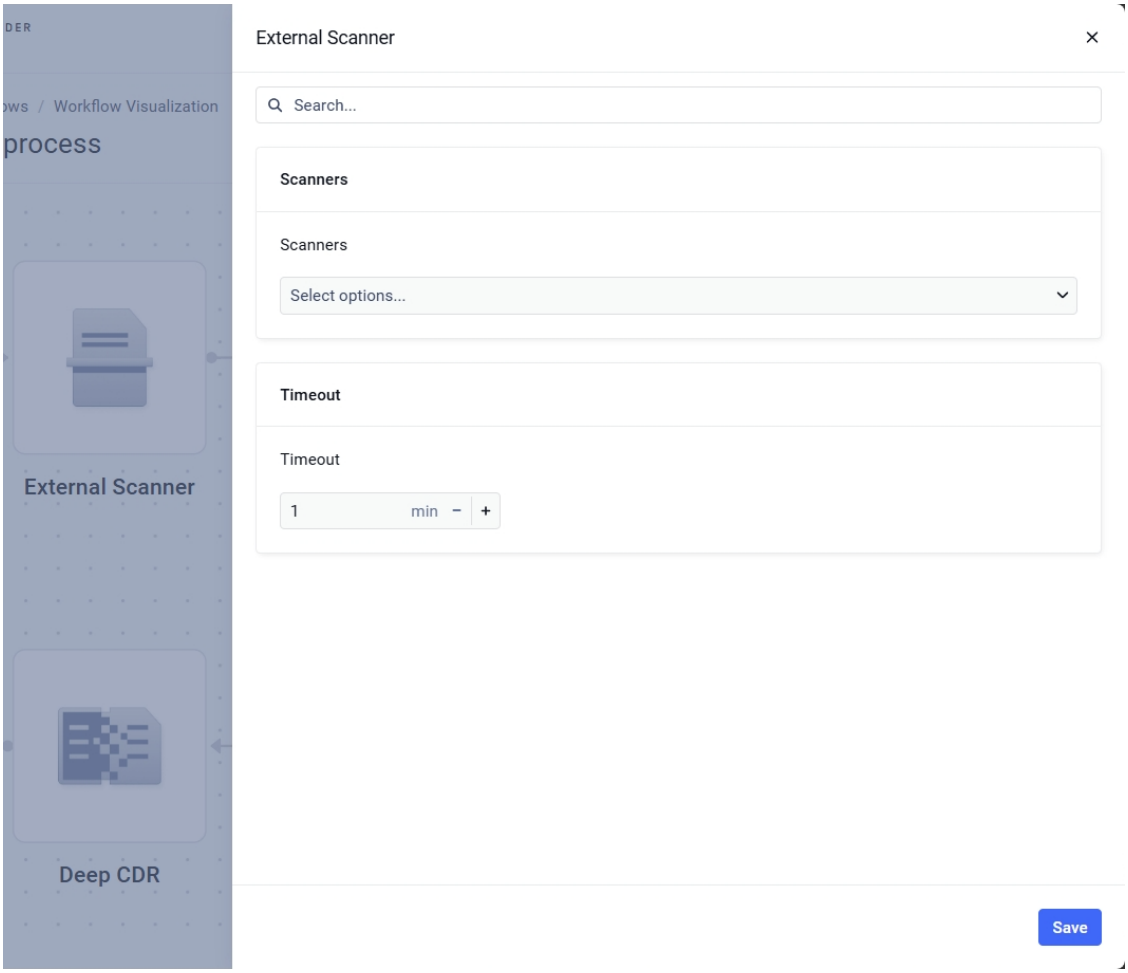
For all information about features powered by Country of Origin, please learn more at Country of Origin



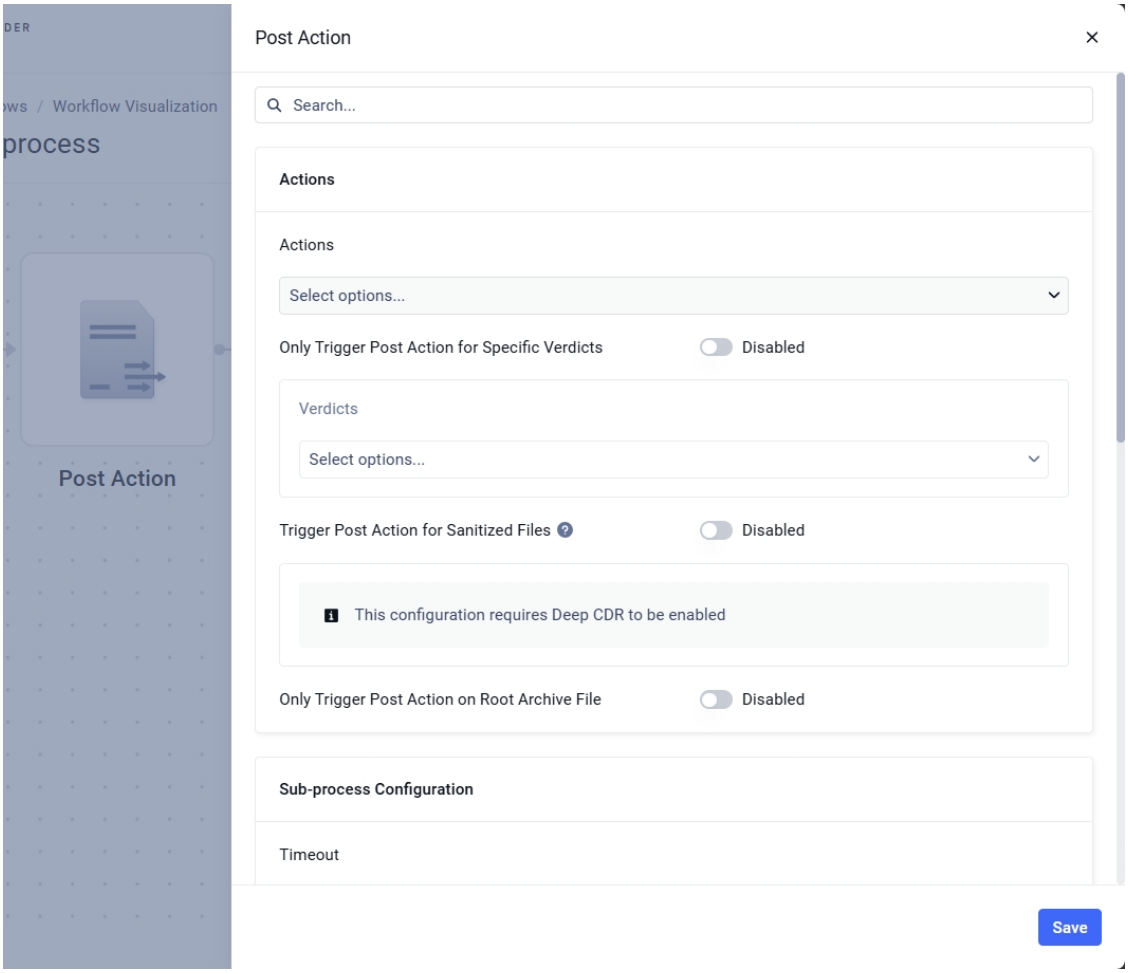
External Scanner and Post Action

One or more applicable external scanner(s) and post action(s) could be selected for desired workflow, and they come with corresponding settings (e.g. timeout). More details at External Scanners and Post Actions.

External Scanner tab:



Post Action tab:



General

By enabling '**Quarantine blocked files**' all of the files which are blocked are automatically copied to the quarantine. For detailed description of the quarantine please see the Quarantine.

By enabling '**Fallback filetype detection to current extension if needed**' (enabled by default), file type detection can use the extension of the currently processed file as a helping hand. For example this could be useful, when analyzing CSV files.

Setting '**Enable file hash for only selected types**' provides a list of supported hash types.

- By default, MD5, SHA1 and SHA256 are selected. SHA512 is unselected to avoid affecting processing performance, despite not much considerable.
- SHA256 cannot be deselected.
- File-based Vulnerability Assessment engine requires SHA1. Therefore, when SHA1 is deselected, it is expected to receive the result "Skipped the assessment" from the engine.

File download timeout: Scan-from-link is the feature supporting users to process file by just specifying a direct download link via header "downloadfrom", instead of sending a payload in HTTP(S) request body. You are able to define timeout for file downloading process of this feature.

By enabling '**Global Timeout**' (disabled by default), application provides a global timeout for entire processing applied to every scan request.

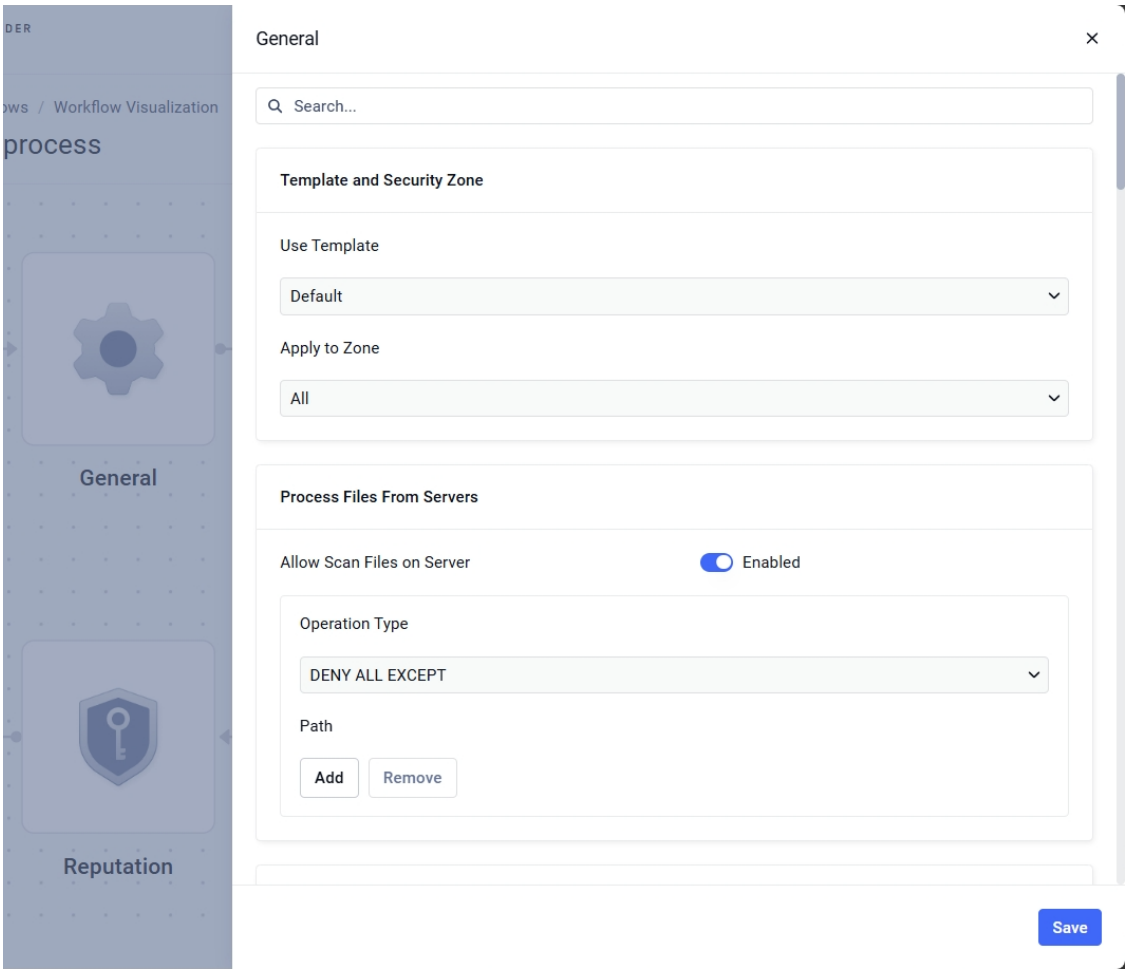
- This is only applicable to Asynchronous mode.
- When a scan request is timed out by this global processing timeout setting, the scan request will be blocked with "Global timeout exceeded" as blocked reason.
- **Note:** *This global processing timeout, when enabled, it should be greater or equal to the sum of all other engine related timeout settings such as Metascan timeout, Deep CDR timeout etc.*

By enabling '**OVERRIDE SCAN RESULTS CLASSIFIED AS ALLOWED**' it is possible to overwrite the default behavior of MetaDefender and determine which scan verdicts should result as allowed.

Scan results checked are marked as allowed.

By default only following verdicts result in allowed status:

- No Threat Detected
- Skipped Clean
- Potentially Vulnerable File
- Yara Rule Matched

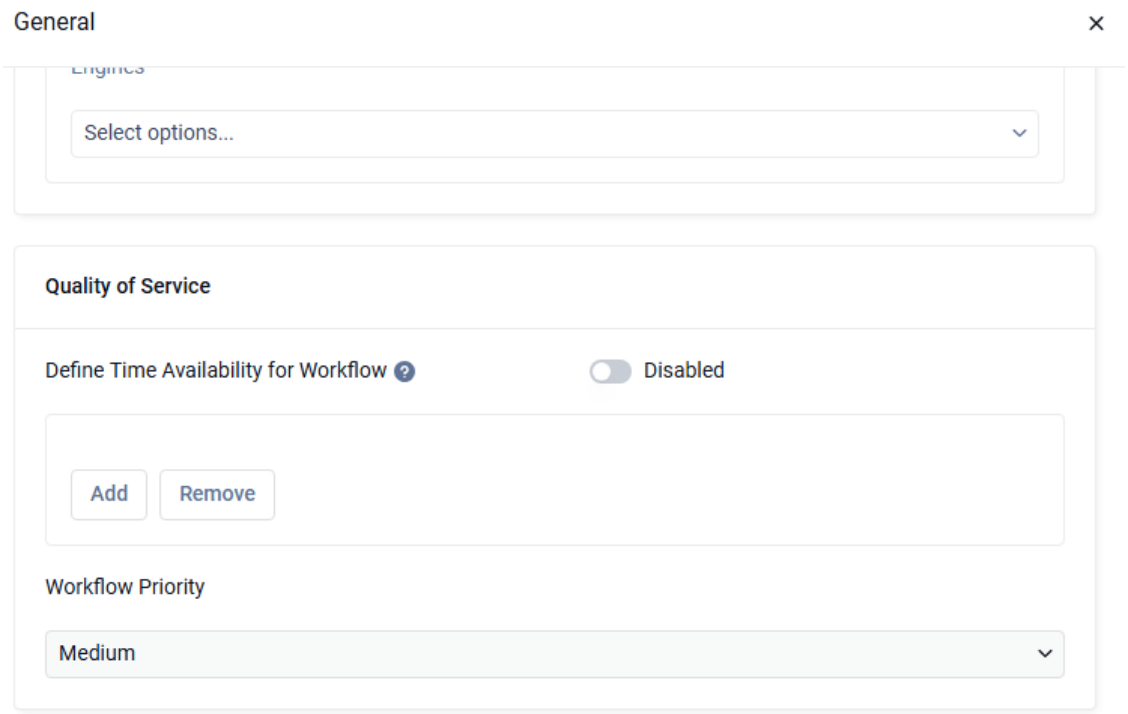


By enabling '**Define time availability for workflow**', it is possible for clients to define time frames in which a specific workflow is ready to use. MetaDefender Core only accepts requests of the workflow rule if they are within available time frames defined by the rule and responds HTTP code 403 otherwise.

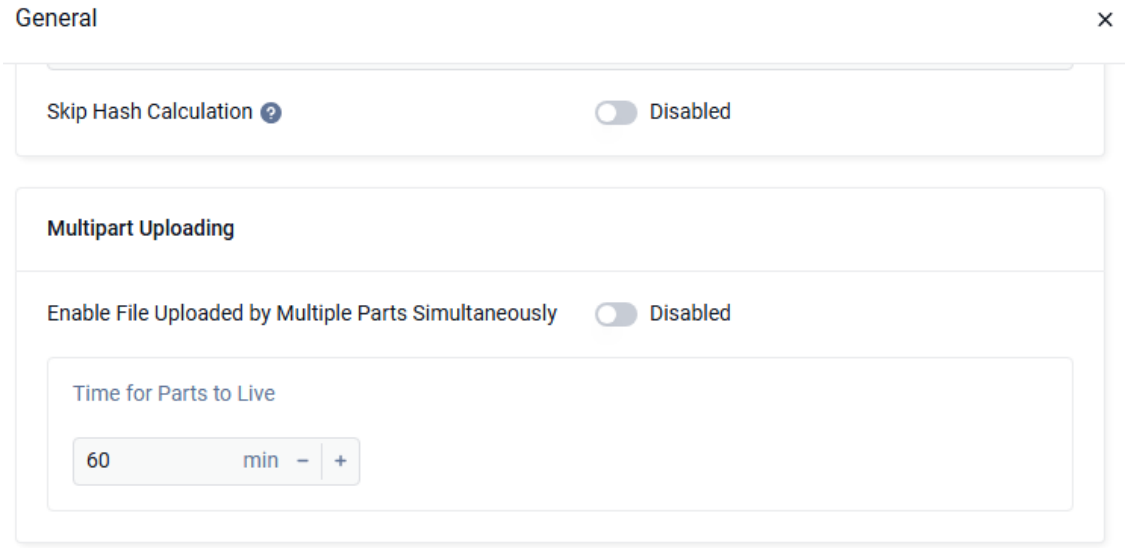
By selecting one of values of '**Workflow priority**', it is possible for clients to prioritize a task coming to engine.

- If there are several tasks of the same priority waiting for an engine to process, the first comes, the first is processed by the engine.
- If tasks of different priorities are waiting for an engine to process, every two tasks of high priority processed by the engine will give a chance to one single task of next lower priority to be processed.

By default, priority of all tasks coming to an engine is Medium.



By enabling '**Multipart uploading**', it is possible for clients to upload files for scan by multiple parts simultaneously. By default, parts have at most 60 minutes to live until the file is combined successfully to scan.

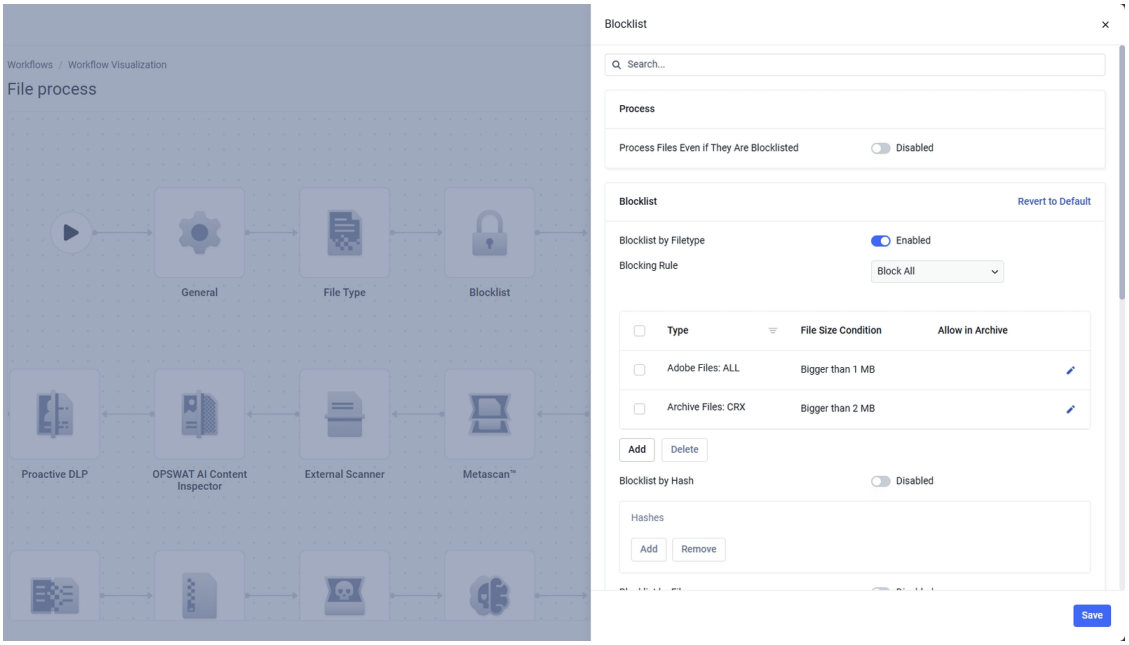


Blocklist by filetype configuration

Overview

The **Blocklist File Type** feature allows you to block files from being processed based on their detected file type. When a file matches a blocklist rule, it is immediately marked as **Blocked** and skips all scan engines.

You can access this feature from the **Blocklist** tab in the workflow/rule configuration page.



Info

Since version 5.18.0, MetaDefender Core introduces a brand-new UI. To align with the updated UI/UX, some minor logic and usability changes have been implemented.

Previously, users were required to select a group first and then choose file types within that group, which made the feature less intuitive. In the updated version, the boundary between groups and file types has been streamlined. While groups are still maintained for backend alignment, users can now directly input file types when adding new items, making the process significantly more user-friendly.

There is also a slight change in UX behavior:

- **Before:** If a user enabled a group, selected specific items, and later disabled the entire group, those sub-items would remain configured when the group was re-enabled.
- **After:** Starting from version 5.18.0, if a group is disabled, all sub-items within that group are cleared. When the group is re-enabled, users will need to reselect the desired items.

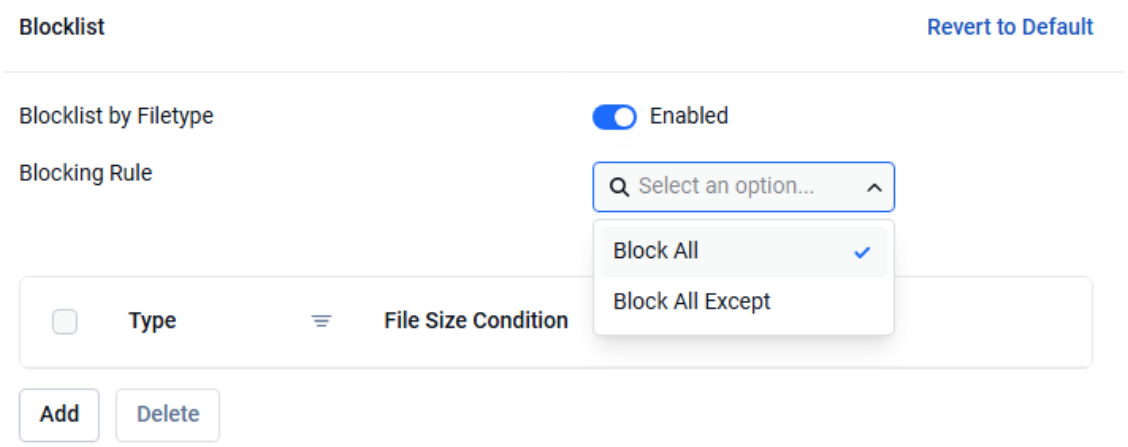
Please note that this change only affects the user experience. The underlying logic remains unchanged—when a group is disabled, its sub-items are not applied, so there is no impact on backend processing or final results.

Blocklist Modes

The blocklist operates in one of two modes, selected from the dropdown at the top-left of the dialog:

Mode	Meaning
Block All	Every file type you add to the list **will be blocked**. All other file types are allowed through for normal processing.
Block All Except	Every file type **not** in the list will be blocked. Only the file types you add are allowed through for normal processing.

Tip: *Think of* ***Block All*** *as a deny-list (block specific types) and* ***Block All Except*** *as an allow-list (only permit specific types).*



Common Controls

Regardless of which mode you choose, the dialog provides the following controls:

- **Condition dropdown** (top-left) - Switch between "Block All" and "Block All Except".
- **Search bar** - Filter the list of configured rules. Type a file type name or extension to quickly find existing rules.
- **Add Rule** button - Add a new file type rule to the list.
- **Save** / **Cancel** buttons - Save your changes or discard them.

Adding a Rule

1. Click **Add Rule**
2. A new row appears in the table.
3. Select a **File Type** from the dropdown (e.g., "Archive Files: XZ", "Executable Files: EXE").
4. Optionally configure a **File Size Condition** (see below).
5. Configure the mode-specific option (**Allow In Archive** or **Allow Child Files**, depending on the mode).

6. Click **Save** to apply.

To remove a rule, click the trash icon in the **Action** column for that row.

> **Info**
> After saving and closing the window, reopening it will display the items reordered alphabetically (grouped accordingly). However, you can easily locate configured items using the search bar, ensuring quick access to your settings.

File Size Condition

Each rule can optionally include a file size condition, which narrows when the rule applies. This lets you block (or allow) a file type only when the file is above or below a certain size.

To enable a file size condition:

1. Toggle the **File Size Condition** switch to ON (blue).
2. Select a condition from the dropdown:
   - **If smaller than** - The rule applies only when the file is smaller than the specified size.
   - **If bigger than** - The rule applies only when the file is bigger than the specified size.
3. Enter a numeric value in the size field.
4. Select the unit: **B** (bytes), **KB** (kilobytes), or **MB** (megabytes).

Examples:

> ***Note:*** *When the file size condition is toggled OFF, the rule applies to all files of the selected type regardless of their size.*

Column	Description
File Type	The file type this rule applies to.
File Size Condition	Optional size-based narrowing of the rule.
Allow In Archive	Toggle to control behavior when this file type is found inside an archive.
Action	Delete the rule.

Allow In Archive

This option is only available in **Block All** mode.

When a blocked file type is found **inside an archive** (e.g., an EXE inside a ZIP file), you may want to handle it differently than when it appears as a standalone file.

Allow In Archive	Behavior
OFF (default)	The file type is blocked immediately, even if it is extracted from an archive.
ON	If the file is inside an archive, it is **not blocked immediately**. Instead, it proceeds through normal processing (all configured scan engines run on it). The final verdict is determined by the scan results, not the blocklist.

When to use this:

- Turn ON **Allow In Archive** when you want to block standalone uploads of a certain file type (e.g., direct EXE uploads) but still want files of that type to be scanned normally when they are extracted from archives. This is useful when the context of being inside an archive changes the risk profile - for example, you might want to block raw EXE uploads but still allow AV engines to scan EXEs found inside ZIP files.

Example scenario:

1. You add a rule: Block All + "Executable Files: EXE" + Allow In Archive = ON.
2. A user uploads `malware.exe` directly -> **Blocked** (file is not inside an archive).
3. A user uploads `package.zip` containing `tool.exe` -> The ZIP is processed normally. When `tool.exe` is extracted, it is **not blocked** and proceeds through normal scanning. If an engine flags it as malicious, it will be marked accordingly.

"Block All Except" Mode - Details

In **Block All Except** mode, the table columns are:

Column	Description
File Type	The file type this rule **allows** (everything else is blocked).
File Size Condition	Optional size-based narrowing of the rule.
Allow Child Files	Toggle to specify which child file types inside archives of this type should also be allowed.
Action	Delete the rule.

Understanding "Block All Except"

This mode inverts the logic: you are building an **allow-list**. Only the file types you add to the list will be permitted. All other file types will be blocked.

Example: If you add only "Archive Files: ZIP" and "Document Files: PDF" to the list, then:

- ZIP files -> **Allowed** (in the list)
- PDF files -> **Allowed** (in the list)
- EXE files -> **Blocked** (not in the list)
- DOCX files -> **Blocked** (not in the list)

Allow Child Files

This option is only available in **Block All Except** mode and addresses an important question: **What happens to files extracted from an allowed archive type?**

Consider this scenario: You allow ZIP files, but a ZIP archive contains an EXE file. Since EXE is not in your allow-list, the extracted EXE would normally be blocked. The **Allow Child Files** option lets you specify exceptions for extracted files.

Allow Child Files	Behavior
OFF (default)	Extracted files follow the normal blocklist rules. If their type is not in the allow-list, they are blocked.
ON	You must select one or more file types from the dropdown. Extracted files matching those selected types will be **allowed through** even though they are not in the main allow-list.

To configure Allow Child Files:

1. Toggle the **Allow Child Files** switch to ON (blue).
2. From the dropdown that appears, select one or more file types that should be allowed when extracted from this archive type.

Example scenario:

1. Mode: Block All Except.
2. You add a rule: Allow "Archive Files: ZIP" with Allow Child Files = ON, and select "Executable Files: EXE" and "Document Files: PDF".
3. A user uploads `package.zip` containing `tool.exe` and `readme.pdf` and `script.bat`:
   - `package.zip` -> **Allowed** (in the allow-list).
   - `tool.exe` (extracted) -> **Allowed** (in the Allow Child Files list for ZIP).

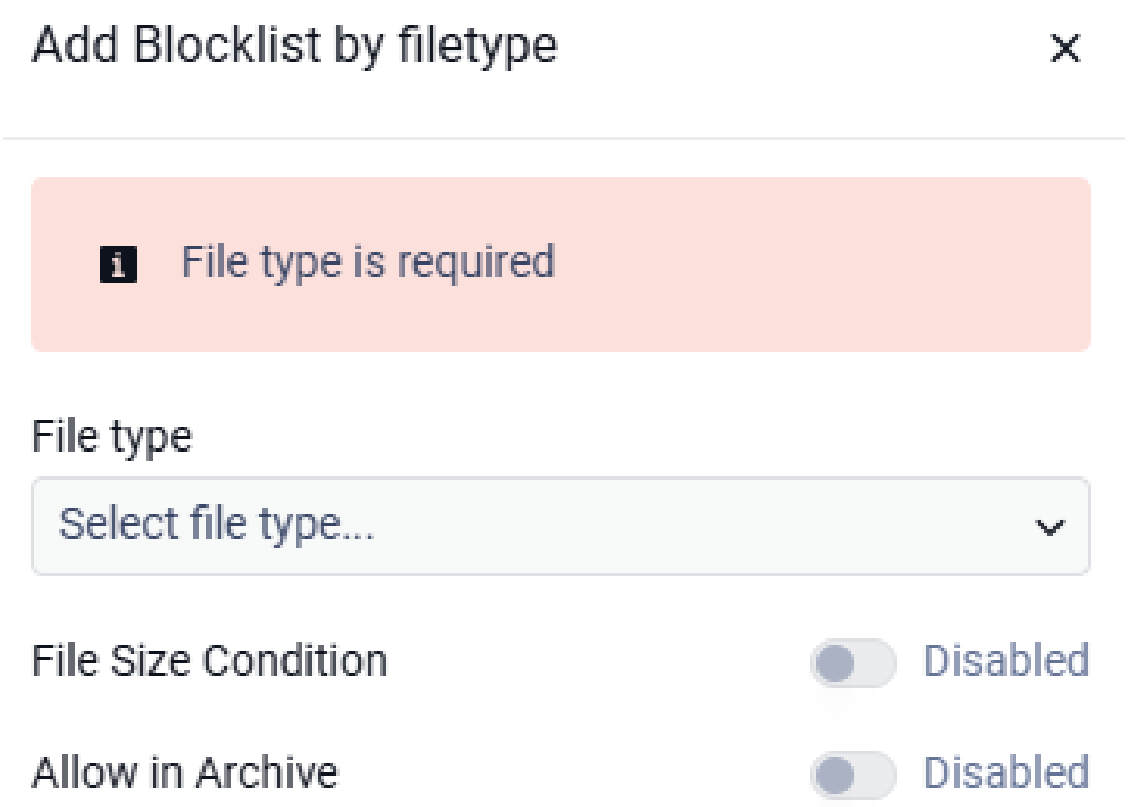
- `readme.pdf` (extracted) -> **Allowed** (in the Allow Child Files list for ZIP).
- `script.bat` (extracted) -> **Blocked** (not in the allow-list and not in Allow Child Files).

Key insight: *Allow Child Files only applies to files that are* ***extracted from an archive****. It does not affect files uploaded directly. A directly uploaded EXE would still be blocked if EXE is not in the main allow-list.*

Searching Configured Rules

Use the **search bar** at the top of the dialog to filter rules that have already been configured. Type a file type name or file extension (e.g., "EXE", "Archive", "PDF") and the list will filter to show only matching rules. This is helpful when you have many rules configured and need to quickly find or verify a specific one.

Error section



The Errors section will automatically disappear once all issues have been resolved.

Clicking the **"Show All Errors"** button will filter the list to display only rows containing errors. In this view, the Blocklist Condition field, Add Rule option, search field, and Save button will be disabled. The button label will also change to **"Show All Rules."**

Clicking **"Show All Rules"** will restore the full list and re-enable all previously disabled controls, returning the interface to its normal state.

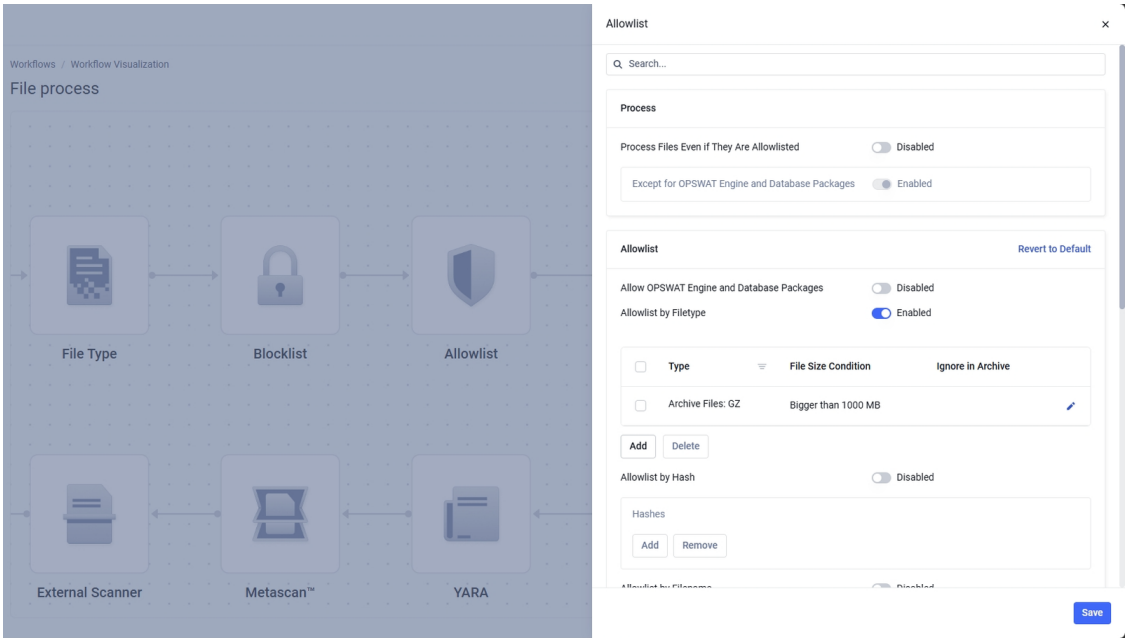
Allowlist by filetype configuration

Overview

The **Allowlist File Type** feature allows you to skip scanning entirely for files of specific types. When a file matches an allowlist rule, it is immediately marked as **Whitelisted** (allowed) and **no scan engines are run on it**. The file receives a clean verdict without being scanned.

You can access this feature from the **Allowlist** tab in the workflow/rule configuration page.

> ***Important:*** *Allowlisting means the file is trusted and bypasses all scanning. Use this only for file types you are confident do not need inspection.*



Info

Since version 5.18.0, MetaDefender Core introduces a brand-new UI. To align with the updated UI/UX, some minor logic and usability changes have been implemented.

Previously, users were required to select a group first and then choose file types within that group, which made the feature less intuitive. In the updated version, the boundary between groups and file types has been streamlined. While groups are still maintained for backend alignment, users can now directly input file types when adding new items, making the process significantly more user-friendly.

There is also a slight change in UX behavior:

- **Before:** If a user enabled a group, selected specific items, and later disabled the entire group, those sub-items would remain configured when the group was re-enabled.
- **After:** Starting from version 5.18.0, if a group is disabled, all sub-items within that group are cleared. When the group is re-enabled, users will need to reselect the desired items.

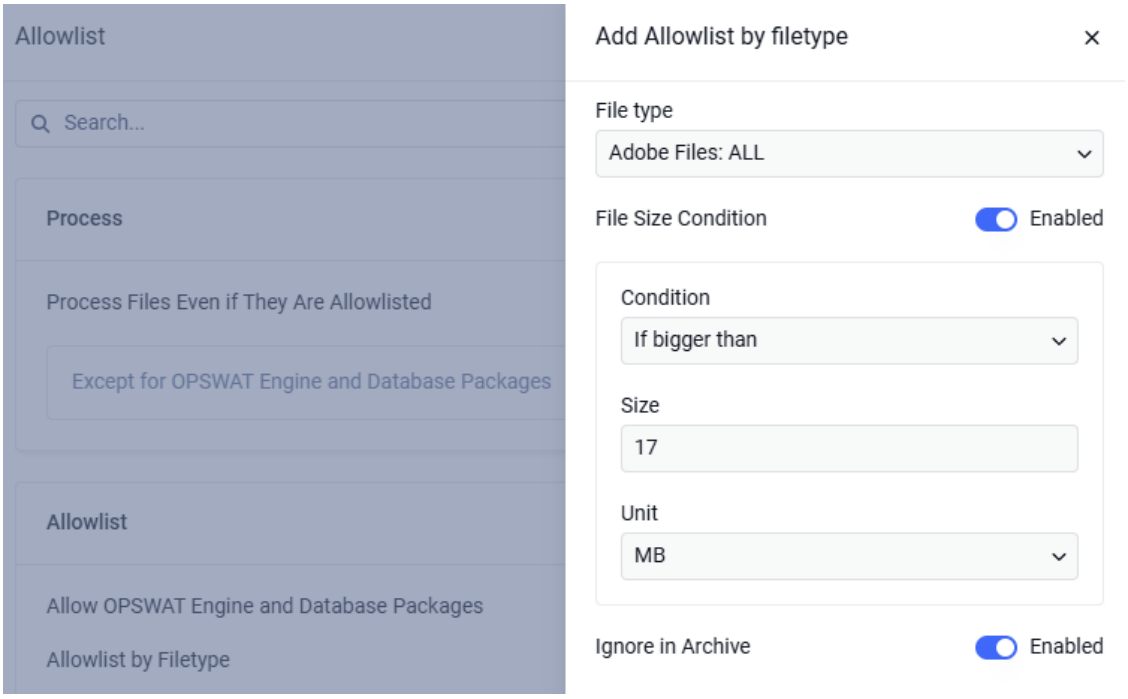
Please note that this change only affects the user experience. The underlying logic remains unchanged—when

a group is disabled, its sub-items are not applied, so there is no impact on backend processing or final results.

How It Differs from Blocklist

The allowlist is simpler than the blocklist — there is no mode selector (no "Block All" / "Block All Except"). You simply add file types that should be trusted and skip scanning.

Feature	Blocklist	Allowlist
Purpose	Prevent specific file types from being processed	Trust specific file types and skip scanning
Verdict when matched	Blocked	Whitelisted
Scan engines run?	No	No
Modes	Block All / Block All Except	Single mode (add trusted types)
Archive-related option	Allow In Archive / Allow Child Files	Ignore In Archive



UI Controls

The Allowlist File Type dialog provides:

- **Search bar** - Filter the list of configured rules by file type name or extension.

- **Add Rule** button - Add a new file type rule.
- **Save** / **Cancel** buttons - Save your changes or discard them.

The table columns are:

Column	Description
File Type	The file type to allowlist (e.g., "Adobe Files: PDF").
File Size Condition	Optional size-based narrowing of the rule.
Ignore In Archive	Toggle to control whether this rule applies to files found inside archives.
Action	Delete the rule.

Adding a Rule

1. Click **Add Rule**
2. A new row appears in the table (on top)
3. Select a **File Type** from the dropdown (e.g., "Adobe Files: PDF", "Image Files: JPEG").
4. Optionally configure a **File Size Condition** (see below).
5. Optionally toggle **Ignore In Archive** (see below).
6. Click **Save** to apply.

To remove a rule, click the trash icon in the **Action** column.

> **Info**
> After saving and closing the window, reopening it will display the items reordered alphabetically (grouped accordingly). However, you can easily locate configured items using the search bar, ensuring quick access to your settings.

File Size Condition

Works the same as in the blocklist. Each rule can optionally include a file size condition to narrow when the allowlist rule applies.

To enable:

1. Toggle the **File Size Condition** switch to ON (blue).

2. Select a condition:

- **If smaller than** - Only allowlist files smaller than the specified size.
- **If bigger than** - Only allowlist files bigger than the specified size.

3. Enter a numeric value.

4. Select the unit: **B** (bytes), **KB** (kilobytes), or **MB** (megabytes).

Examples:

Configuration	Effect
PDF + "If smaller than 5 MB"	PDF files under 5 MB are allowlisted (skip scanning); larger PDFs are scanned normally
JPEG + (size condition OFF)	All JPEG files are allowlisted regardless of size

Note: When the file size condition is OFF, the rule applies to all files of the selected type regardless of size.

Ignore In Archive

This option controls whether the allowlist rule applies to files found **inside an archive** (extracted files).

Ignore In Archive	Behavior
OFF (default)	The allowlist rule applies to all matching files, whether they are uploaded directly or extracted from an archive. Both are marked as Whitelisted.
ON	The allowlist rule is **ignored** for files extracted from archives. Only directly uploaded files of this type are allowlisted. Extracted files proceed through normal scanning.

Why would you use this?

When you trust a file type as a standalone upload (e.g., you trust directly uploaded PDFs from your users), but you still want to scan that same file type when it is embedded inside an archive. Archives can be used to disguise malicious content, so scanning extracted files provides an extra layer of security.

Example scenario:

1. You add a rule: Allowlist "Adobe Files: PDF" + Ignore In Archive = ON.
2. A user uploads `report.pdf` directly -> **Whitelisted** (skips scanning, clean verdict).
3. A user uploads `package.zip` containing `invoice.pdf` -> The ZIP is processed normally. When `invoice.pdf` is extracted, the allowlist rule is **ignored** because it came from an archive. The PDF proceeds through normal

scanning by all configured engines.

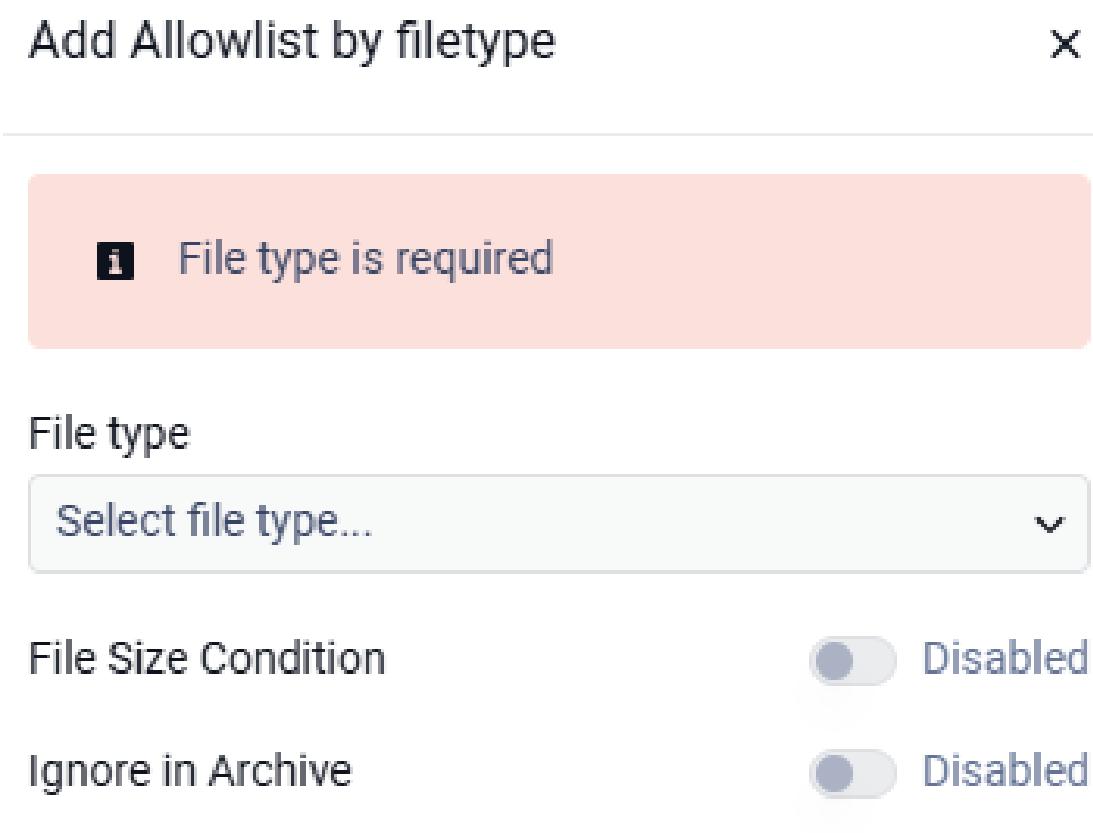
Another example (Ignore In Archive = OFF):

1. You add a rule: Allowlist "Image Files: JPEG" + Ignore In Archive = OFF.
2. A user uploads `photo.jpg` directly -> **Whitelisted**.
3. A user uploads `images.zip` containing `photo.jpg` -> The extracted `photo.jpg` is also **Whitelisted** (the rule applies everywhere).

Searching Configured Rules

Use the **search bar** at the top of the dialog to filter rules that have already been configured. Type a file type name or extension (e.g., "PDF", "Image", "JPEG") and the list will filter to show only matching rules.

Error section



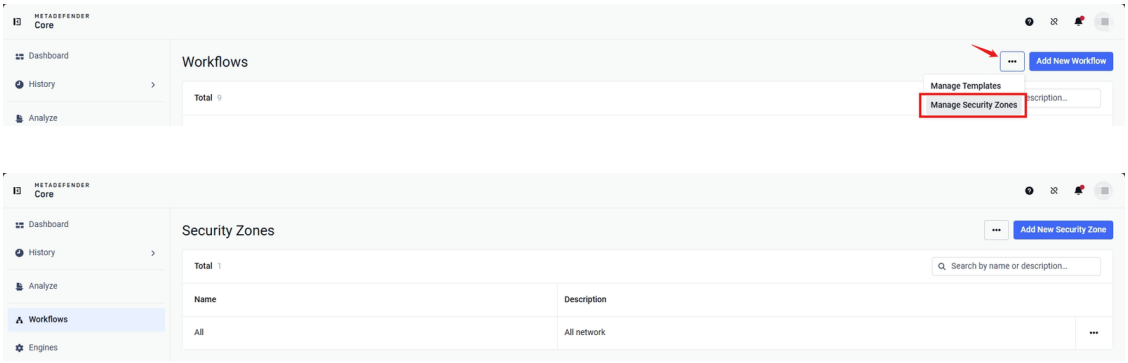
The Errors section will automatically disappear once all issues have been resolved.

Clicking the **"Show All Errors"** button will filter the list to display only rows containing errors. In this view, the Blocklist Condition field, Add Rule option, search field, and Save button will be disabled. The button label will also change to **"Show All Rules."**

Clicking **"Show All Rules"** will restore the full list and re-enable all previously disabled controls, returning the interface to its normal state.

Security zone configuration

The Security zone page is found under **Workflows > Manage Security Zones** after successful login.



The following actions are available:

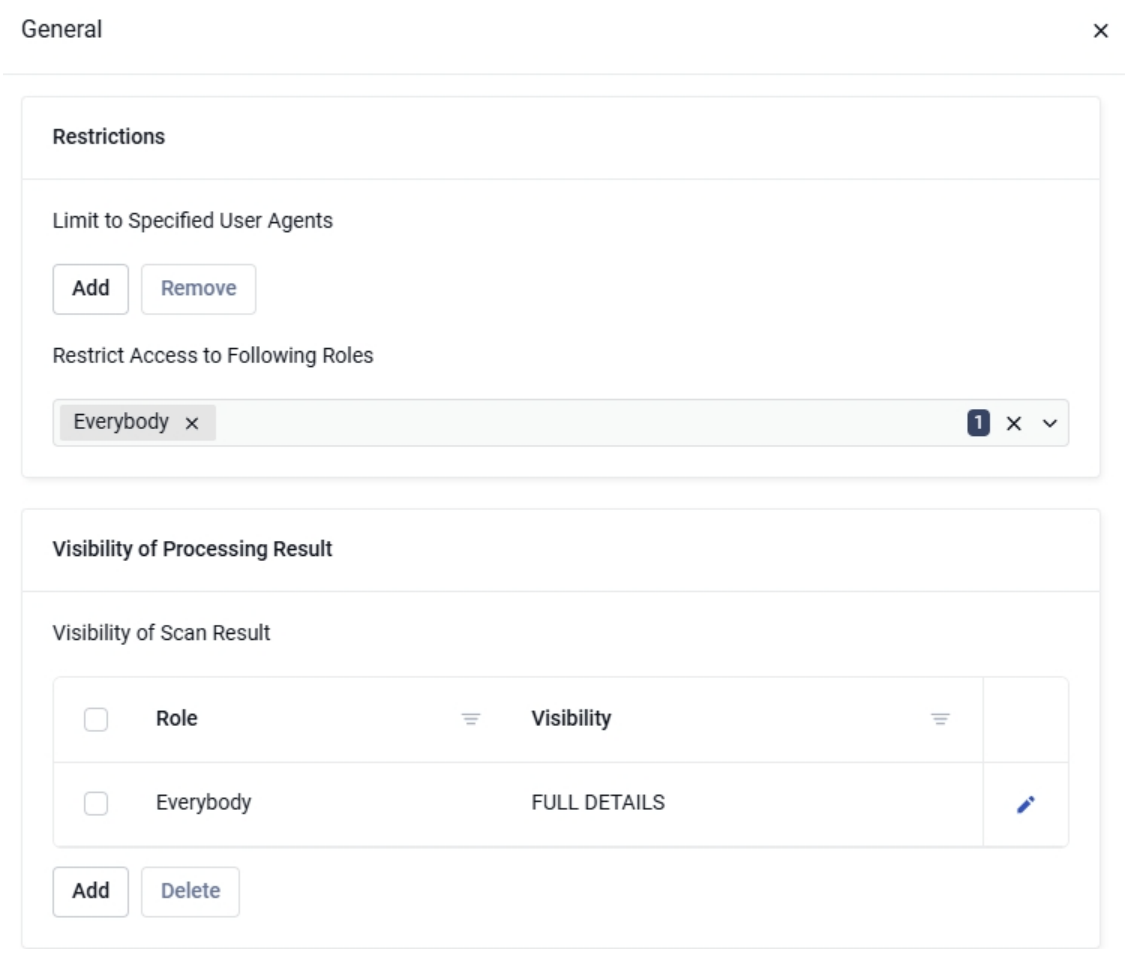
- new zones can be added
- existing zones can be viewed
- existing zones can be modified
- existing zones can be deleted

Each zone contains a name, description and multiple network masks. Both IPv4 and IPv6 network zones are supported Workflow Configuration Template

Workflow rule configuration

The Workflow rule page is found under **Workflow > Manage Workflows** after successful login.

The rules represent different processing profiles.



The following actions are available:

- new rules can be added
- existing rules can be viewed
- existing rules can be modified
- existing rules can be deleted

Rules combine workflow templates and security zones and describe which workflows are available in a specified security zone. Multiple rules can be added for the same security zone.

Restriction

Once clicking on a rule, a window pops up where beside the rule properties all the chosen rule's options are shown on the different tabs.

On this page it is possible to enable local file scanning by checking the **ALLOW SCAN FILES ON SERVER** checkbox.

By enabling this feature, a local scan node is able to scan the files at their original place if the files' location is allowed in the list below the checkbox. For example if this list has *C:\data* in it, then all files under that folder (e.g.: *C:\data\not_scanned\JPG_213134.jpg*) are allowed to be scanned locally if it is choosen . For testing this local scan feature, please note:

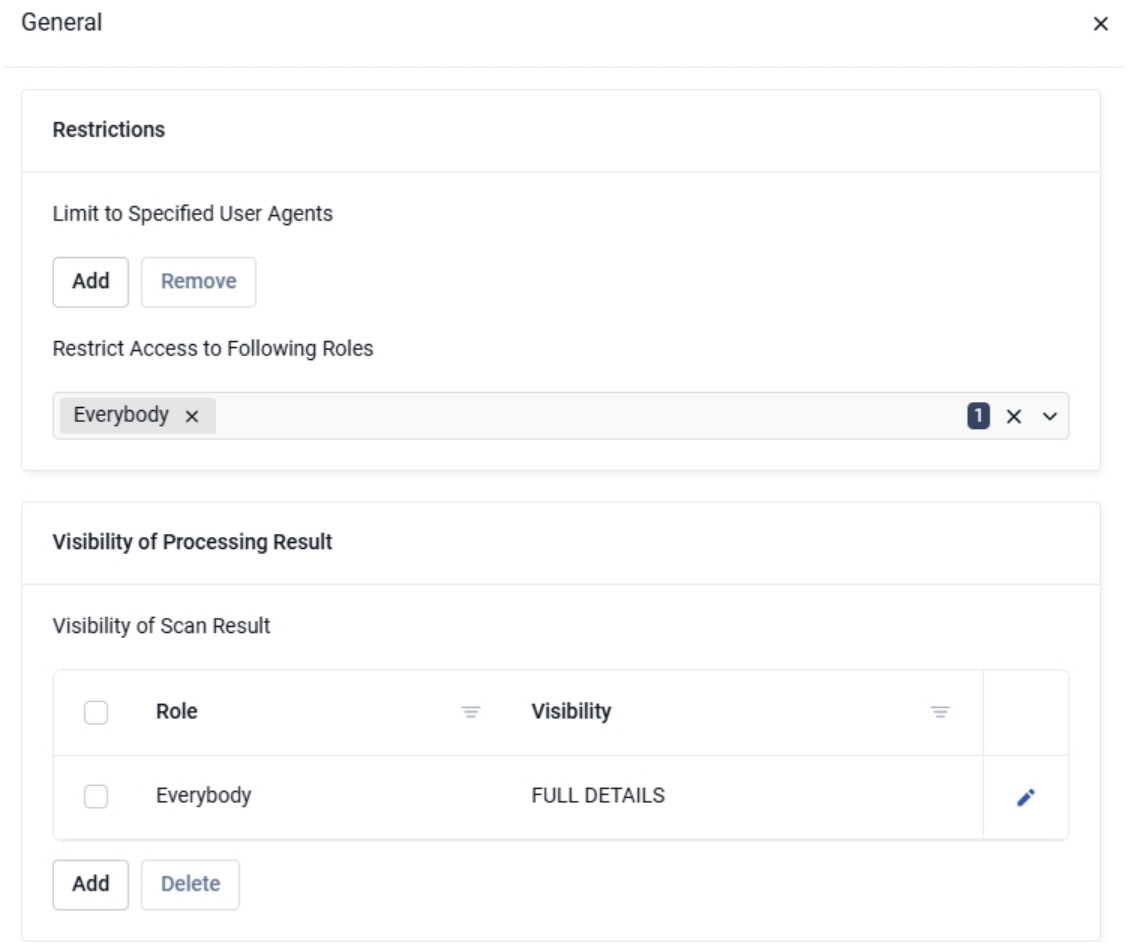
- Need to use "filepath" header while submitting a file via REST, see more at POST File Analysis.
- MetaDefender Core's web scan (localhost:8008) is not an applicable tool to test because it does not allow you to

customize the scan request's header.

Various accessibility options can be set on this page. You can define one of three visibility levels for the scan results for each role in the **VISIBILITY OF PROCESSING RESULT** field:

- **Full details**: all information for a scan is displayed.
- **Per engine result**: Scan details are displayed except per engine scan time and definition date.
- **Overall result only**: Only the overall verdict is displayed.

Processing result report will be generated dynamically based on the defined Role and Visibility settings.



There are also two special roles - **Every authenticated** refers to any logged in user, while **Everybody** refers to any user. Without belonging to any role specified within the rule, the user has no access to view the scan results. The usage of the rule to given roles can also be restricted with the **RESTRICT ACCESS TO FOLLOWING ROLES** field.

Clicking on a tab it is possible to overwrite a property that was previously defined inside the workflow template.

An option if changed will only overwrite the specific property for the underlying rule and makes no modification on the original workflow template that was choosen by the rule.

This means that several rules can be created using the same workflow template overwriting different options while the untouched properties will remain as they were set in the workflow template.

Rules are processed in order, the first matching rule will be used for the request. You can change order of rules via drag&drop in the Web Management Console. If there is no rule that matches for the client (source IP address), then

the scan request will be denied.

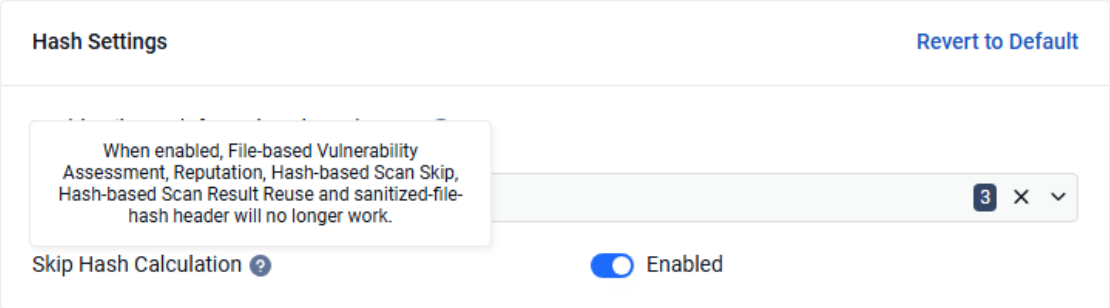
Skip hash calculation

Info

"SKIP HASH CALCULATION" setting could be enabled under "General" tab on each desired workflow rule to make this feature triggered (disabled by default).

When enabled, expecting the following features will no longer work:

- File-based Vulnerability Assessment feature since the Vulnerability engine requires file hash value as an input.
- Reputation feature since this engine requires file hash value as an input.
- "sanitized-file-hash" header since this contains SHA256 hash value of sanitized file.
- Scan result reuse for the same hash feature since SHA256 hash value is required.



Support an option to skip hash calculation on every processing file regardless it is individual file or a part of archive.

Expecting to be used against big file processings to significantly reduce overall processing time.

In JSON scan result,

- it reveals empty values at md5, sha1, sha256, sha512.
- is_skip___hash key will be set to true should be used as a flag for client integration.

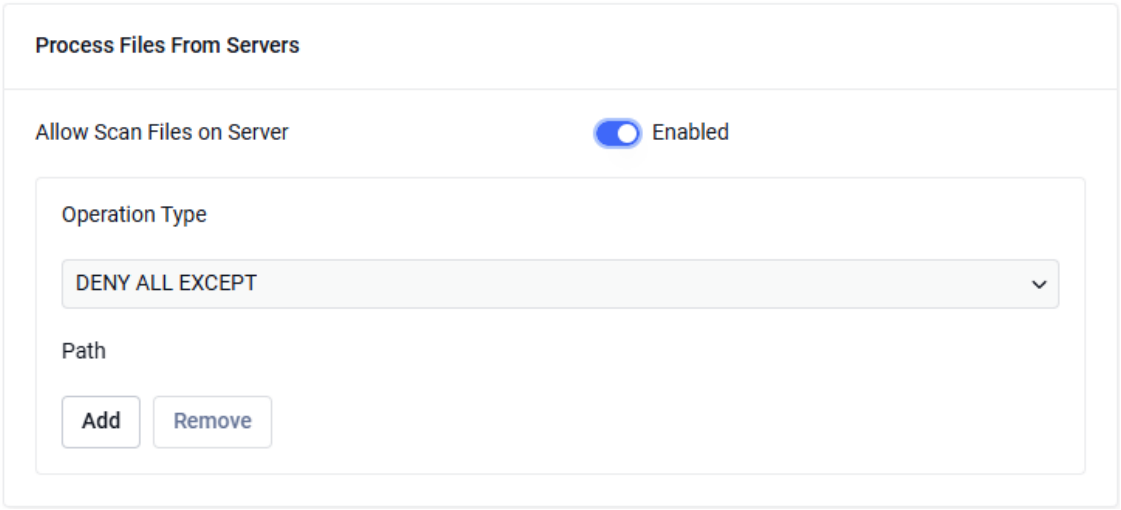
Local scan enablement

By default, MetaDefender Core will always require client to upload payload data over HTTP(S) for processing, which we call it as "remote scan" with REST interface supported.

However, if the file resides on the same machine with MetaDefender Core, or resides on a local networked drive where MetaDefender Core could access and read without any authentication required, the product will support clients to specify absolute file path for processing, no need for payload uploaded in HTTP(S) body then.

Info

"ALLOW SCAN FILES ON SERVER" option could be enabled under desired workflow rule - General tab



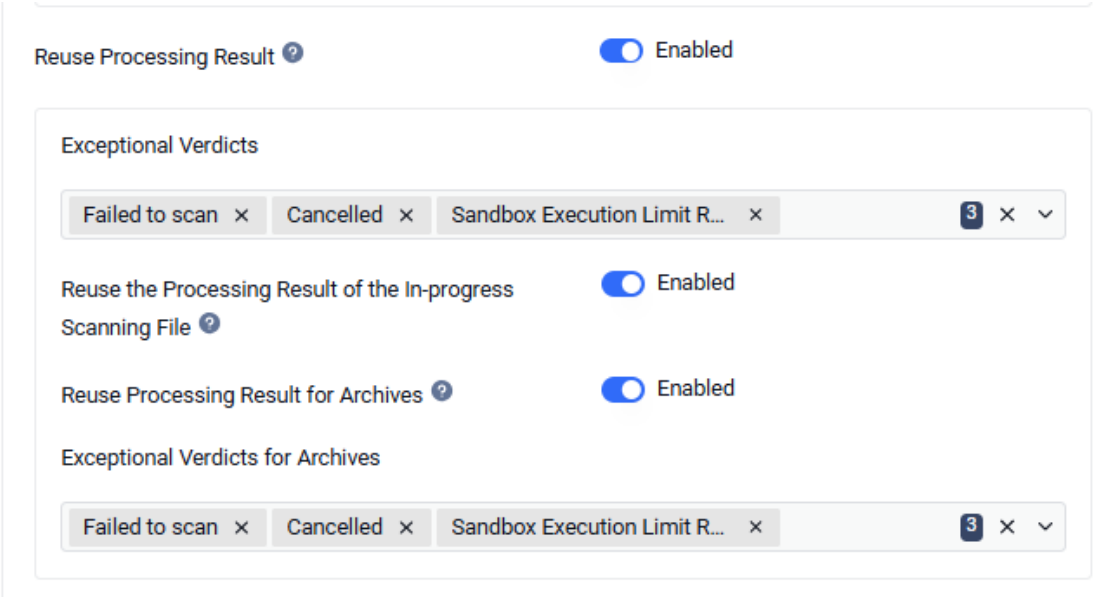
Info

Starting with MetaDefender Core 4.19.1, the path under above setting could accept UNC path (besides local folder path)

From client side, expecting to use filepath header under HTTP(S) file request - POST File Analysis

Reuse scan results for the same hash

The new setting under "General" section in MetaDefender Core workflow rule (disabled by default) will let authorized users configure the product to automatically reuse eligible processed results with the same hash for all other in-progress submitted requests.



MetaDefender Core customers can also specify to skip using this feature, and instead start a new processing flow as usual, on certain prior scan verdicts e.g. Failed to scan.

For now, MetaDefender Core will consider a scan request eligible for reusing the results of an earlier same-hash request when it meets all the following conditions:

- The setting "Reuse processing result" is enabled.
- The setting "Skip hash calculation" is disabled.
- The user of the earlier scan request is anonymous or the same as the current user.
- The workflow rule used for the earlier scan request must be the same as that of the current one.
- The list of verdicts that are configured on the new workflow rule must not contain the verdict of the earlier scan request.
- Request headers:
  - `metadata` header must be the same as that of the earlier scan request.
  - `engines-metadata` header must be the same as that of the earlier scan request.
- **No** engines or their databases have been updated since the earlier scan request was initiated until now.
- **No** engine status or engine configurations have been changed since the earlier scan request was initiated until now.
- **No** external scanners have been modified since the earlier scan request was initiated until now.
- **No** workflow configurations have been modified since the earlier scan request was initiated until now.
- If the earlier scan request includes a Deep CDR sanitized file, it must be present in the database as well as on disk.
- If the earlier scan request includes a Proactive DLP processed file, it must be present in both the database and on disk.
- If the "Discard processed files when the original file is blocked" feature is enabled, and processed files from Deep CDR or Proactive DLP are discarded, the conditions for reusing the scan result are **not** met.
- The lists of possible file types detected by file type engine in the earlier and current requests are the same.

By default, reuse applies to individual files. To reuse results for archives as well, enable **Reuse Processing Result for Archives**. This can significantly speed up repeated submissions of the same archive.

- When **Reuse Processing Result for Archives** is enabled, you can separately choose which prior verdicts should never be reused for archives.
- Reuse also works **inside** other archives: if an archive you submit contains a nested archive that was already processed, the nested archive's result is reused while the rest of the outer archive is processed normally.
- Aside from the exceptional verdicts and all the conditions above, the table below shows when an archive is re-scanned:

Re-scanned Condition	Why the archive is re-scanned
The archive, or any file inside it, is **password-protected / encrypted**.	Currently a stored result can't re-validate the password supplied with the new request, so the archive must be extracted and scanned again.
A stored archive result that exceeded the **expansion-factor** limit is encountered **nested inside another**	The expansion-factor allowance is based on the top-level file's size, so the result may differ at a different

Re-scanned Condition	Why the archive is re-scanned
archive.	position.
The archive **extraction timed out** on the earlier scan.	An extraction timeout depends on system load at the time and may not happen again, so the result is treated as non-repeatable and the archive is re-scanned.
A **skip-by-hash (allow/block list) entry** that affected a file inside the archive has **changed or expired.**	The stored decision for that file is now out of date, so the archive is re-scanned to apply the current allow/block list.

Warning

Regarding to external scanner and post action, MetaDefender Core cannot detect any modification inside the scanner and/or script running the post action, so customers are advised to remove the old scanner/script and add a new one if scanner/script gets changed instead of keeping the old name for new scanner/script.

FileType, OPSWAT Alin AI Deflection, Country of Origin and File Structure Validation always run on the newly submitted file, because they are evaluated before MetaDefender Core decides whether an earlier result can be reused. Every other engine result is taken from that earlier request.

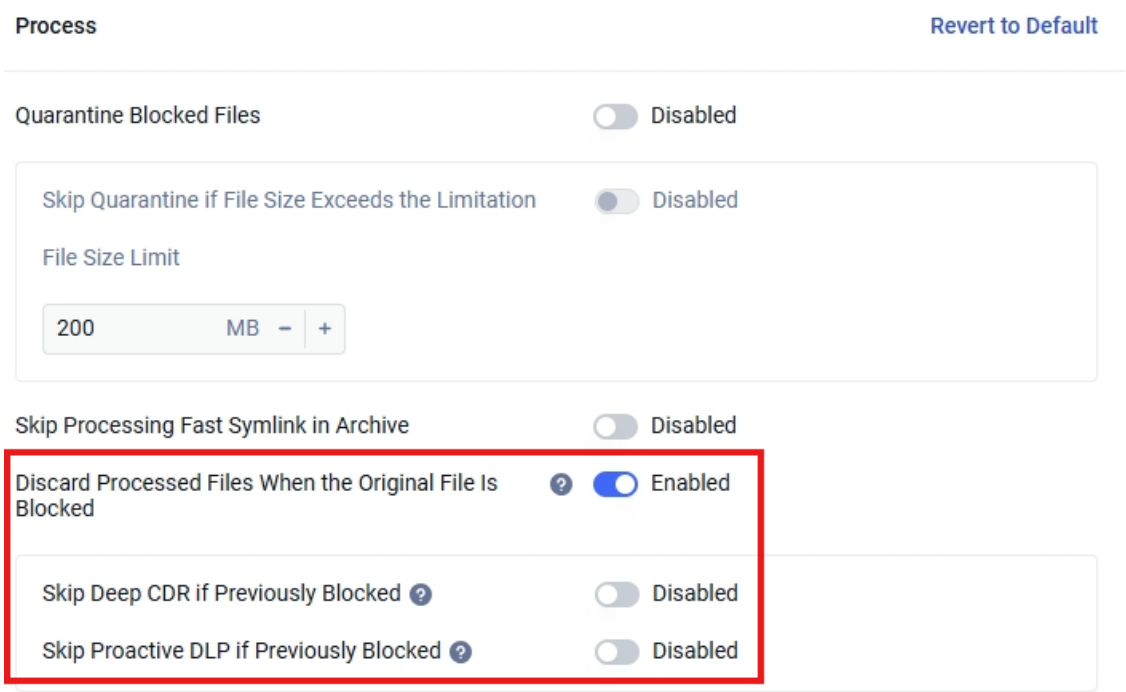
Besides that, MetaDefender Core customers have an option to let the product wait for other in-progress request (with the same hash), until it finishes and then automatically resuse to apply the final result on those waiting requests. **This behavior does not apply to archive files.**

If the current request satisfies all conditions to reuse the processing result of an in-progress file, the current scan request will wait until:

- The file completing its processing will wake all other requests waiting for its result. If the current request reuses the processing result successfully, it will wake all others and so on.
- Reaching the wait timeout (5 minutes, cannot adjust for now), do rescan the file.

Discard processed files when the original file is blocked

This setting, located under the 'General' section in the MetaDefender Core workflow rule (disabled by default), allows you to discard processed files generated by Deep CDR or Proactive DLP if the original file is blocked.



In addition, there are two sub-options: 'Skip Deep CDR if previously blocked' and 'Skip Proactive DLP if previously blocked.' These options allow you to decide whether to **not** process the original file with Deep CDR or Proactive DLP if it was previously blocked by prior processes. These options can help improve performance by skipping Deep CDR or Proactive DLP processing in certain cases. However, if you still need the processing results from Deep CDR or Proactive DLP for specific purposes, it is recommended **not** to enable these options.

Note that these sub-options do not apply to the sanitization of archive files processed by the Archive Compression engine.

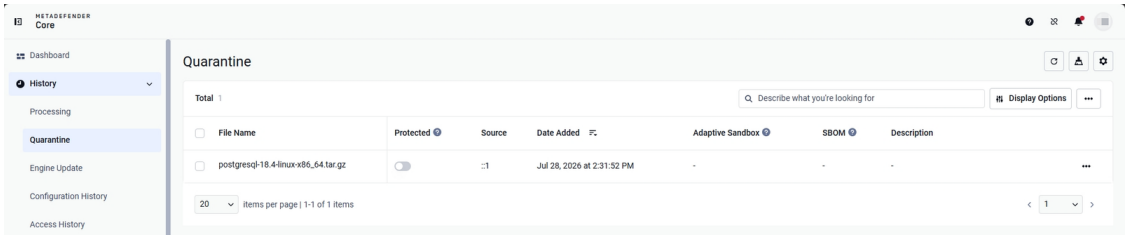
In cases where a processed file is discarded, the scan results of the original file will not be available for 'Reuse scan result', as mentioned earlier.

Quarantine

Options

The quarantine is for keeping blocked files in a separated place. It can be used by configuring workflows (see Advanced configuration for the Workflow Template).

On the **Quarantine** page (**History** ▯ **Quarantine**), the following operations can be performed on the quarantined files:



1. By clicking on the ... item, details appear.

2. Pinned files won't be removed on clean-ups. Use the pin icon to do so.

3. For removing the files from the list, please use the bin icon.

4. Files can be downloaded by clicking the download icon.

The *Check with Adaptive Sandbox, Check with SBOM,* the *Pin*, *Unpin* and *Delete* operations can also be performed in bulk using the check boxes before the filenames and clicking the action icons above the file list.

Check with Adaptive Sandbox

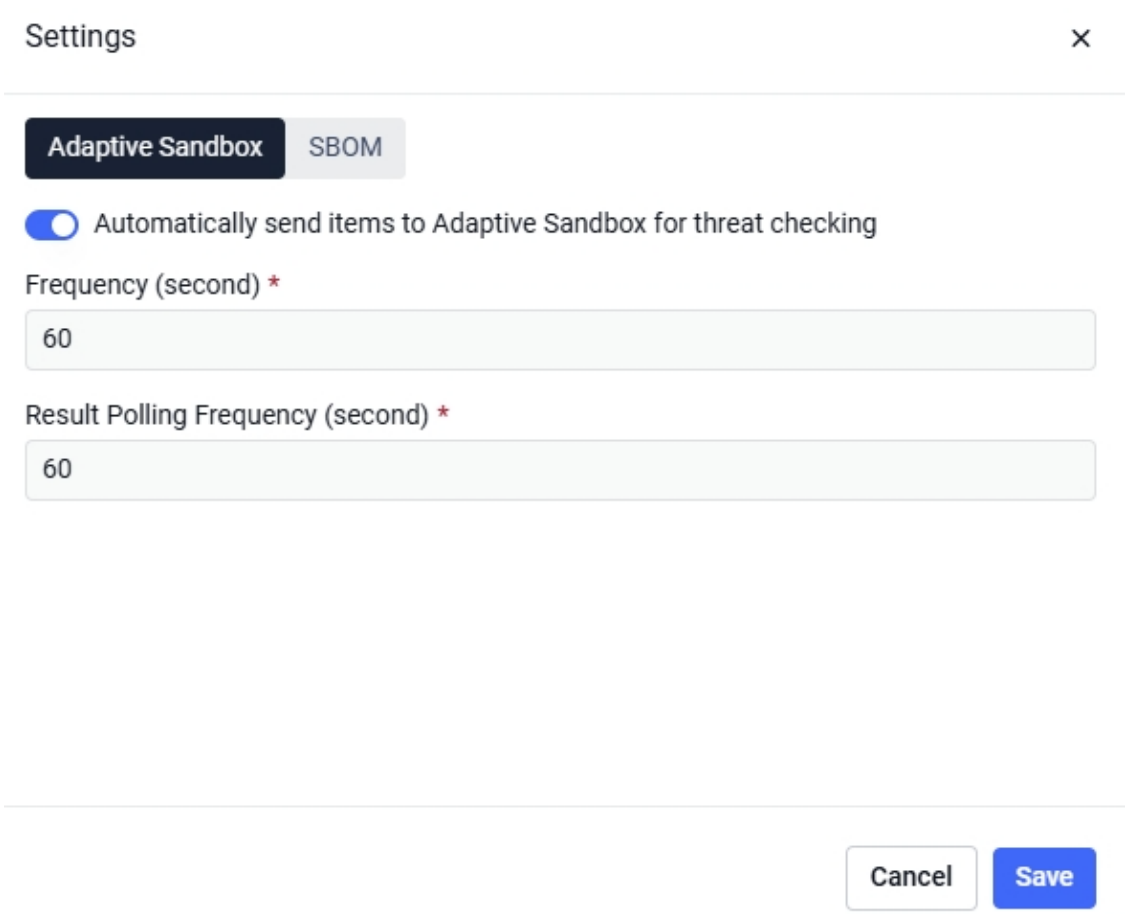
Files in the quarantine can be uploaded to Adaptive Sandbox to retrieve deep analysis result.

This feature requires the **Adaptive Sandbox** technology to be licensed, and enabled in workflow.

Quarantine items may be sent to Adaptive Sandbox:

1. Manually select the file and click on the `Check with Adaptive Sandbox`button, or

2. Automatically, driven by the configuration under Quarantine settings



Check with SBOM

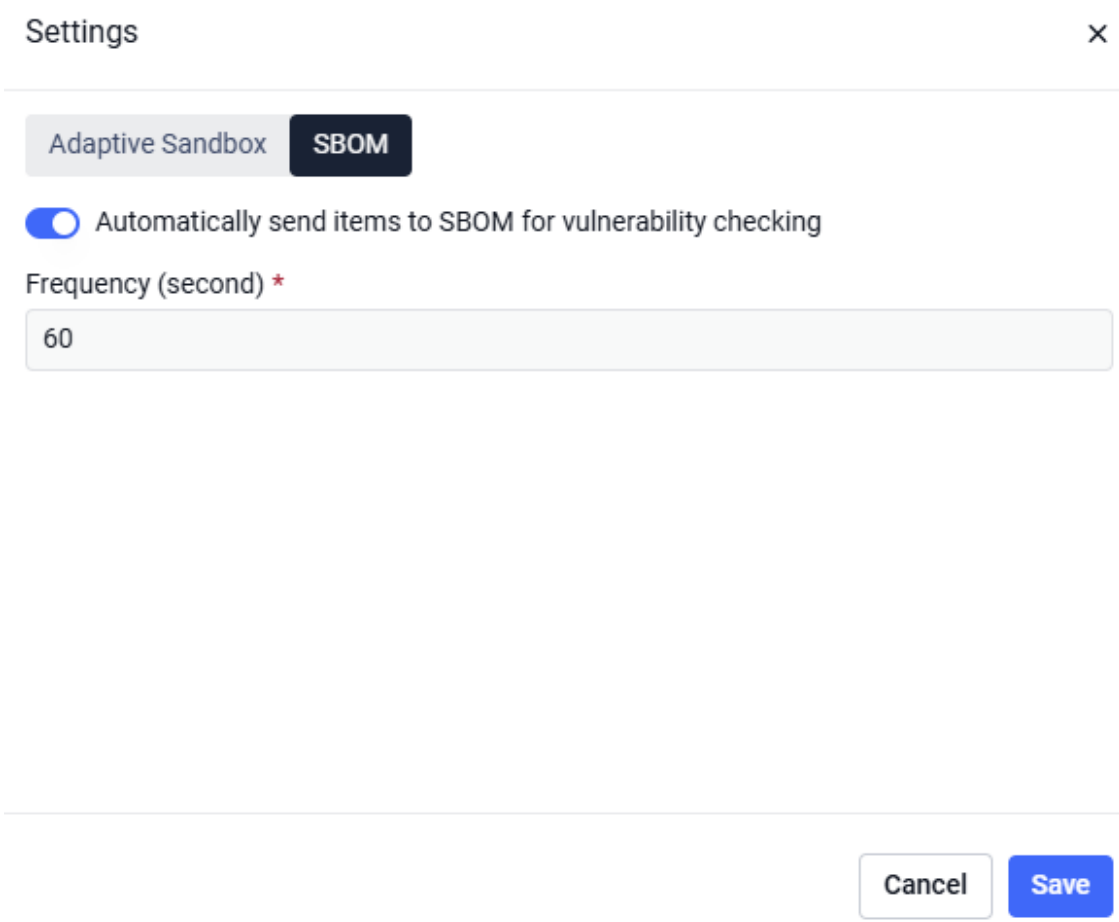
Files in the quarantine can be uploaded to SBOM engine to retrieve deep analysis result.

This feature requires the **SBOM** technology to be licensed, and enabled in workflow.

Quarantine items may be sent to SBOM:

1. Manually select the file and click on the `Check with SBOM`button, or

2. Automatically, driven by the configuration under Quarantine settings



MetaDefender Core Performance Tuning Guide

The performance fine tuning guide describes how to configure the MetaDefender Core application and build the appropriate computing server to gain optimal file processing performance.

This document is intended for system administrators.

Performance metrics

While processing files on MetaDefender Core, service performance is measured by various metrics. Some of them are commonly used to define performance levels, including:

Performance metrics	Description
Throughput (number of processed objects per hour vs. number of processed files per hour)	On MetaDefender Core, meaning of “files” and “objects” are not the same. • “files”: exclusively refers to original files submitted to MetaDefender Core. These could be either archive or non-archive file formats. For archives, depending on

Performance metrics	Description
	archive handling settings, MetaDefender Core may need to extract them and process all nested files inside as well. For example, one archive file could contain millions of nested files inside. • "objects": refers to any individual files that MetaDefender Core must process. These could be separate original files submitted to MetaDefender Core, or extracted files coming from an archive. The number of processed objects is considered to be a more accurate throughput metric to measure MetaDefender Core performance. The primary metric used to measure average vs peak throughput of a MetaDefender Core system is "processed objects per hour."
Submission load (number of successful requests per second)	This performance metric measures the load generated by a test client application that simulates loads submitted to MetaDefender Core. A submission is considered successful when the client app submits a file to MetaDefender Core and receives a dataID, which indicates that the file has successfully been added to the Queue. Submission load should measure both average and peak loads.
Average processing time per object	The primary metric used to measure processing time of a MetaDefender Core system is "avg processing time (seconds/object)."
Total processing time (against certain data set)	Total processing time is a typical performance metric to measure the time it takes to complete the processing of a whole dataset.

Factors impacting MetaDefender Core performance

MetaDefender Core file processing performance can be influenced by a variety of factors related to software, hardware, and the operating system. Here are the key factors and the guidelines:

Hardware factors and guidelines

It is recommended to use a standard benchmarking tool such as Geekbench or PassMark to check for any

hardware-related performance issues.

These are a few key hardware components that could impact MetaDefender Core and system performance:

CPU

OPSWAT performance testing is based on cloud (AWS instances) and on-premises (Dell servers) systems and recommends CPUs with a Geekbench score of >1050 for single core performance and >3700 for multi-core performance (Geekbench).

The number of vCPU cores needed depends on the AV engine package and other OPSWAT engines in use (e.g. Deep CDR, Proactive DLP). Customers should review our package efficacy report to determine the appropriate engine package.

The minimum requirement is 8 vCPUs for Metascan only, and 12 vCPUs if both Metascan and Deep CDR technologies are needed. Please refer to our documentation on recommended system configurations for more information.

RAM

Sufficient memory is essential for handling large files, multiple concurrent scans, or a large number of engines. Insufficient RAM can lead to swapping and slowdowns. DDR4 or DDR5 RAM is recommended for optimal speeds. The minimum memory requirement is at least 16GB of RAM to support Metascan and Deep CDR. Please refer to our documentation on recommended system configurations for more information.

Storage

Storage speeds affect how quickly files are read and written during the scanning process. This factor is even more important when processing archive files on MetaDefender Core since it involves a lot of archive extraction tasks to create and write nested files.

High-speed storage (M2, SSD with >100K IOPS) is recommended. For virtualized environments, RAID 1 is recommended as it has a faster IO write speed than RAID 5.

The minimum storage requirement is at least 70GB of free space. Please see recommended storage requirements for more information.

Network

When connected to a client such as MetaDefender Kiosk for network scanning, the speed and latency of the network connection can impact performance, especially if large files are being transferred. A minimum of 1 Gbps (125 MB/sec) is required and 5Gbps is highly recommended to avoid potential network-related issues while downloading files.

Other hardware settings

For on-premises systems, BIOS firmware settings related to power management, processor, memory and disk settings could also vastly impact overall system performance. BIOS settings should be configured for maximum performance (avoiding any power optimized settings).

Operating system guidelines

OS Version

The version and architecture (32-bit or 64-bit) of the operating system can impact performance, with newer versions and 64-bit architectures typically offering better performance and support for more memory.

These are the recommended operating systems:

- Windows: Windows Server 2022, 64-bit
- Linux: Linux Red Hat Enterprise 9, Debian 11 or Ubuntu 22.04

Background Processes

The number of background processes and services running on the operating system can consume resources and impact scanning performance. Any nonessential background processes should be suspended.

File System Type

Different file systems (e.g. NTFS, FAT32, ext4) have varying performance characteristics that can affect how quickly files are accessed and scanned. We recommend NTFS on Windows and ext4 on Linux.

Windows Power profile are a collection of hardware and system settings, including processor power management, that regulate performance and power consumption. “Balanced” or “Power Saver” profiles can throttle CPU processing and can significantly reduce performance. Your system power profile should be set to “Performance” settings (e.g. “Best Performance”, “High Performance, “Ultimate Performance”).

Software factors

MetaDefender Core version: It is always recommended to use the latest MetaDefender Core version for the best performance and to avoid running into legacy stability issues while operating.

Engines: The number of engines configured to process files can greatly impact performance. More engines provide better security but could slow down the scanning process.

MetaDefender Core configuration: Scan settings, such as deep scans, heuristic analysis, and archive scanning, can affect performance. More thorough scans require more resources and time.

File characteristics (file type, file size): The types of files being scanned (e.g., executables, archives, multimedia files) can impact performance since some file types are more complex than others.

Concurrent Scans: The number of simultaneous scans being performed can impact overall performance, as resources are divided among the concurrent tasks.

Integration Method: How MetaDefender Core is integrated into other applications (e.g. via APIs) can also influence performance, especially if the integration is not optimized.

Performance Tuning

Understanding requirements

MetaDefender Core and its engines utilize system resources as much as possible to boost throughput while not exceeding a pre-defined threshold (e.g. 90-100%) continuously for extended periods to maintain system stability.

Prior to making any performance tuning, the following considerations need to be made:

- Expected outcome (i.e. a baseline for performance)
  - Throughput rate (processed objects/hour)
  - Avg. processing time (seconds/object)
- Use case (Multiscanning, Deep CDR, Adaptive Sandbox, etc) and package (MetaDefender Core 8, MetaDefender Core 16, etc)

- Environment (virtually/physically hosted on-premises or in the cloud) and deployment (native installation, docker, Kubernetes, AMI)
  - Number of vCPUs and CPU speed
  - Memory
  - Free storage space
  - Additional factors such as power profile, BIOS settings

System, Application, Database Tuning

Performance tuning can be broken down into three components:

1. System and Environment Tuning
2. MetaDefender Core Application Tuning
3. MetaDefender Core Database Tuning

To identify potential bottlenecks in specific circumstances, it is important to understand the problem (i.e. system resource issue, application bottleneck) to determine the best course of action.

System and Environment Tuning

Please see system environment recommendations below:

Environment	Use Case	CPU Cores	Memory	Free Disk Space*	Additional Environment tuning
Windows - Multi-scanning	MetaDefender Core 8	8	8 GB	>166 GB	Network speed: >1Gbps BIOS settings: max performance Operating System: Windows Server 2022, 64-bit File system: NTFS Windows Power Profile: Best/High/ Ultimate performance

Environment	Use Case	CPU Cores	Memory	Free Disk Space*	Additional Environment tuning
	MetaDefender Core 12	16	16 GB	>174 GB	
	MetaDefender Core 16	16	16 GB	>182 GB	
	MetaDefender Core 20	32	16 GB	>190GB	
	MetaDefender Core MAX	32	32 GB	>270GB	
Linux-multi-scanning	MetaDefender Core 5	4	4 GB	>160 GB	Network speed: >1Gbps BIOS settings: Max Performance Operating Systems: Linux Red Hat Enterprise 9, Debian 11 or Ubuntu 20.04 File system: ext4
	MetaDefender Core 10	8	8 GB	>170 GB	
	MetaDefender Core MAX	16	16 GB	>190 GB	
CDR and DLP	Deep CDR and Proactive DLP	+4	+8 GB		There are additional vCPU and memory requirements for different configurations.

Environment	Use Case	CPU Cores	Memory	Free Disk Space*	Additional Environment tuning
					For example, MetaDefender Core 8 + CDR + DLP requires 12 vCPUs and 16GB RAM.
Embedded Sandbox engine	<5000 scans/ day	+4	+4 GB		
	10000 scans/ day	+8	+8 GB		
	25000 scans/ day	+16	+16 GB		

vCPUs*: CPU speed with a Geekbench score >1050 for single core performance and >3700 for multi-core performance.

Free Disk Space*: Free storage space requirements depend on the types of files, use cases, and throughput. See storage requirements documentation for more information.

MetaDefender Core Application Tuning

MetaDefender Core Tuning Process

Like any other high-performance applications, MetaDefender Core performance tuning involves a systematic approach to identifying and resolving performance bottlenecks. Here is a typical procedure for tuning the performance of an application:

1. **Define performance goals:**
    1. Identify key performance indicators (KPIs) with metrics such as throughput, total processing time, and resource utilization thresholds.
    2. Set performance targets: establish acceptable performance thresholds for KPIs.
2. **Define constraints:** Understand targeted environments, use cases, and application and system constraints. Understand what can and can't be changed. Understand what can be changed and what can't be.
3. **Establish a baseline:** measure, capture, and document performance levels using monitoring tools.
4. **Identify bottlenecks and propose remediation plan:**
    1. Analyze logs to identify culprit(s) causing performance bottlenecks, while monitoring resource utilization (CPU,

memory, disk, network) to identify resource constraints.

2. Determine the impact of each identified issue on overall performance, and then focus on issues with the highest impact on performance. This issue remediation may also require tuning MetaDefender Core application settings.

5. **Design test case and perform load testing:** Ensure all KPIs are measured, captured, and logged for analysis.
6. **Review and repeat:** Analyze captured performance test results, identify issues, continuously iterate and calibrate application settings, and re-run performance load tests until the performance target is achieved.

MetaDefender Core Tuning Guide

Setting Name and how to set	Description	Default value	Recommended settings and when to change
MetaDefender Core temporary folder Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global\tempdirectory` Linux: File `/etc/ometascan/ometascan.conf` => `tempdirectory`	Location where all temporary files (e.g. during extraction) are written into and read from, crucial for IO read and write.	Inside installation folder	When large number of temp files are written (e.g. processing large archive files) Temp folder should be kept in high speed storage such as M2, SSD or RAM disk.
MetaDefender Core queue size Use MetaDefender Core v5 via REST API to update: https://docs.opswat.com/mdcore/knowledge-base/how-can-i-configure-the-maximum-queue-size-in-metadefender-core-	All submitted scan requests, when accepted, are designated as a task slot sitting in MetaDefender Core queue. For archive file submissions, MetaDefender Core needs to extract archive files to nested files for individual and separate scanning. Each nested file scanning task will be tracked as a separate slot in MetaDefender Core queue.	500	In high throughput use cases and when system has >32 vCPU cores and CPU utilization is low. Increase queue size incrementally until desired throughput/ speed/utilization is achieved. When processing large archive files and when system has >32 vCPU cores with many small, nested files. Increase queue size to 5000-10000 depending on size of archive files.

Setting Name and how to set	Description	Default value	Recommended settings and when to change
Skip hash calculation Workflow -> General -> Process -> “Skip hash calculation”	MetaDefender Core will calculate hash checksum against every submitted file for file integrity validated by users when needed after scanning. Hash calculation might require significant CPU resources and add significant overhead to overall processing time, especially for large files.	Skip hash calculation = <<disabled>>	When content of files doesn’t change often in high throughput environments Skip hash calculation = <<enabled>>
Max file sizes setting Workflow -> General -> Max sizes -> File scan	Larger files lead to longer processing time for temporary file creation and access, as well as file hash calculation. Large files are usually in archive compressed file format, which require MetaDefender Core and its engines to extract and process nested files within.	200 MB	In environments with limited resources, with large file processing causing performance bottlenecks Set max file to limit processing large files (e.g. 1000MB)
Reuse processing result (i.e. cache processing results) Workflow -> General -> Process -> Reuse processing result	MetaDefender Core is supposed to process any files regardless of how many times the same files are re-submitted. There’s an available setting under MetaDefender Core workflow to allow MetaDefender Core to reuse processing results on the same file hash when meeting certain conditions	disabled	In use cases including email security, supply chain and cross domain most likely to reuse the same files (or even nested in different archive files) submitted repeatedly to MetaDefender Core for scanning Set value to enabled.

Setting Name and how to set	Description	Default value	Recommended settings and when to change
	(configurations and engines are not changed from the last result)		
Engine specific handling limits Workflow -> Archive -> Max recursion level, max number files, max total size Workflow -> Deep CDR -> Advanced -> specificFiletype -> Recursive Level	For example, archive handling limits (max recursive level, max total extracted files, max total size of extracted files) will tremendously impact overall file processing. A higher limit means MetaDefender Core needs to dig deeper to process them.	depends on module	When resource utilization is very high (i.e. 100% CPU utilization) and files contain many archived files. Incrementally reduce level and observe resource utilization.
Engine parallel count Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\internal\parallelcount` Linux: File `/etc/ometascan/ometascan.conf` => `parallelcount`	MetaDefender Core needs to open a certain number of threads on the system to communicate and send tasks to each engine underneath. The max number of threads to be allowed is defined as the parallel count setting on each engine, and this limit can be configured differently for each engine. This limit can be increased to push more tasks to engines when applicable. However, increasing this limit does not always mean higher throughput produced by engines. Too many threads opened might overload engines.	parallelcount=20 parallelcount_7z=-1	When observed throughput is low, the system has >32 vCPU cores, and resource utilization is low. Incrementally increase parallel count and observe throughput. When scanning large archive files, increase Archive extraction engine parallel count to 15. When MetaDefender Core uses remote sandbox, lower parallel count to 5. Note: parallel_count should not exceed 50

Setting Name and how to set	Description	Default value	Recommended settings and when to change
	We recommend lowering the parallel count for the sandbox engine when MetaDefender Core uses remote sandbox.		
Engine scan timeout Workflow -> Metascan -> Timeouts	MetaDefender Core controls engine processing timeouts according to set configurations. By default, 1 minute is allocated for each Metascan AV engine, and 10 minutes for the entire Metacan AV scanning stage. Under normal conditions, Metascan AV engines only take milliseconds to around 30 seconds to scan a file. Lowering this timeout threshold to seconds will help MetaDefender Core move on quickly without waiting on any unexpectedly slow scanning from any engine. At the same time, to ensure Metascan AV scanning outcome quality, consider setting Metascan AV failure threshold to reject scans where a set number of Metascan AV engines times out (fails).	Metascan • Per AV engine scan=1 min • Global scan=10 min	In high throughput use cases, set AV engine scan timeout: • Per AV engine scan = 5 seconds • Global scan = 10 seconds
Processed (sanitized)	Similar to temporary file	Inside installation folder	Recommended to use

Setting Name and how to set	Description	Default value	Recommended settings and when to change
file storage Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global\sanitizepath` Linux: File `/etc/ometascan/ometascan.conf`=> `sanitizepath`	storage, when enabled with our advanced threat prevention, Deep CDR, MetaDefender Core will write and store sanitized output files in a designated location, which is set to the inside of the MetaDefender Core folder by default.		high speed local disk for sanitized files storage for optimal processing. Temp folder should be set to a high-speed disk such as M2, SSD or RAM disk.
Max number of database connections Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\internal\db_connection` Linux: File `/etc/ometascan/ometascan.conf` => `db_connection`	MetaDefender Core opens connections to communicate and work with PostgreSQL database server. By default, 10 connections are opened, adjustable via configuration.	10	In a high throughput environment when database utilization is low. Increasing this limit could help in case of performance bottlenecks due to slow database processing.
Logging channels Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger` loglevel wineventlog_level syslog_level Linux: File `/etc/ometascan/ometascan.conf` loglevel syslog_level	MetaDefender Core supports various channels for operational logging including product own log files, syslog, Windows event log.	loglevel=info	Unless required, disabling unsupervised logging channels (e.g. Windows event log) should help increase overall performance.

MetaDefender Core Database Tuning

MetaDefender Core uses PostgreSQL relational database to store application settings and processing results. Most product transactions on MetaDefender Core require a database query and update (e.g. existing scan result lookup,

new scan result insert). It's crucial to keep the product database well-maintained and robust for the most optimal scanning throughput, and to achieve the best user experience for overall product performance.

The following measures should be implemented for database tuning:

- Database deployment model
- Hardware considerations
- PostgreSQL configurations
- MetaDefender Core configurations
- Routine database maintenance
- Monitoring and calibration

Database deployment model

There are two main modes on MetaDefender Core deciding how PostgreSQL database is leveraged:

- Persistent mode (Stateful): MetaDefender Core will store any scan results into PostgreSQL database for further lookup.
- Non-persistent mode (Stateless): MetaDefender Core will not store any scan results into PostgreSQL database. For every scan request transaction, it returns scan results back to the client side via webhook callback and finalizes transactions without storing anything in database. There is no need to consider performing database tuning in this mode.

In the persistent mode, depending on business requirements, users are supported with two different database deployment models:

- Standalone database: Each MetaDefender Core instance has its own PostgreSQL database to operate with, and when deploying multiple MetaDefender Core instances using the standalone database model, those PostgreSQL database servers are isolated and not connected to each other.
- Shared database: MetaDefender Core instances will share the same PostgreSQL database server for configurations and historic scan results. This ensures that all MetaDefender Core instances will always use the same configurations and have full access to other MetaDefender Core instances' scan histories. This provides a consolidated data warehouse for client query and a consolidated view for IT administration.

Consider these pros and cons when choosing shared database for usability over performance:

	Pros	Cons	When to consider
Standalone database	• Optimal performance on database side. • Quick and simple deployment. • Reduce cost of deployment maintenance with	• Less transparent backend infrastructure for client integrations. • Requires additional setup for centralized scan data repository.	• Scanning performance is the top priority. • Client-side integration is flexible (able to setup sticky session or webhook callback

	Pros	Cons	When to consider
	sophisticated product upgrade strategies. • Avoid single point of failure on database side without further PostgreSQL high availability setup. • Isolate data from different servers to help with security privacy compliance.	• Requires additional setup for configurations auto sync.	for API integration). • Small or very large scale.
Shared database	• Visibility and manageability with centralized scan data view and access. • Product configurations are auto synced among instances. • Simplified API integration.	• Prone to having database bottlenecks causing performance degradation. • Requires additional setup for database high availability. • More complex product deployment with more dependencies among instances. • High cost of system maintenance	• Usability is more important with centralized scan results view, and configurations auto synced. • Simplified client-side integration. • Have dedicated database system admin resources. Suitable for small to medium scale.

Hardware consideration

- **Memory**: Ensure your server has sufficient RAM. More RAM allows for larger caches, reducing storage I/O.
- **CPU**: Multi-core CPUs improve parallel query performance. Choose CPUs with higher clock speeds for better performance.
- **Storage**: Use SSDs for faster read/write operations. RAID configurations (e.g., RAID 1+0) can provide a good balance between performance and redundancy.

PostgreSQL configurations

The most crucial prerequisite for optimal PostgreSQL performance is having plenty of RAM capacity and high-speed storage I/O.

Modify the PostgreSQL configuration file (postgresql.conf) to optimize performance based on your hardware and workload. Users can find this conf file at **<PostgreSQL install location><version>\data\postgresql.conf**

Memory Settings

- shared_buffers: This is the amount of memory dedicated to caching data. Set to 25-40% of available RAM.
- work_mem: Allocate enough memory for sorting and hashing operations. A starting point can be 4-16 MB, adjusted based on workload.
- maintenance_work_mem: Use a larger value (e.g., 64-512 MB) for maintenance operations like vacuuming and index creation.
- effective_cache_size: Set to 50-75% of total system memory. This helps PostgreSQL estimate the likelihood of cached data.
- autovacuum_vacuum_threshold: Use a large value (e.g. 100000) for less redundant vacuum execution.
- autovacuum_vacuum_scale_factor: Set to 0 to ensure that auto-vacuum can execute properly and reduce storage space usage.

Checkpoint Settings

- checkpoint_timeout: Increase to 15-30 minutes to reduce the frequency of checkpoints.
- wal_buffers: Increase to 16 MB or higher for better write-ahead logging performance.

WAL (Write-Ahead Logging) Settings

- wal_level: Use minimal if not using replication; otherwise, replica or logical may be needed.
- max_wal_size: Increase to reduce the frequency of WAL file recycling (we recommend greater than or equal to 2GB).

MetaDefender Core configurations

There are a few MetaDefender Core configurations that vastly impact to database performance:

- Data retention setting
  - Reference: https://docs.opswat.com/mdcore/configuration/data-retention
  - Time frequency deciding when to mark old database-relevant records as “deleted” and not usable for data query anymore. The actual data wipe-up will be triggered later to reclaim disk space when PostgreSQL runs auto vacuuming, or when running database maintenance utility.
  - Lowering this threshold should keep database size manageable over time (combined with regular database maintenance practice) which eventually helps with overall performance, however the downside is usability and security compliance to keep data for a sufficient period of time.
- Database connection pool
  - Reference: https://docs.opswat.com/mdcore/configuration/metadefender-configuration#internal-section (db_connection setting).
  - Maximum number of simultaneous connections to be opened by MetaDefender Core application to send SQL queries and work with PostgreSQL database server.

- Increasing this threshold should utilize better PostgreSQL database server capacity when applicable, however this should be considered carefully with PostgreSQL bottlenecks in a shared database model.

Routine database maintenance

Run database maintenance utility on regular basis: https://docs.opswat.com/mdcore/operating/database-maintenance---metadefender-core-4-19-0-or-above

This utility covers the following important tasks:

- **Vacuuming**: vacuum the database to reclaim storage and update statistics.
- **Analyze**: run ANALYZE to update statistics for the query planner.
- **Reindexing**: reindex tables to reduce bloat, especially if you notice degraded performance.

Monitoring and calibration

- **pg_stat_statements**: Enable this extension to track query performance and identify slow queries.
- **pgTune**: Use pgTune PGTune - calculate configuration for PostgreSQL based on the maximum performance for a given hardware configuration or similar tools to generate optimized configuration settings based on your server specifications.

Example via practice

Let's look at an example that demonstrates the performance tuning process for MetaDefender Core to understand how we can achieve optimal results.

Problem statement: Scanning a large archive file wsusscn2.cab (size is 638MB with a total of 849.5K nested files inside) on MetaDefender Core 5.10.0 bundled with 20 AV engines with an expected total processing time of roughly 1 hour on a Dell PowerEdge R640 server running Windows Server 2022.

Based on this information, let's define our goals and constraints:

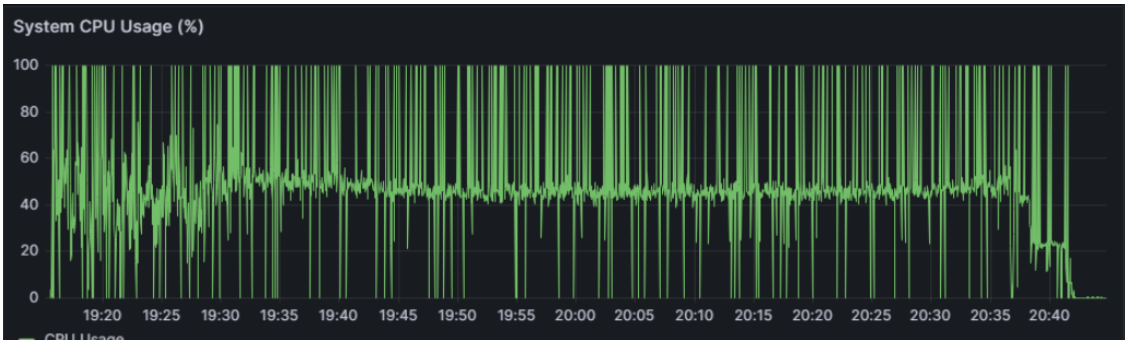
- **Performance goals:**
  - KPIs: the crucial metric to be measured here is the total processing time. CPU resource utilization should be also captured.
  - Performance target: total processing time is 1 hour or less, with an acceptable margin of error of 10% (i.e. 1 hour 6 minutes), and average CPU utilization should not be less than 50%, while the peak should be 100%.
- **Constraints:**
  - **Environment:**
    - The performance test will be run and measured on the bare metal of the Dell PowerEdge R640 server (2 sockets, 40 CPU cores, 512GB RAM, 2x 480GB SSD RAID-1)
    - Windows Server 2022 is known as a reliable OS for high-performance applications such as MetaDefender Core.

- **Application:**
  - MetaDefender Core version 5.10.0
  - 20 AV engine standard package
- **Use-case:** Submitting the sample file to MetaDefender Core directly via GUI, so no client application logic should be revised here.
- **What can be changed:** MetaDefender Core settings, Windows OS power saving settings, Dell server BIOS settings.
- **What can't be changed:** Windows Server OS version, Dell PowerEdge R640 specs, sample .cab file, 20 AV engines, MetaDefender Core version.

Now that we've defined all goals and constraints, we can proceed with running and capturing baseline performance metrics to understand what we can achieve with MetaDefender Core default settings.

Establish baseline

The initial performance test run with MetaDefender Core default settings reveals that total processing time is 1 hour 32 minutes, with the average CPU utilization at just 47%.



Now that we understand our baseline, we can adjust our settings to reach the performance goal of 1 hour (+10%) while keeping average CPU utilization at 50% or above.

Identify bottlenecks

The next step is identifying bottlenecks. Equipped with our baseline data, there are a few key log items we want to analyze logs further in this step:

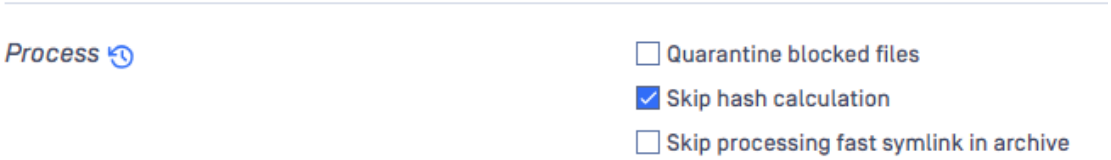
- MetaDefender Core queue utilization is one of the crucial things to analyze, especially with archive file processing. By default, MetaDefender Core provisions 500 slots for the queue, capped at 25% of those slots for archive extraction. Obviously, these conditions are not optimal for this sample archive file test run. This sample file contains multiple levels of sub-archive files inside, thus when it is capped at 125 slots (25% out of 500), a lot of sub-archive files inside must wait in the queue due to the limited slot allocation for archive extraction at some points.
- Hash calculation: MetaDefender Core is supposed to calculate hash checksums against every single file, including nested ones recursively inside the original archive. This leads to increased resource and time consumption. The sample file contains a total of 849.5K files inside, thus the total amount of time spent on hash calculation is measured in minutes.
  - Since this use-case is to submit the sample file to MetaDefender Core locally, the file integrity check can be

omitted for smaller total processing time.

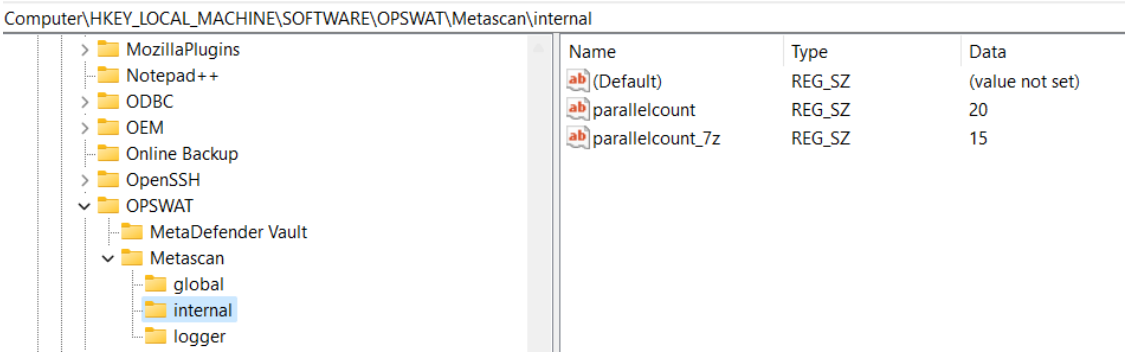
- Parallelcount: Based on findings in logged data, the archive extraction task seems to take more time than necessary and tends to be timed out. Without any tuning, MetaDefender Core does not have any limit in opening threads to push tasks into its archive engine for extraction. When it comes to situations with numerous nested archive files needing to be extracted simultaneously, this could become an IO bottleneck and impact overall processing performance.
- AV engine timeout: By default, MetaDefender Core waits for each AV engine to return with scan results before finalizing each AV scanning stage and moving to the next. Under high loads, there are numerous AV engine timeouts found in the application log. This also means waiting time in these circumstances are also related to engine timeouts, and consequently adding significant overhead on overall performance time.

Based on this analysis, we may implement the following remediation plan:

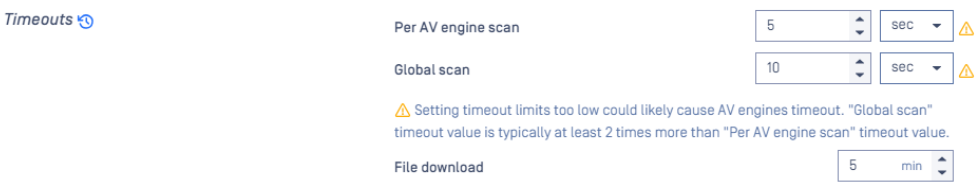
- Queue size: increased to 10,000 to ensure enough slots in the queue are allocated for archive extraction.
- Hash calculation: enabled “Skip hash calculation.”



- Parallelcount: limited number of threads for extraction (parallelcount_7z) to 15, not unlimited.



- AV scan timeout: reduced to less than 1 minute for “fail fast” approach, which allows MetaDefender Core to quickly recover and move on if any AV engines time out.

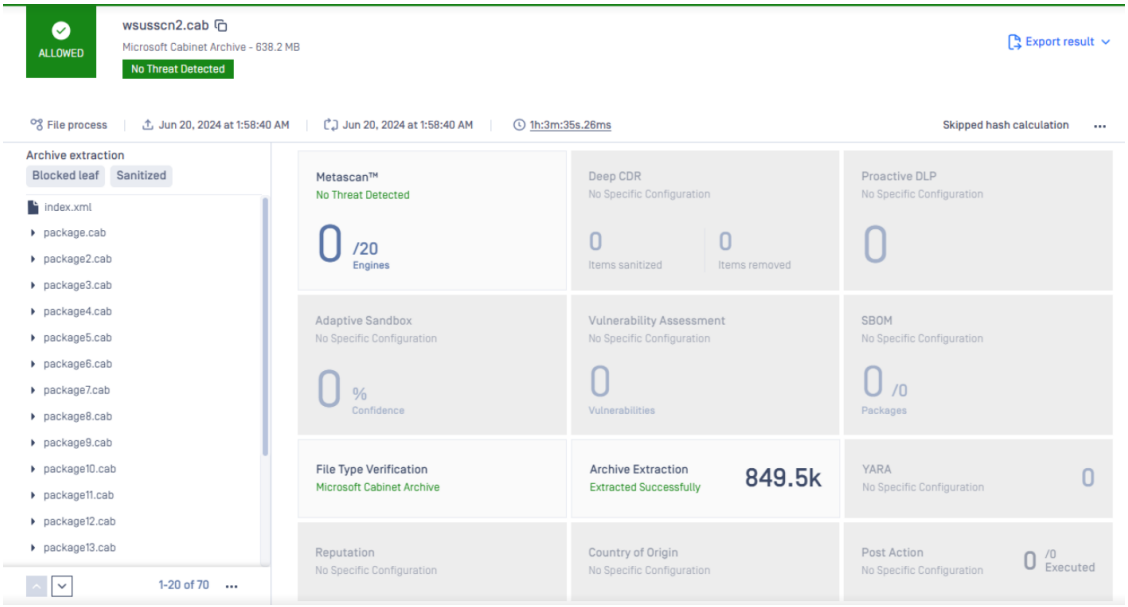


Design test case and perform load testing

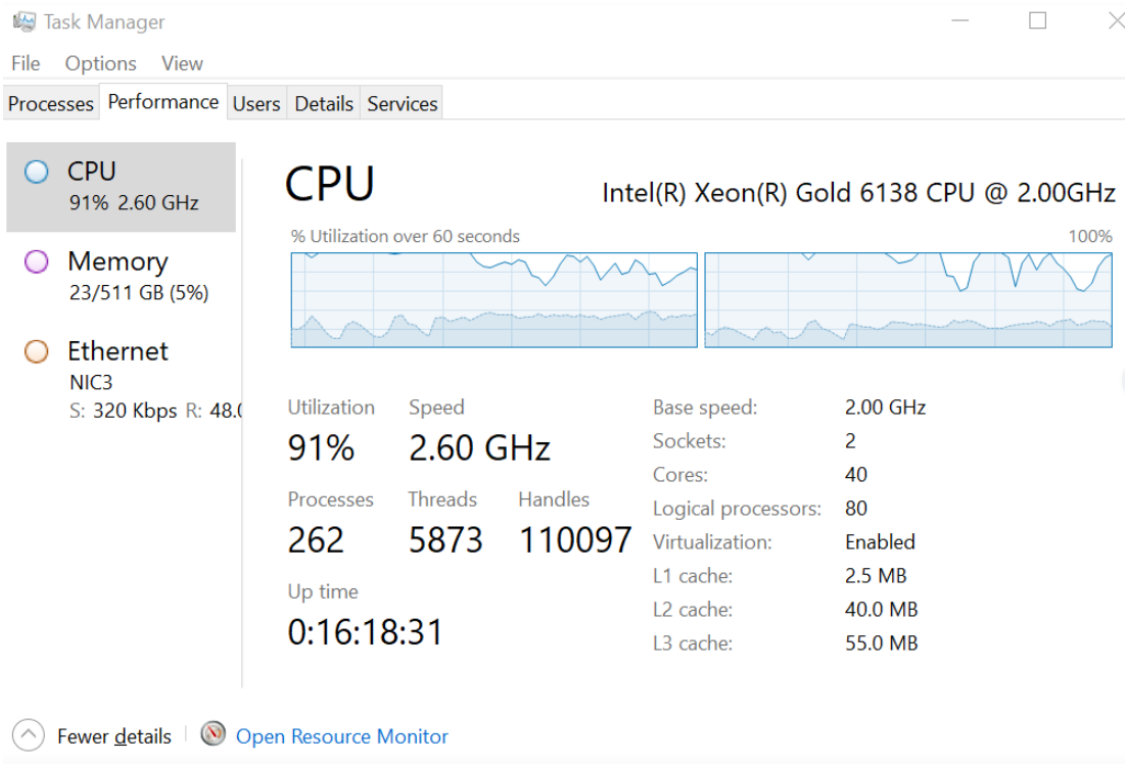
The same test case should be used, now with fine-tuned settings value as mentioned in previous step.

Review and repeat

The new performance test result comes back as 1 hour and 3 minutes, which is meeting the performance goal.



The CPU resource utilization is also monitored and we can see much higher utilization levels.



Performance monitoring

After MetaDefender is Core is tuned, it is crucial for the system to have a monitoring mechanism to ensure all metrics are monitored, in order to detect potential performance issues and remediate in a timely manner.

In MetaDefender Core context, some relevant parameters can be monitored for performance measurement:

Monitoring parameter	Recommendation
System resources (CPU, RAM, Disk)	System resources utilization should be kept within acceptable range and issue alerts when exceeded for an extended time period. Some examples of system resource monitoring: • Alert when average CPU utilization exceeds 95% for 15 minutes or more • Average memory used is less than 90% of available system memory • Free storage space falls below 10%
MetaDefender Core active performance	Can be collected via using a supported API https://docs.opswat.com/mdcore/metadefender-core#activeperformance MetaDefender Core performance metrics that should be monitored: • Avg. processing time • Avg. queue time • Avg. waiting time • Total number of successful results • Total number of processed objects • Total processed file size Alert setting should be configured after observing baseline and set proper threshold.
MetaDefender Core health check	Can be collected via a using supported API https://docs.opswat.com/mdcore/metadefender-core#healthcheck Including MetaDefender Core's • queue size • status

User management

To manage the users of the MetaDefender Core go to the **User Management** menu in the Web Management Console.

The Users and groups tab lists the existing users in the system.

Default user

After installation, a default local admin user needs to be created under LOCAL user directory following the welcome wizard

Special user accounts

Some user accounts are reserved in the product for system internal usage. These accounts are documented in this section.

> **Warning**
> The special accounts documented in this section are for internal usage. Do not directly modify these accounts through the user management functions cause it may give unexpected results.

SYSTEM/management account

The SYSTEM/management account is reserved for Central Management

When the product is connected to *Central Management* as a managed instance, then this account is automatically created by *Central Management* at the first successful connection with the following parameters:

Username	Password	Name	Email	Roles
management	*N/A*	MetaDefender Central Management	management@localhost	Adminis

All consecutive connection attempts are performed by *Central Management* using the SYSTEM/management account.

Functions

Besides listing existing users and AD groups the **Users** tab provides the following functions:

- Add new user or AD group
- Modify (and view) existing user's or AD group's properties
- Delete existing user or AD group

Add new user from a Local type user directory

To add a new user from a Local type user directory click the ADD NEW USER button and select a Local type user directory in the USER DIRECTORY drop down list.

The field ASSIGN TO ROLES lists all the roles that are assigned to this user.

Warning

As long as TLS is not configured for the Web Management Console, passwords are sent clear-text over the network. To set up TLS see Configuring TLS.

Warning

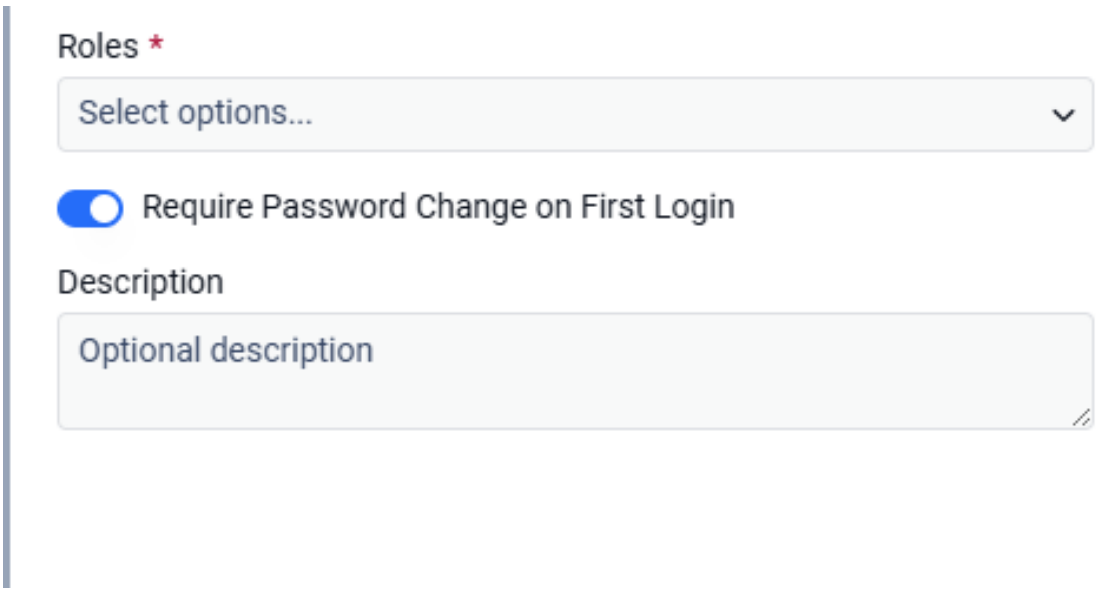
If enhanced password policy is enabled for the user directory this user belongs to, then the new password must fulfil the password complexity requirements listed on the User directories](/configuration/user-directories) page.

The APIKEY value provides access to the MetaDefender Core REST API for this user with no authentication. If no such functionality is needed for the user then this field can be left blank.

There are two ways to have an APIKEY for a user.

- generating by using *Generate* button next to APIKEY field,
- typing one that matches the following criterias:
  - The length of the API key must be exactly 36 characters.
  - It must contain numeric and lower case letter characters only [0-9a-z].
  - It must contain at least 10 lower case letter characters.
  - It must contain at least 10 numeric characters.
  - It is allowed to contain at most 3 consecutive lower case letter characters (e.g. "abcd1a2b3c..." is invalid).
  - It is allowed to contain at most 3 consecutive numeric characters (e.g. "1234a1b2c3..." is invalid).

Admin can also request to the users, they need to change the password on the First Login.



Add new users from an Active Directory type user directory

To add a new user from an Active Directory type user directory click the ADD NEW USER button and select an Active Directory type user directory in the USER DIRECTORY drop down list. Select USER as the ACCOUNT TYPE.

Provide the name of the account and click the *FIND ACCOUNT* button to look up the account in the Active Directory. If the lookup succeeds, then the ACCOUNT DISPLAY NAME and the DISTINGUISHED NAME fields are filled automatically.

> **Note**
>
> Do provide the account name precisely. There is no functionality to look up similar names or partial matches.

The field ASSIGN TO ROLES lists all the roles that are assigned to this user.

Add new group from an Active Directory type user directory

> **Info**
>
> The purpose of adding an Active Directory group to the MetaDefender Core is to assign role(s) to all the users in that Active Directory group.
>
> The users of the Active Directory group can authenticate with their Active Directory credentials in MetaDefender Core Web Management Console and will be assigned with the roles of the group.

To add a new group from an Active Directory type user directory click the ADD NEW USER button and select an Active Directory type user directory in the USER DIRECTORY drop down list.

> **Note**
>
> Select GROUP as the ACCOUNT TYPE.

Provide the name of the group and click the *FIND ACCOUNT* button to look up the group in the Active Directory. If the lookup succeeds then the ACCOUNT DISPLAY NAME and the DISTINGUISHED NAME fields are filled automatically.

> **Note**
>
> Do provide the account name precisely. There is no functionality to look up similar names or partial matches.

The field ASSIGN TO ROLES lists all the roles that are assigned to all users of this group.

Assign roles to a user or an Active Directory group

Role(s) must be assigned to users and Active Directory groups in order they can use the Web Management Console.

The field ASSIGN TO ROLES in the **Add/assign new user(s)** and **Modify user** dialogs lists all the roles that are assigned to the user.

The following is the role assignment policy:

1. At least one role must be assigned to a user or Active Directory group
2. Optionally multiple different roles can be assigned
    1. In this case the highest available permission applies to each function. Example:

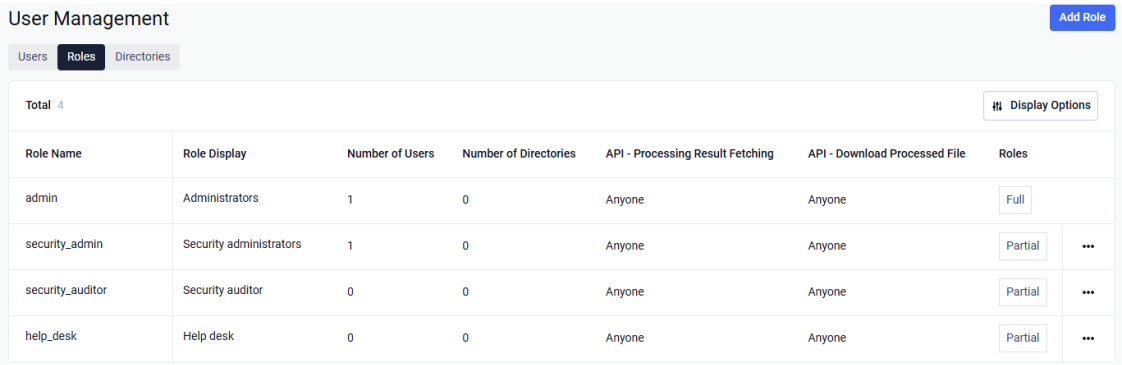
Roles assigned	Effective permissions	
	Full permission	Read only permission
security_admin	Scan history, Update history, Security rules, Security zones, Analysis workflows, Scan nodes, Engines, Update settings, Scan settings	
security_auditor		All except External settings
security_admin AND security_auditor	Scan history, Update history, Security rules, Security zones, Analysis workflows, Scan nodes, Engines, Update settings, Scan settings	Config history, Data retention, User management, License

Delete user

> **Note**
> Active sessions of the deleted user will be aborted at the time of the next interaction with the server.

Roles

Roles can be assigned to users. This simplifies controlling permissions. The Roles tab lists the existing roles in the system.



Default roles

After installation the following default roles are created with the following parameters:

Role name	Display name	Default member username	Permissions on functionality	Permissions on API level
admin	Administrators	admin	Full on all functions	Be able to fetch scan result submitted by anyone Be able to download processed file where original file was submitted by anyone
security_admin	Security administrators		Full on Scan history, Update history, Security rules, Security zones, Analysis workflows, Scan nodes, Engines, Update settings, Scan settings functions	Be able to fetch scan result submitted by anyone Be able to download processed file where original file was submitted by anyone
security_auditor	Security auditor		Read-only on a ll except External settings functions	Be able to fetch scan result submitted by anyone Be able to download processed file where original file was submitted by

Role name	Display name	Default member username	Permissions on functionality	Permissions on API level
				anyone
help_desk	Help desk		Read-only on Scan history, Update history, Security rules, Security zones, Analysis workflows, Scan nodes, Engines, Scan settings functions	Be able to fetch scan result submitted by anyone Be able to download processed file where original file was submitted by anyone

Permissions on functionality

Each role has a set of rights associated to it. Each of these rights represent the level of access to the appropriate function of Metadefender Core Web Management Console. A right can be set to one of three different states:

- **None:** users of this role have no right to access the given function of Metadefender Core Web Management Console. The menu belonging to the function is not displayed for the users of this role.
- **Read-only:** users of this role have right to access the given function for observation purposes only. Users of this role can, however, not effectuate any modification or any change to the function.
- **Full:** users of this role have full access to the given function, including viewing any data belonging to it and modifying its configuration.

Permissions on API level

Each role has a set of rights pertaining to REST API access level, including following REST endpoints:

Processing result fetching:

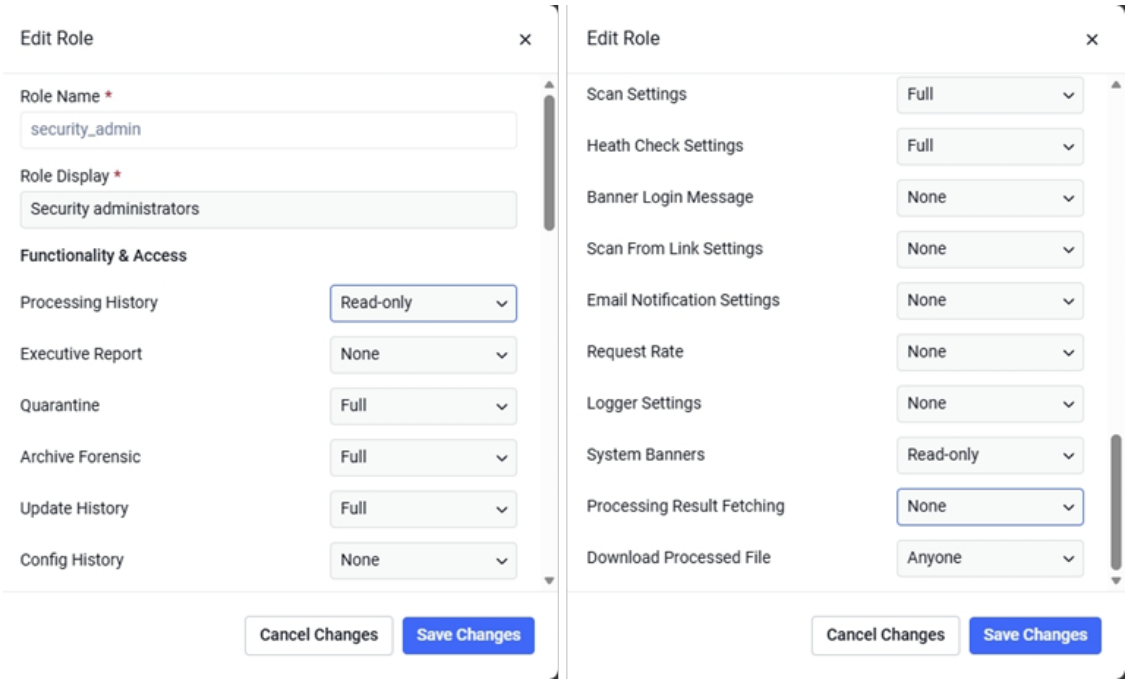
- GET /hash/<md5, sha1, sha256> (GET File Analysis)
- GET /file/<data_id> (GET File Analysis)
- GET /file/batch/<batch_id> (GET Status of Batch Analysis)
- GET /stat/log/scan (Leveraged by Core management console)
- GET /stat/log/scan/export (Leveraged by Core management console)

Download processed file:

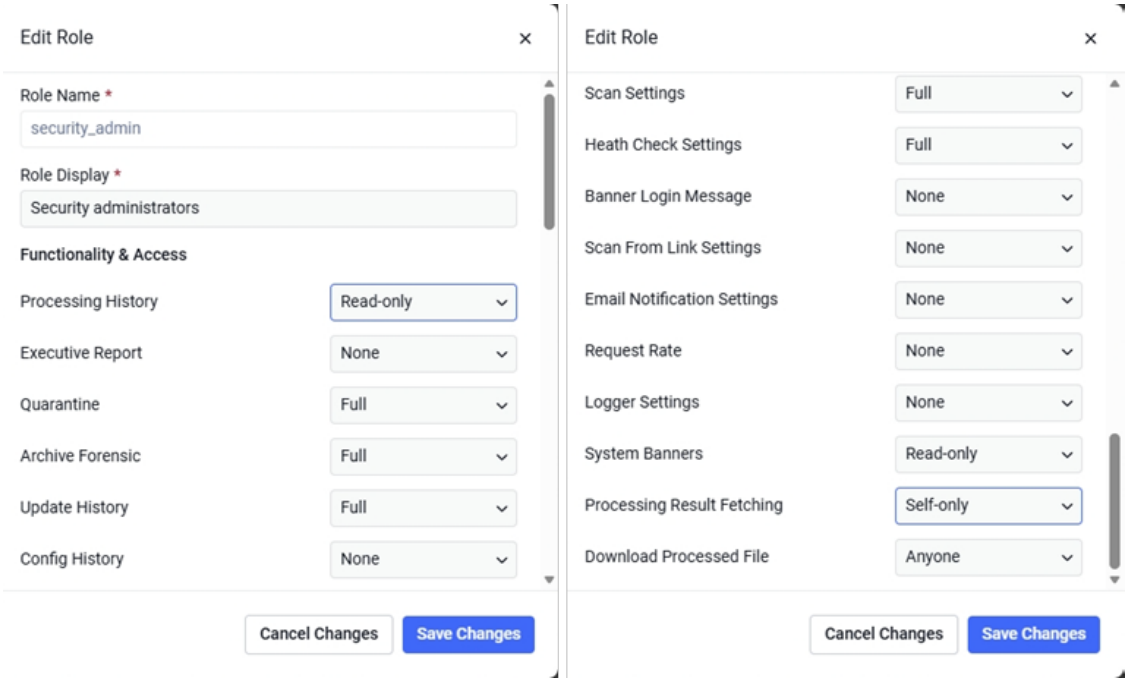
- GET /file/converted/<data_id> (GET Download Sanitized Files)
- GET /file/processed/<data_id> (Leveraged by Core management console)

A right can be set to one of three different states:

- **None:**
  - Users of this role have no right to access the given REST APIs (return "Access denied" error) and relevant functionalities on Metadefender Core Web Management Console
  - **Note:** When "NONE" is selected for "Processing result fetching", "Processing history" menu item under Dashboard will automatically switch to "READ-ONLY" right, and "FULL" right will instead be disabled for selection ("FULL" right can be only enabled for selection back when this option is switched to "ANYONE")



- **Self-only:**
  - Users of this role only have right to access the given REST APIs and relevant functionalities on Metadefender Core Web Management Console where the scan requests were submitted by themselves only
  - Users of this role have no right to access the given REST APIs (return "Access denied" error) where scan requests were submitted by anyone else
  - **Note:** When "SELF-ONLY" is selected for "Processing result fetching", "Processing history" menu item under Dashboard will automatically switch to "READ-ONLY" right, and "FULL" right will instead be disabled for selection ("FULL" right can be only enabled for selection back when this option is switched to "ANYONE")



- **Anyone:** Users of this role have full access to the given REST APIs and relevant functionalities on Metadefender Core Web Management Console where the scan requests were submitted by anyone

Functions

Besides listing existing roles, the **Roles** tab provides the following functions:

- Add new role
- Modify (and view) existing role
- Delete existing role

> **Note**
>
> The default role **Administrators** cannot be deleted or modified.

Modify role

> **Warning**
>
> The users' permissions won't be modified during the session, even if one of their roles are modified in the meantime.
>
> For example:
>
> 1. A user is assigned to the role *security_admin* and has Full permissions on *Config history*
> 2. She can see *Config history* changes.
> 3. During her session the *Config history* permissions are set to *None* for the *security_admin* role.

4. The logged in user can still select the *Config history* menu and can see the configuration changes there.

Then new permissions will be effective only after a logout and new login.

Delete role

Note

A role cannot be deleted as long as it is assigned to any user.

As a consequence, deleting a role cannot affect active sessions of users.

User directories

Users can be organized into separate user directories. User directories help to enforce the following login policies:

1. Lockout after a number of consecutive failed login attempts
2. Disable logins for all users of the user directory

The Users tab lists the existing user directories in the system.

Default user directory

After installation a default user directory is created with the following parameters:

User directory type	Name	Number of failed logins before lockout	Lockout time [minutes]
Local	LOCAL	3	5
Local	SYSTEM	0	0

Two types of user directories exist in MetaDefender Core:

1. Local
2. Active Directory/LDAP

Local type user directories

Local type user directories allow creating users that locally exist on the MetaDefender Core.

To protect user accounts of a local user directory against brute force password breaking attacks, policy settings may be applied:

- **Number of failed logins before lockout**: After this number of consecutive failed login attempts the account gets

locked.

- **Lockout time [minutes]**: The account remains locked for the given minutes.
  - When the lockout time elapses, the account lock gets released automatically.
  - Users with appropriate permission may release the account lock earlier using the *RELEASE LOCKOUT button*.
- **Enable enhanced password policy**: Check out more at Password Policy.

Active Directory/LDAP type user directories

LDAP and Active Directory type user directories allow users defined in an LDAP or Active Directory to access MetaDefender Core.

These types of user directories do not provide the possibility to define login policies; these policies may be defined in the LDAP or Active directory directly.

Functions

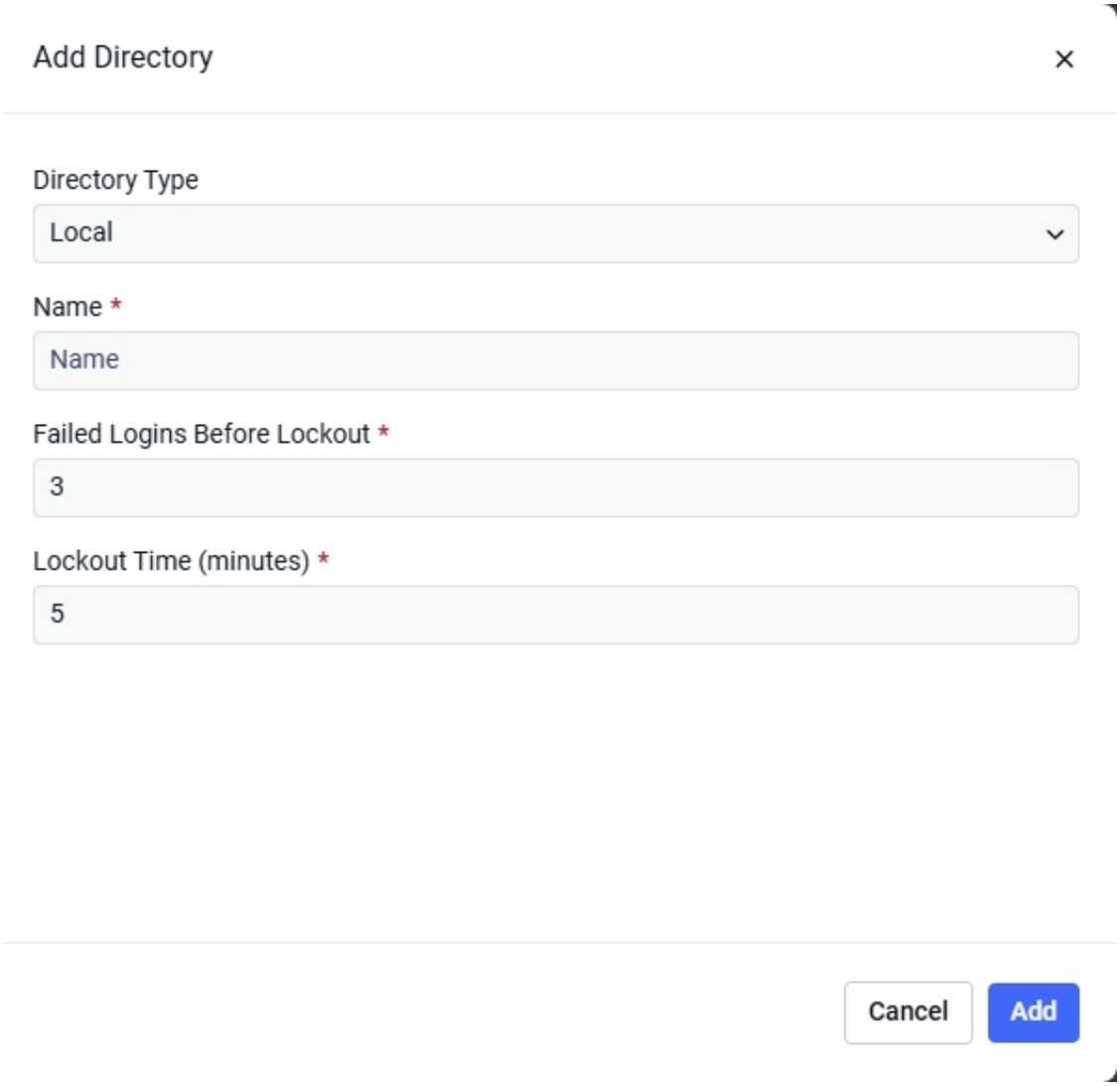
Besides listing existing user directories the **User directories** tab provides the following functions:

- Add new user directory
- Modify (and view) existing user directory
- Delete existing user directory
- Enable or disable existing user directory
- Unlock locked accounts

Add new Local type user directory

Click the *ADD NEW USER DIRECTORY* button and select **Local** in the USER DIRECTORY TYPE drop down list.

For explanation of the **Number of failed logins before lockout** and **Lockout time [minutes]** fields read the Local type user directories section.



Add new Active Directory

Click the *ADD NEW USER DIRECTORY* button and select **Active Directory** in the USERDIRECTORY TYPE drop down list.

The USERNAME and PASSWORD values should be the name as DN (distinguished name) and password of a user who has permissions to do searches in the directory.

Warning

As long as TLS is not configured for the Web Management Console, passwords are sent clear-text over the network. To set up TLS see Enabling HTTPS.

Warning

As long as ENCRYPTION field is set to *None* there is no encryption used between the MetaDefender Core and the Active Directory server. All passwords and other information are sent clear-text over the network.

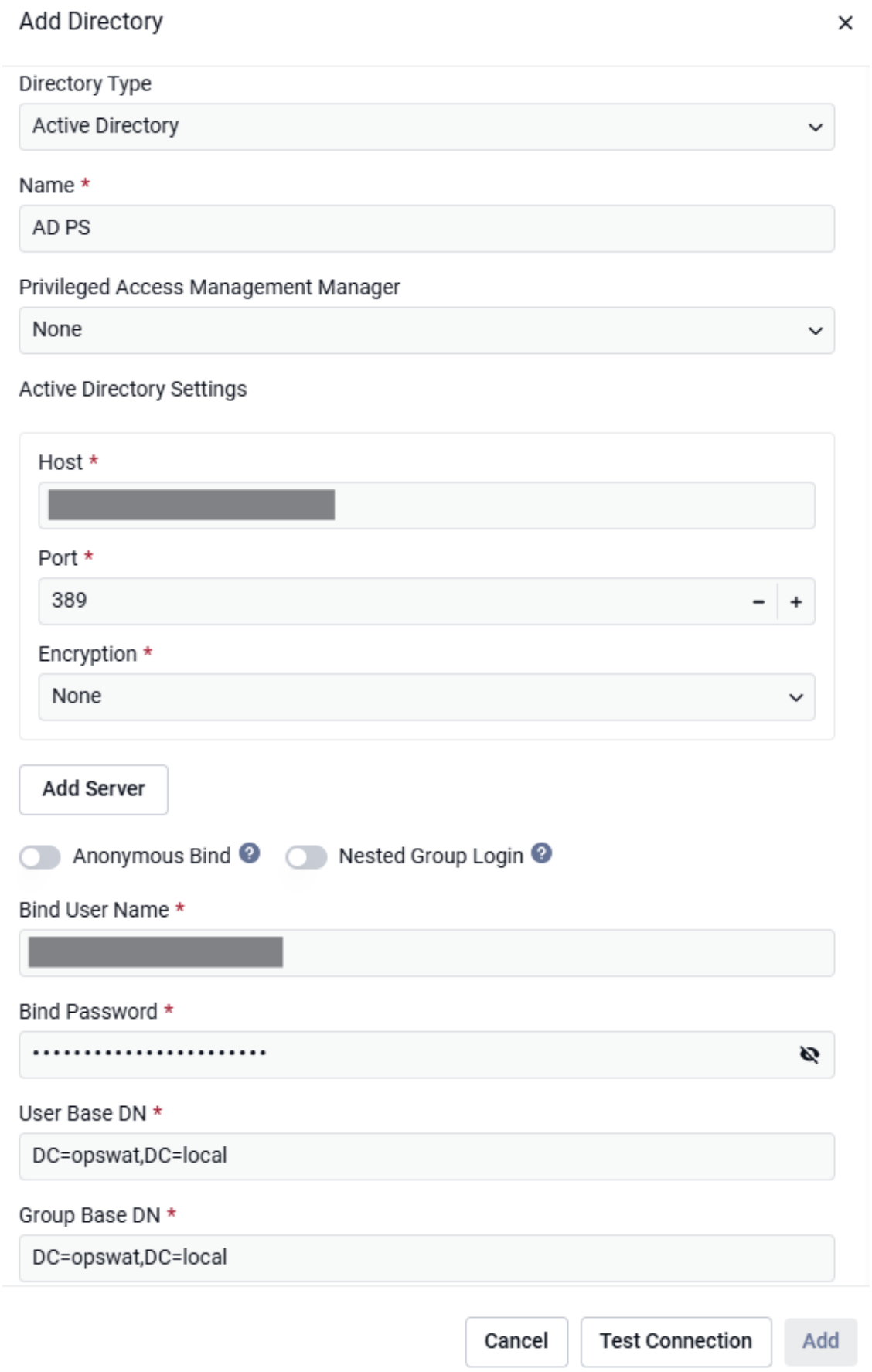
Use *StartTLS* or *SSL* as ENCRYPTION whenever possible and don't forget to install the certificate of the issuer of the AD server's certificate on the server that runs MetaDefender Core.

The USER BASE DN and the GROUP BASE DN values should provide the entries in the Active Directory tree where user and group entity lookups should be started. For tips about finding the proper values for these fields see Active Directory/LDAP attributes.

In the section "Privileged Access Management Manger"

- Select None - if you don't have any external application for manage AD accounts
- Select BeyondTrust Password Safe - if you use Password Safe for manage AD accounts - More details in here: Integrate Active Directory with BeyondTrust Password Safe

Click the *TEST* button to test the Active Directory settings. If the test succeeds, then the user directory can be added to the list with the *ADD* button.

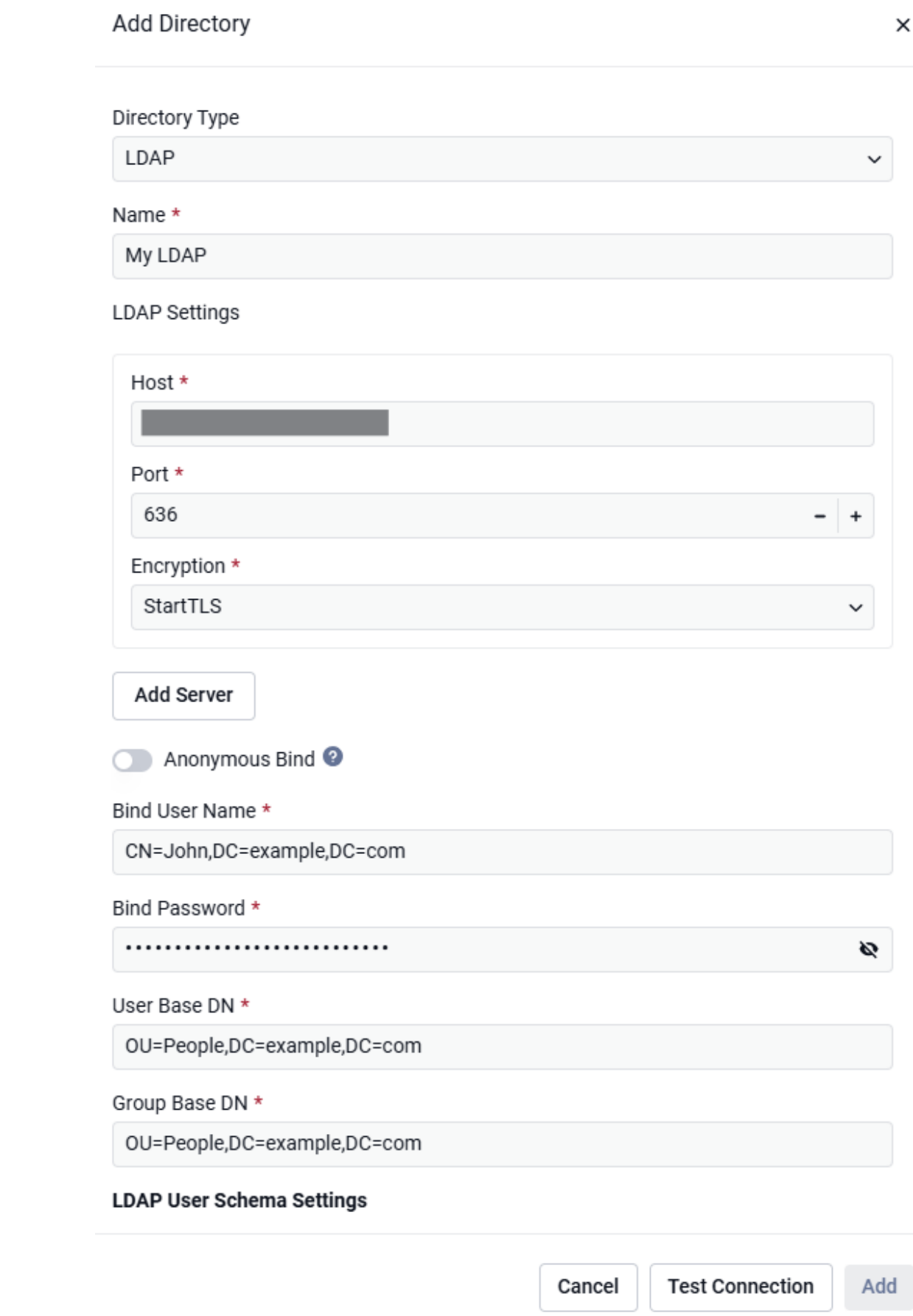


Add new LDAP Directory

Click the *ADD NEW USER DIRECTORY* button and select **LDAP** in the USERDIRECTORY TYPE drop down list.

The following information should be given to configure an LDAP user directory:

- **bind username**: The name as DN of a user who has permissions to do searches in the LDAP directory.
- **user base DN**: The DN from where all users can be reached.
- **group base DN**: The DN from where all groups can be reached.
- **user object class**: The name of the object class (objectClass) that is for user objects. (e.g. posixAccount or person)
- **user account attribute**: The name of the LDAP attribute that contains the login name of the users.
- **group object class**: The name of the object class (objectClass) that is for group objects. (e.g. posixGroup or group)
- **group account attribute**: The name of the attribute that contains the name of the group of the users.


Add Directory
Directory Type
LDAP
Name *
My LDAP
LDAP Settings
Host *
Port *
636
Encryption *
StartTLS
Add Server
Anonymous Bind
Bind User Name *
CN=John,DC=example,DC=com
Bind Password *
User Base DN *
OU=People,DC=example,DC=com
Group Base DN *
OU=People,DC=example,DC=com
LDAP User Schema Settings
Cancel
Test Connection
Add

Note

Please note that using only DC components for the user/group DNs may result in searches to be executed from the top of the directory information tree and potentially slow down LDAP server responses a lot and thus have an impact on MetaDefender Core password validation. The rule of thumb here is that the more specific the user/group DN the faster the server response.

Taking the above example into consideration: a user search DN of "OU=People,DC=example,DC=com" could potentially result in much faster server response than "DC=example,DC=com" and should be preferred assuming all users reside under "OU=People,DC=example,DC=com" in the directory information tree.

Note

Please also note that users and groups may reside in different parts of the directory information tree, as a consequence applying the same, more specific DN both as USER BASE DN and GROUP BASE DN may cause MetaDefender Core not to find group accounts in the directory information tree. So, these DNs should be chosen carefully.

Unlock locked accounts

All the locked user accounts that belong to a Local type user directory, can be released clicking the *RELEASE LOCKOUT* button.

Notes

The currently logged on user cannot disable the user directory to which her account is assigned to. For example, the admin user cannot disable the LOCAL user directory.

The currently logged on user cannot delete the following:

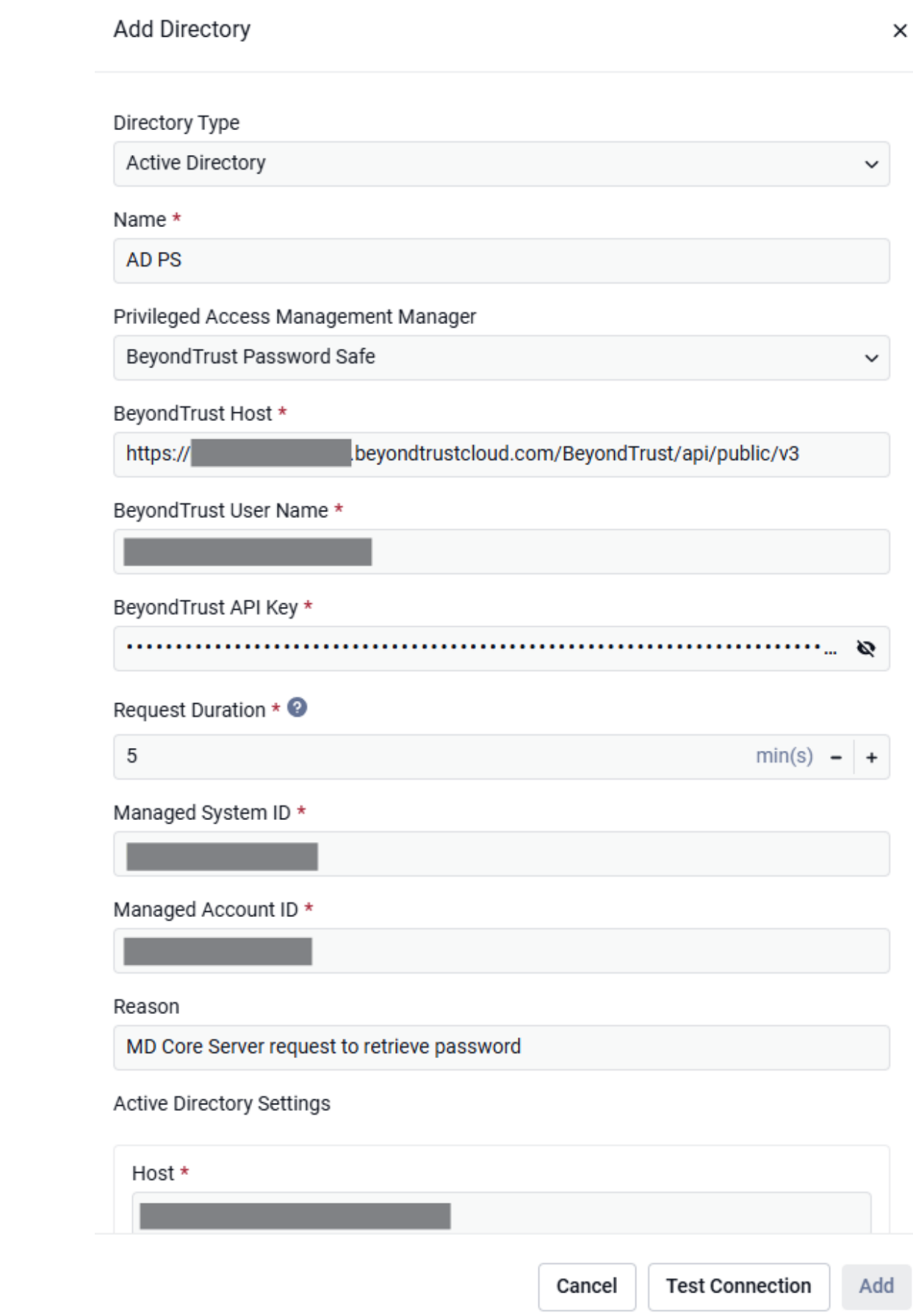
- Her own user account. For example, the admin user cannot delete the admin user account.
- The user directory to which her account is assigned to. For example, the admin user cannot delete the LOCAL user directory.

Integrate Active Directory with BeyondTrust Password Safe

Starting from version 5.16.1 - MetaDefender Core can integrate with BeyondTrust Password Safe to manage Active Directory (AD)'s admin credential.

Note: Does not support for LDAP Directory

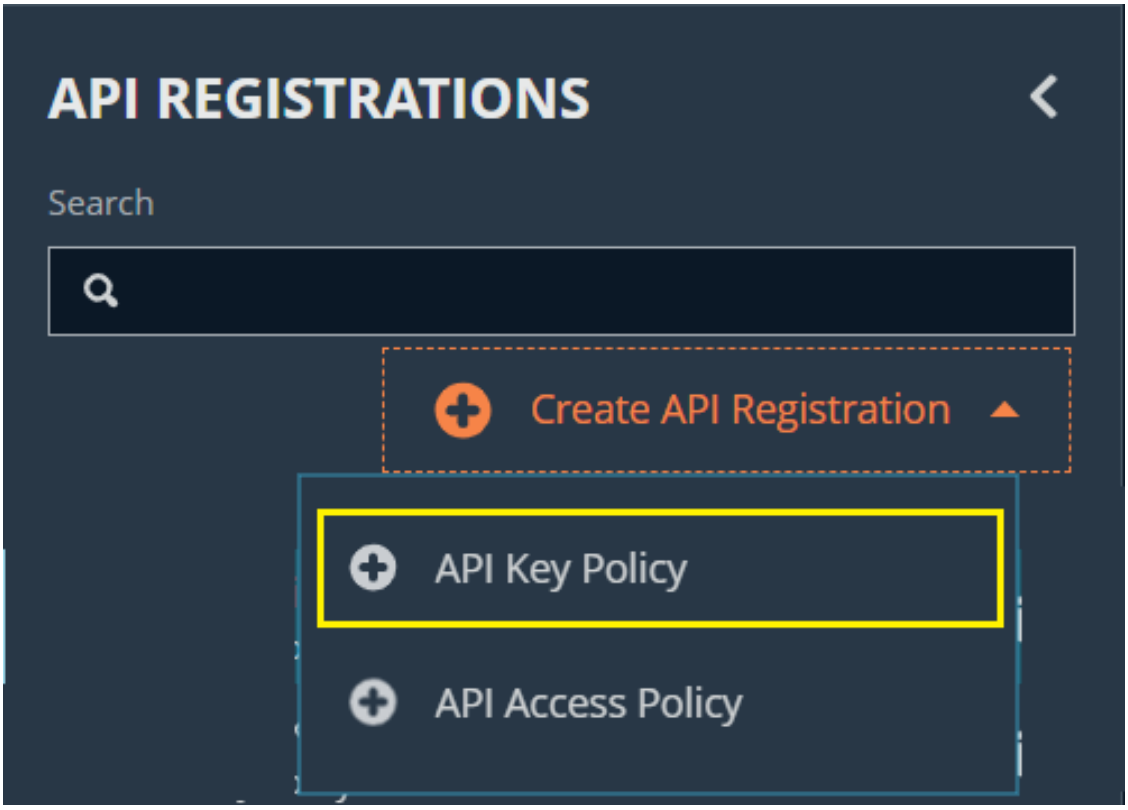
When adding Active Directory, please select **BeyondTrust Password Safe** in section "Privileged Access Management Manager"


Add Directory
Directory Type
Active Directory
Name *
AD PS
Privileged Access Management Manager
BeyondTrust Password Safe
BeyondTrust Host *
https://[redacted].beyondtrustcloud.com/BeyondTrust/api/public/v3
BeyondTrust User Name *
[redacted]
BeyondTrust API Key *
••••••

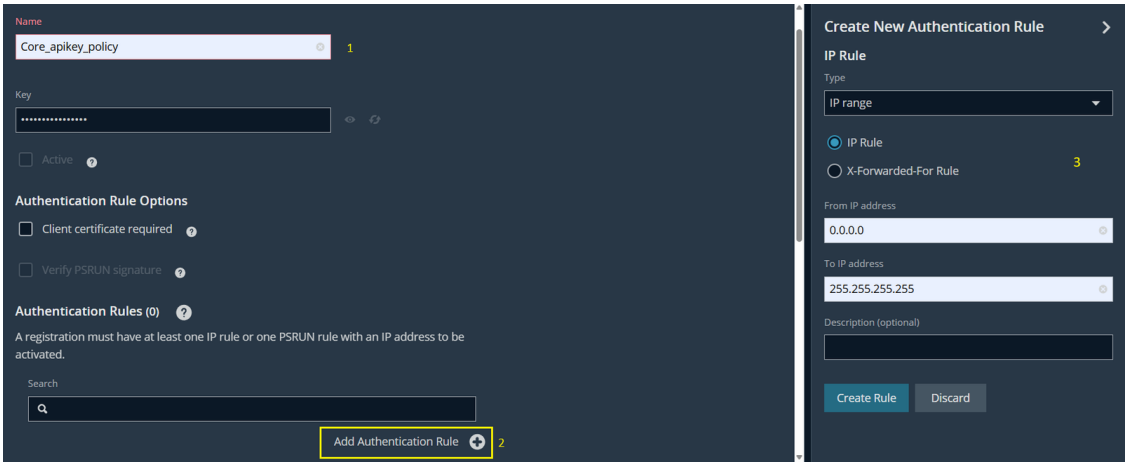
- BeyondTrust Host
- BeyondTrust Username: **your Password Safe login account.**
- BeyondTrus API Key
- Managed System ID
- Managed Account ID

Get BeyondTrust Host and BeyondTrust API Key from Password Safe.

- Login to Password Safe.
- Navigate to Configuration General API Registrations Create API Registration API Key Policy.



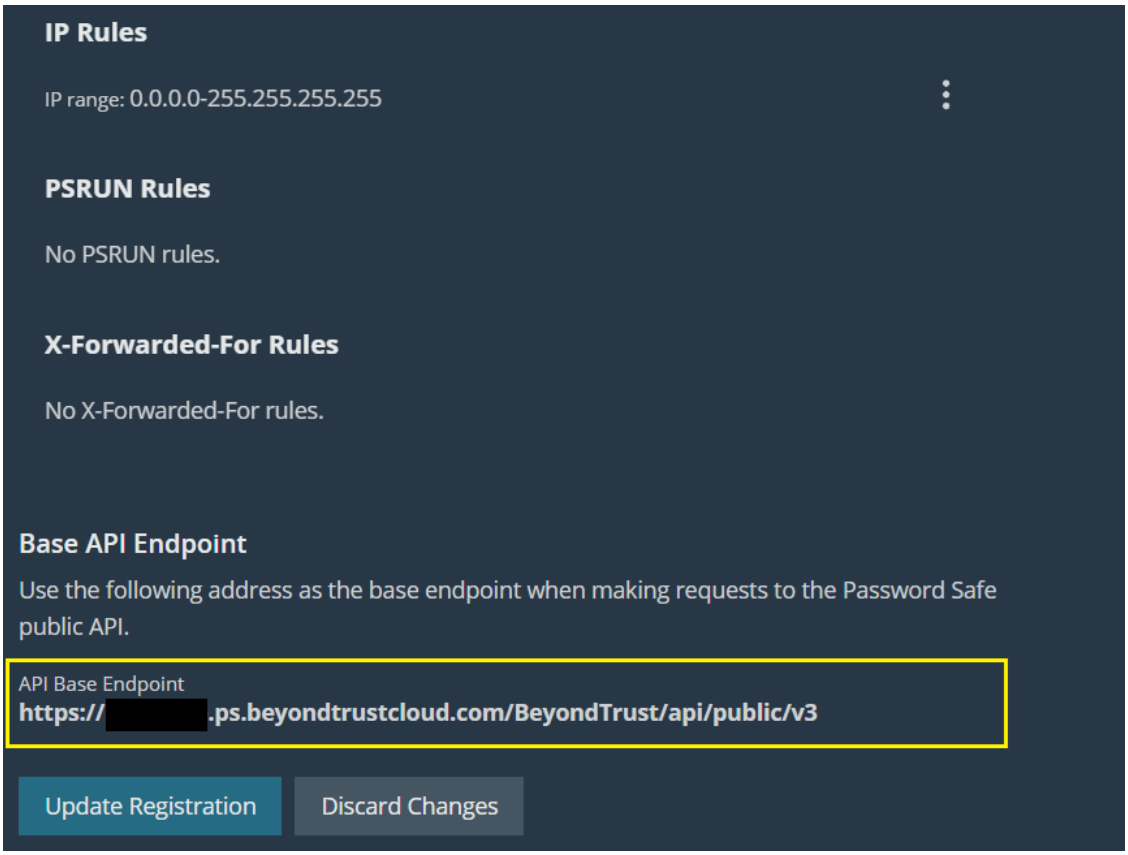
Fill the name for the new API Key policy and create new Authentication Rule as below



- Click Create Registration button to finish.
- Click on the new API Key policy we just created to view details → show and copy the key as BeyondTrust API Key.

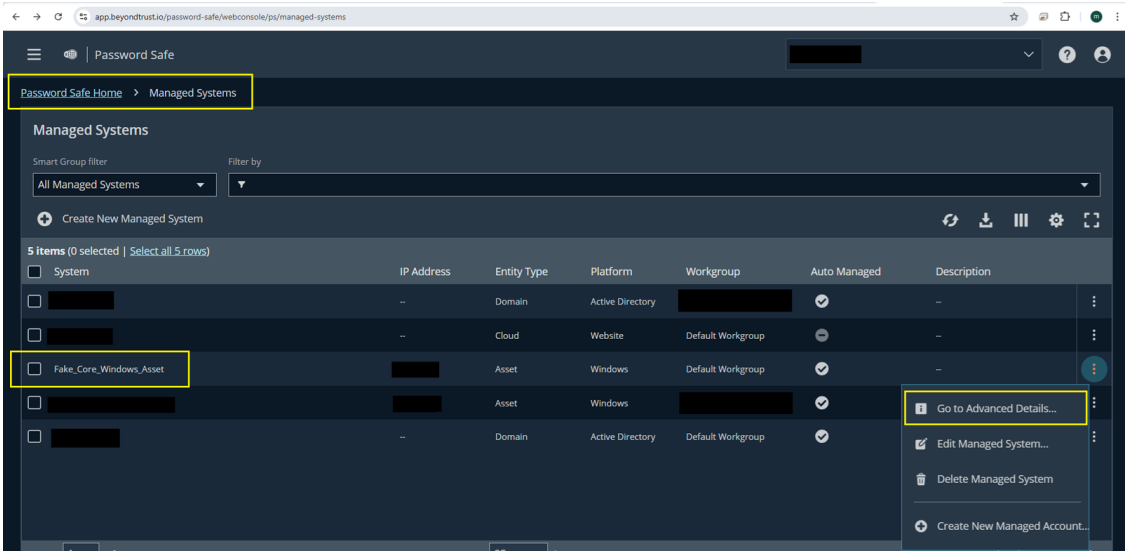


- Continue to scroll down to bottom → copy API Base Endpoint as BeyondTrust Host.

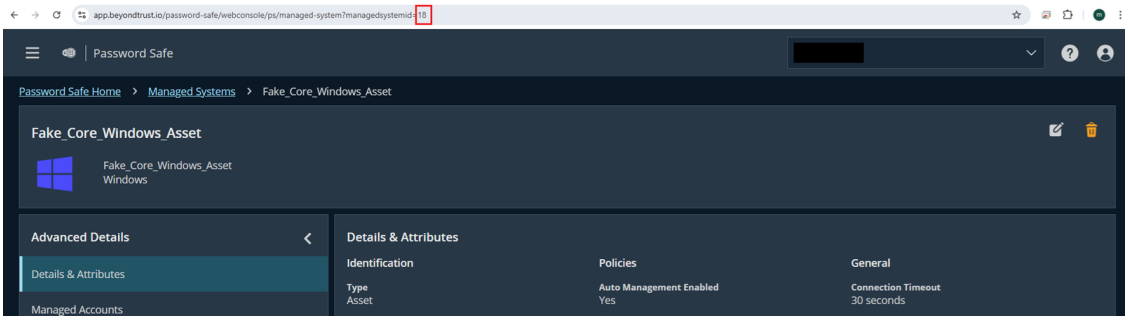


Get Managed System ID and Managed Account ID of managed AD account from Password Safe

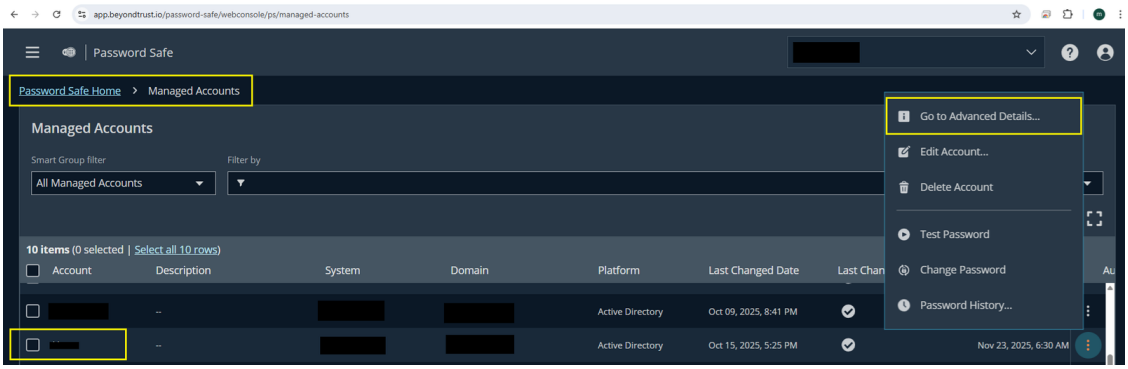
- Login to Password Safe.
- Navigate to Managed Systems → select your relevant system → click the 3-dots options on the right → select Go to Advanced Details...



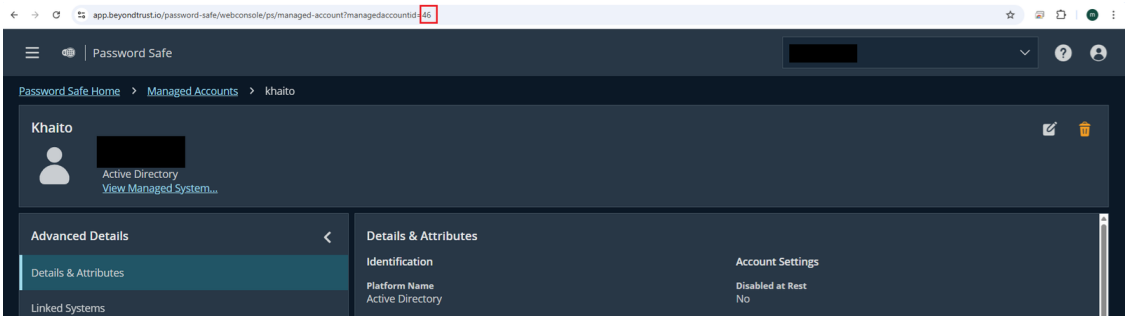
- The **Managed System ID** will show up in the URL



- Next, navigate to Managed Accounts → select your AD account → click the 3-dots options on the right → select Go to Advanced Details...



- The **Managed Account ID** will show up in the URL



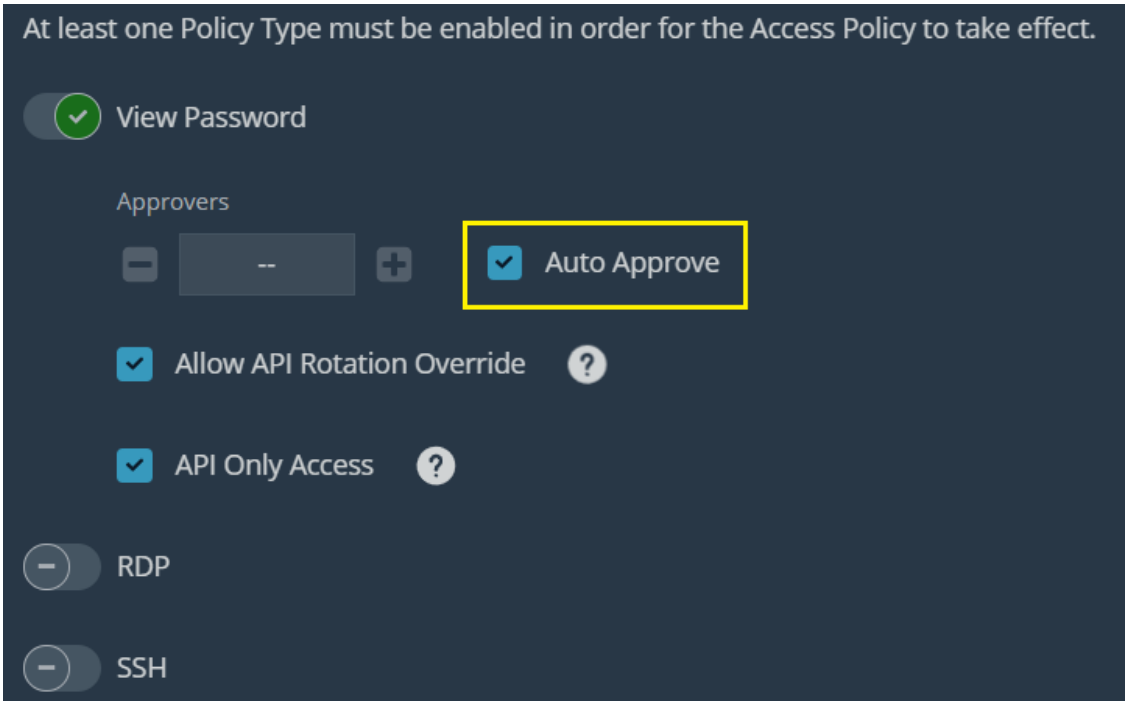
Info

When integrating AD with Password Safe, the users don't need to input "Bind password" - because this password will be managed by Password Safe, users don't need to care about this field anymore.

Enable auto-approve in Password Safe

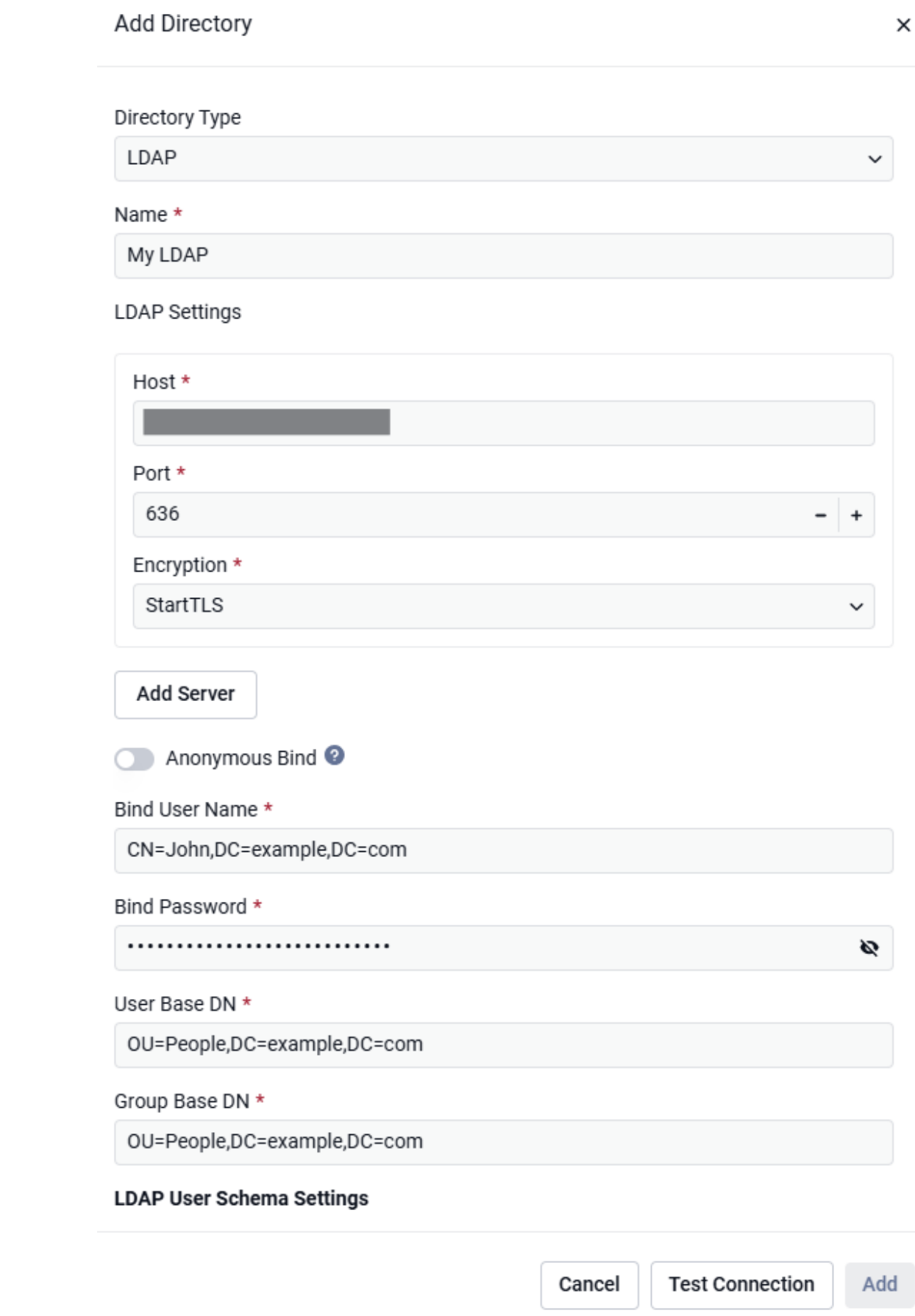
We highly recommend that the user should enable auto-approve in Password Safe for seamlessly integration with Active Directory

1. Login to Password Safe.
2. Navigate to Privileged Access Management Policies Access Policies edit your Access policy enable Auto Approve



Active Directory/LDAP attributes

This page contains tips on how to obtain the USERNAME and the USER BASE DN and GROUP BASE DN attributes when creating an Active Directory type user directory.


Add Directory
Directory Type
LDAP
Name *
My LDAP
LDAP Settings
Host *
Port *
636
Encryption *
StartTLS
Add Server
Anonymous Bind
Bind User Name *
CN=John,DC=example,DC=com
Bind Password *
User Base DN *
OU=People,DC=example,DC=com
Group Base DN *
OU=People,DC=example,DC=com
LDAP User Schema Settings
Cancel
Test Connection
Add

Username

All three attributes should be expressed with a valid LDAP syntax.

Normally a domain administrator should provide these values, however there is a way to get the USERNAME as a LDAP DN, that is needed for the MetaDefender Core to do searches in the directory information tree, and it is as follows:

Log on to a Windows server machine that has connectivity to the Active Directory

1. Choose a user that is intended for this purpose (ie: has rights to do searches in the tree)

2. Open a Command window with elevated rights (Run as Administrator)

3. Assuming example.com as domain and John Smith with account name john.smith as the user, type the following:

```
Code

> dsquery user domainroot -samid john.smith
```

or

```
Code

> dsquery user domainroot -name John Smith
```

Info

The commands above will return the correct DN for the user in question. The DN should look something like this:

`CN=John Smith,OU=People,OU=Engineering,DC=example,DC=com`

Please note, the actual user DN will not look exactly like the above example, but will depend on the structure of the underlying directory information tree in the Active Directory server.

User base and group base DN

Once the user DN is obtained, an easy way to get the DNs for the user and group searches is by taking all the DC parts of the user DN and leaving the rest out, which results in the following DN:

```
Code

DC=example,DC=com
```

Note

Please note that using only DC components for the user/group DNs may result in searches to be executed from the top of the directory information tree and potentially slow down AD server responses a lot and thus have an impact on Metadefender Core password validation. The rule of thumb here is that the more specific the user/group DN the faster the server response.

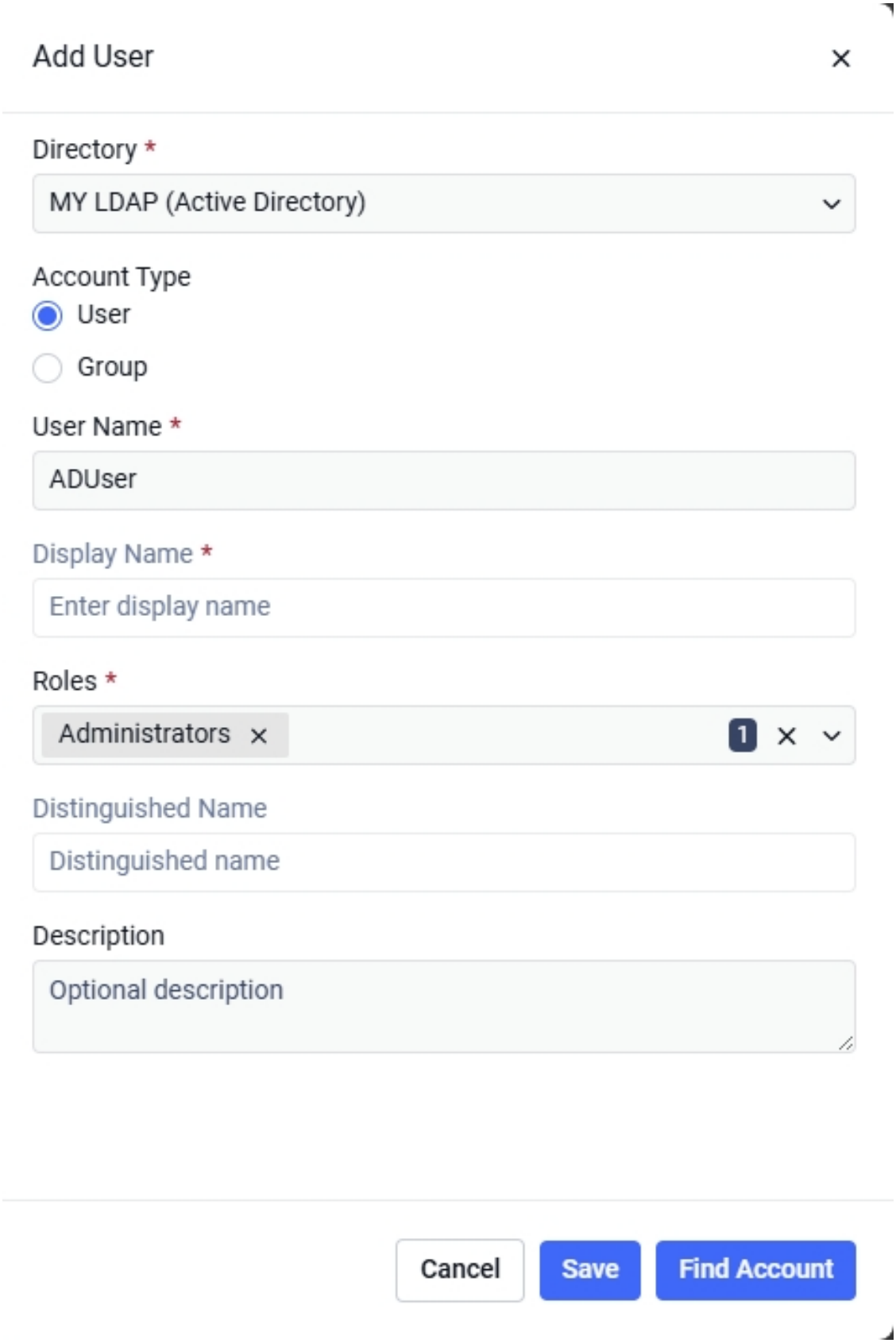
Taking the above example into consideration: a user search DN of "OU=People,OU=Engineering,DC=example,DC=com" could potentially result in much faster server response than "DC=example,DC=com" and should be preferred assuming all users reside under "OU=People,OU=Engineering,DC=example,DC=com" in the directory information tree.

Note

Please also note that users and groups may reside in different parts of the directory information tree, as a consequence applying the same, more specific DN both as USER BASE DN and GROUP BASE DN may cause Metadefender Core not to find group accounts in the directory information tree. So these DNs should be chosen carefully.

Add User & Group Active Directory

In User Management, add a user or group to Active Directory (AD).



Login with Active Directory

To log in with an Active Directory (AD) user account, you typically use the following format:

1. **Directory name**: Directory name could be

1. the name of your AD directory - the name that user has configured on management console
2. or the domain name

2. **AD Account:** An AD user or a member of an AD group that you have recently added to the directory.

Login Format:

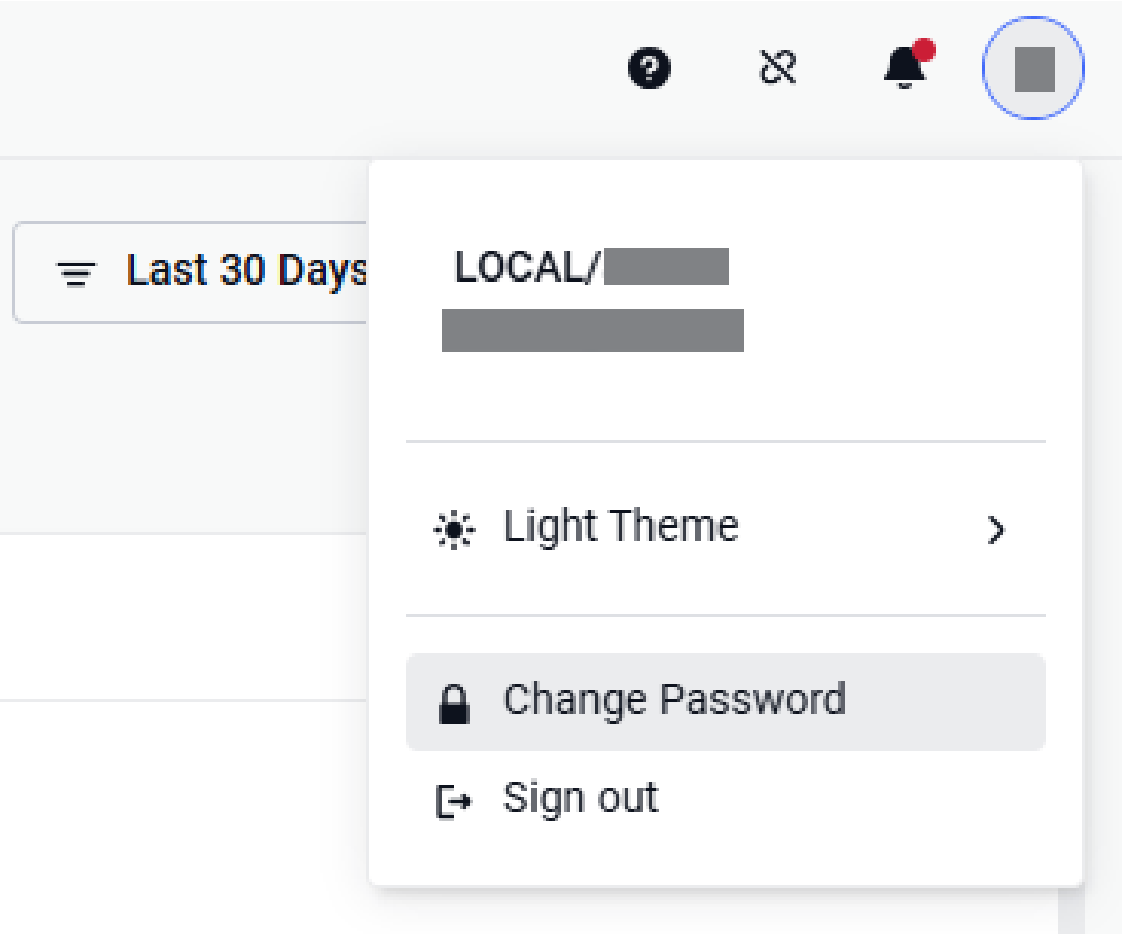
```
[DirectoryName]\[AD account]
```

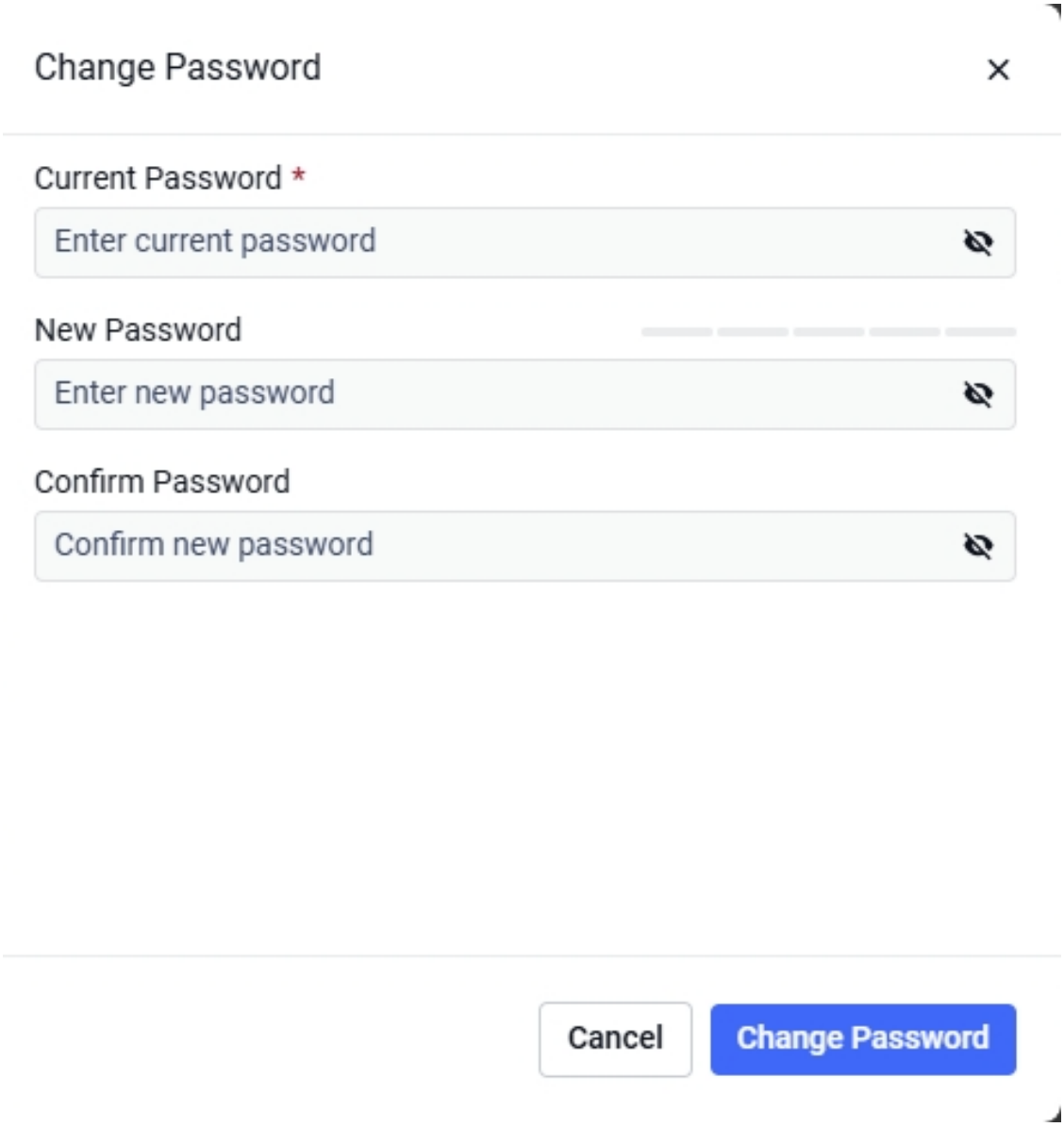
For example, if your AD directory is `My Directory` and your domain name is `mycompany` and your username is `ADUser`, you would log in as:

```
My Directory\ADUser
mycompany\ADUser
```

Change user password

The current user can change their own password, the option could be found under their username on the top right





Important notes

Warning

As long as TLS is not configured for the Web Management Console, passwords are sent clear-text over the network. To set up TLS see Enabling HTTPS.

Warning

If enhanced password policy is enabled for the user directory this user belongs to, then the new password must fulfill the password complexity requirements listed on the User directories) page.

Single Sign-On (SSO)

MetaDefender Core supports Single Sign-On (SSO) authentication where users can configure the product to

integrate with various widely used Identity Providers (IdP) on both SAML 2.0 and OpenID Connect protocols.

> **Info**
> MetaDefender Core consistently verifies the time frame validity of the sign-in session, so the system time on the device running MetaDefender Core needs to be configured accurately.

Learn more integration guidelines with the following specific IdPs.

IdP	OIDC	SAML
Amazon Cognito	Supported	
ForgeRock	Soon	Supported
Keycloak		Supported
Microsoft Active Directory Federation Services (ADFS)	Supported	Supported
Microsoft Entra ID	Supported	Supported
Okta	Supported	Supported
OneLogin	Supported	Supported
Ping Identity - PingFederate	Supported	Supported
Ping Identity - PingOne	Supported	Supported

User directory priority at login

MetaDefender Core can authenticate users against more than one user directory (for example, Local, LDAP, or Active Directory). Each directory has a priority order. That order controls which directory is used when a user signs in.

Plain username (no directory prefix)

If the user enters only a username (for example, `jsmith`), MetaDefender Core uses the highest-priority directory — the first directory in the configured order.

Username with directory name

If the user enters `DIRECTORY/USERNAME` (for example, `CORP-LDAP/jsmith`), Core looks up that directory. If more than one directory could match, Core tries them in priority order and uses the first one where authentication succeeds.

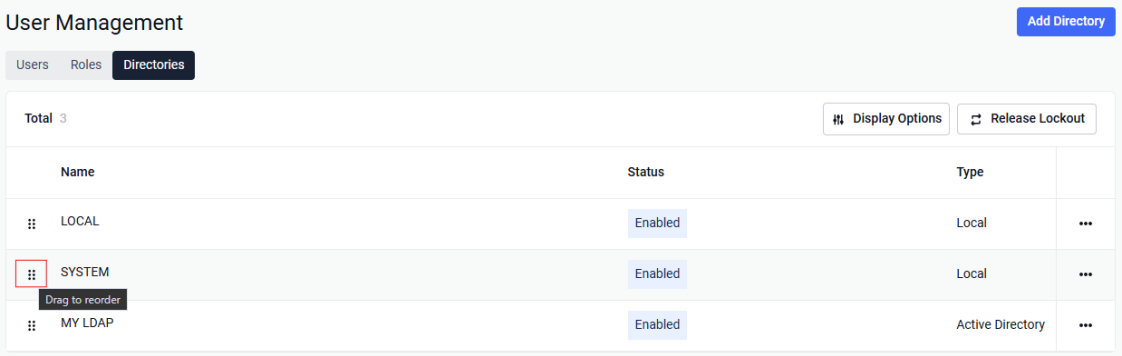
Why order matters

- Default login: Users who sign in without a directory prefix are checked against the top-priority directory only.
- Multiple LDAP/AD directories: Priority decides which directory is tried first when several could apply.
- Same username in different directories: Use `DIRECTORY/USERNAME` so Core knows which directory to use.

Managing directory order

In the Management Console, user directories are listed in priority order (highest first). Administrators can change that order to control which directory is used for plain-username logins.

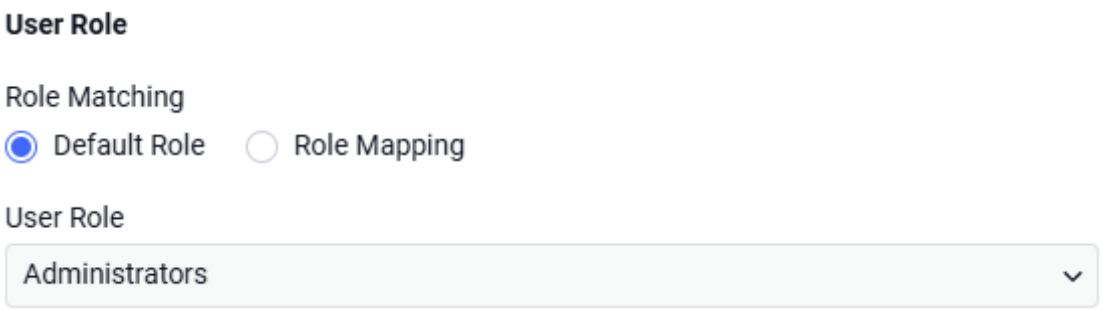
Tip for administrators: Put the directory most users should use (for example, your main Active Directory) at the top. Use `DIRECTORY/USERNAME` when users must sign in through a specific directory.



User role mapping

Motivation

MetaDefender Core supports two types of role assignments for signed-in users. The first is `Default role`, which is the standard option and fits most simple use cases. If this option is selected, a fixed role is assigned to users when they successfully sign in to MetaDefender Core with their accounts. This role is specified when the SAML or OIDC directory is created. In the picture below, Administrators role will be assigned to every signed-in user.

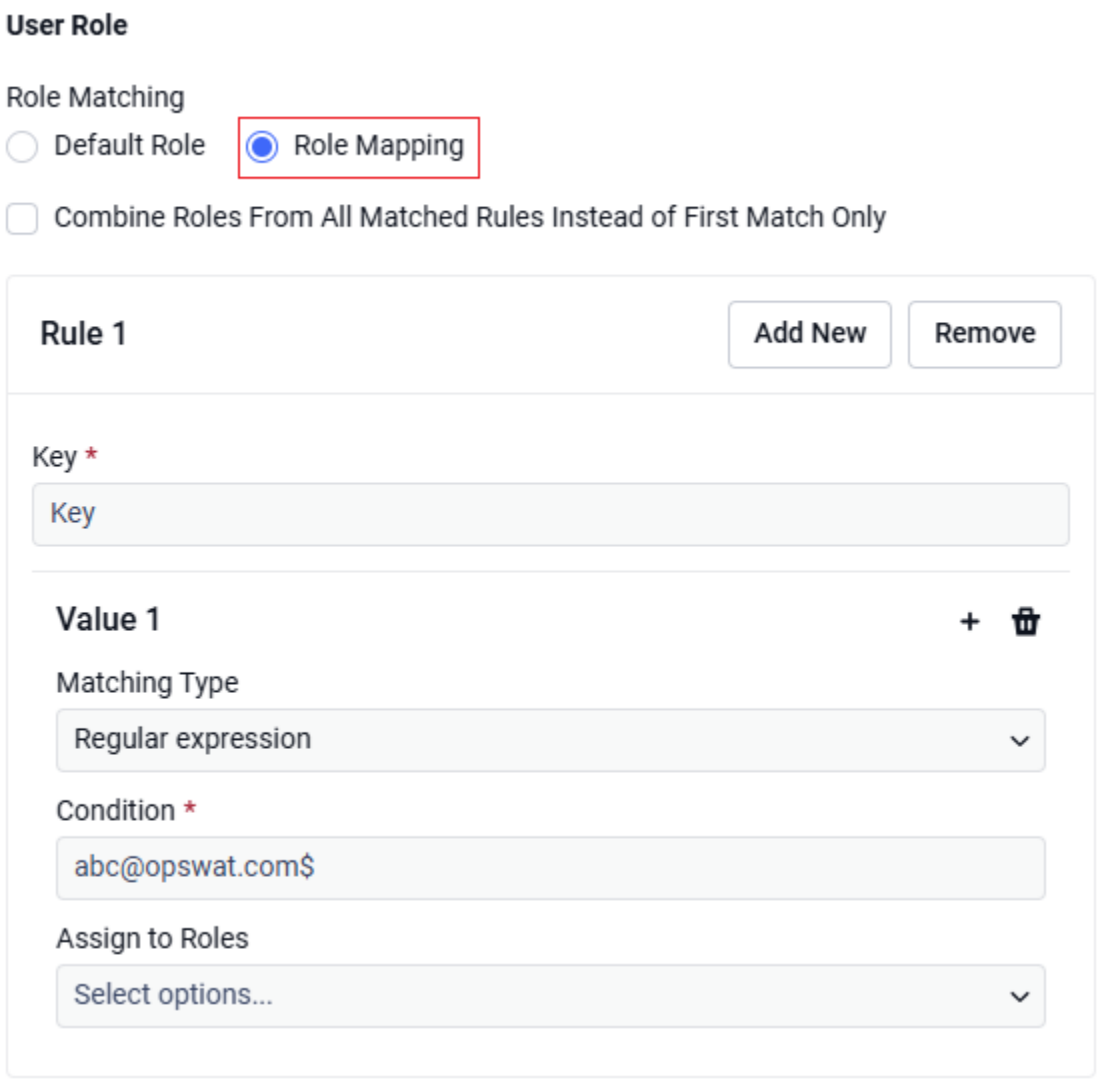


However, in some organizations where employees are structured into groups with separate privileges, assigning the same role to all users in MetaDefender Core does not make sense. One common use case is that the Editor role is assigned to a group of system experts who can control and update MetaDefender Core configurations to meet the organization's needs, while the Viewer role is assigned to the team leaders who can only view the configurations. For all other user groups, they should only use MetaDefender Core features without the ability to log in, view, or update

configurations. MetaDefender Core supports Role Mapping to help build flexible rules for assigning appropriate roles to groups of users based on attributes provided by the IdP when the user successfully signs in.

General idea

Role Mapping consists of a series of rules that are checked one after another by MetaDefender Core until the first one matches the attribute values provided by the IdP or until none match, causing MetaDefender Core to refuse the user's sign-in. The following shows a sample plain rule.



Each rule includes a single key name from which to take the value, a list of conditions, and roles that will be assigned to the user if their conditions are satisfied. The key name is the attribute name which is defined by the IdP and can be modified or created by the system administrator on the IdP page. In the current version of MetaDefender Core, only Regular Expressions can be selected as the matching type. The input under `Condition` is a valid regular expression that MetaDefender Core will use to match against the value of the key. Multiple matching rules can be added by clicking `Add New` button. If multiple rules are defined, each rule is checked sequentially, following their defined order.

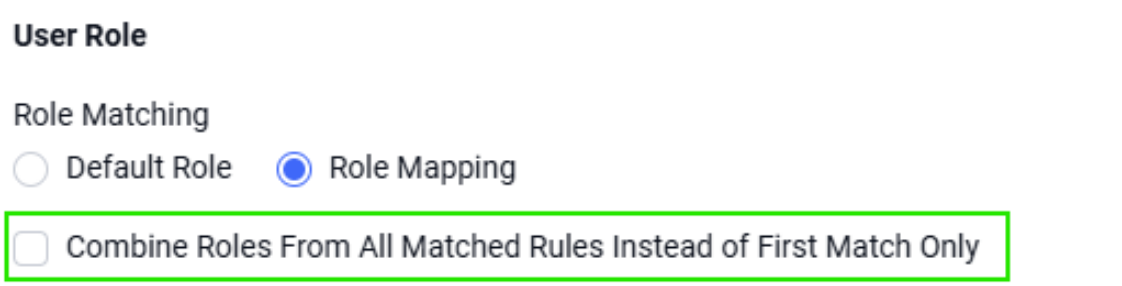
There should be a case where a single key can be used to match multiple conditions. The system administrator can click the plus button next to `VALUE` title to add more conditions. Conditions are checked in the order they are listed when the rule is evaluated by MetaDefender Core.

How User Roles Are Assigned Based on Matched Conditions

Info

Starting from **MeteDefender Core 5.18.0**, user can decide how user roles are assigned during role mapping using the option `Combine roles from all matched rules instead of first match only`:

- **Disabled (default):** Roles from the first matching condition are assigned to the user.
- **Enabled:** If a user matches multiple conditions, roles from all matching rules are combined and assigned instead of only the first match.



Building rules

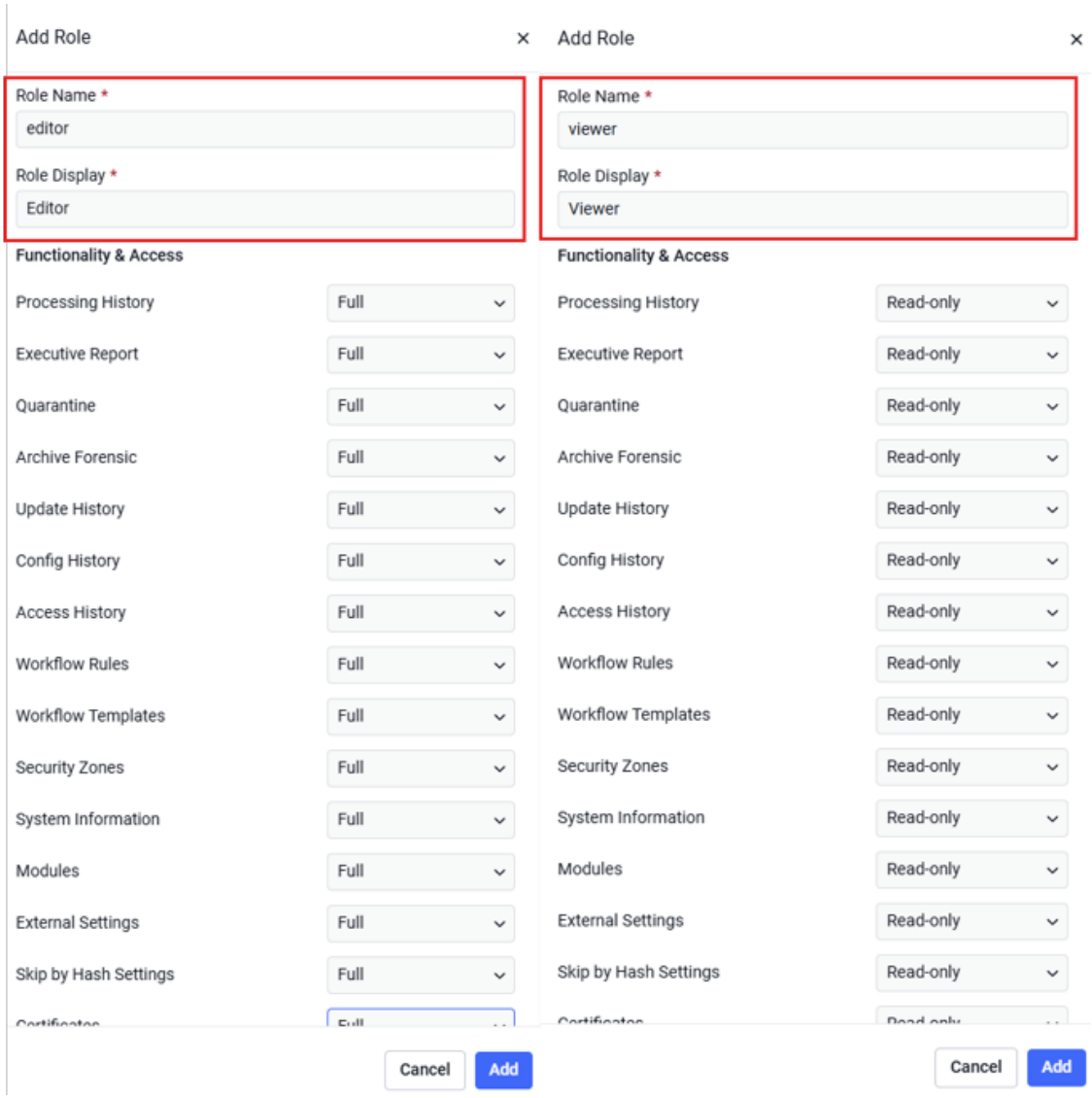
We remind you of the use case mentioned in Motivation section, in which the Editor role is assigned to the group of system experts, the Viewer role to the group of team leaders, and no role to other groups.

Info

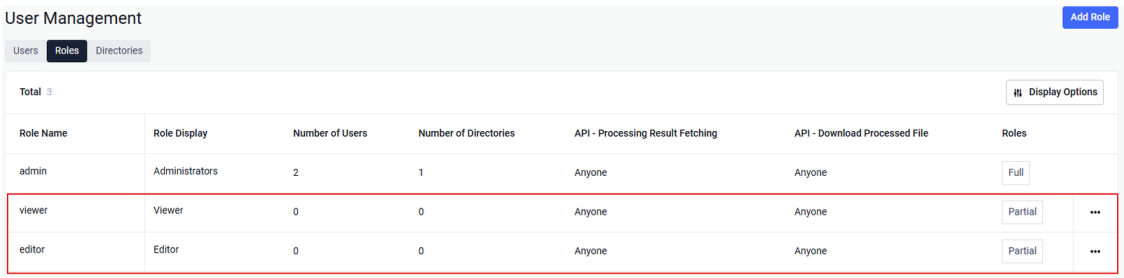
You are advised to successfully set up SSO integration between your preferred IdP and MetaDefender Core using the default role option before attempting to apply role mapping.

Suppose the system administrator has already created an attribute named "group" in the preferred IdP and set its value either *expert*, *leader*, or *normal_user*. The new attribute must be configured to be returned in the authentication response from the IdP when a user signs in successfully. The following are steps to build role mapping rules based on "group" attribute in MetaDefender Core.

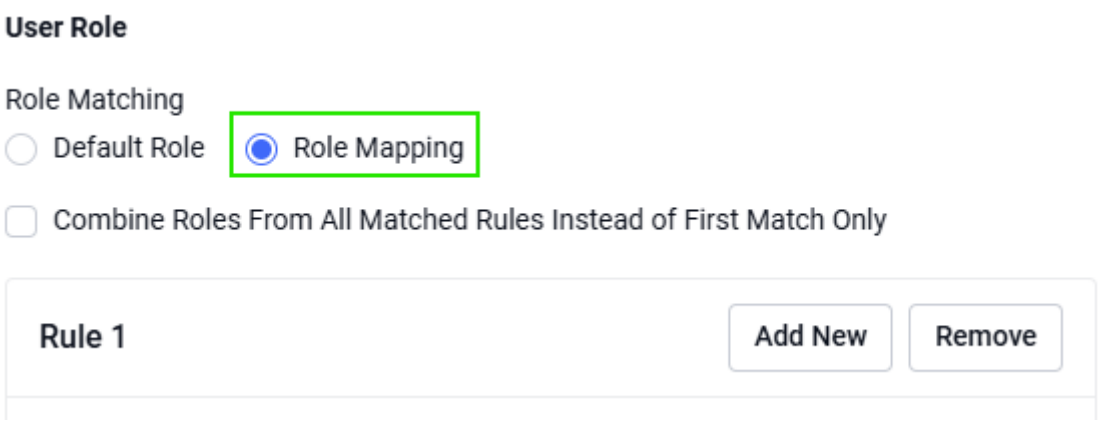
1. Sign in to MetaDefender Core.
2. From the dashboard, navigate to `User Management` > `Roles`.
3. Click `Add Role` to add Editor role and assign it appropriate rights.
4. Do the same for Viewer role.



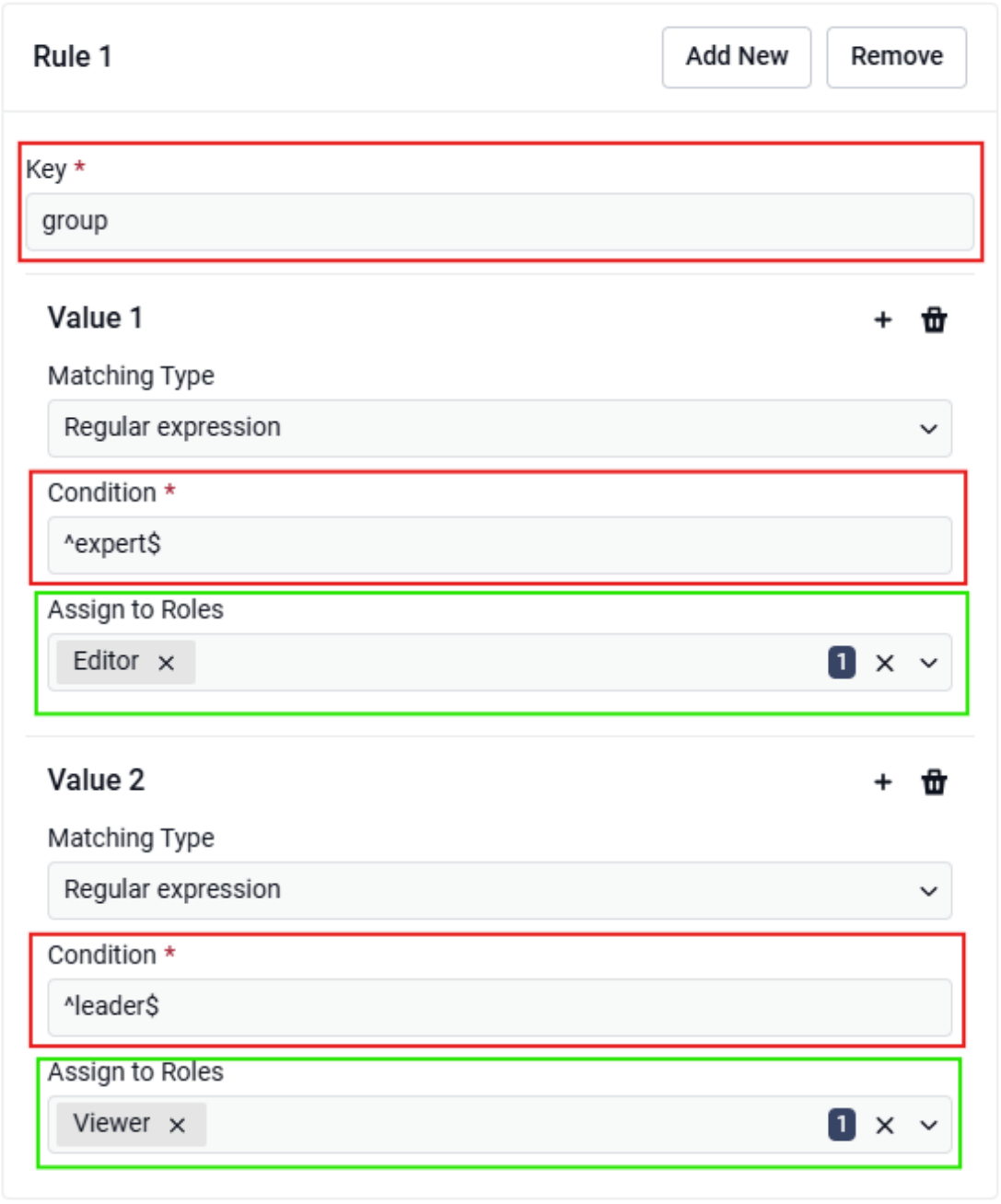
5. Click `Add` to complete the assignment of rights to the role.



6. Create a new SAML or OIDC directory and follow steps to set up the SSO integration between your preferred IdP and MetaDefender Core. At the step of user role selection, choose `Role mapping` option.

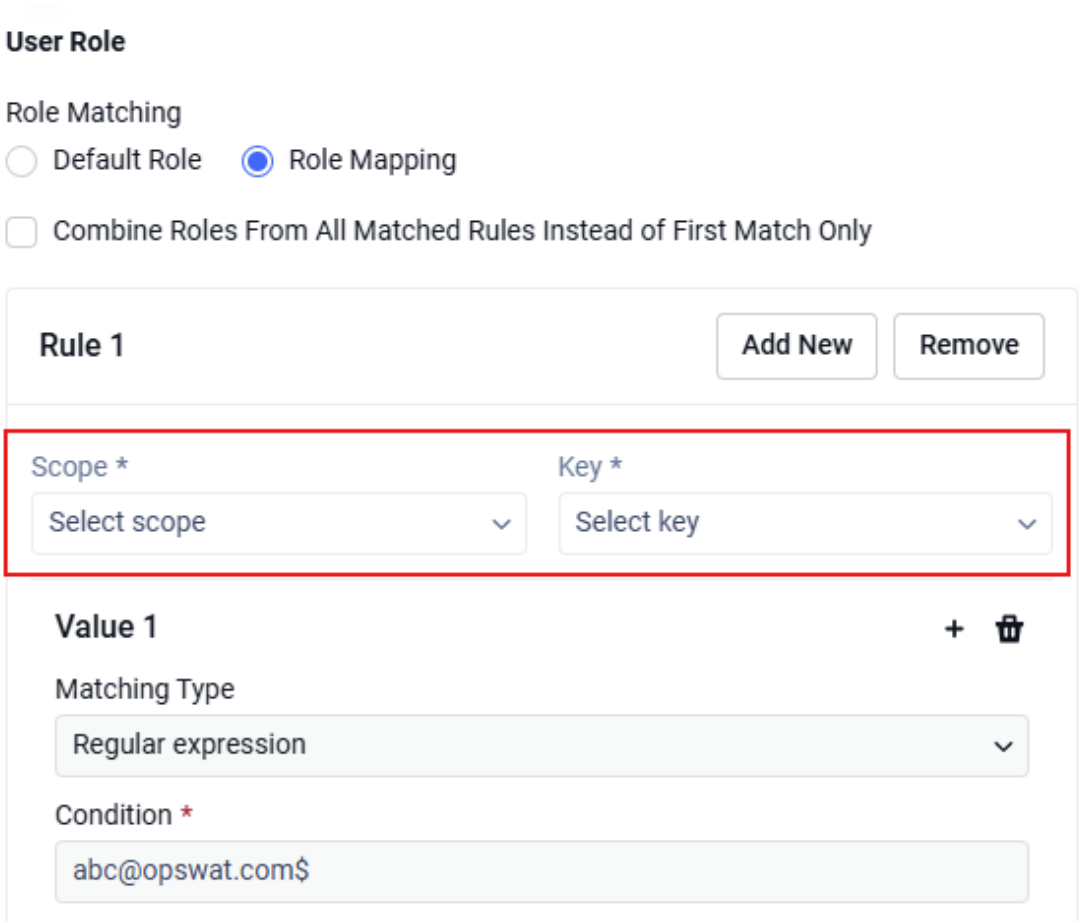


7. Fill in `Key` field with "group".
8. Click the plus button to define an additional `VALUE 2`.
9. `VALUE 1` will attempt to match the value of "group" key against the regular expression ^expert$, while the second will do so against ^leader$. If the first matches perfectly, the user will be assigned the Editor role, while the latter will assign the Viewer role. We do not define a rule for *normal_user*, which means users in this group will always fail to sign in to MetaDefender Core.



10. Click `Add` to complete the settings.
11. In `User Management` screen of MetaDefender Core, toggle the new created directory. A dialog box appears to confirm the action. Once `Enable` is clicked, all sessions are expired immediately.
12. Click `Login` from the home screen of MetaDefender Core and sign in with your account registered in IdP. If everything goes right, the MetaDefender Core dashboard is displayed with the user identity set in the top right corner. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

For the OIDC directory, as a key belongs to a specified scope, the system administrator must specify the key and its associated scope when defining the matching rules. Please refer to the IdP documentation to determine which key belongs to which scope.



Multiple Login URLs

Overview

MD Core now supports **multiple** login/redirect URLs as part of the SSO authentication flow. This enhancement allows deployments spanning multiple domains or client applications to dynamically route users to the correct redirect endpoint during login.

Key benefits

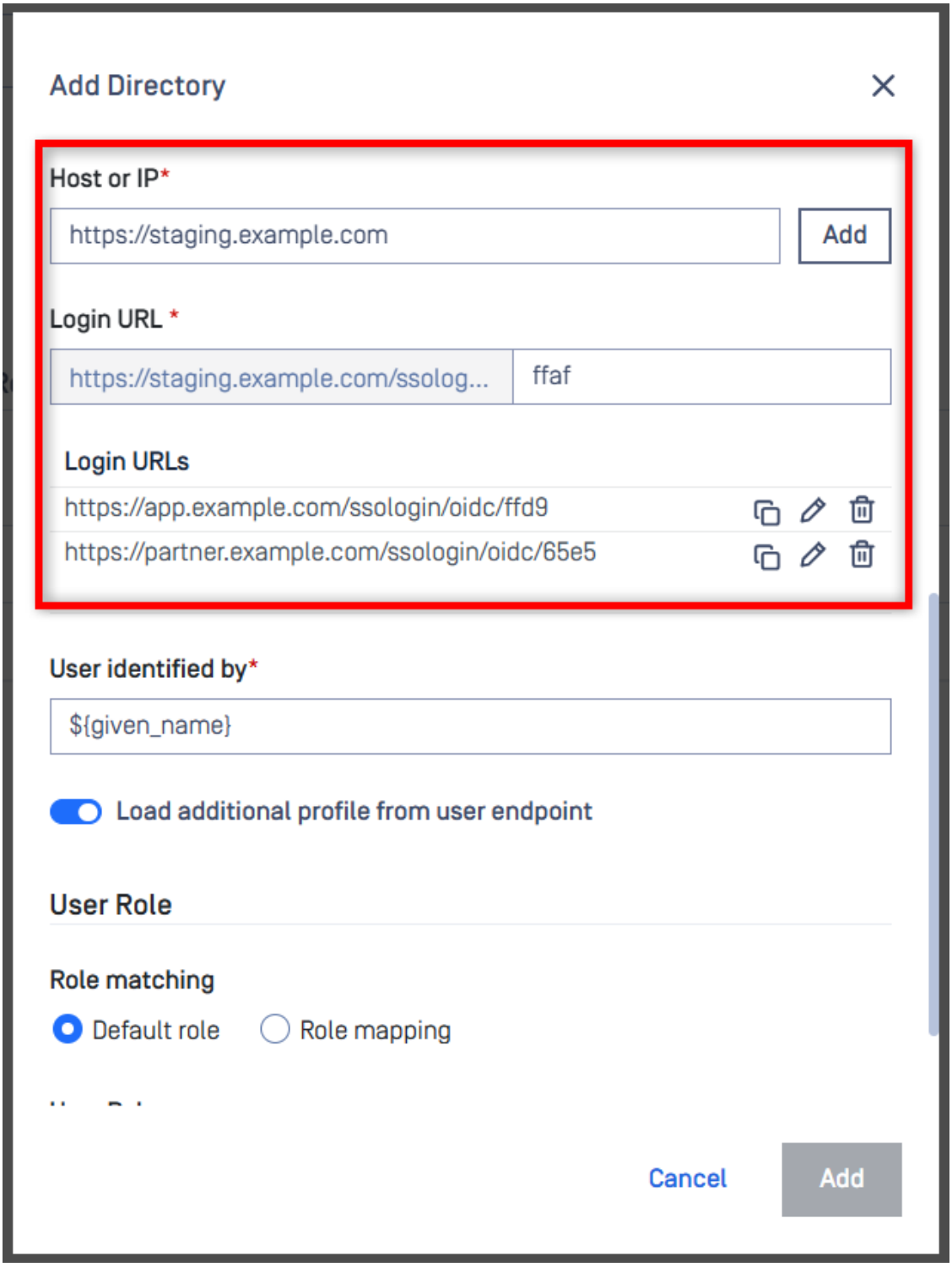
- Supports multi-domain SSO scenarios.
- Enables multiple client application redirects.
- Improves authentication flexibility without custom logic.

Info

- The URLs must be registered on both MD Core and the Identity Provider.
- Unregistered redirect URLs are rejected for security reasons.

Configuration Steps

1. Sign in to the MD Core dashboard using an administrator account.
2. Go to `User Management` > `Directories`.
3. Click `Add Directory` , or choose an existing SSO directory to modify.
4. If you are adding a new directory, select **OIDC** or **SAML** as `Directory Type` and fill in the required configuration fields.
5. Under the `Host or IP` section, enter your login URL and click `Add` .
6. In your Identity Provider configuration panel, add the **same login URL** as the **redirect/callback URL**.
7. Repeat steps 5–6 for any additional login URLs you need to support.



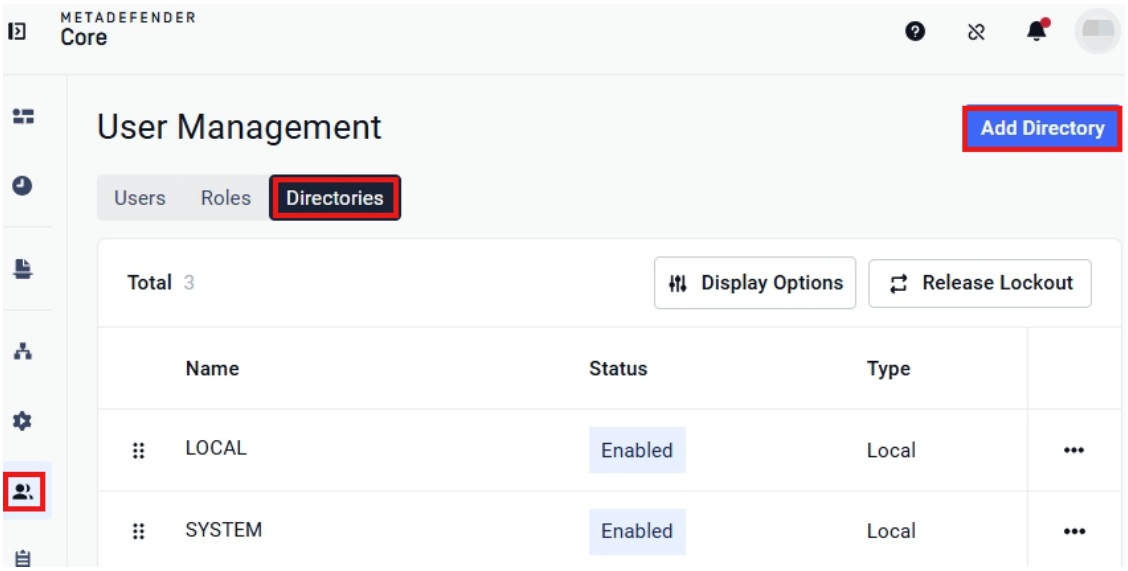
Amazon Cognito

The instructions guide users on how to configure Amazon Cognito OIDC Single Sign-on for MetaDefender Core.

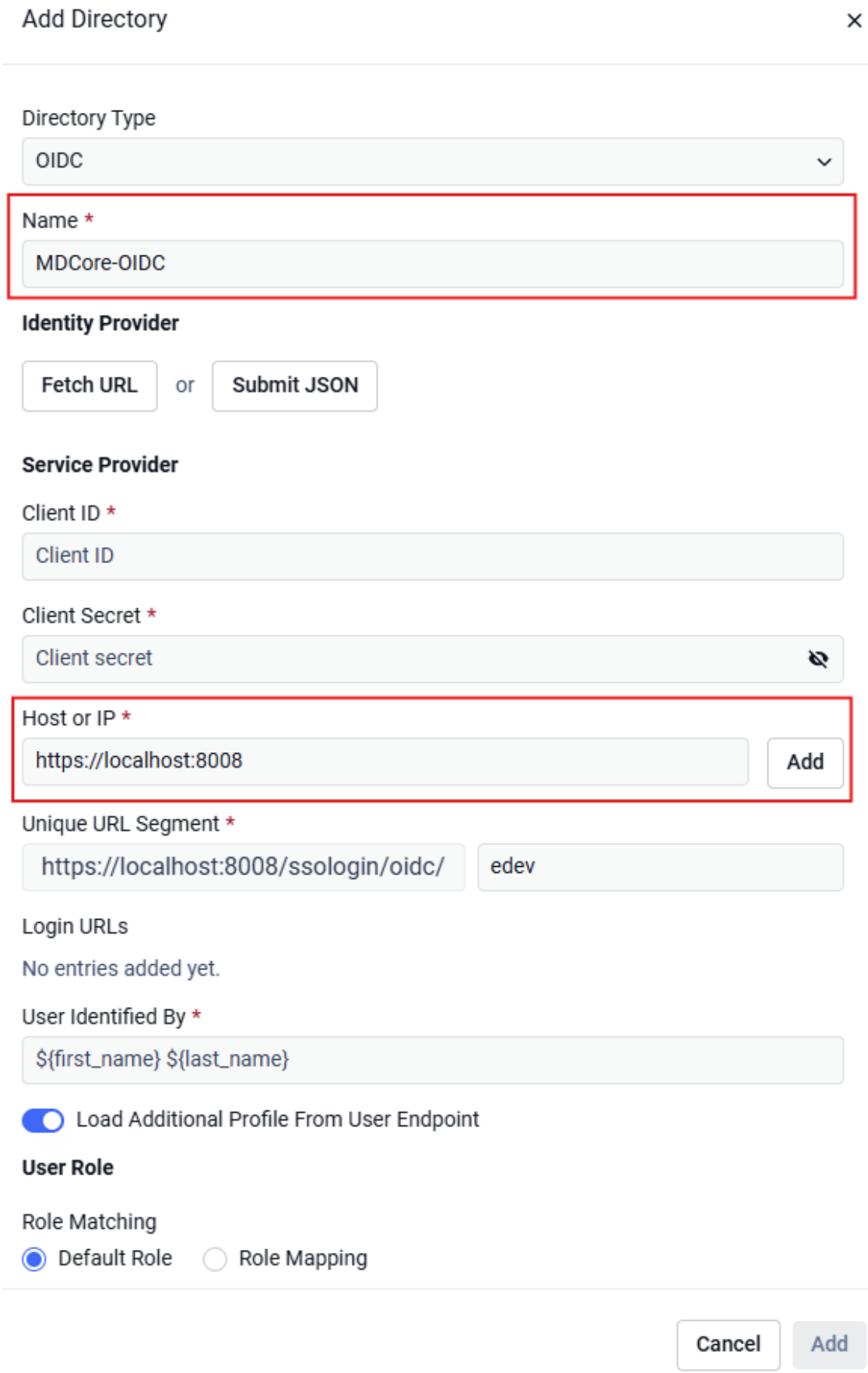
Configure OIDC SSO

Create OIDC directory in MetaDefender Core

1. Sign in to MetaDefender Core.
2. In the dashboard, click on `User Management` in the left sidebar.
3. In `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.



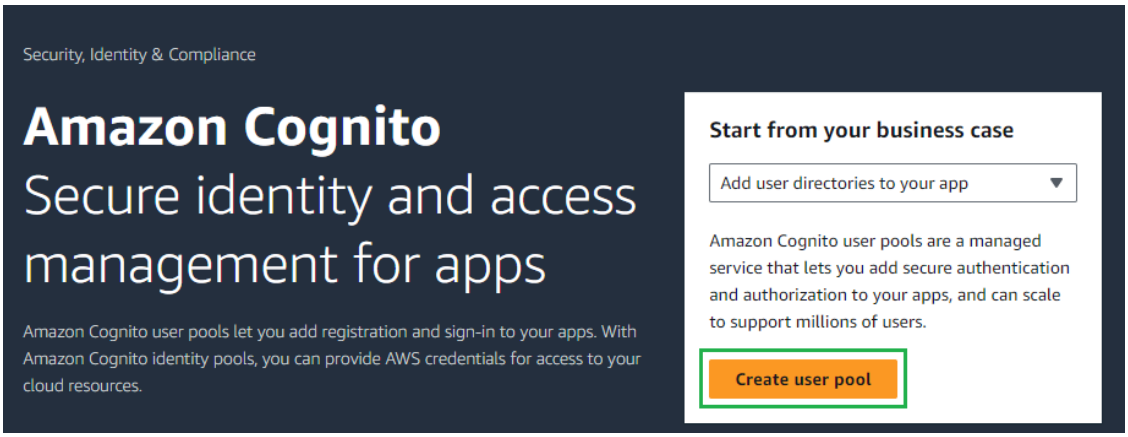
4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCore-OIDC.
5. Enter `Host or IP` where MetaDefender Core is hosted, for this example use http://localhost:8008.
6. Copy the string generated under `Login URL` and store it as *reply_uri*.



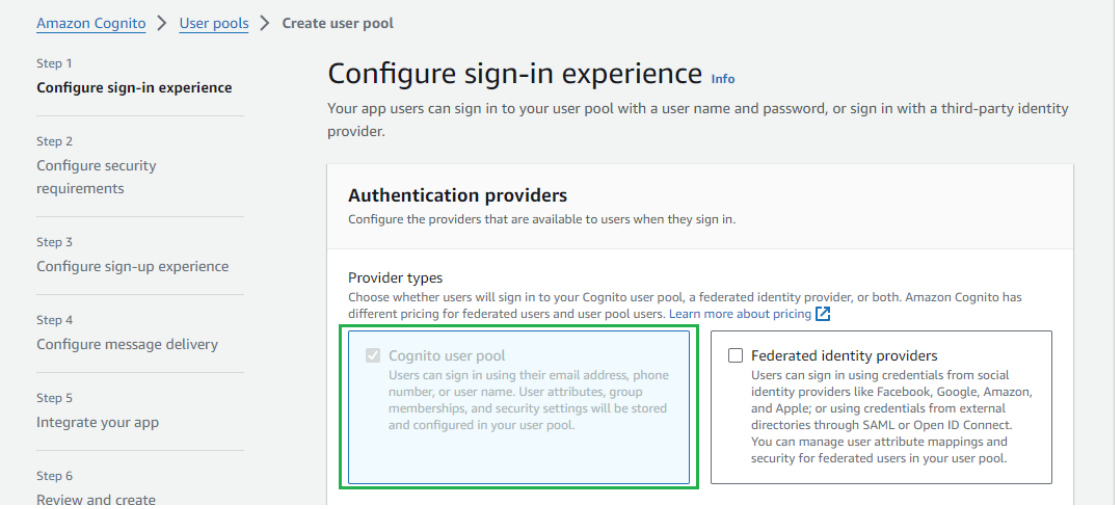
Create Amazon Cognito user pool

1. Access https://aws.amazon.com/cognito/v2/home and sign in with your account.

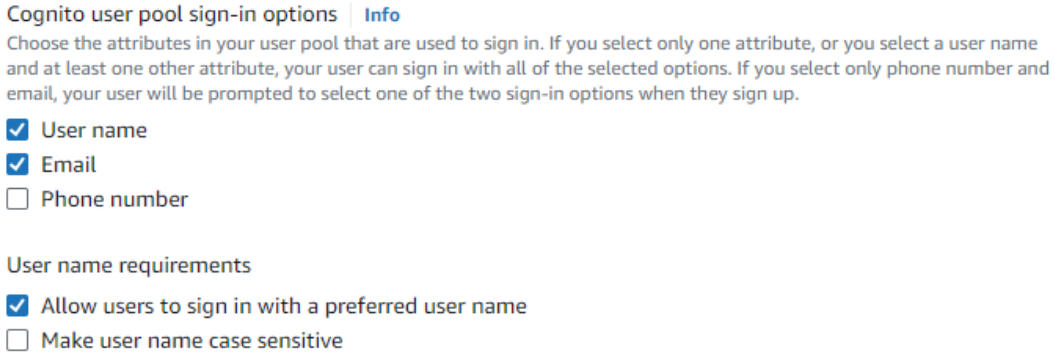
2. On the main page, click `Create user pool`.



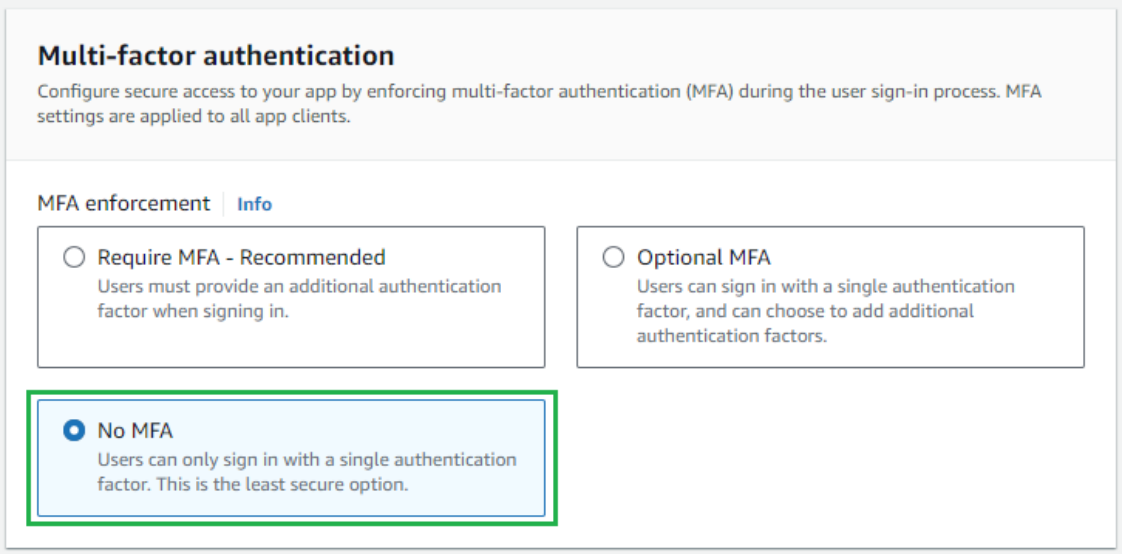
3. In `Step 1 - Configure sign-in experience`, select `Cognito user pool` for `Provider types`.



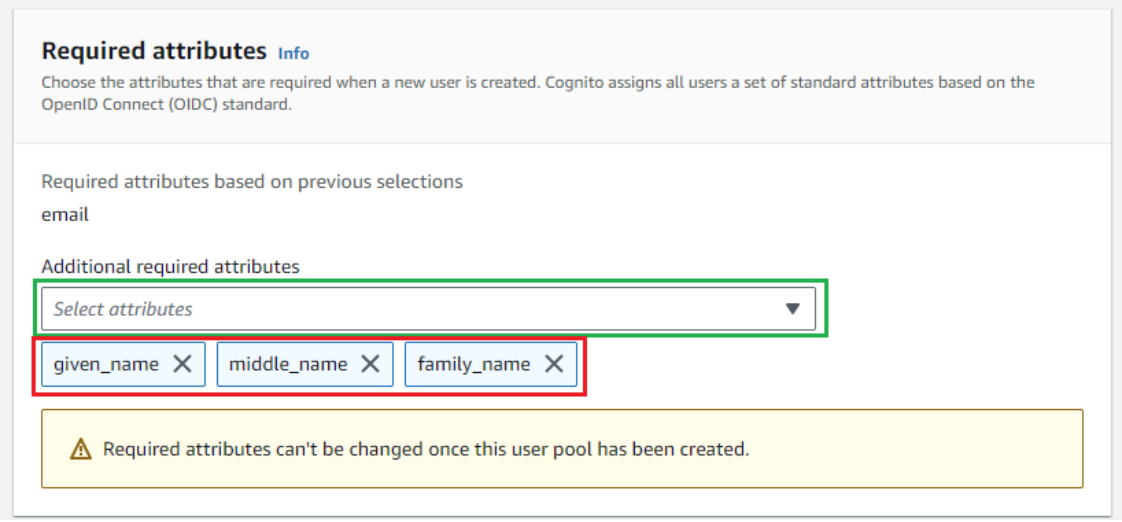
4. Tick `User name`, `Email`, and `Allow users to sign in with a preferred user name`, then click `Next`.



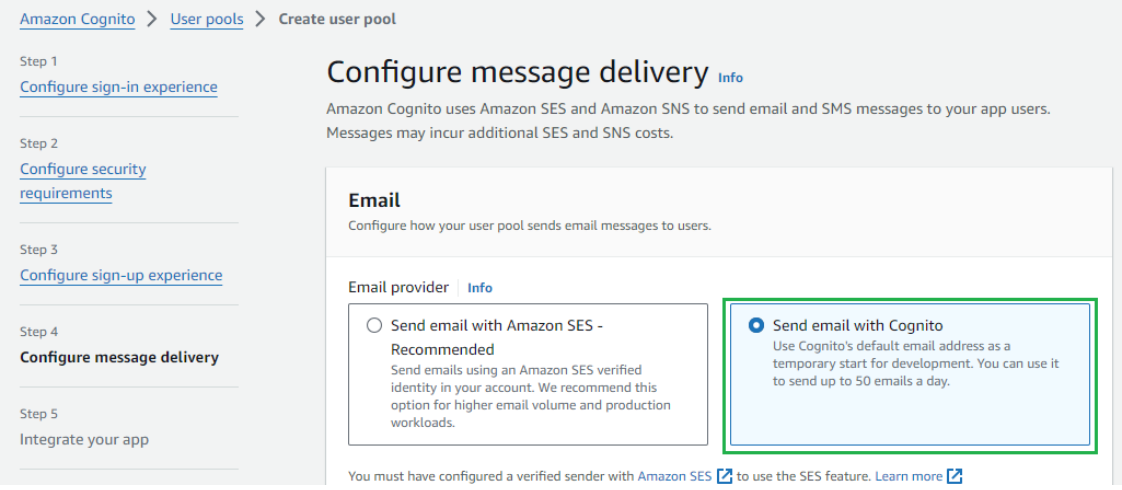
5. In `Step 2 - Configure security requirements`, select `No MFA` for `MFA enforcement`, leave all others as default , and click `Next`.



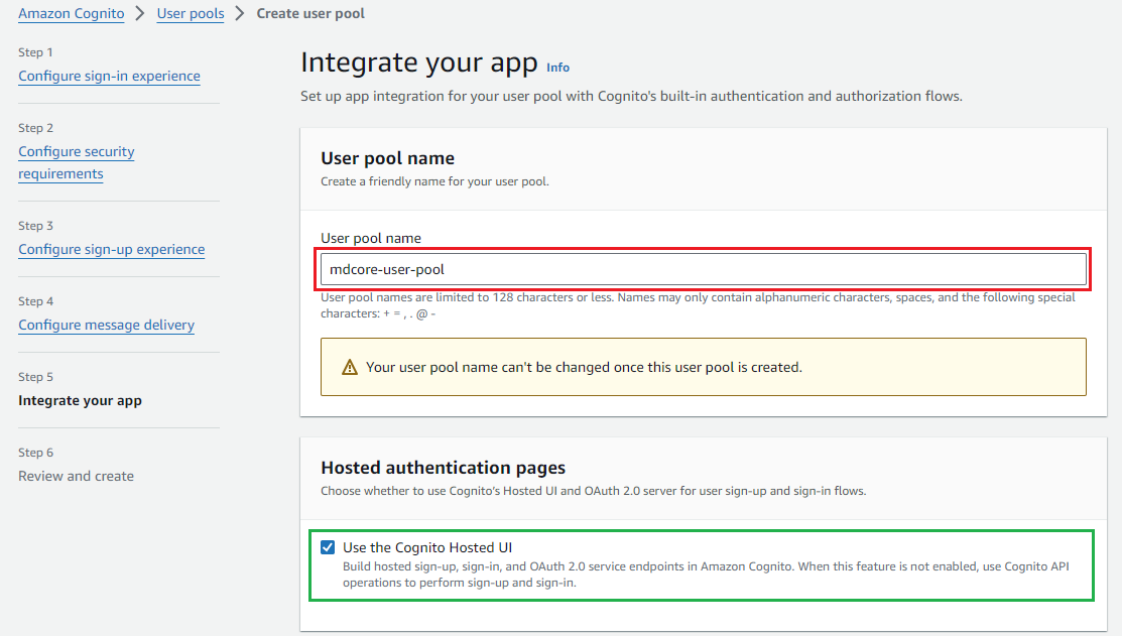
6. In `Step 3 - Configure sign-up experience`, navigate to `Required attributes` and add additional attributes: given_name, middle_name , and family_name. Click `Next`.



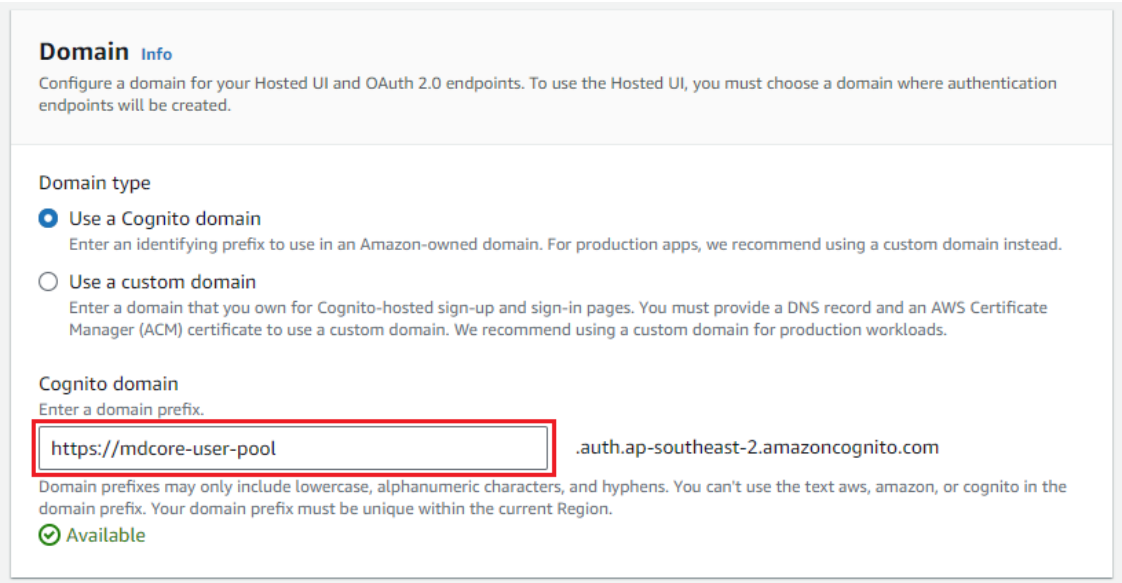
7. In `Step 4 - Configure message delivery`, choose `Send email with Cognito` and click `Next`.



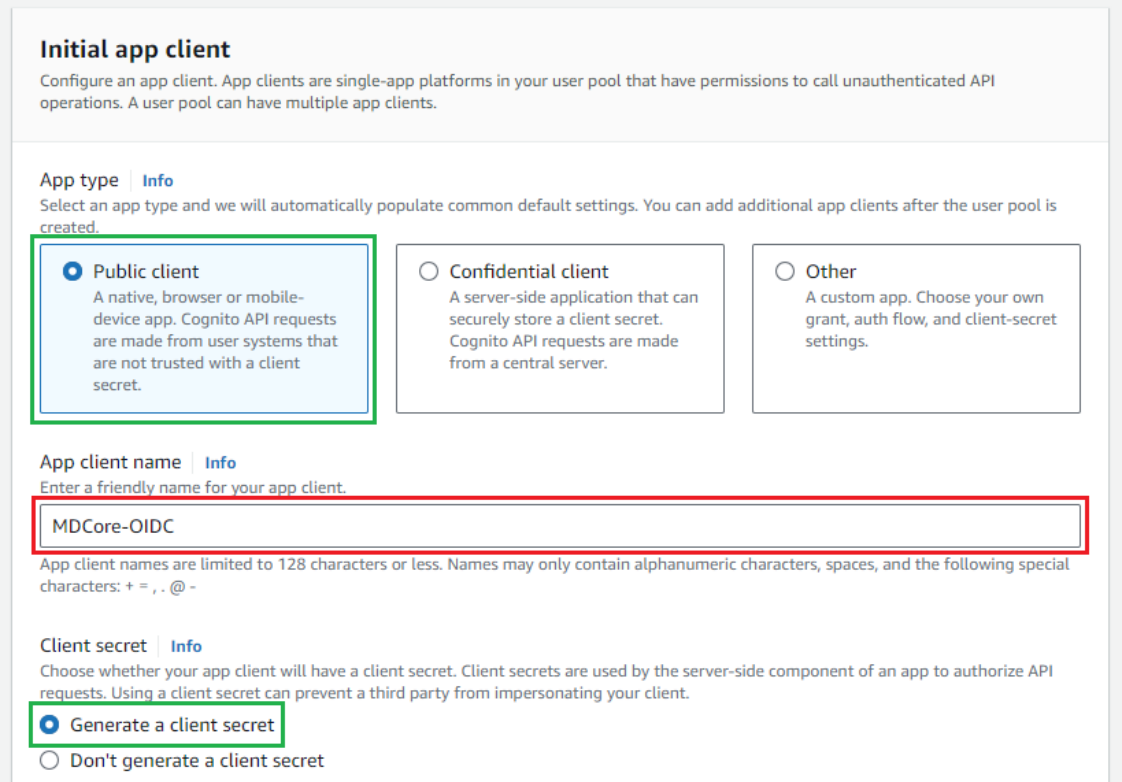
8. In `Step 5 - Integrate your app`, enter the user pool name and tick `Use the Cognito Hosted UI`.



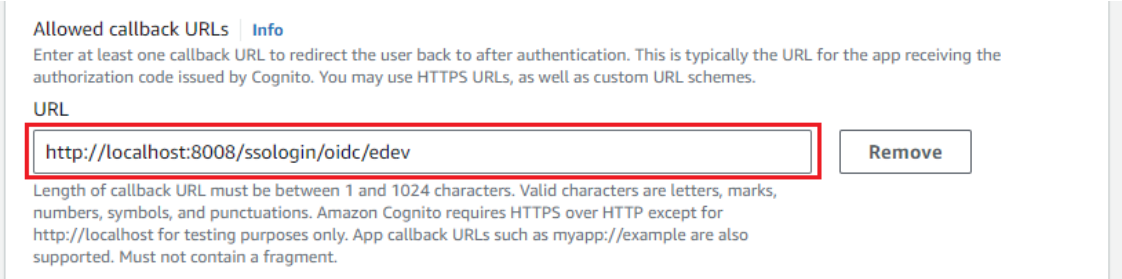
9. Enter your domain to `Cognito domain`.



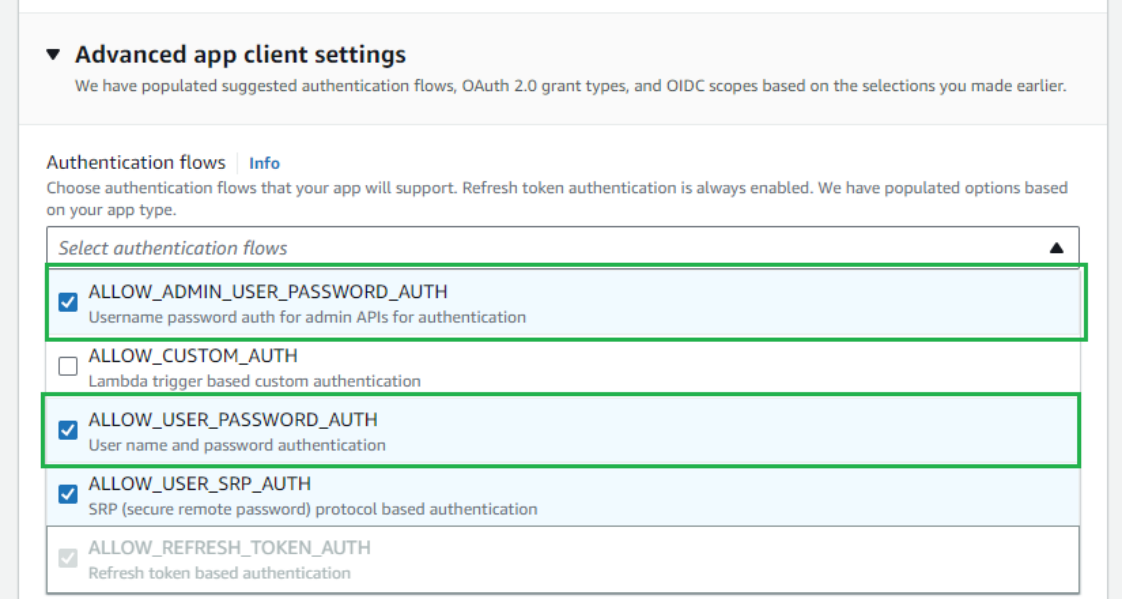
10. Under `Initial app client` section, select `Public client` for `App type`, enter `App client name`, and select `Generate a client secret` for `Client secret`.



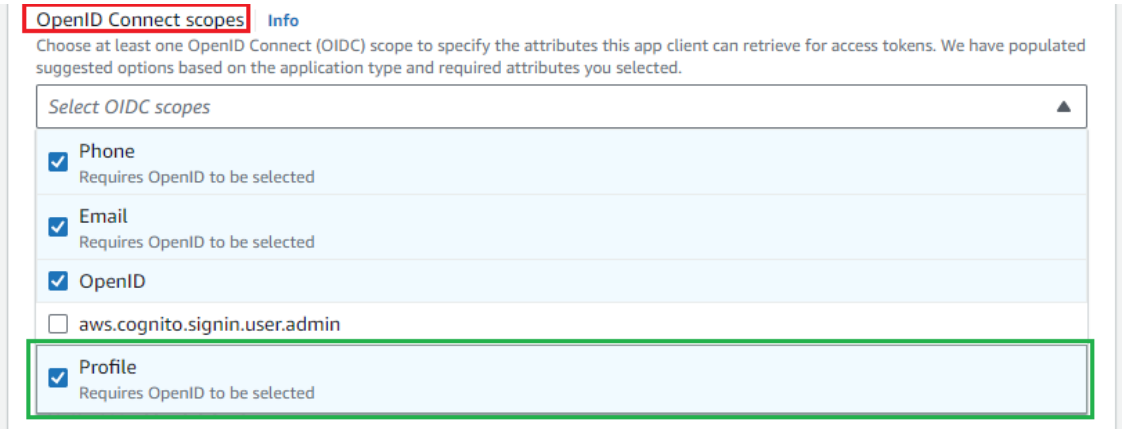
11. Under `Allowed callback URLs` section, fill in `URL` box with the value of *reply_uri*.



12. Expand `Advanced app client settings`, then stick to ALLOW_ADMIN_USER_PASSWORD_AUTH and ALLOW_USER_PASSWORD_AUTH for `Authentication flows`.



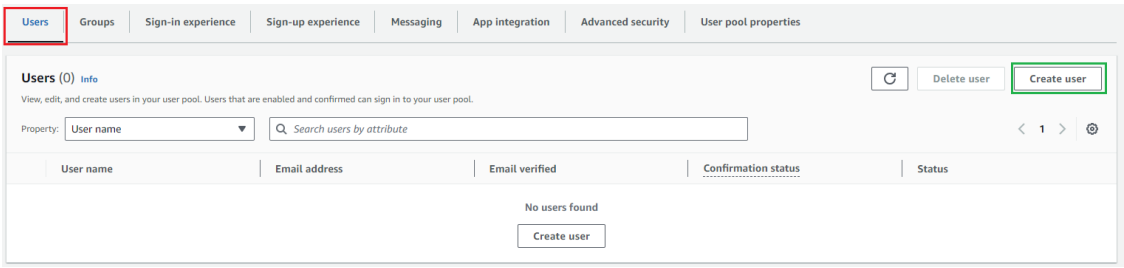
13. Navigate to `OpenID Connect scopes` and add `Profile` scope.



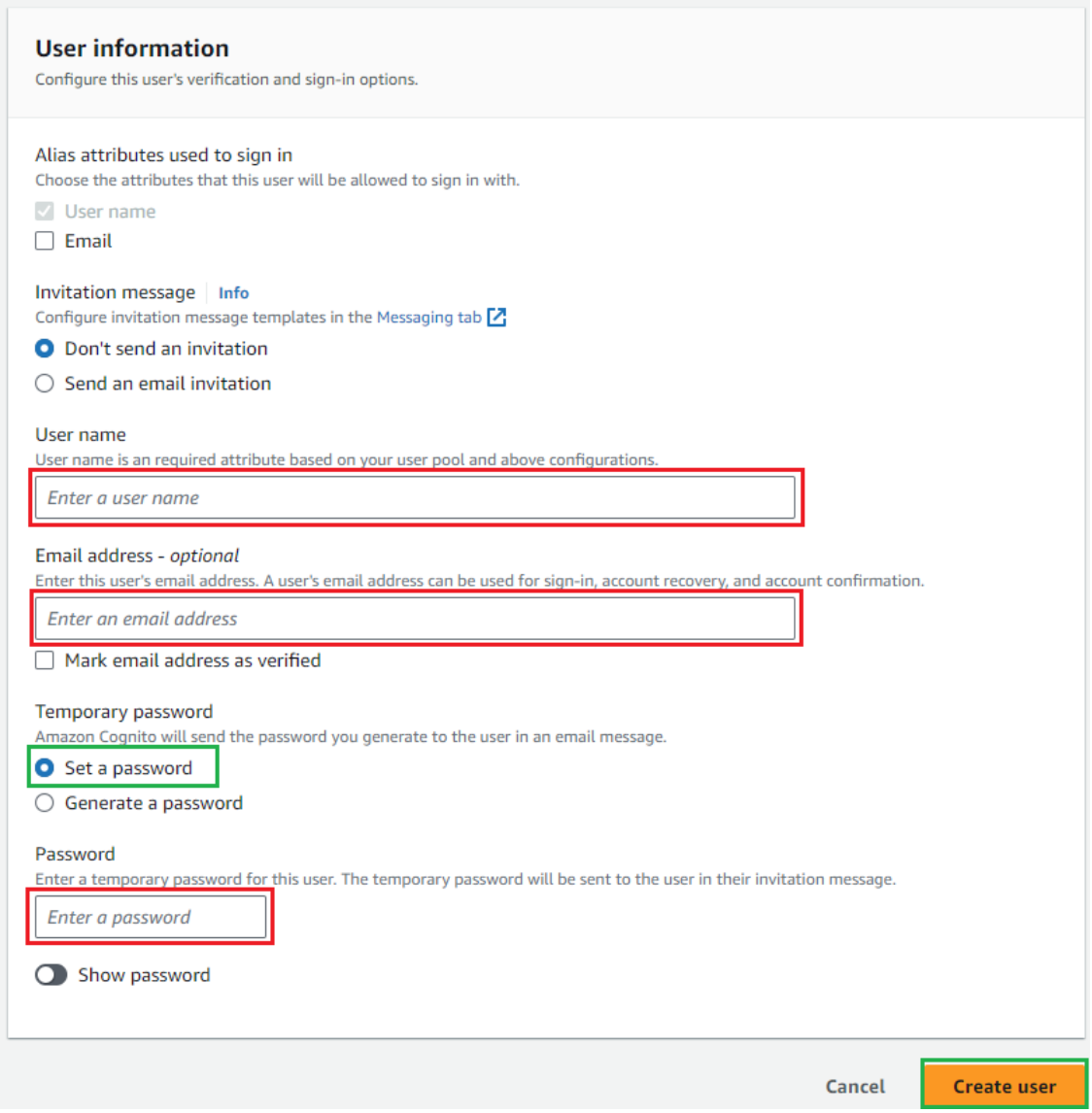
14. Click `Next` in the final review page and click `Create user pool`.

Create Amazon Cognito user

1. In Amazon Cognito, click `User Pools` in the left sidebar and select the user pool that has been created.
2. In `Users` tab, click `Create User`.

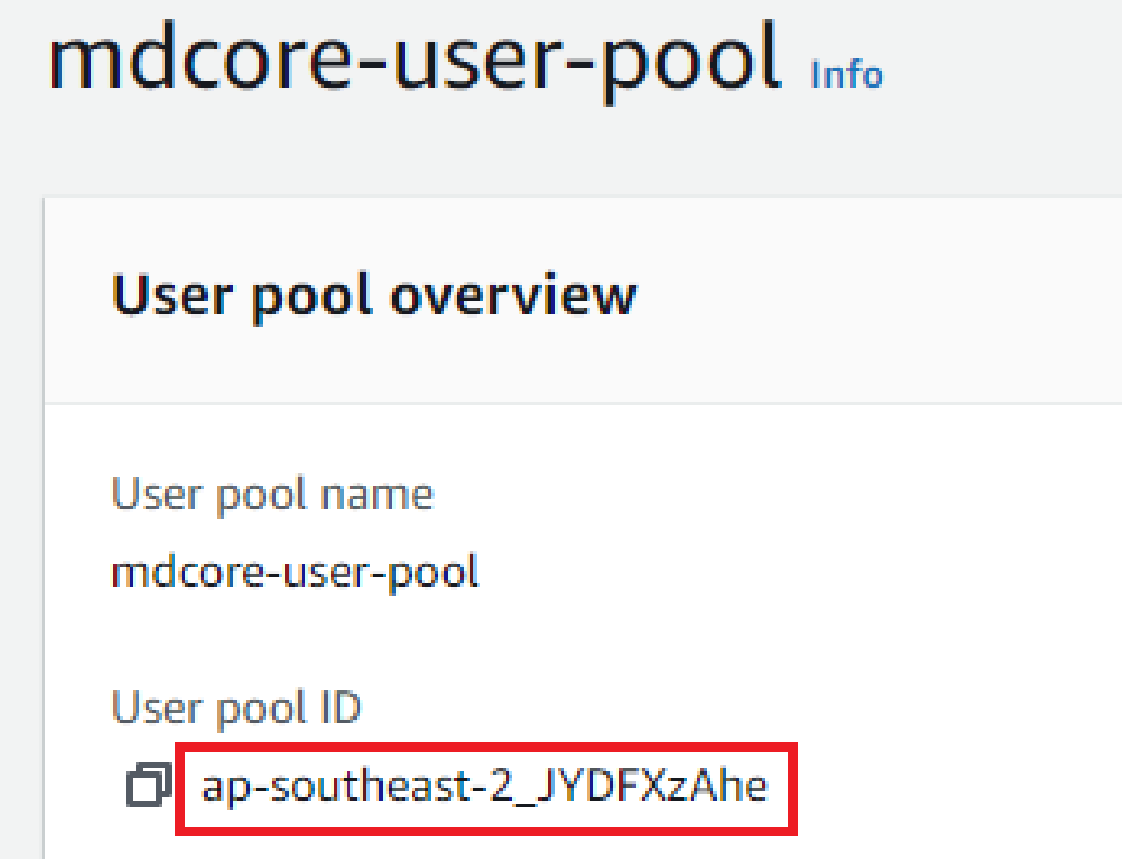


3. Enter the user name, and optional email address; stick to `Set a password`, and enter your temporary password. Click `Create User` to complete.



Download and modify ODIC metadata

1. Start your favorite web browser and enter a URL in the following format: `https://cognito-idp.{region_where user_pool_is_created}.amazonaws.com/{user_pool_id}/.well-known/openid-configuration`, where `user_pool_id` can be found in `User pool overview`.



2. Save the responded content to a file on disk.

3. Open the file in your favorite editor.

4. Add the following content to the beginning of the file, right after the first open brace.

Code · json

```
"claims_supported": ["middle_name", "given_name", "family_name"],
```

5. Save the file.

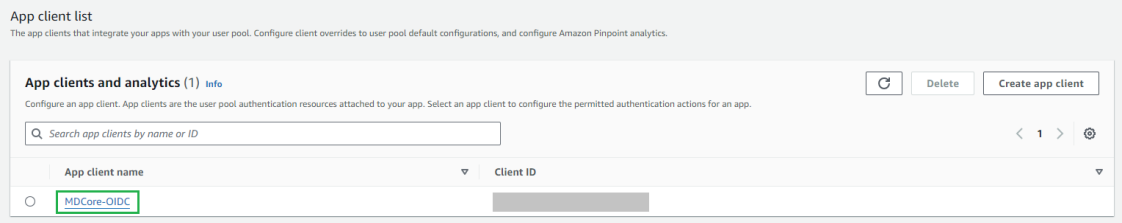
Info

`claims_supported` includes items that are selected for `Required Attributes` at step 6 of Create Amazon Cognito user pool stage.

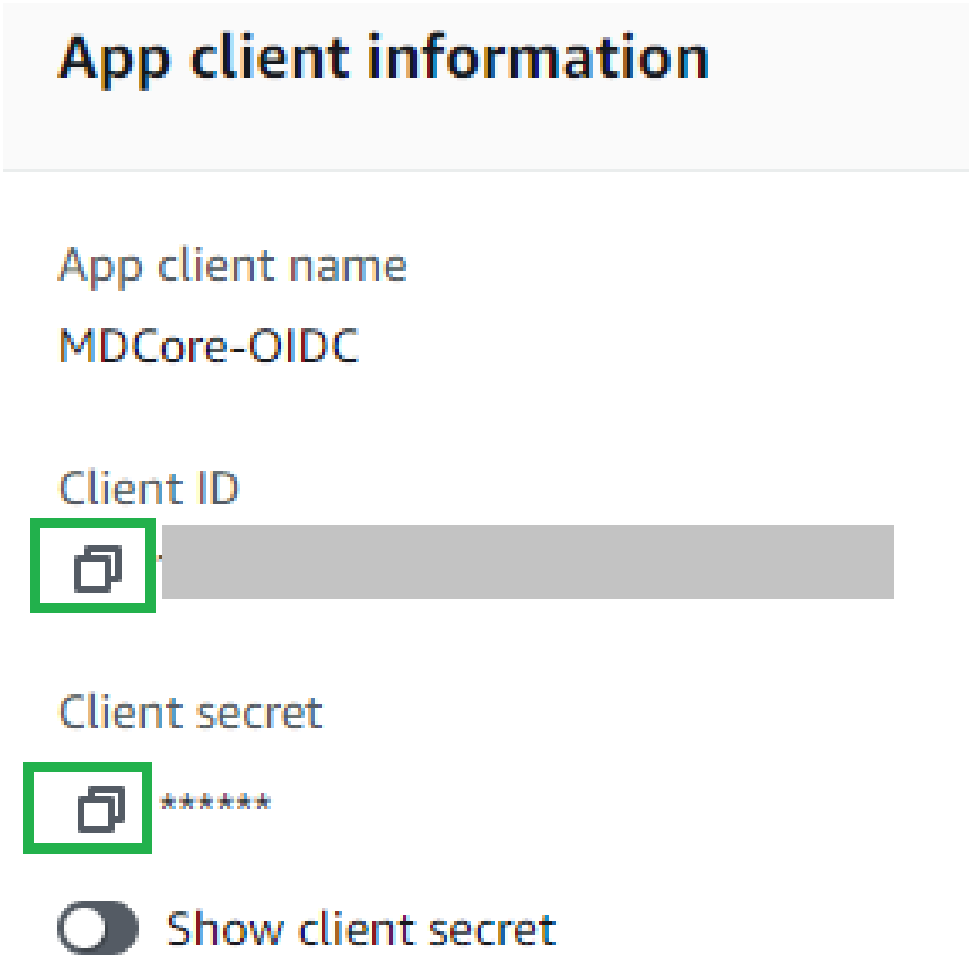
Complete the configurations

1. In Amazon Cognito, click on `User Pools` in the left sidebar and select the user pool that has been created.

2. In `App Integration` tab, navigate to `App Client list` and select the app client that has been created.



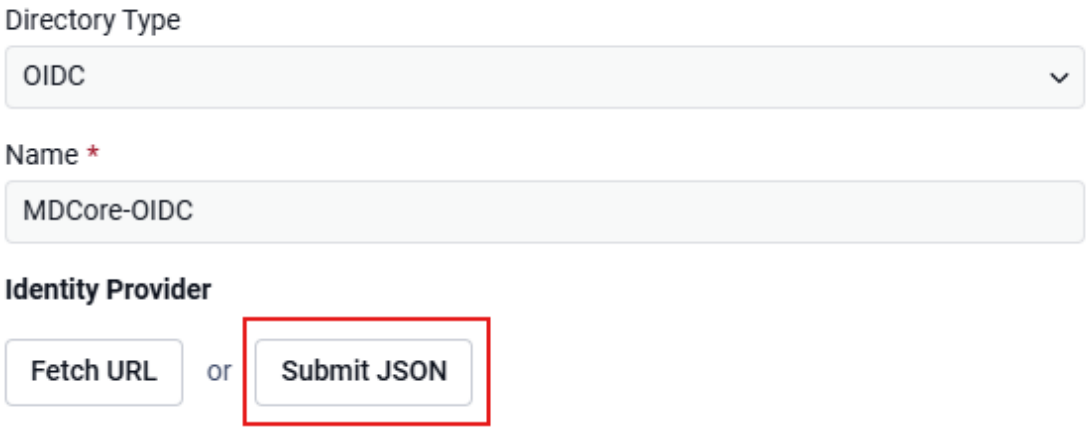
3. Copy `Client ID` and `Client Secret`.



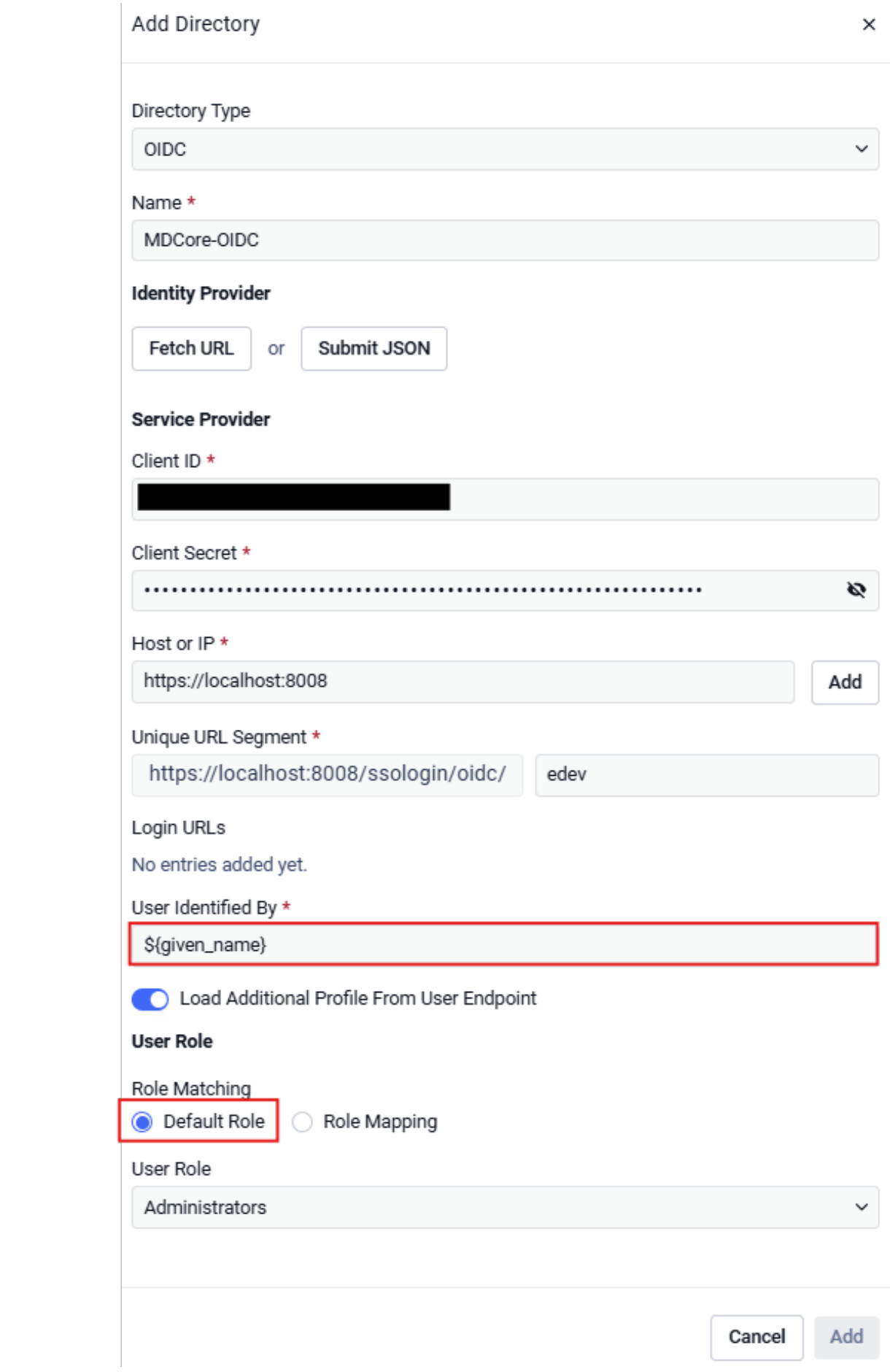
4. Switch to MetaDefender Core, under `Service Provider`, paste `Client ID` and `Client Secret` from the previous step into their respective fields.



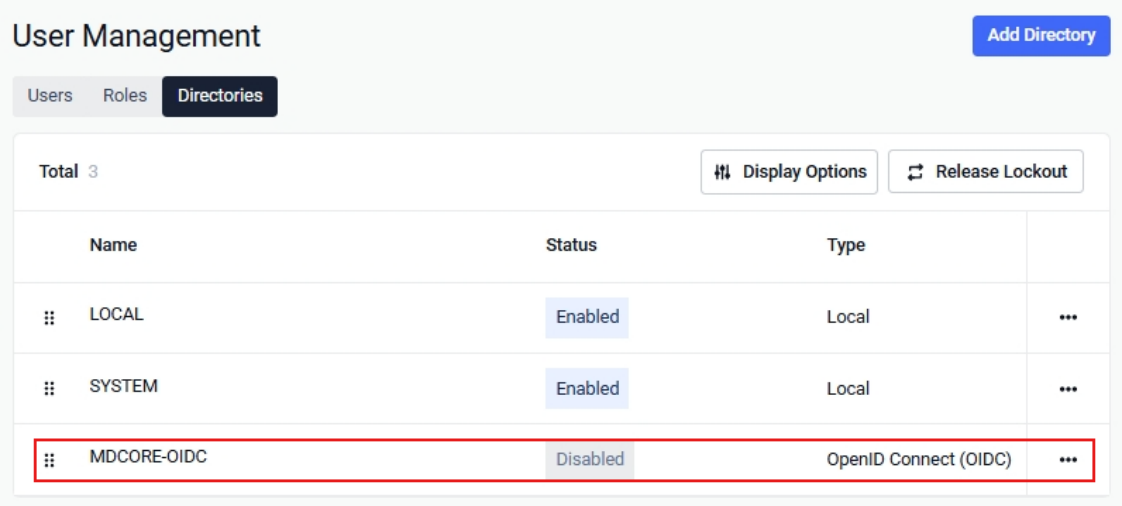
5. In `Identity Provider`, click `Submit JSON` button and upload the file that was modified in the previous stage. Wait a moment for MetaDefender Core to set Amazon Cognito as its IdP.



6. Enter `User identified by` with ${given_name} and select the appropriate role.

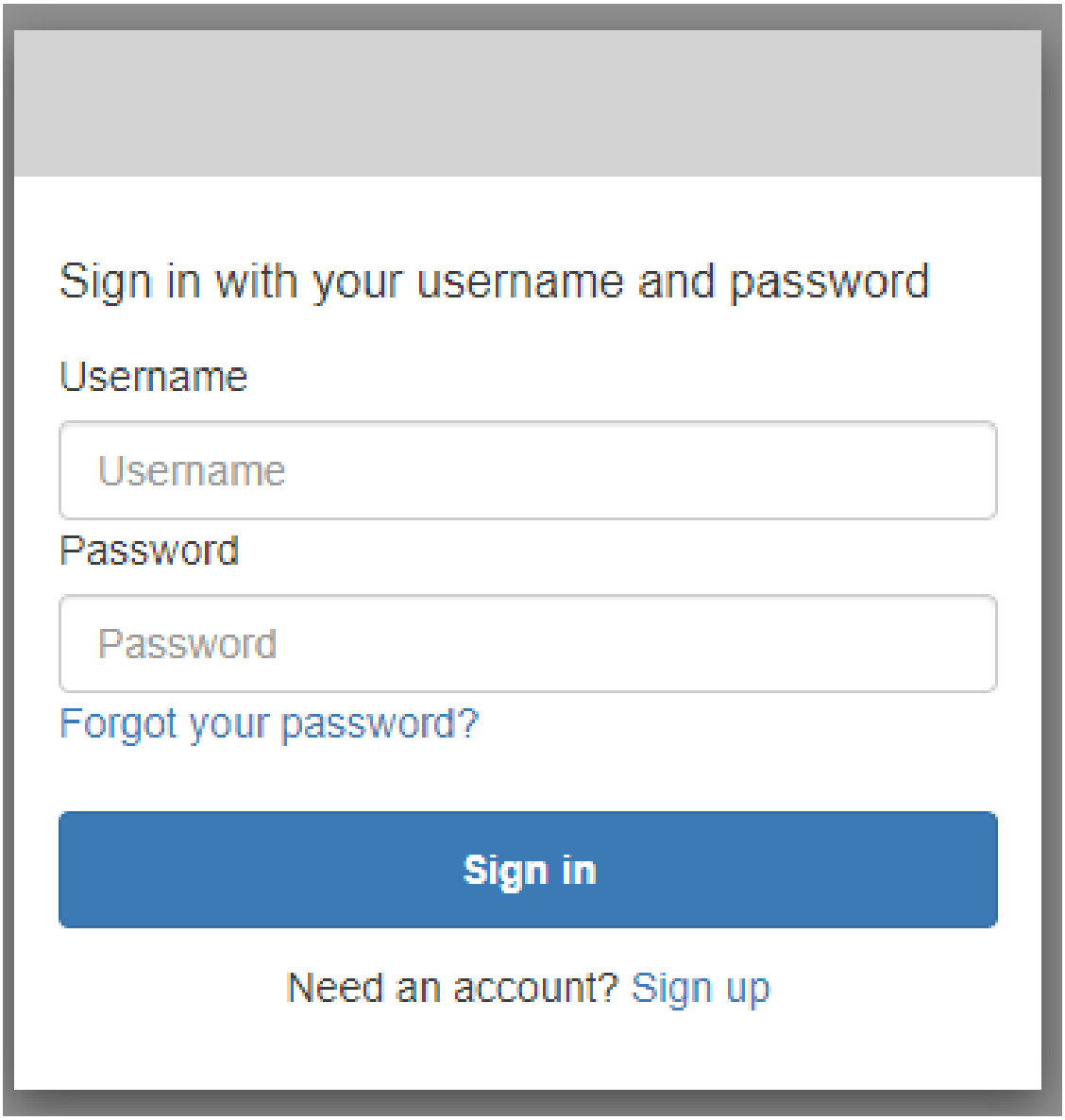

Add Directory
Directory Type
OIDC
Name *
MDCore-OIDC
Identity Provider
Fetch URL
or
Submit JSON
Service Provider
Client ID *
[redacted]
Client Secret *
••
Host or IP *
https://localhost:8008
Add
Unique URL Segment *
https://localhost:8008/ssologin/oidc/
edev
Login URLs
No entries added yet.
User Identified By *
${given_name}
Load Additional Profile From User Endpoint
User Role
Role Matching
Default Role
Role Mapping
User Role
Administrators
Cancel
Add

7. Click `Add` to complete.
8. In `User Management`, toggle the new directory. A dialog box will appear to confirm the action. Once `Enable` is clicked, all sessions will be expired immediately.

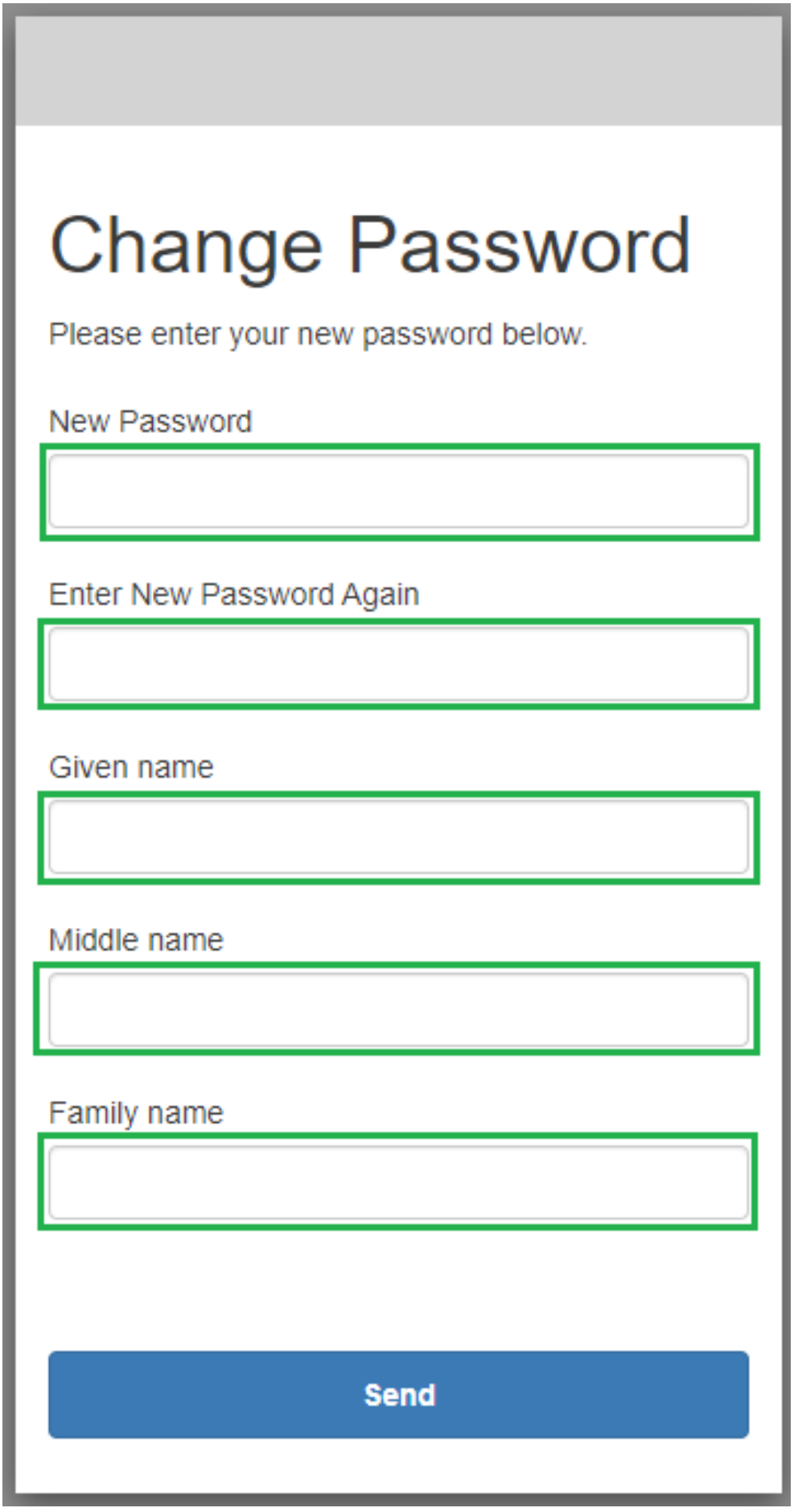


Test the integration

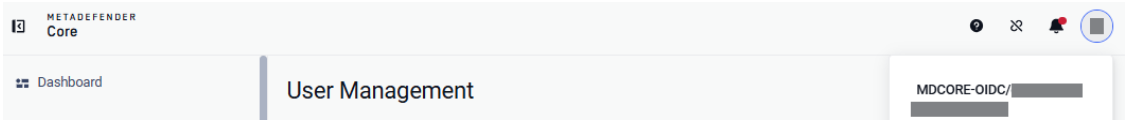
1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to Amazon Cognito page.



2. Change the password, enter other required fields, and click `Send`.


Change Password
Please enter your new password below.
New Password
Enter New Password Again
Given name
Middle name
Family name
Send

3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

ForgeRock

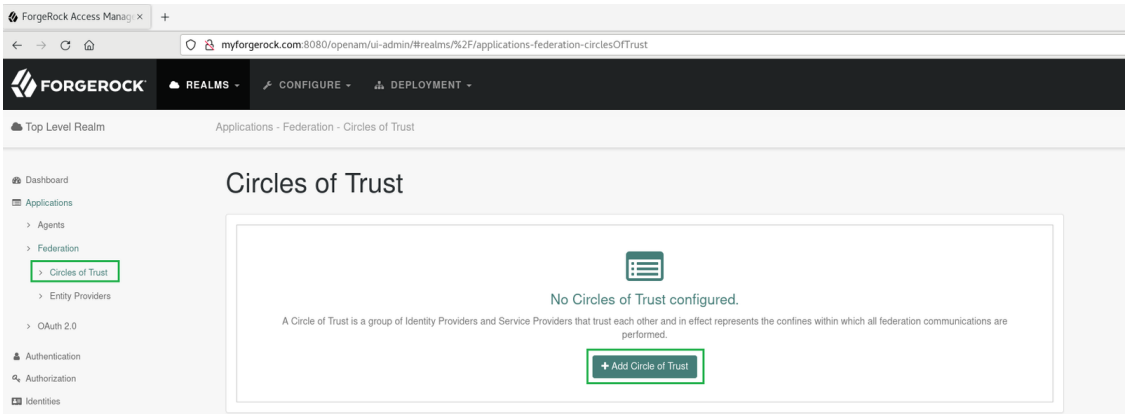
The instructions guide users on how to configure ForgeRock SAML Single Sign-on for MetaDefender Core.

Configure SAML SSO

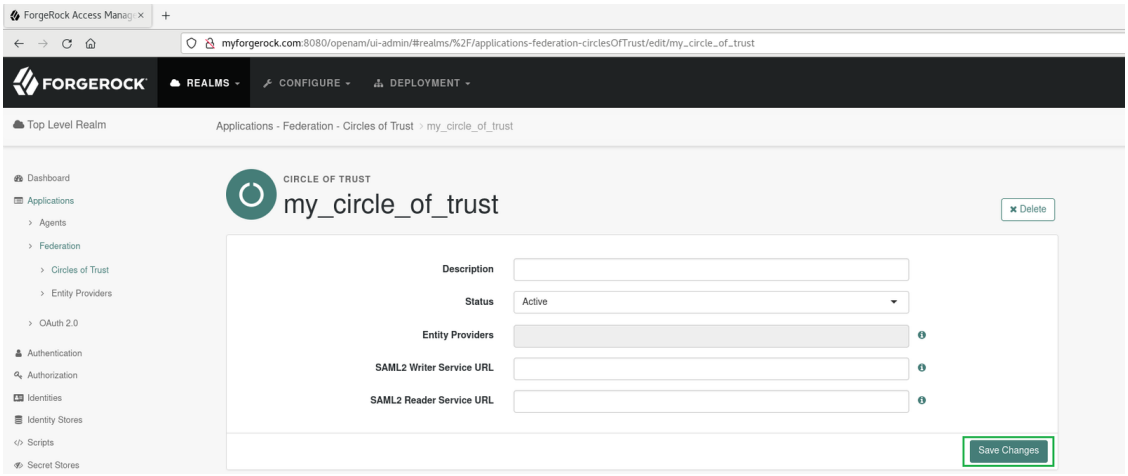
> **Info**
>
> The integration is demonstrated using ForgeRock version 7.2.0.

Configure Identity Provider in ForgeRock

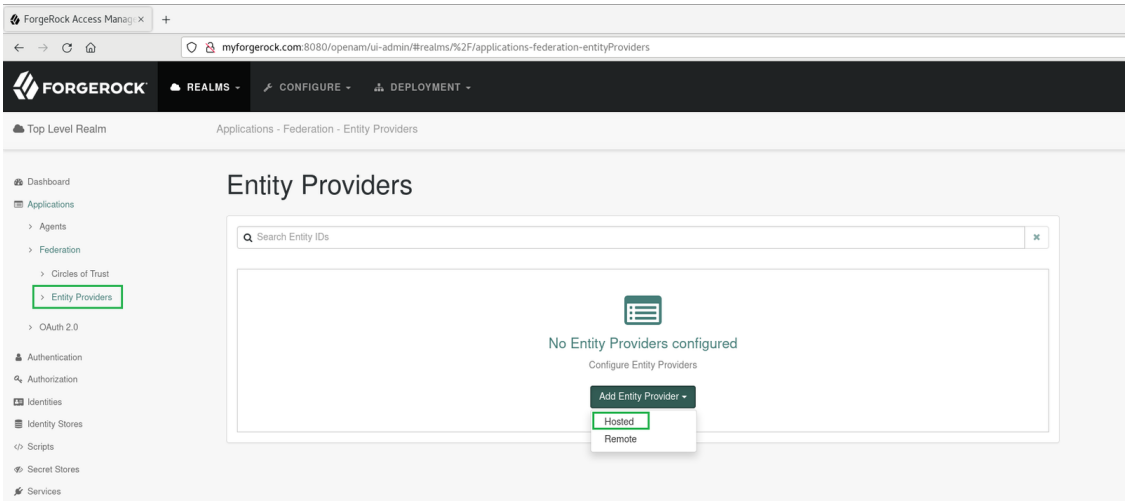
1. Access ForgeRock console.
2. On `Realms` page, select `Top Level Realm` located in the upper left corner.
3. Go to `Applications` in the sidebar, expand `Federation`, choose `Circles of Trust`, and press `Add Circle of Trust`.



4. Input the name, 'my_circle_of_trust' for instance, then select `Create` and `Save Changes`.

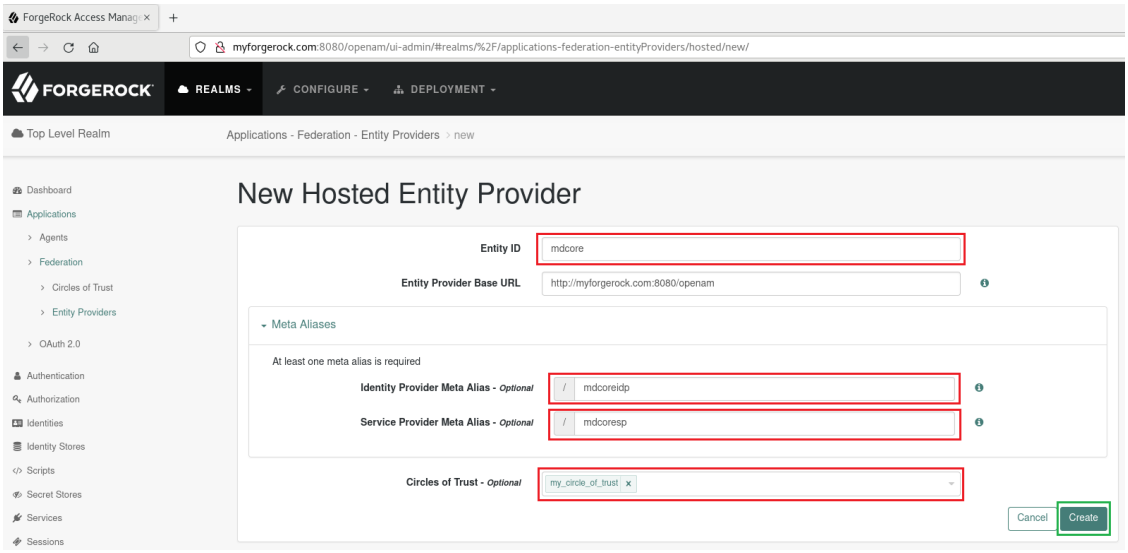


5. Select `Entity Providers` from `Federation` in the side bar, select `Add Entity Provider`, and pick `Hosted`.



6. Input 'mdcore' for `Entity ID`, 'mdcoreidp' for `Identity Provider Meta Alias`, 'mdcoresp' for `Service Provider Meta Alias`, and 'my_circle_of_trust' for `Circles of Trust`, then click `Create`.

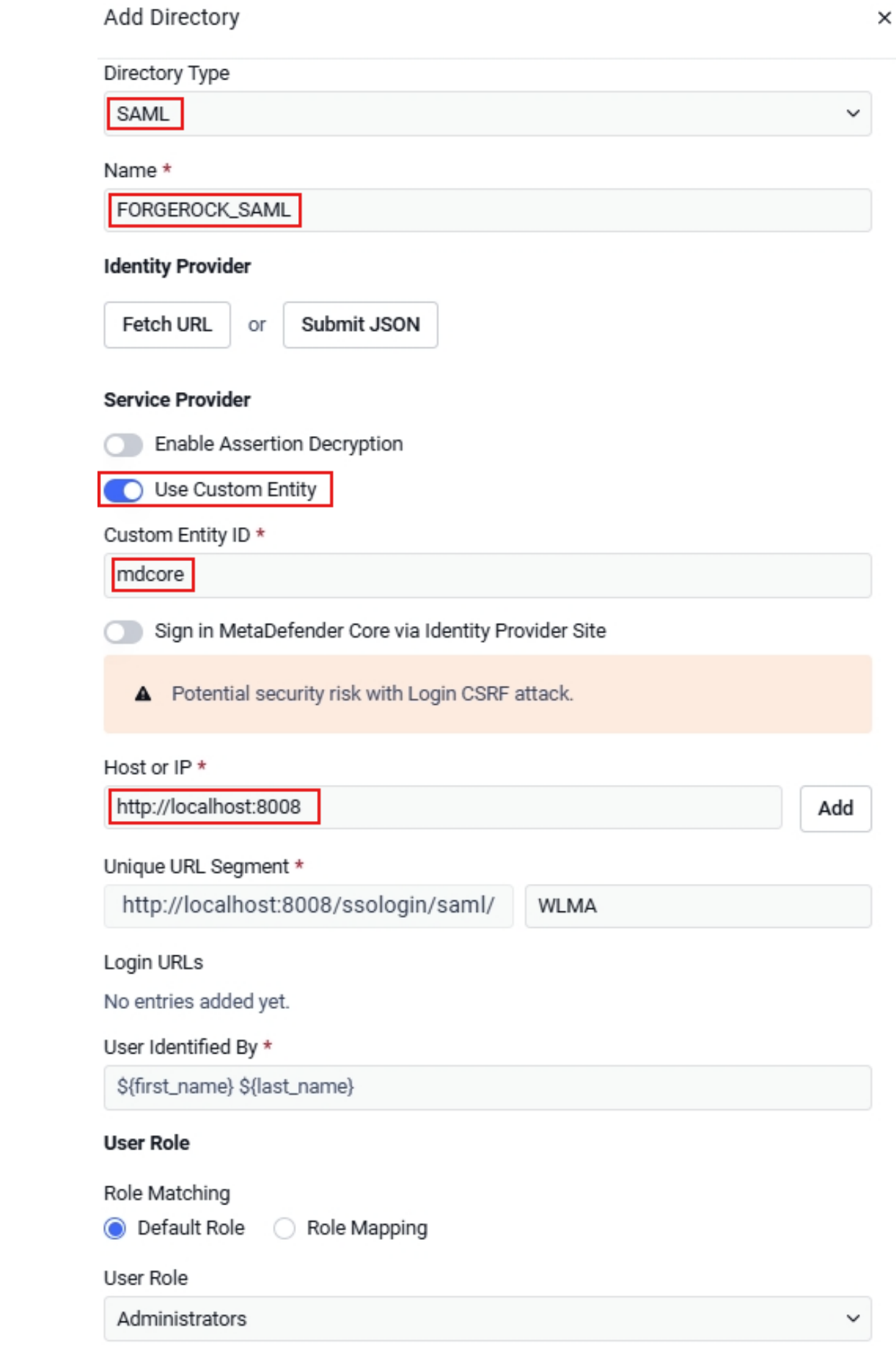
> **Info**
>
> The value of `Entity ID`, 'mdcore' in this case, will later be used to configure `Custom entity ID` in MetaDefender Core.



Create SAML directory in MetaDefender Core

1. Access MetaDefender Core console.
2. On the dashboard, select `User Management` from the sidebar.
3. On `User Management` page, choose `Directories` tab and then click on `Add Directory` in the upper right corner.
4. On `Add Directory` page, select `SAML` for directory type.
5. Input the name of the new folder, such as 'FORGEROCK_SAML'.
6. In `Service Provider` area, toggle `Use Custom Entity ID` option and type 'mdcore' in `Custom entity ID` box.
7. Fill in `Host or IP` with the address of the machine where MetaDefender Core is located, using 'http://localhost:8008' as an example.
8. Click the copy button beside `Login URL`.

> **Info**
>
> The copied login URL will be used subsequently in `Location` setting of ForgeRock.

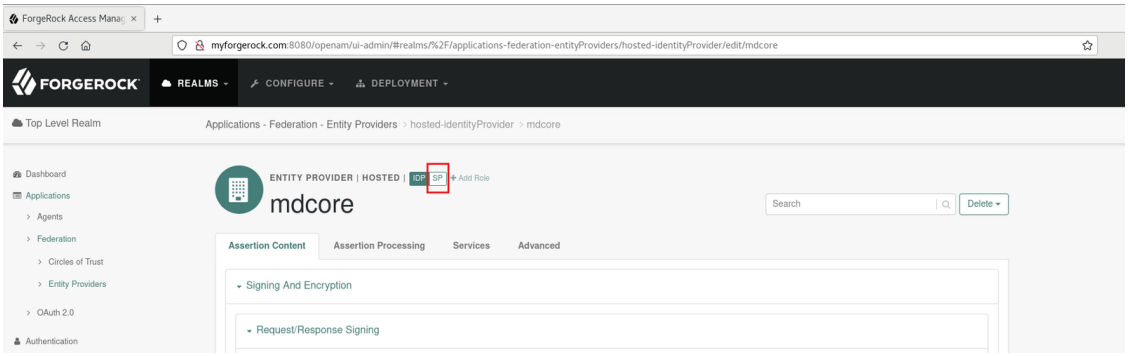

Add Directory
Directory Type
SAML
Name *
FORGEROCK_SAML
Identity Provider
Fetch URL
or
Submit JSON
Service Provider
Enable Assertion Decryption
Use Custom Entity
Custom Entity ID *
mdcore
Sign in MetaDefender Core via Identity Provider Site
Potential security risk with Login CSRF attack.
Host or IP *
http://localhost:8008
Add
Unique URL Segment *
http://localhost:8008/ssologin/saml/
WLMA
Login URLs
No entries added yet.
User Identified By *
${first_name} ${last_name}
User Role
Role Matching
Default Role
Role Mapping
User Role
Administrators

Warning

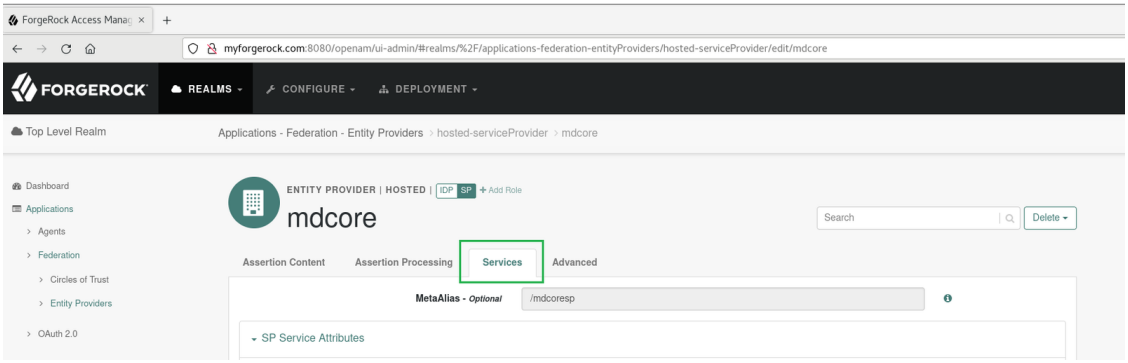
The setup in MetaDefender Core is incomplete, please refrain from clicking outside of `Add Directory` screen.

Configure Service Provider in ForgeRock

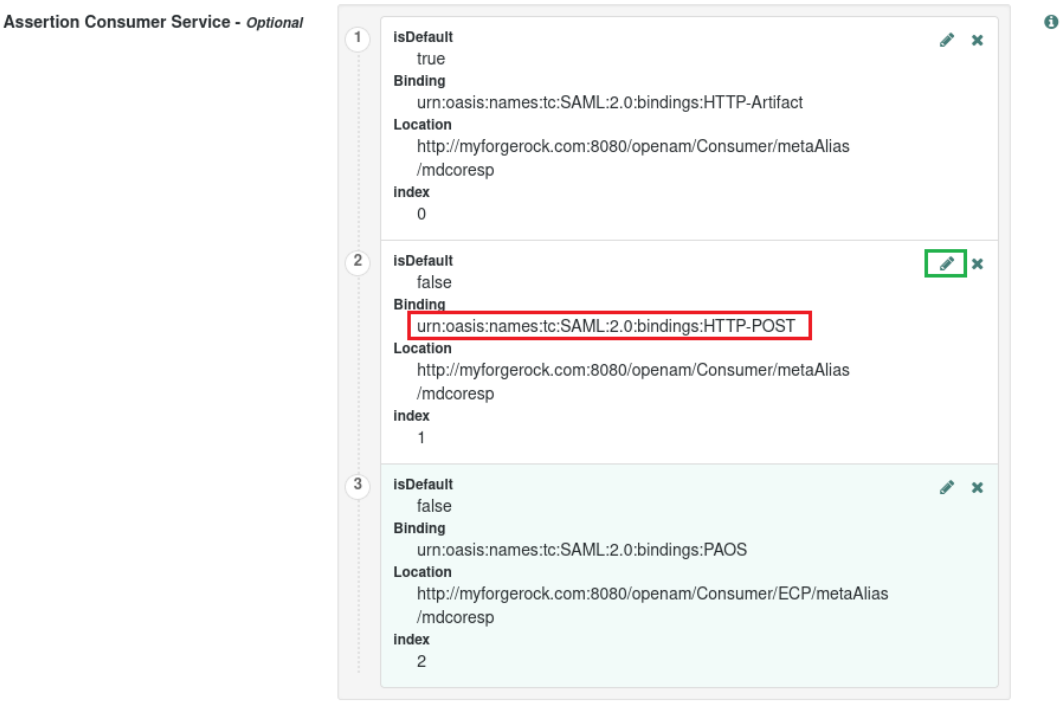
1. In the ForgeRock console, switch to `SP` for the entity provider 'mdcore'.



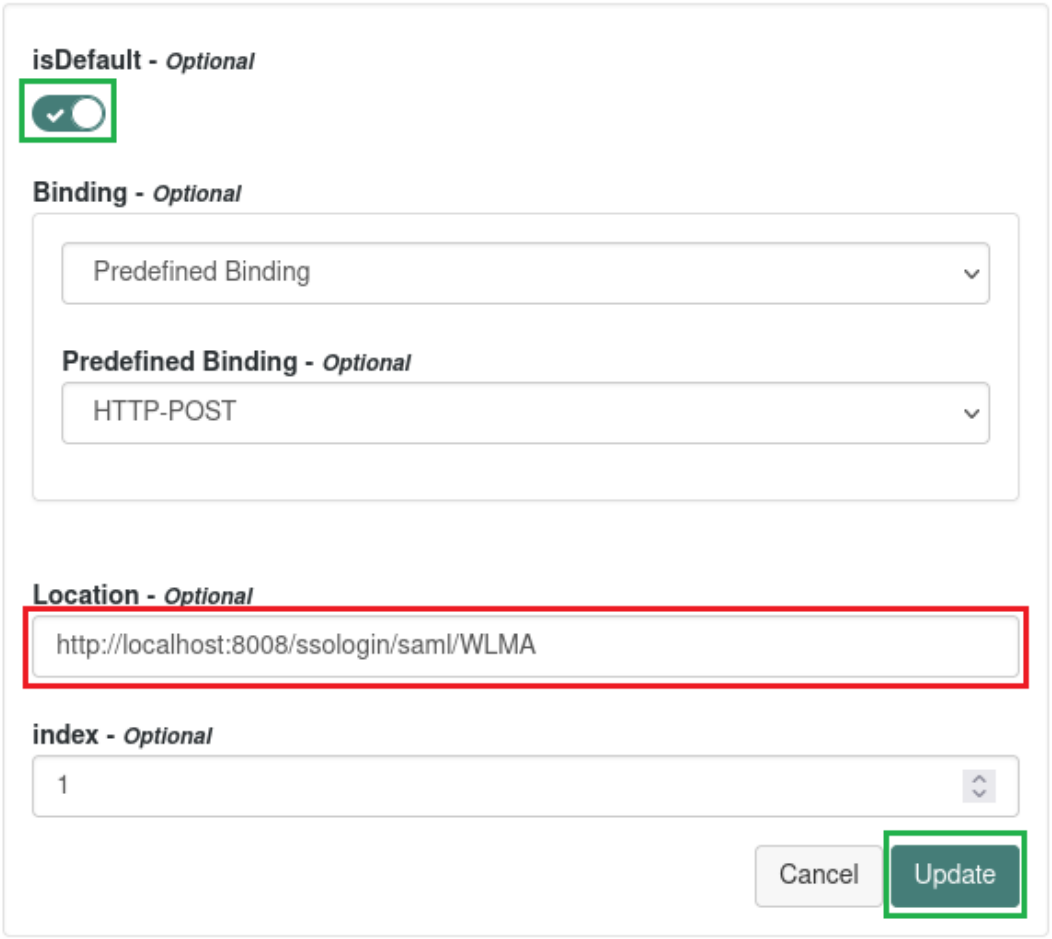
2. Click on `Services` tab.



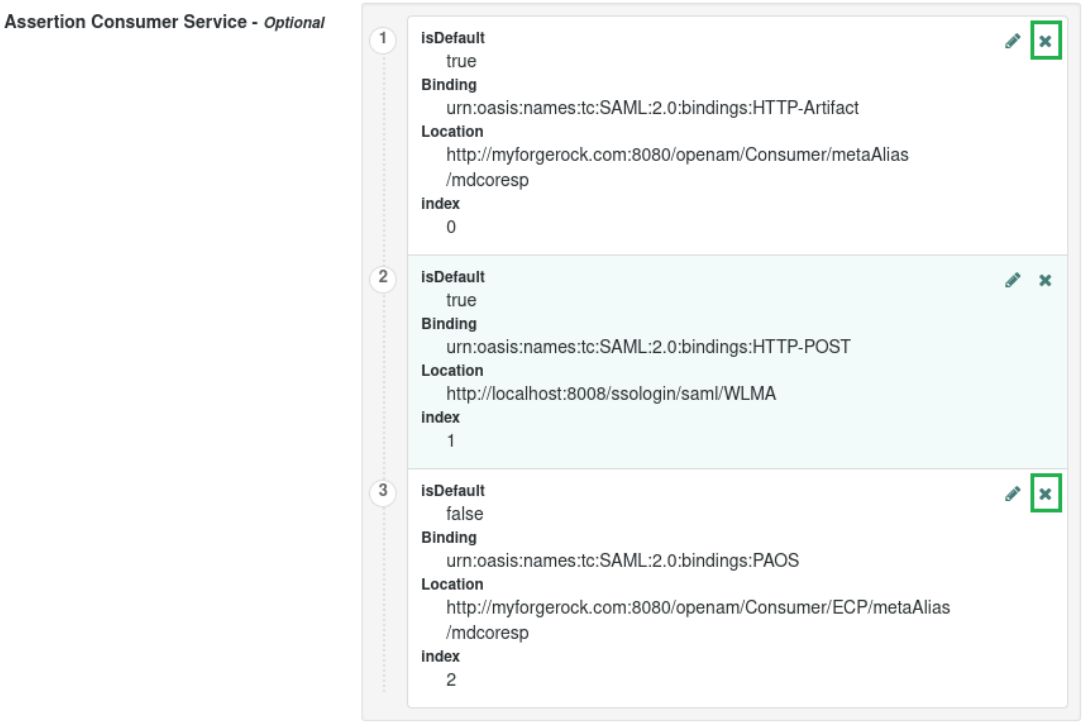
3. Go to `Assertion Consumer Service` area, choose HTTP-POST binding entry, and tap the edit icon located in the top right.



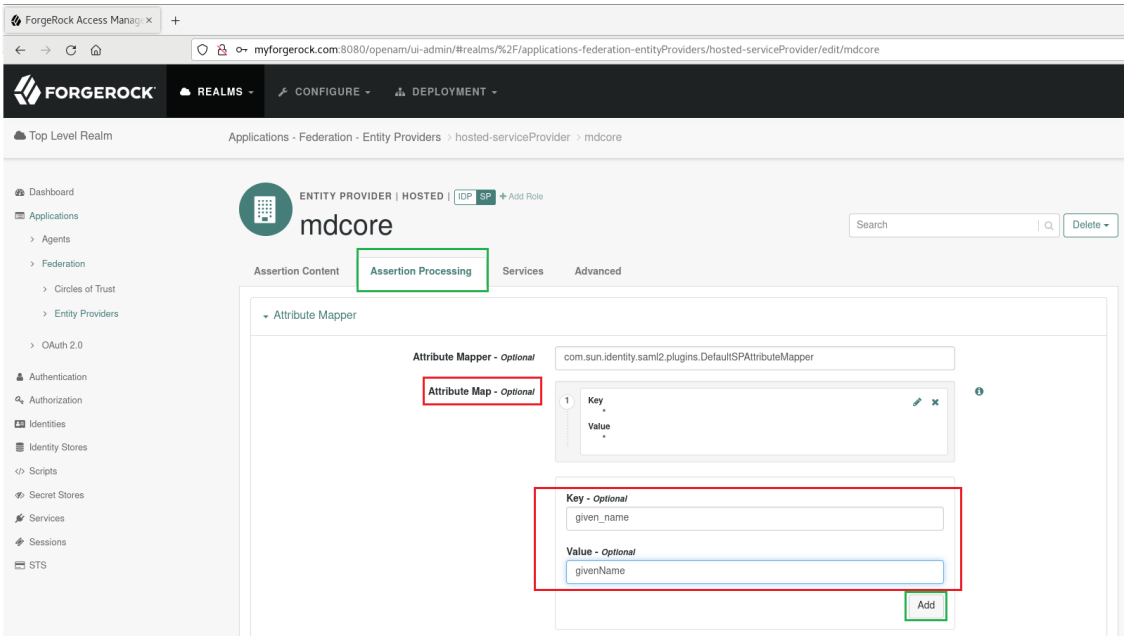
4. Input login URL from MetaDefender Core into `Location`, toggle `isDefault` switch in the upper left, and press `Update`.



5. Remove HTTP-Artifact and PAOS bindings, then press `Save Changes`.



6. Select `Assertion Processing` tab, navigate to `Attribute Map`, enter 'given_name' and 'givenName' as `Key` and `Value`, accordingly.



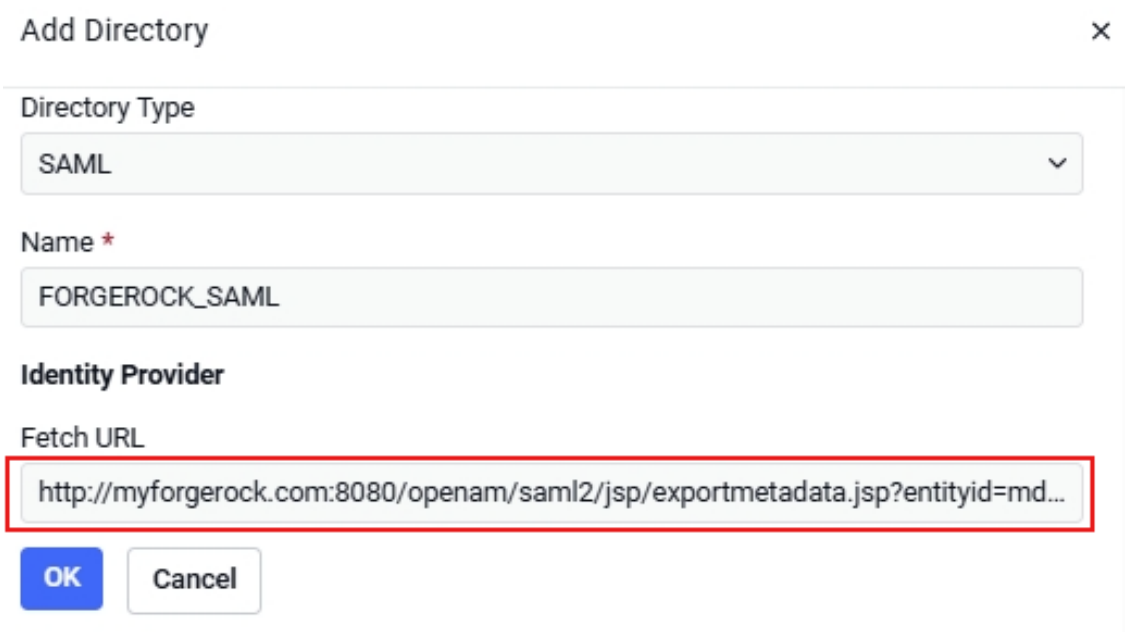
7. Press `Add` and `Save Changes`.

Complete configuration in MetaDefender Core

1. Go back to MetaDfender Core, in `Add Directory` page, navigate to `Identity Provider` section and click on `Fetc`

`h URL`.

2. Enter the SAML Metadata URL from ForgeRock into the field beneath `Fetch URL`, click `OK` and wait a moment for MetaDefender Core to assign ForgeRock as its IDP.



Info

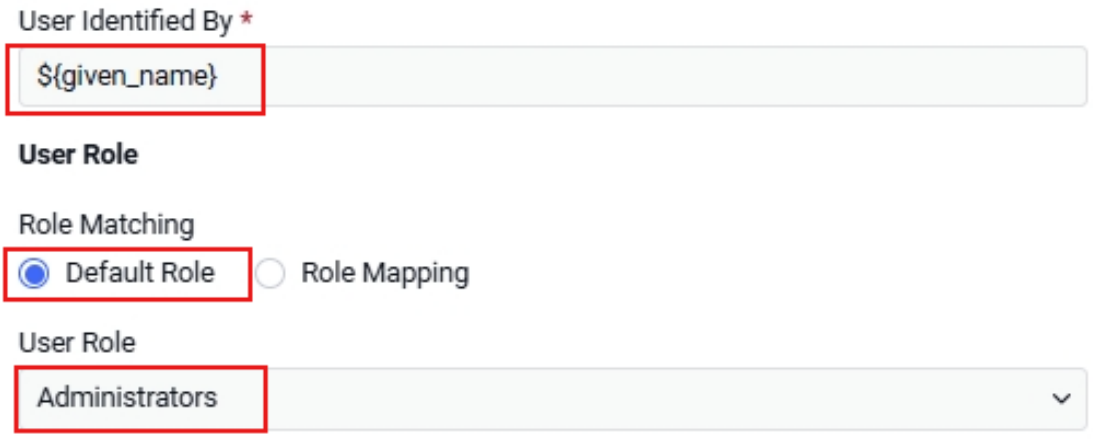
The structure of the metadata URL is as outlined below:

`http://<forgerock`*server>:<port>/openam/saml2/jsp/exportmetadata.jsp?entityid=<entity*`id>`

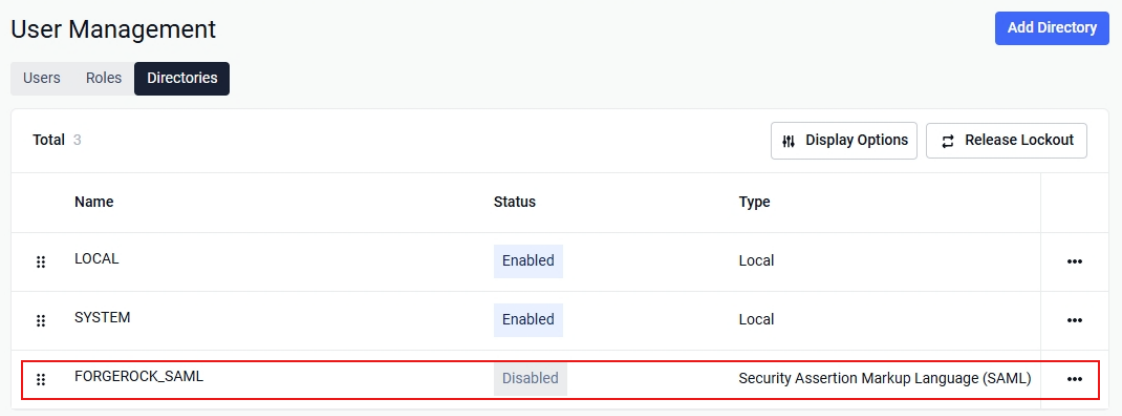
In which, `forgerock_server` and `port` correspond to the host and port of the machine hosting ForgeRock server; `entity_id` refers to the value of `Entity ID` configured in ForgeRock. In this guide, ForgeRock is located at 'myforgerock.com:8080' and the `Entity ID` is 'mdcore'; thus, the metadata URL becomes

`http://myforgerock.com:8080/openam/saml2/jsp/exportmetadata.jsp?entityid=mdcore`

3. Populate `User identified by` field with '${given_name}', as set up in ForgeRock during the earlier step.
4. Choose the appropriate role for the user in `User Role` and click `Add` to finish the process.



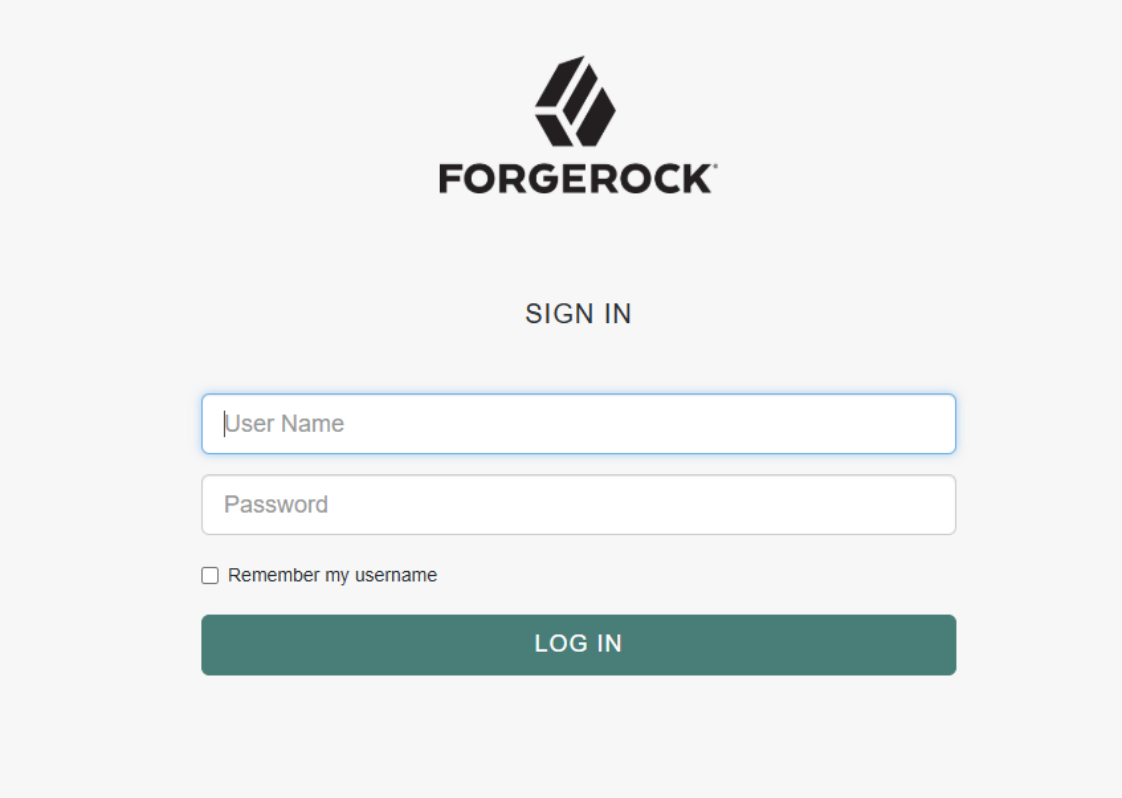
5. On `User Management` screen, switch on the new directory, 'FORGEROCK_SAML' in this case.



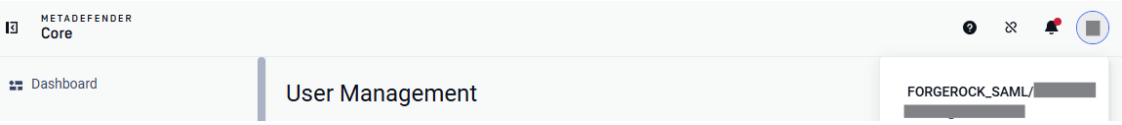
6. A confirmation dialog box will pop up to verify the action. Upon clicking `Enable`, all sessions will be terminated instantly, and ForgeRock will be utilized for user authentication.

Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to ForgeRock Sign-in page.



2. Sign in with your ForgeRock account.
3. If all proceeds smoothly, MetaDefender Core dashboard will appear with the user's identity in the upper right corner.



4. Otherwise, visit backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Keycloak

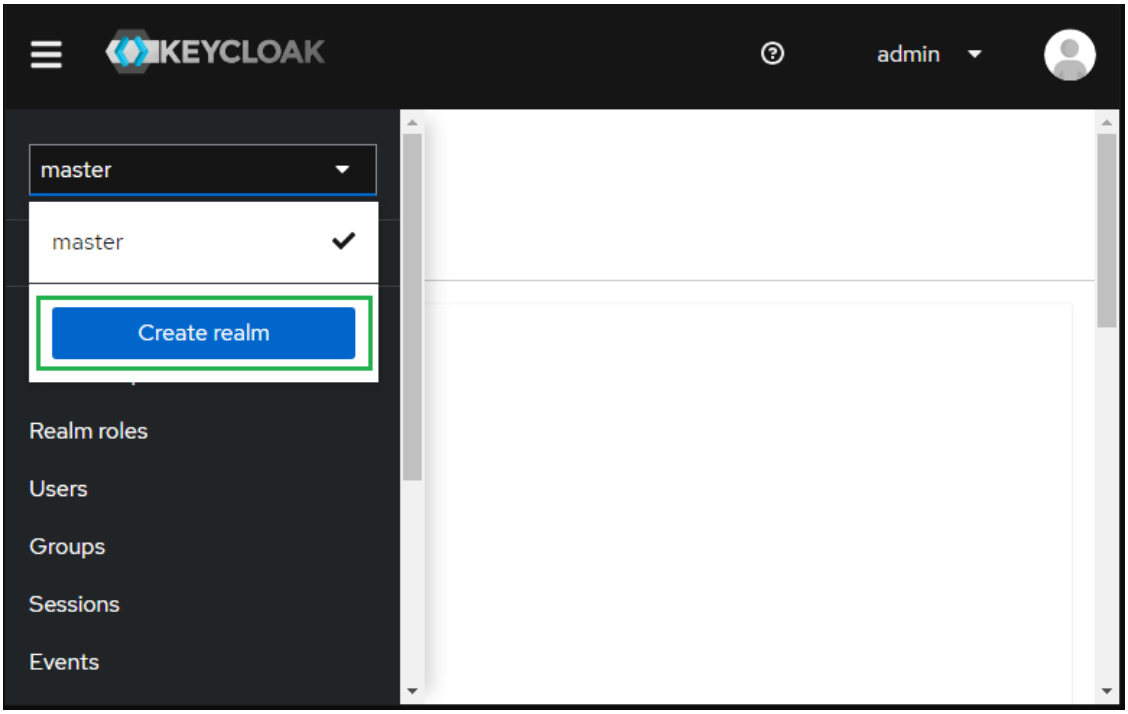
The instruction guides users on how to configure Keycloak SAML Single Sign-on for MetaDefender Core.

Configure SAML SSO

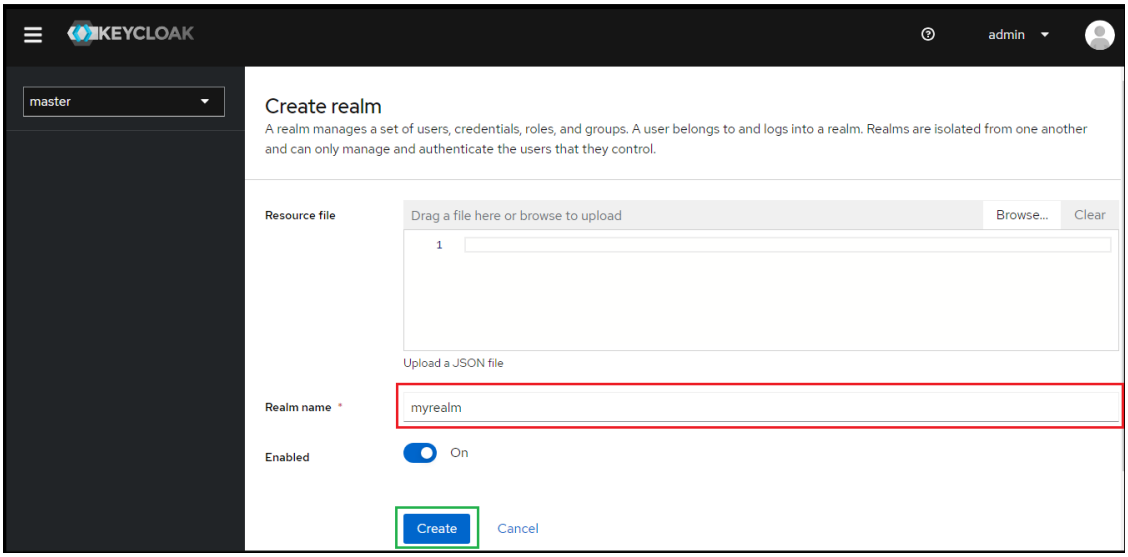
Create a realm in Keycloak

1. Sign in to Administrator Console, drop the list in the top left corner, and click `Create realm`.

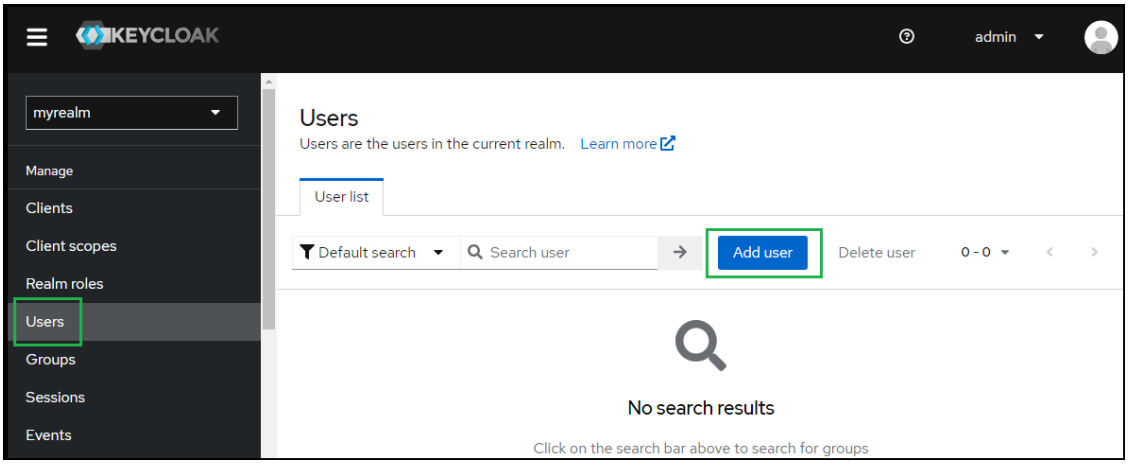
> **Info**
>
> Keycloak initially includes a single realm `master` which is used for managing Keycloak only and not for any applications.



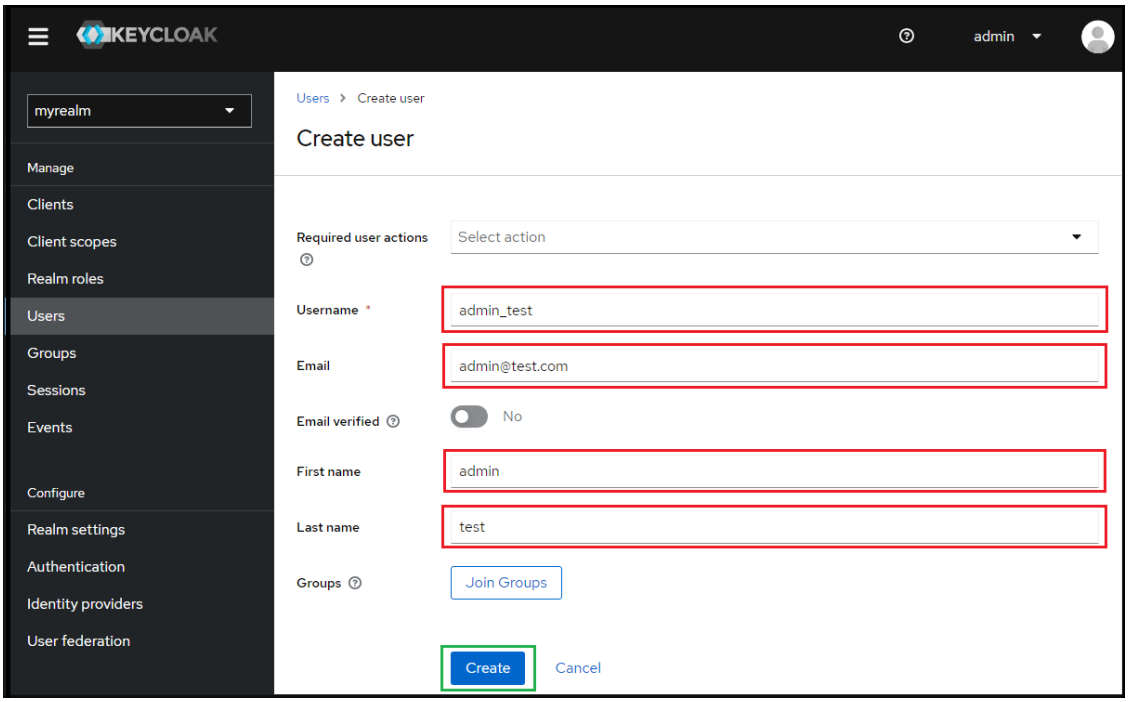
2. Enter `Realm name` e.g. myrealm and click `Create`.



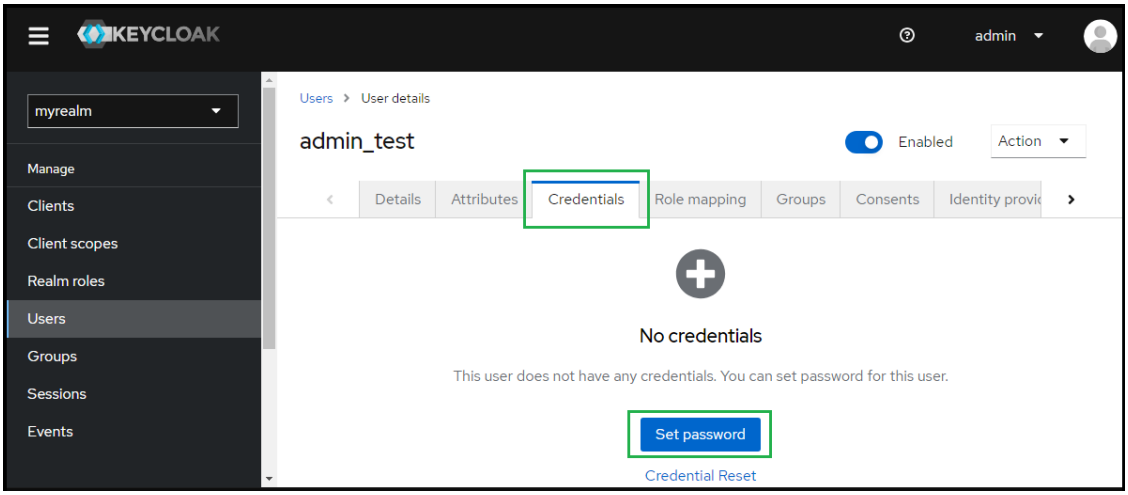
3. Select `Users` in the left sidebar and click `Add user`.



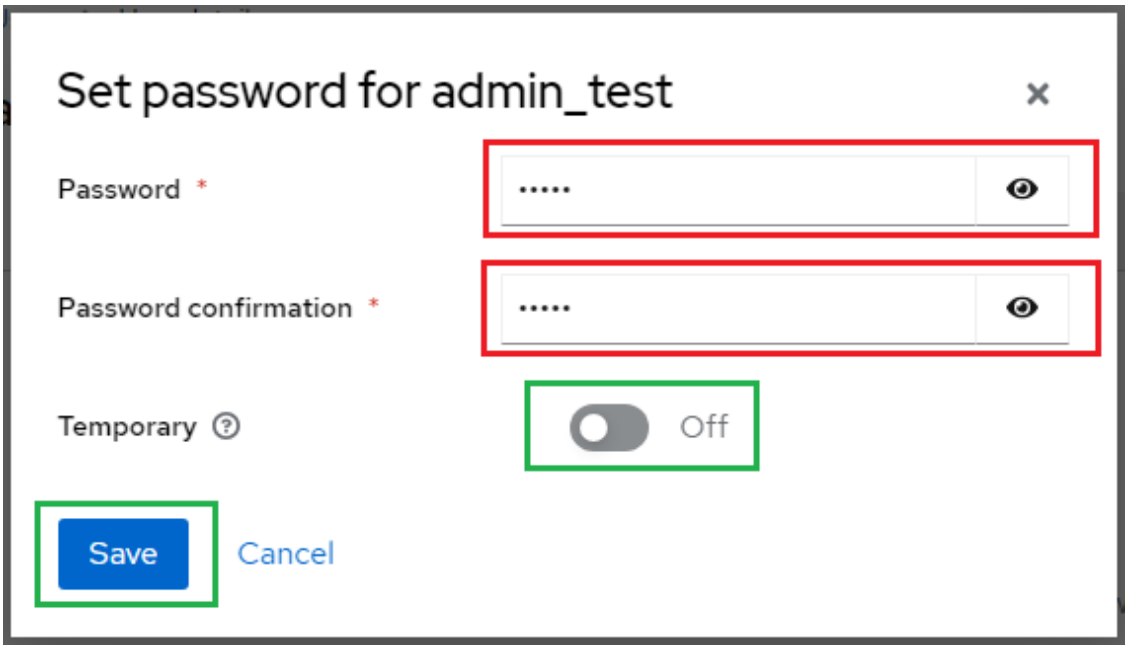
4. Enter values for `Username`, `Email`, `First name` and `Last name`; then click `Create`.



5. Under `User details`, select `Credentials` tab and click `Set password` to create a password for the user created in the previous step.

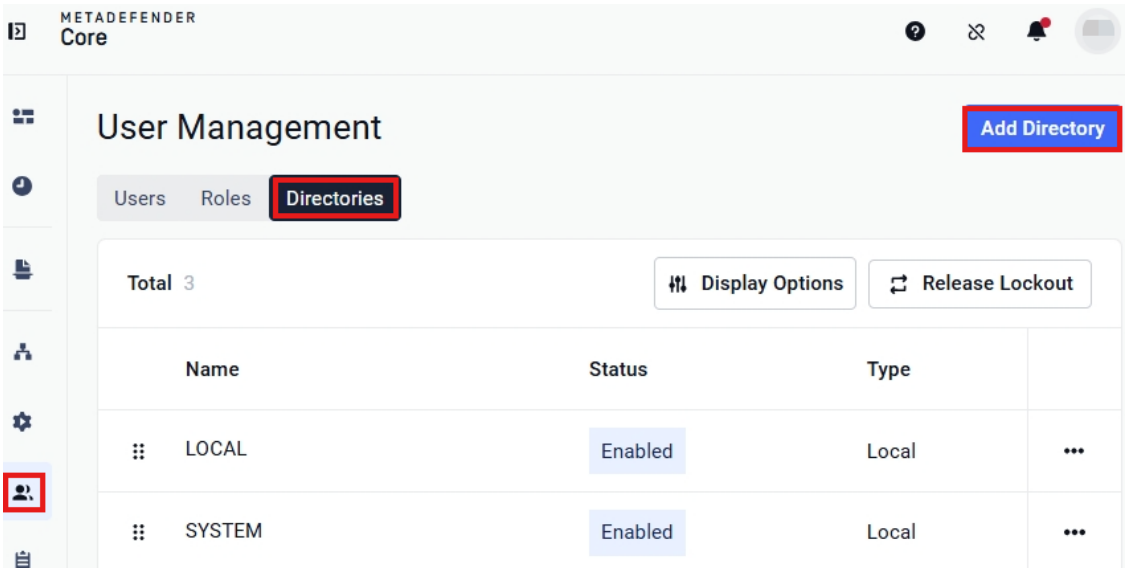


6. Enter the password and toggle `Temporary` to Off, then click `Save`.

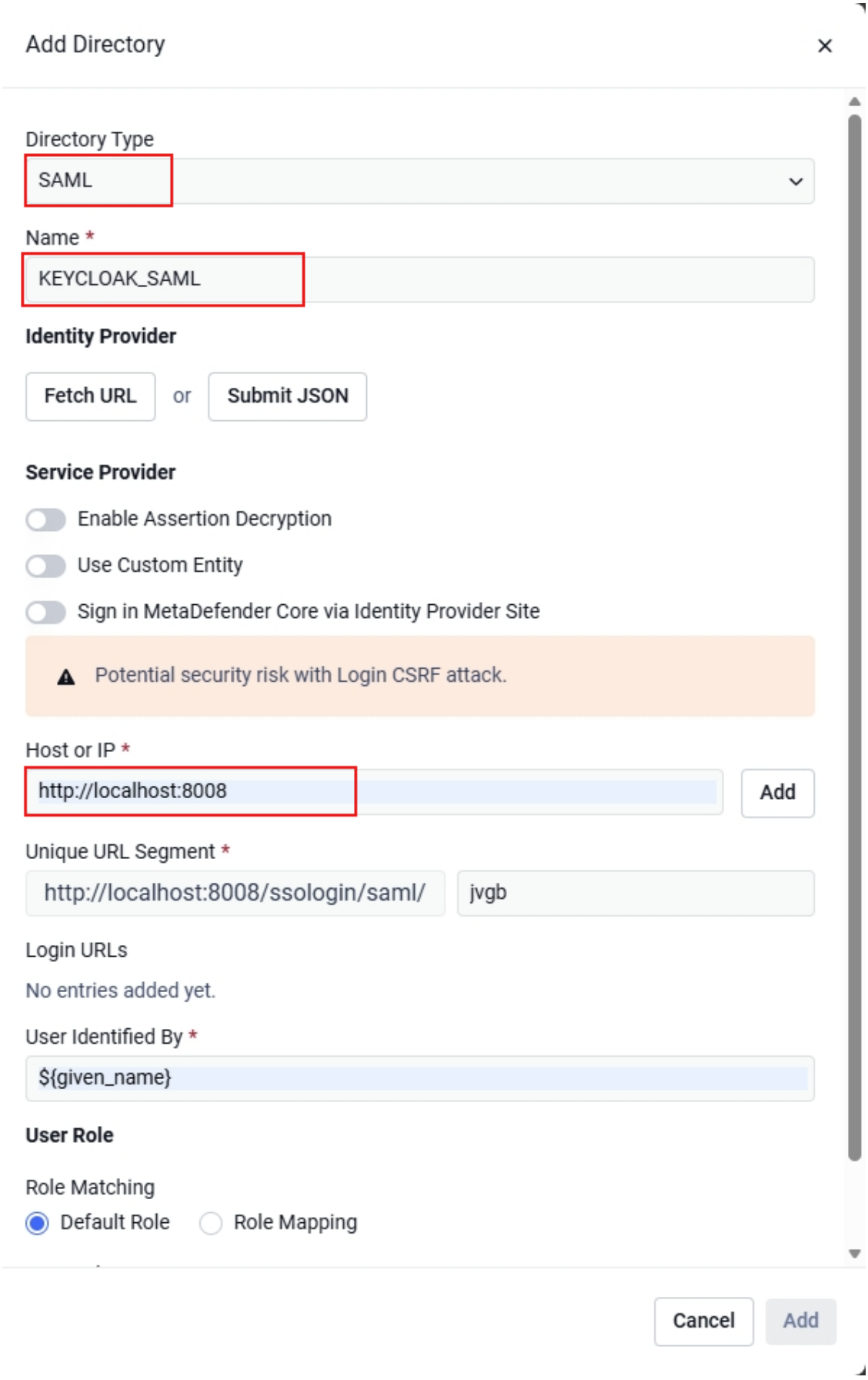


Create SAML directory in MetaDefender Core

1. Sign in to MetaDefender Core management console.
2. Under `Dashboard`, click `User Management` in the left sidebar.
3. Under `User Management`, select `Directories` tab and hit `Add directory` in the top right.

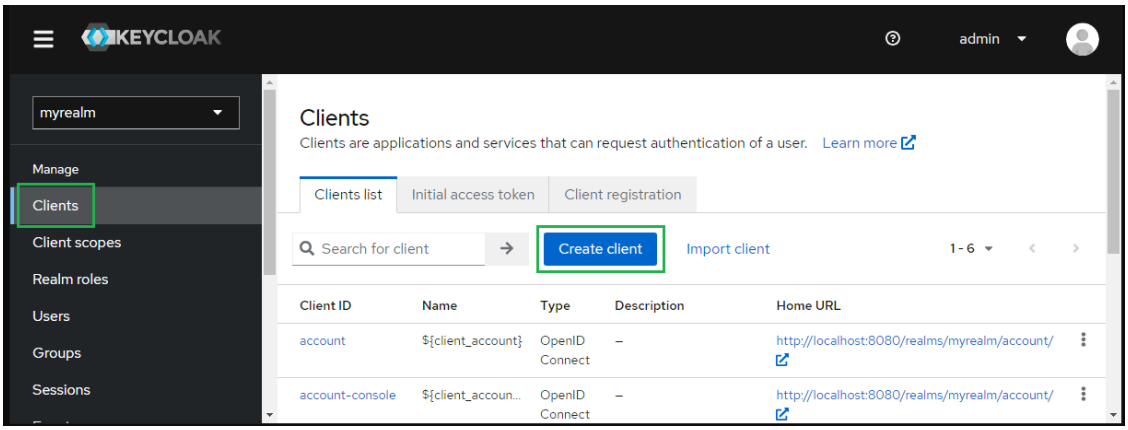


4. On `Add Directory` page, select SAML in `Directory type`.
5. Fill `Name` of the new directory, such as KEYCLOAK_SAML.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, using http://localhost:8008 as an example.
7. Copy the value of `Login URL`.

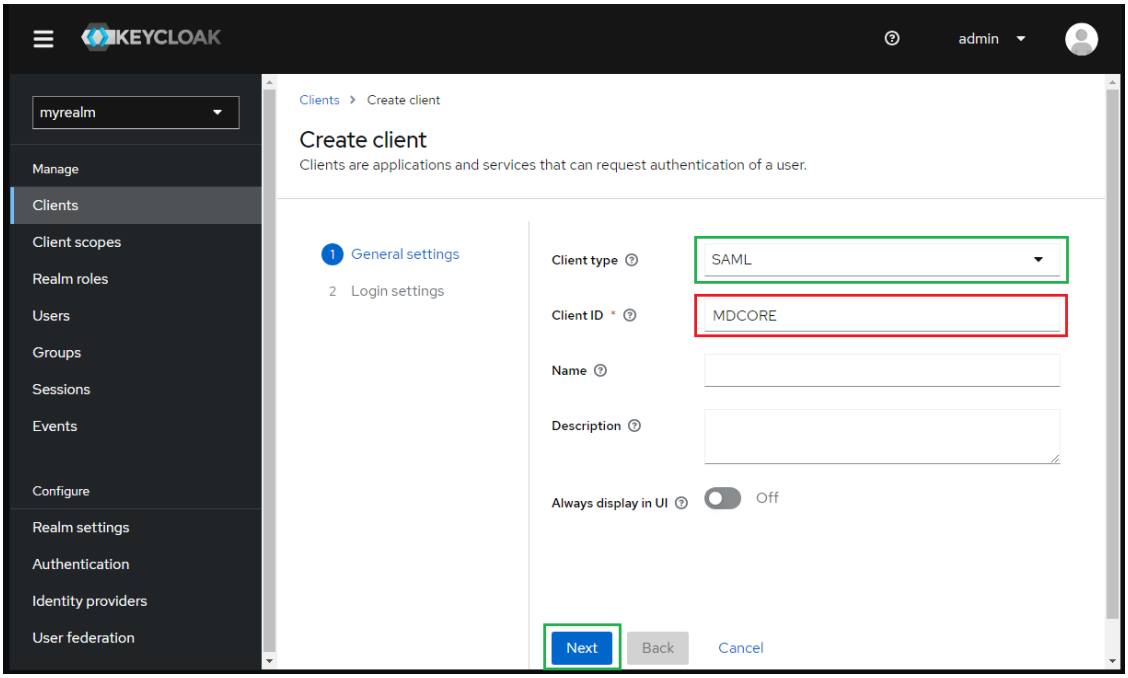


Create Keycloak application

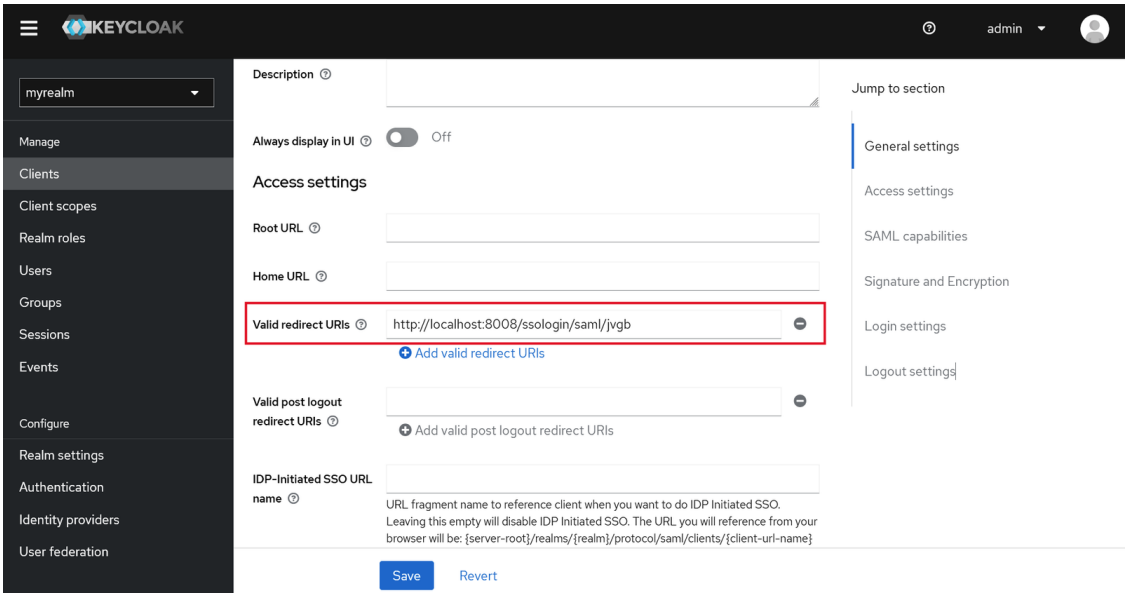
1. On screen `myrealm`, select `Clients` in the sidebar and click `Create client`.



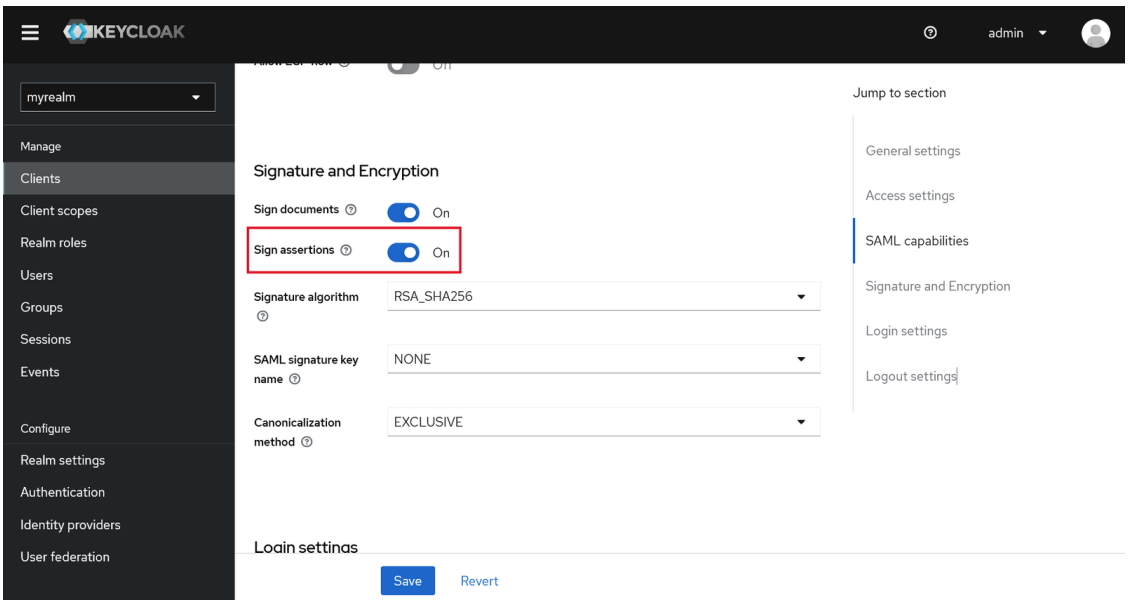
2. Choose **SAML** for `Client type` and enter **MDCORE** for `Client ID` then click `Next` and `Save`.



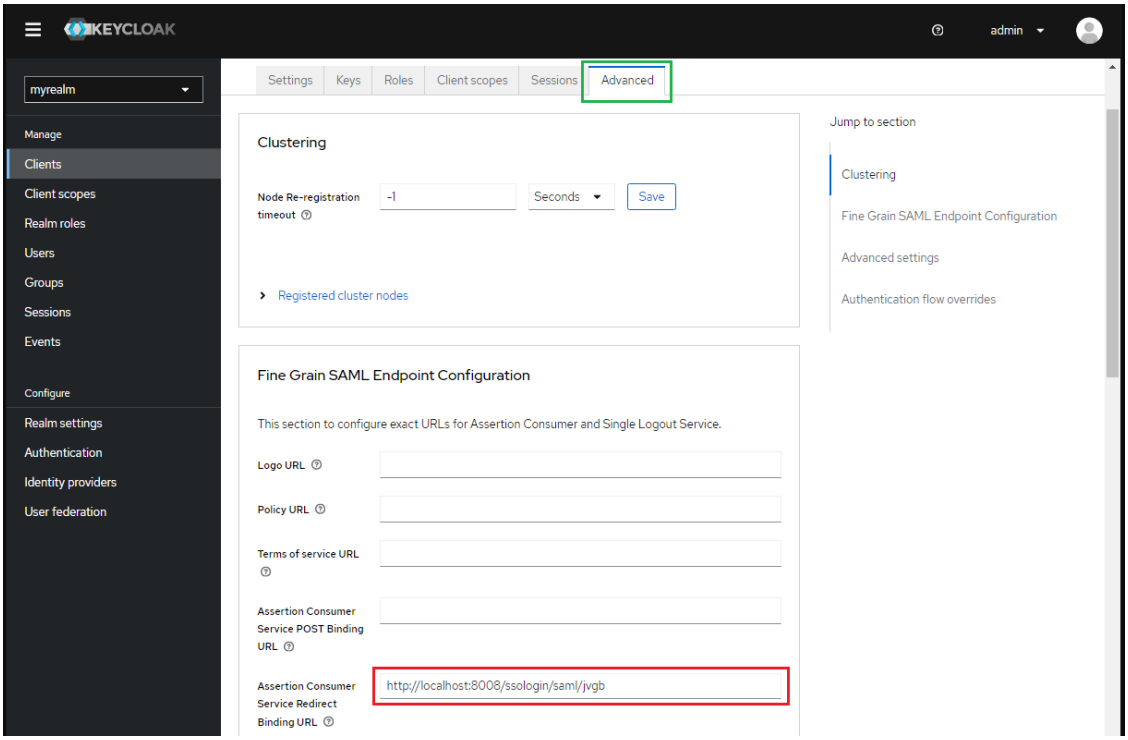
3. Under `General Settings`, find the `Valid Redirect URIs` field, then paste the **Login URL** from MetaDefender Core into the corresponding input box.



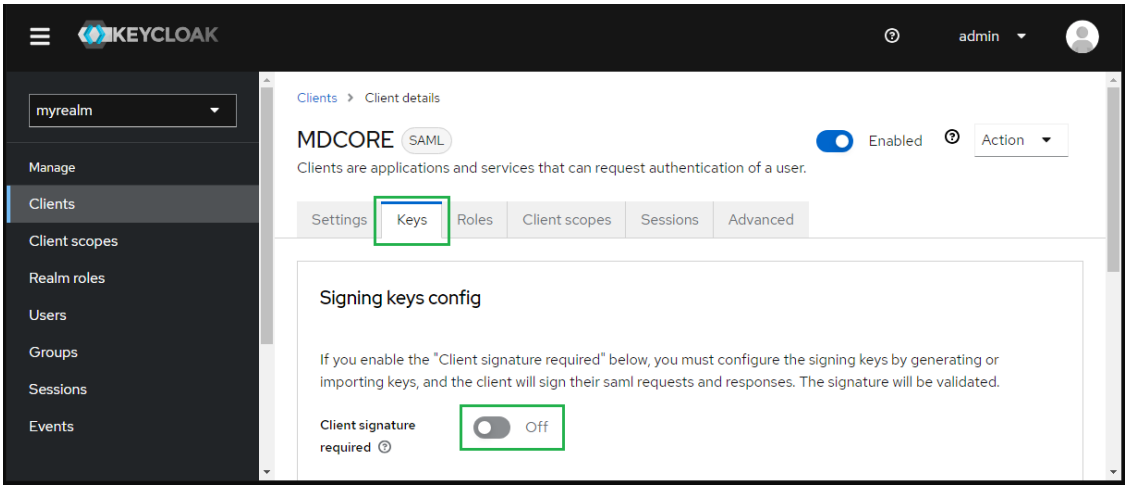
4. Find `Signature and Encryption` section, enable `Sign assertions` and click `Save`.



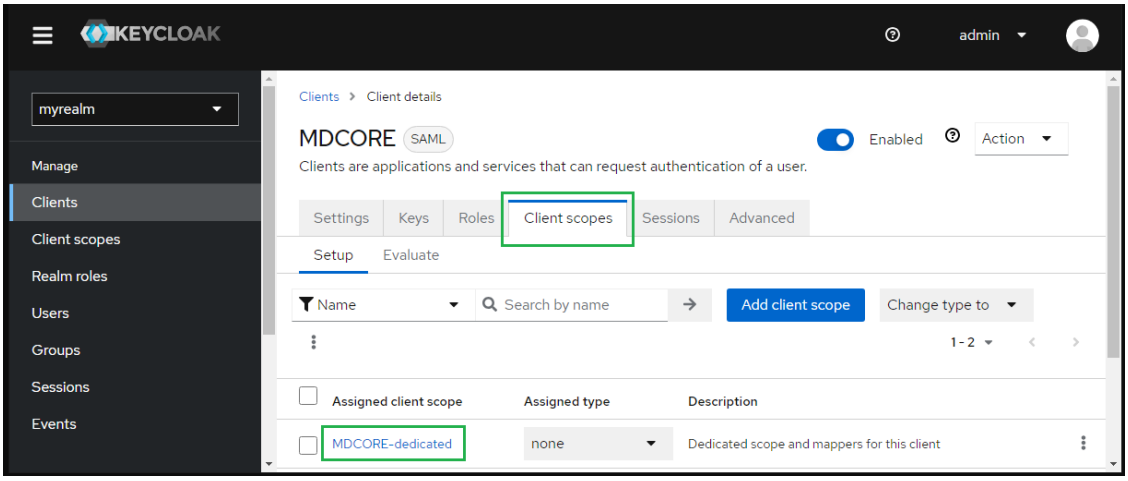
5. Go to `Advance` tab and paste the value of **Login URL** from MetaDefender Core to `Assertion Consumer Service Redirect Binding URL` and click `Save`.



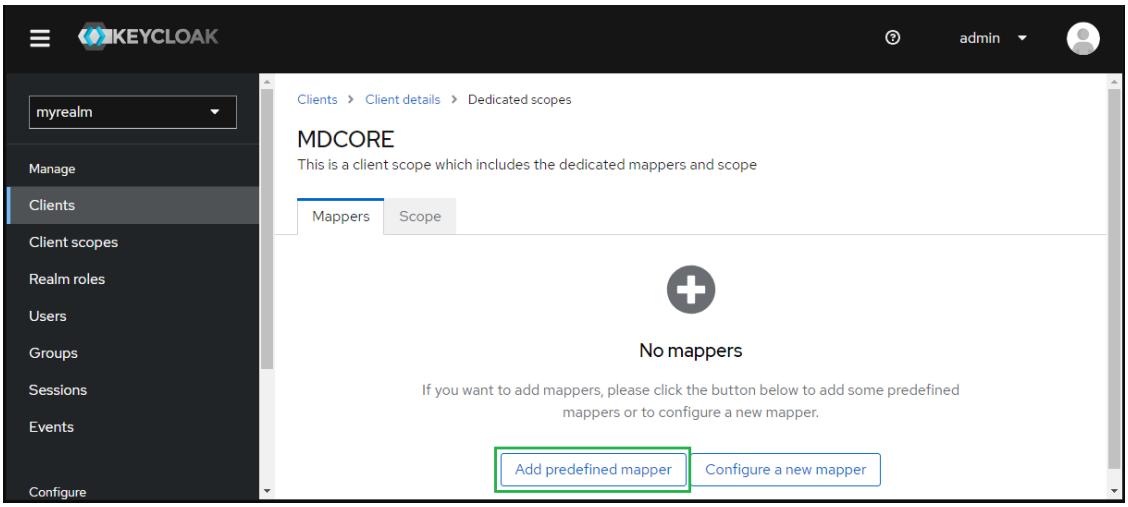
6. In `Keys`tab, toggle `Client signature required` to Off.



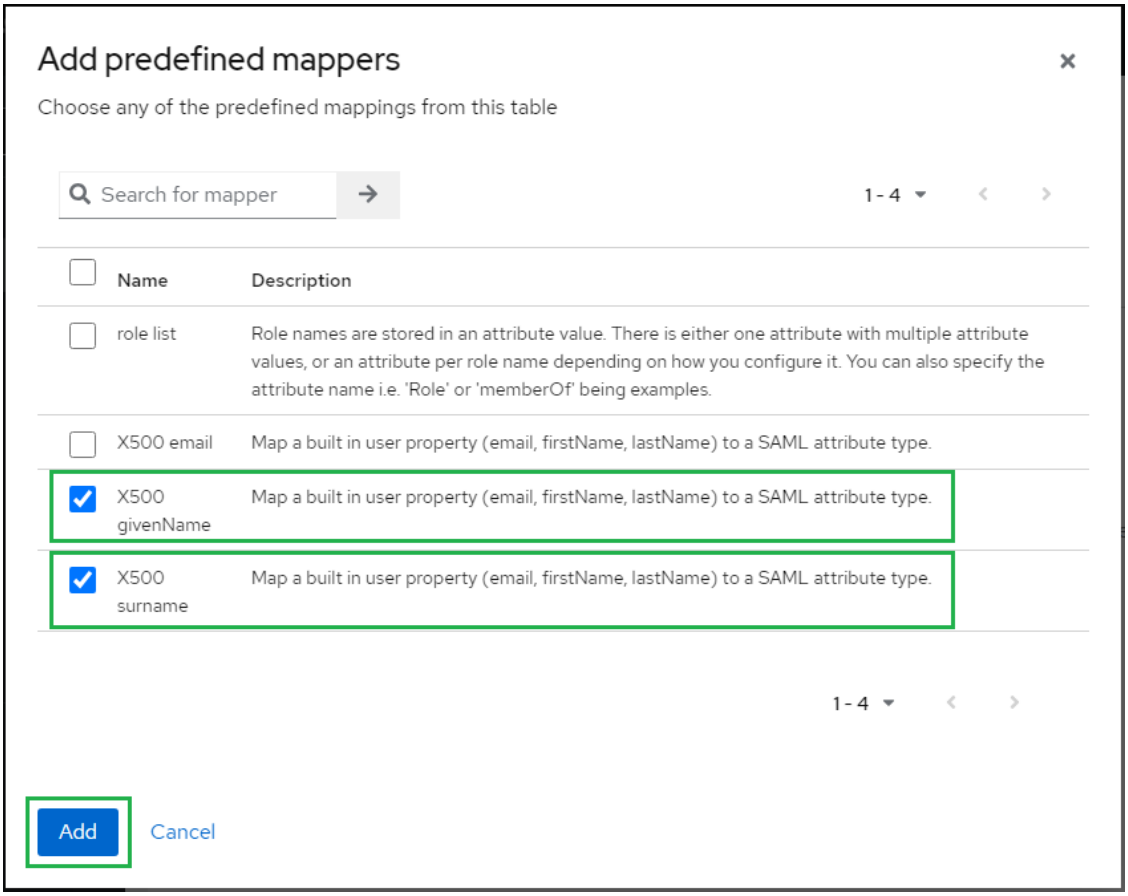
7. In `Client scopes` tab, select `MDCORE-dedicated`.



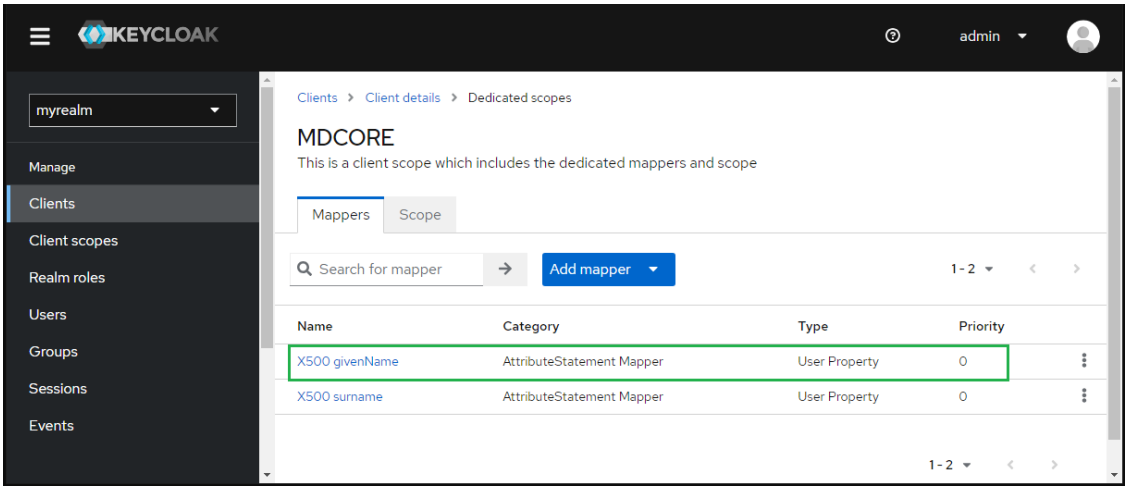
8. Under `Dedicated scopes`, navigate to `Mappers` tab and click `Add predefines mapper`.



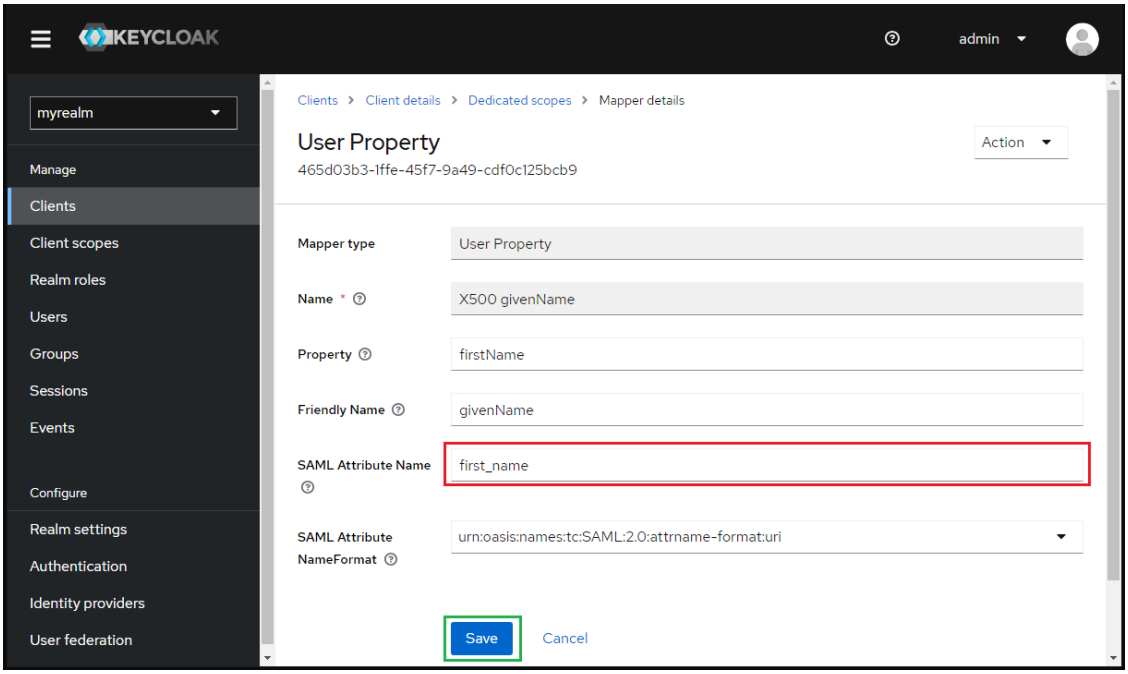
9. Check `X500 givenName` and `X500 surname`, then click `Add`.



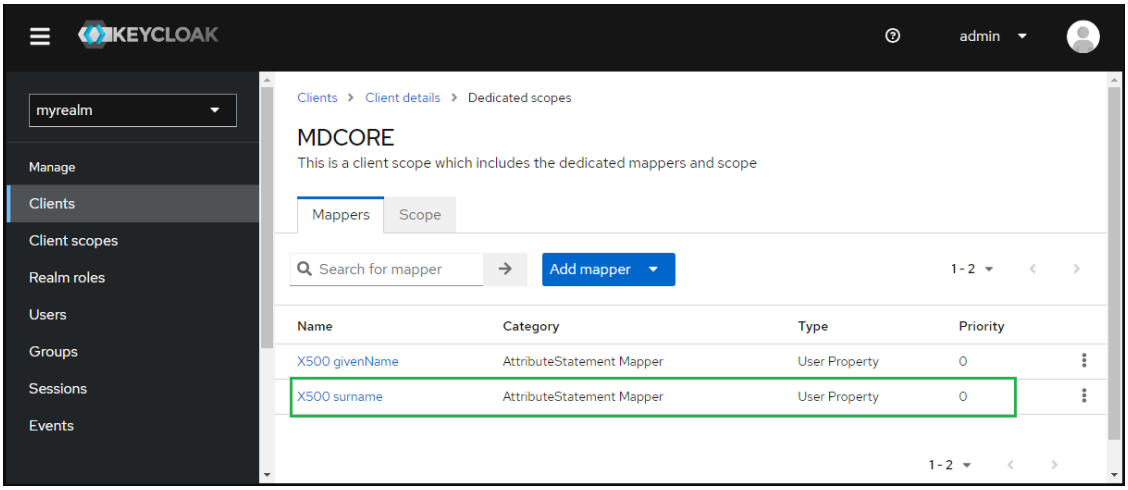
10. Back to `Mappers` tabs, click on `X500 givenName`.



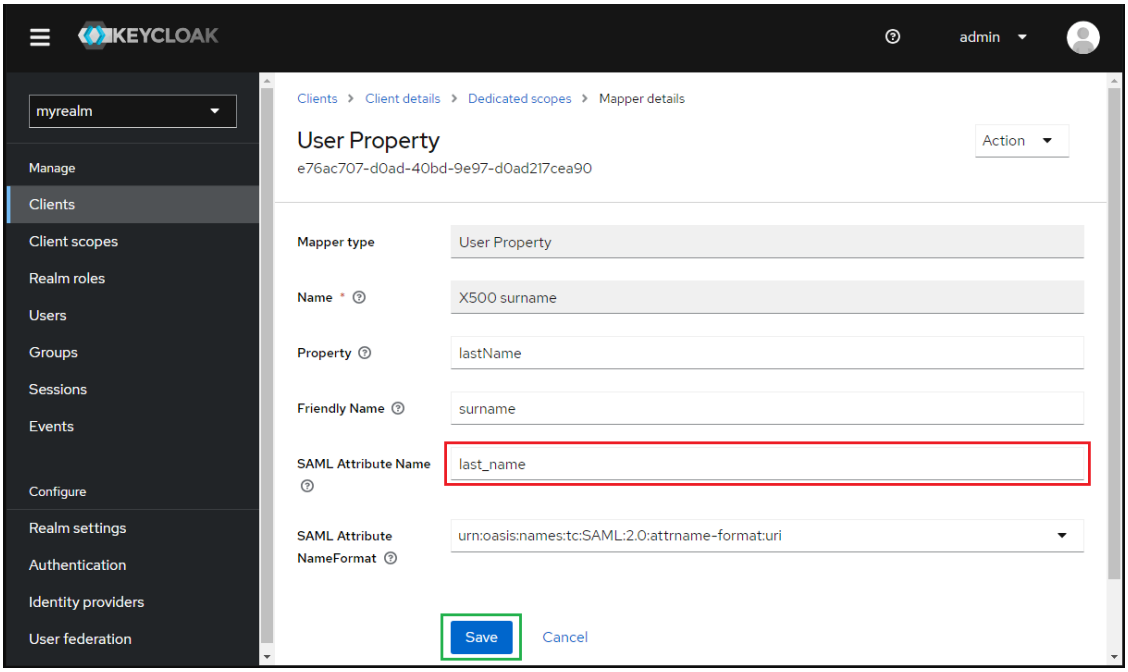
11. Enter first_name to `SAML Attribute Name` and click `Save`.



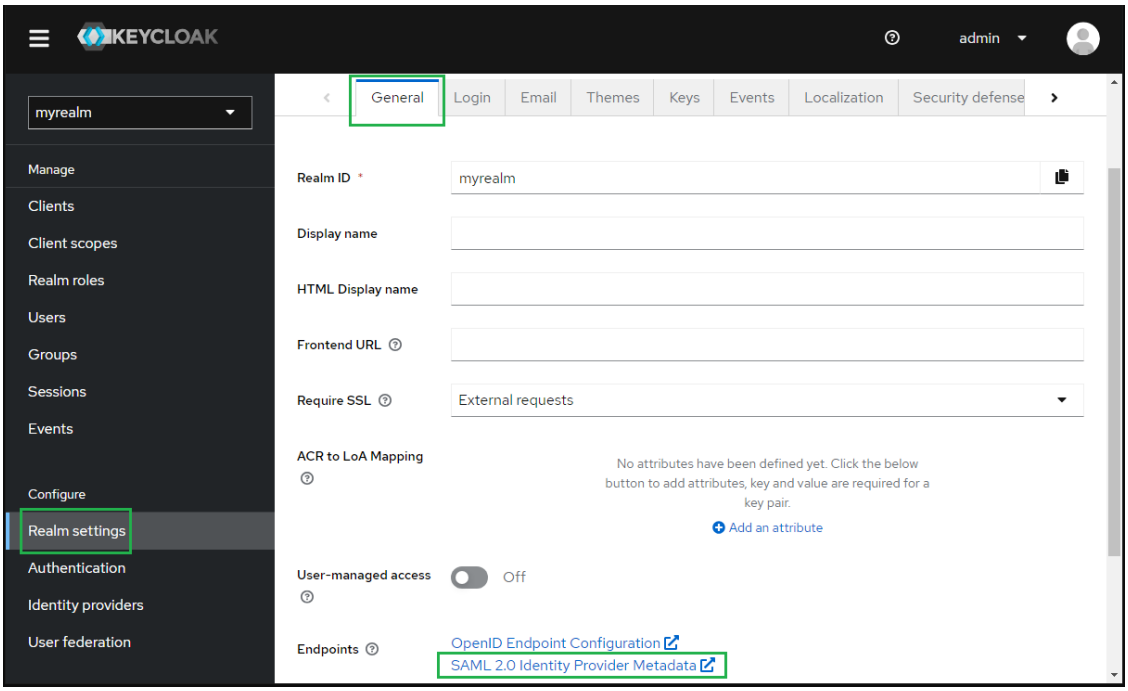
12. Click `X500 surname` in `Mappers` tab.



13. Enter last_name for `SAML Attribute Name` and click `Save`.

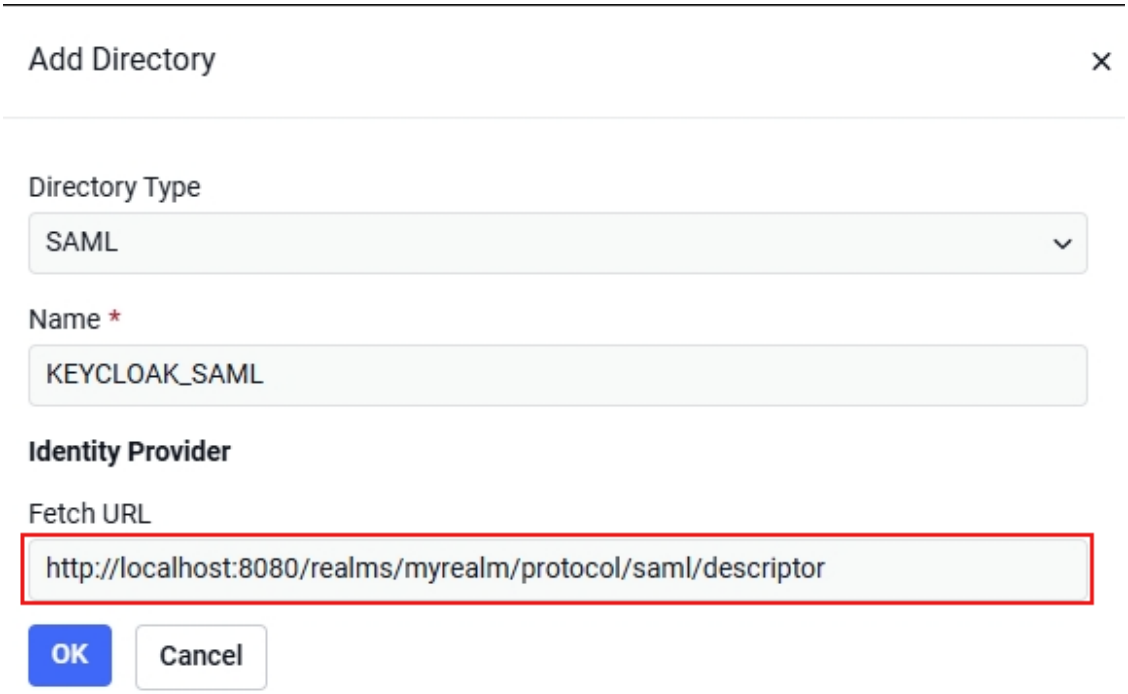


14. Select `Realm settings` in the sidebar, navigate to `General` tab, click `SAML 2.0 Identity Provider Metadata` and copy the metadata link.



Complete configuration in MetaDefender Core

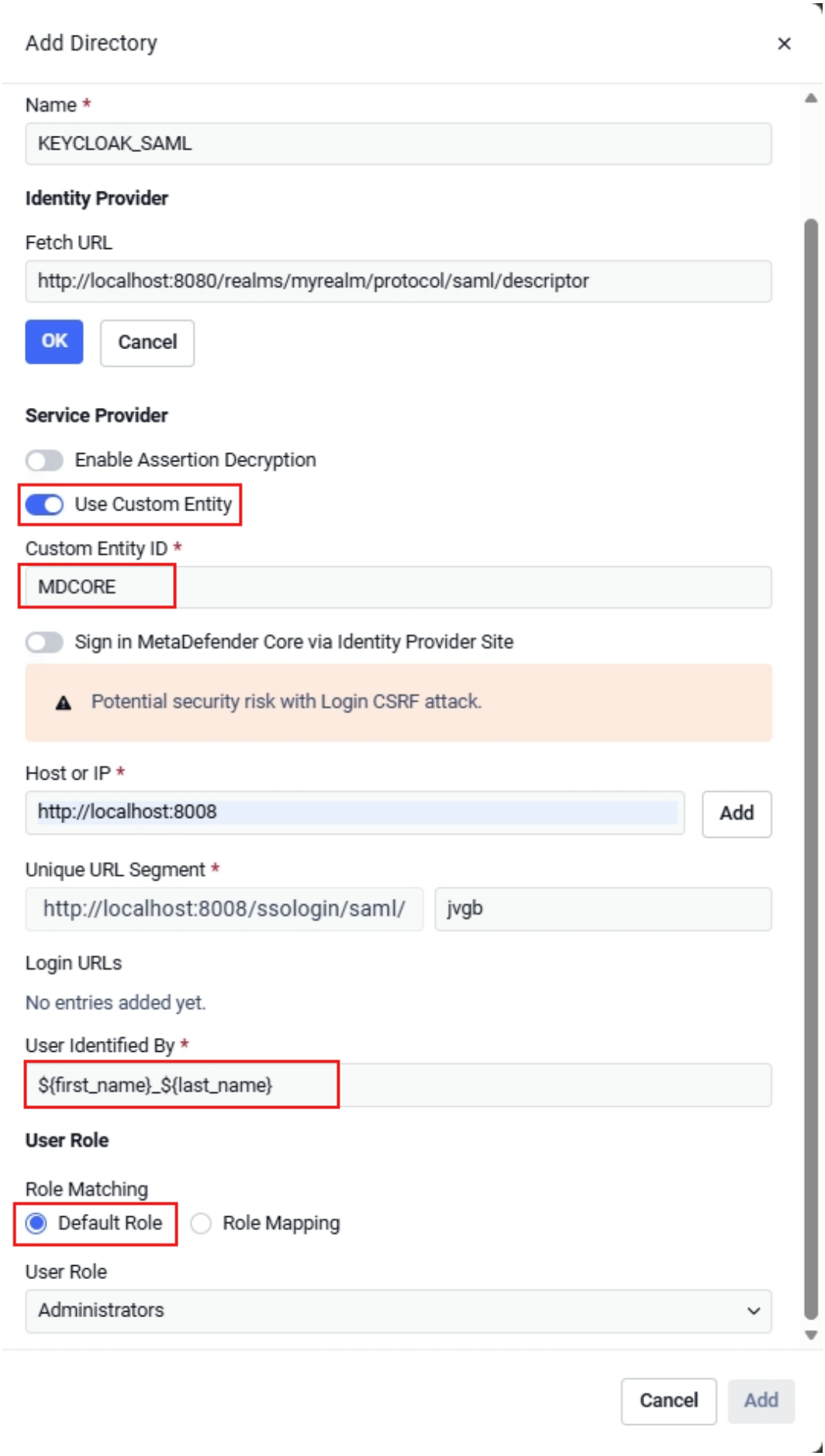
1. Switch to MetaDefender Core screen, under `Identity Provider`, click `Fetch URL`.
2. Paste the metadata link from Keycloak to the box under `Fetch URL` and click `OK` to ensure MetaDefender Core can set Keycloak as its IdP.



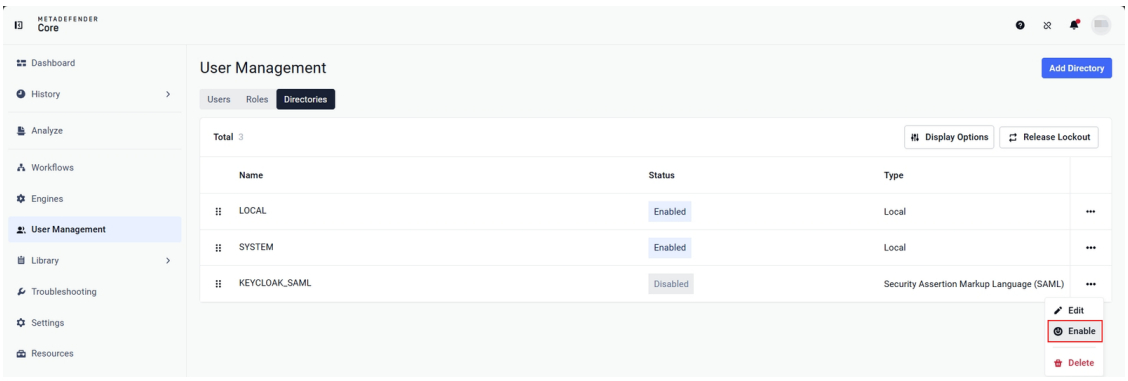
3. Under `Service Provider` section, enable `Use Custom Entity ID` and enter MDCORE in `Custom Entity ID` field.

> **Info**
>
> The ID set for `Use Custom Entity ID` must match `Client ID` created by Keycloak.

4. Fill in the user identity under `Use Identified by` with ${first_name}_${last_name}, for example.
5. Select the appropriate role for the user under `User Role`.
6. Click `Add` to complete the settings.

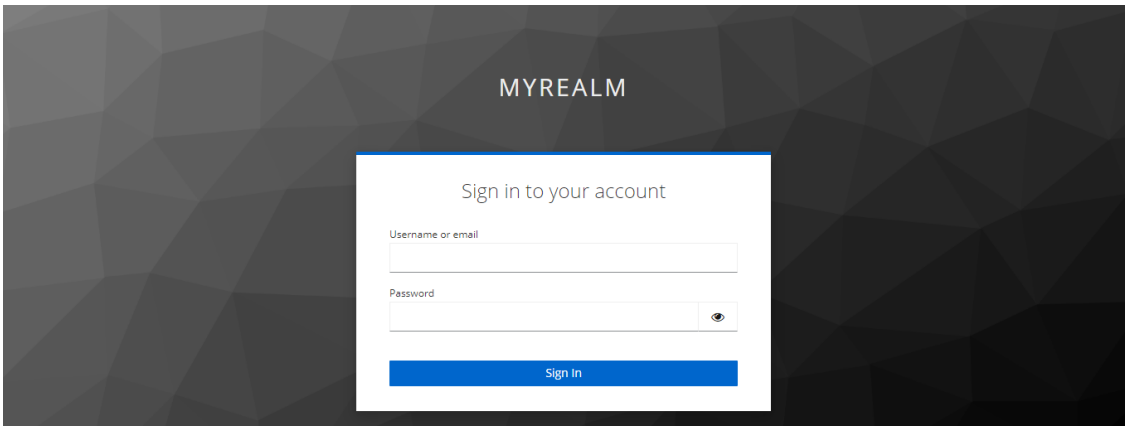

Add Directory
Name *
KEYCLOAK_SAML
Identity Provider
Fetch URL
http://localhost:8080/realms/myrealm/protocol/saml/descriptor
OK
Cancel
Service Provider
Enable Assertion Decryption
Use Custom Entity
Custom Entity ID *
MDCORE
Sign in MetaDefender Core via Identity Provider Site
Potential security risk with Login CSRF attack.
Host or IP *
http://localhost:8008
Add
Unique URL Segment *
http://localhost:8008/ssologin/saml/
jvgb
Login URLs
No entries added yet.
User Identified By *
${first_name}_${last_name}
User Role
Role Matching
Default Role
Role Mapping
User Role
Administrators
Cancel
Add

7. On `User Management` screen, toggle the new directory, *KEYCLOAK-SAML* in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all existing sessions will expire immediately.

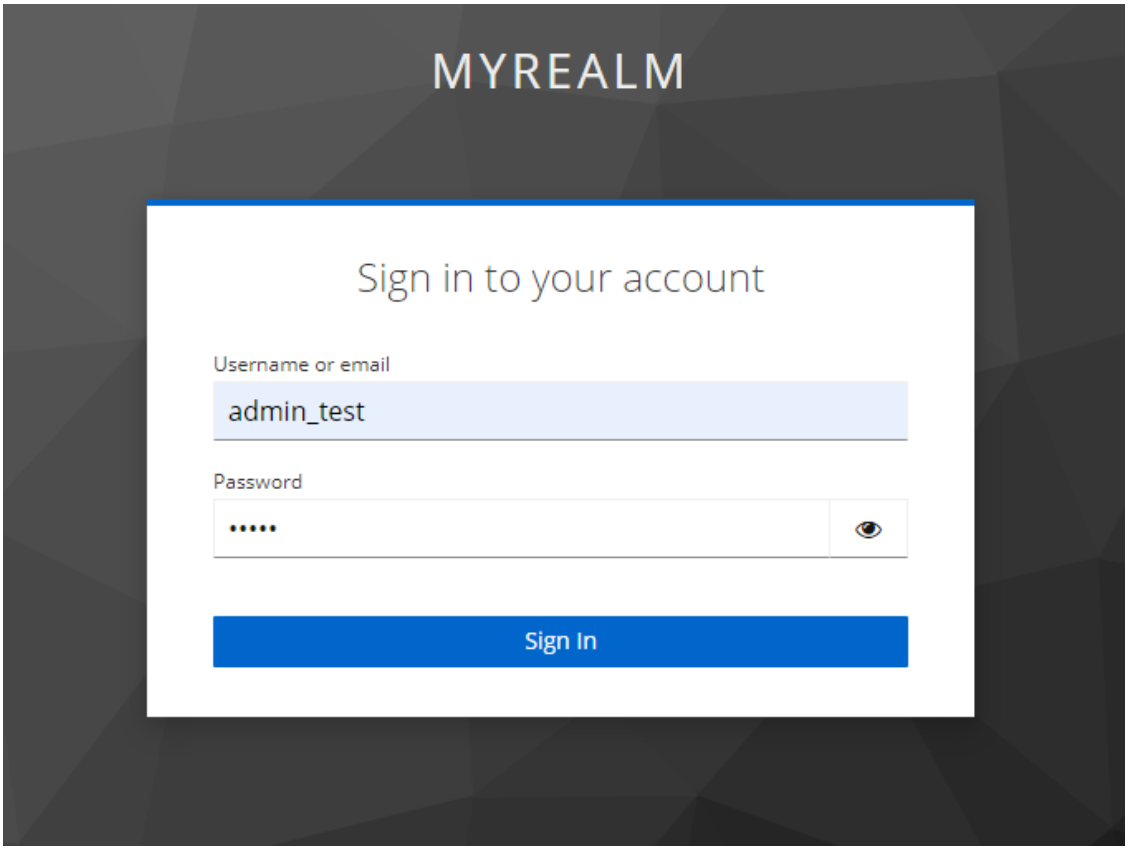


Test the integration

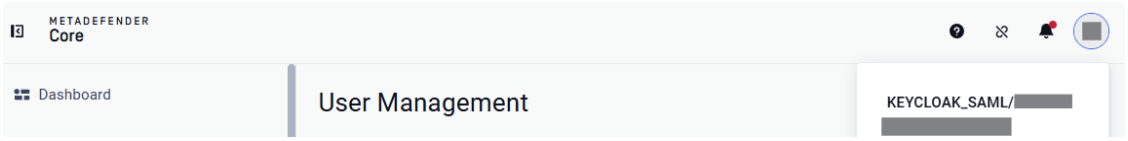
1. Click `Login` from the home screen of MetaDefender Core; the user is redirected to Keycloak page.



2. Sign in with the account registered in Keycloak.



3. If everything goes well, MetaDefender Core dashboard will be displayed with user identity set in the top right corner.



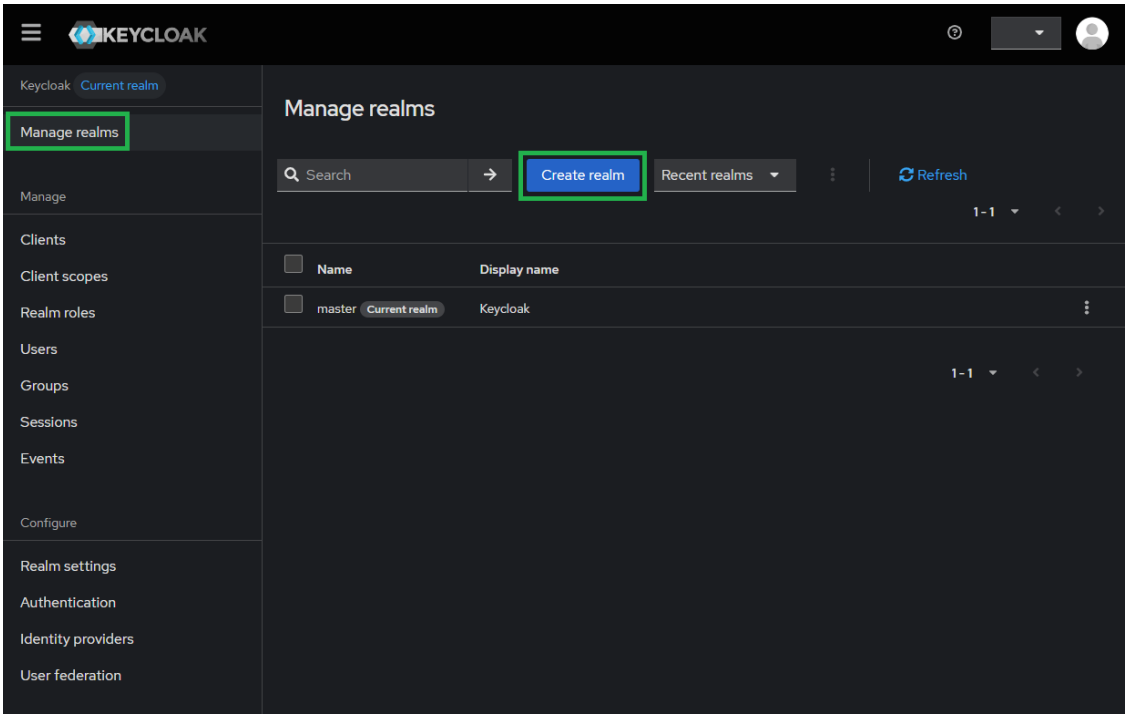
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Configure OIDC SSO

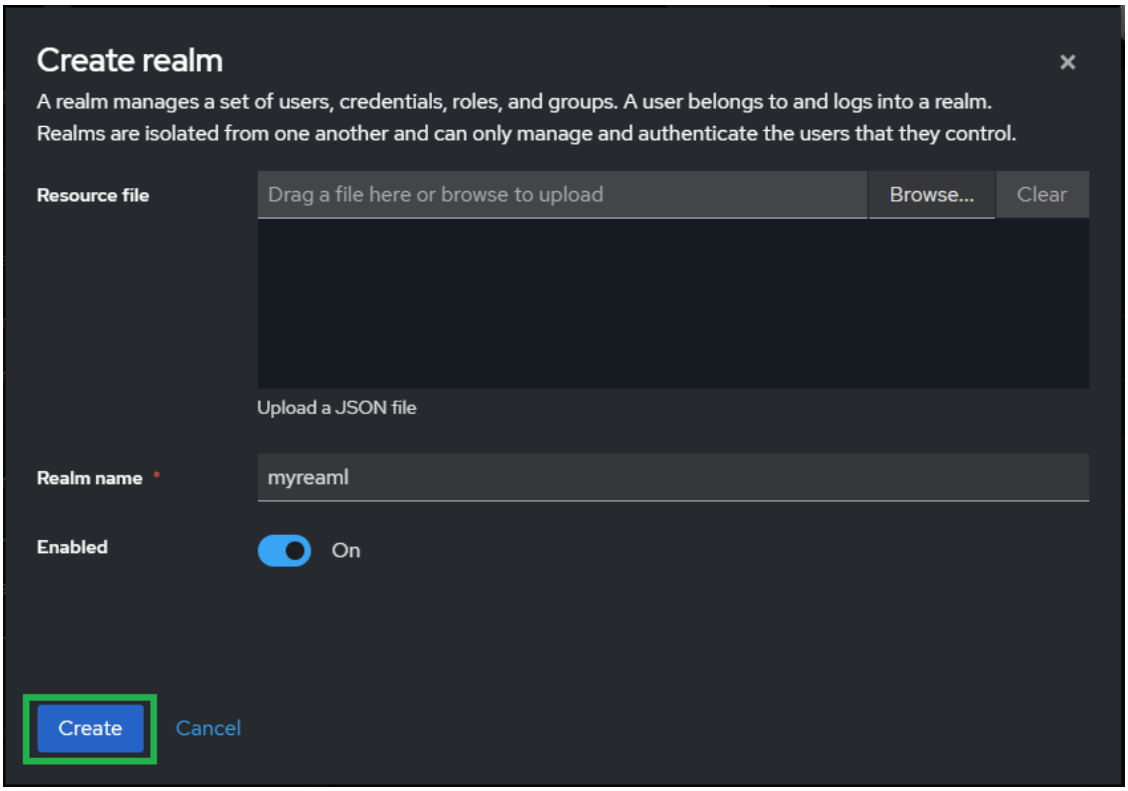
Create a realm in Keycloak

1. Sign in to Administrator Console, drop the list in the top left corner, and click `Create realm`.

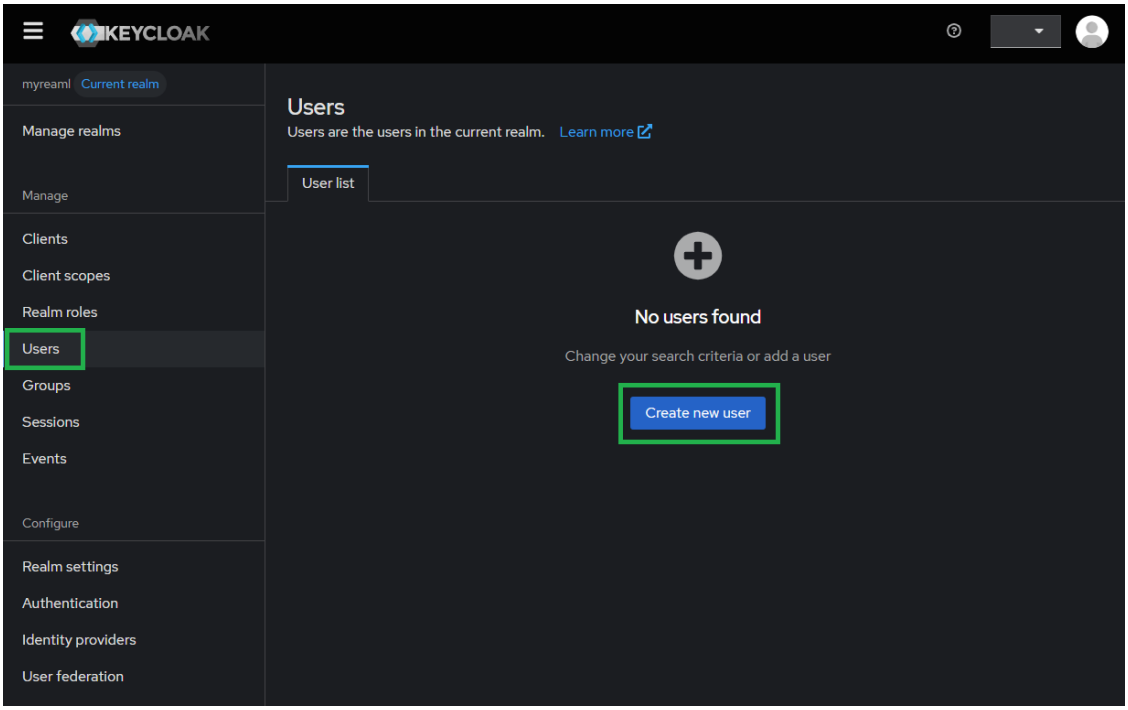
> **Info**
>
> Keycloak initially includes a single realm `master` which is used for managing Keycloak only and not for any applications.



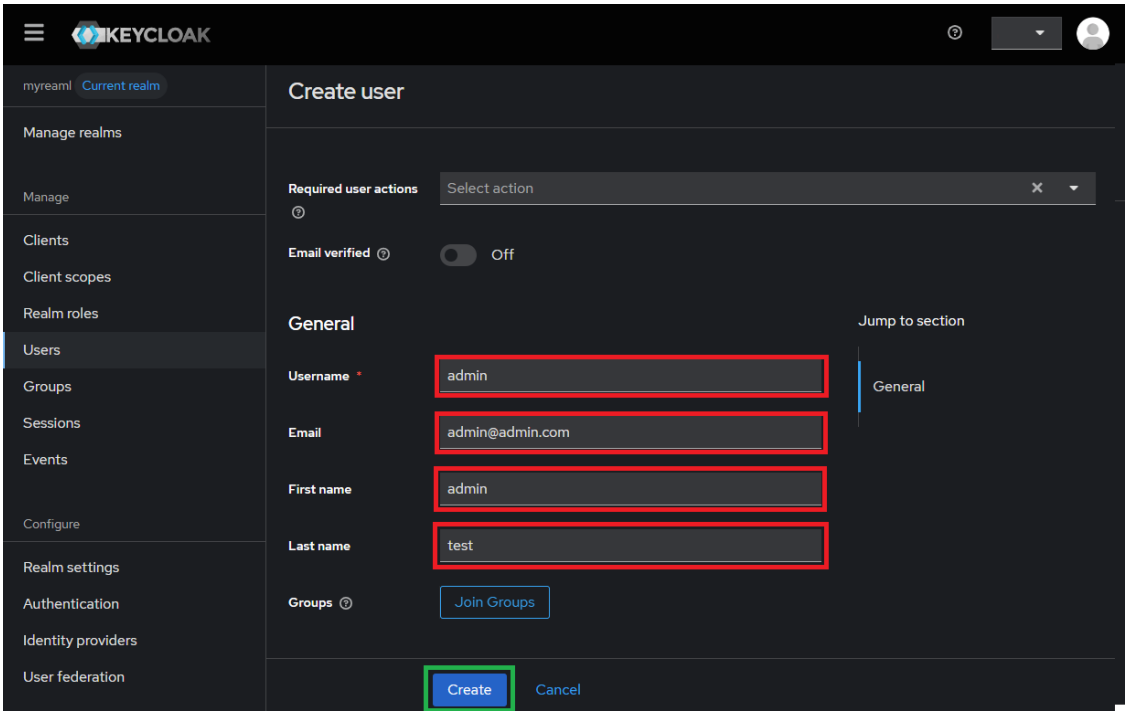
2. Enter `Realm name` e.g. myrealm and click `Create`.



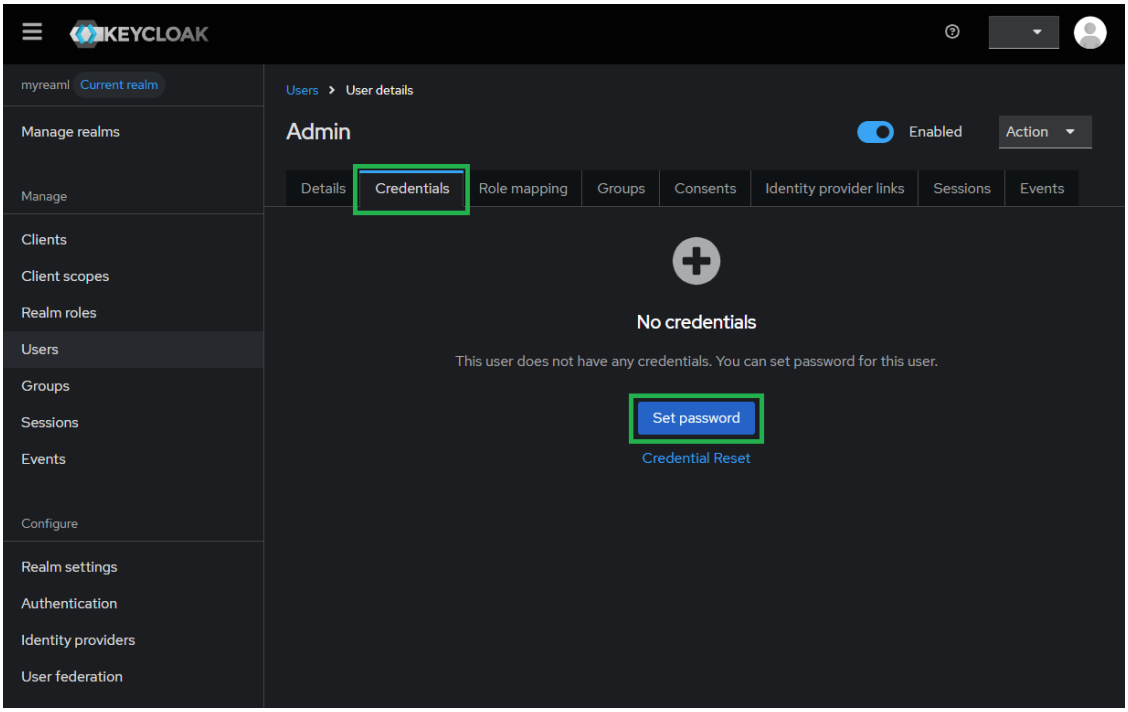
3. Select `Users` in the left sidebar and click `Create new user`.



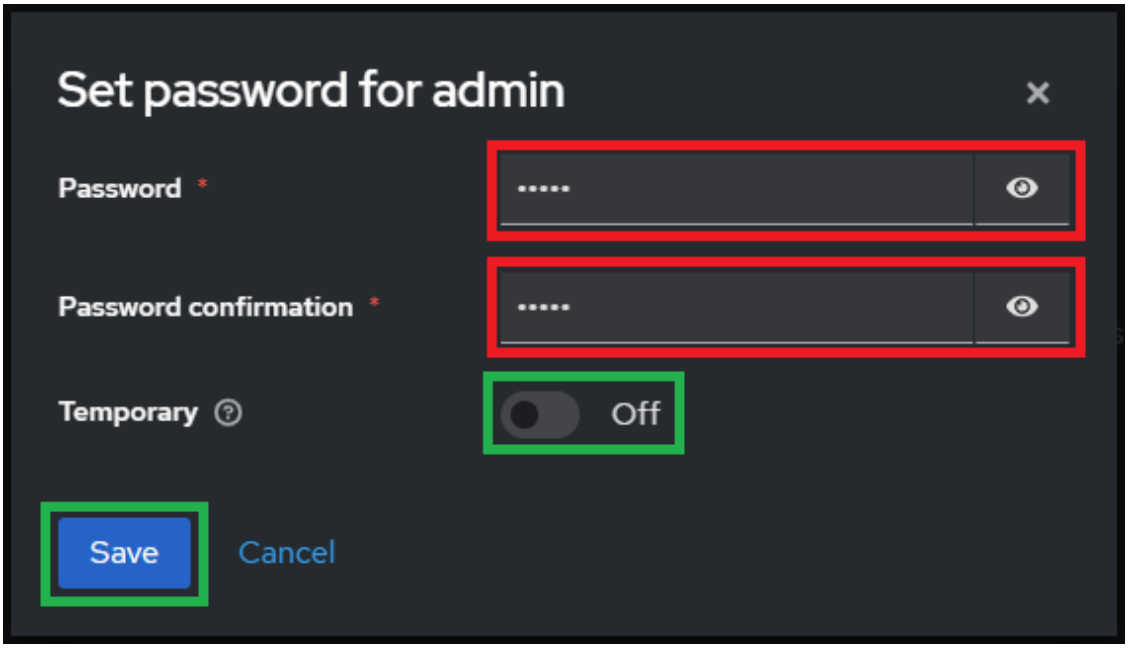
4. Enter values for `Username`, `Email`, `First name` and `Last name`; then click `Create`.



5. Under `User details`, select `Credentials` tab and click `Set password` to create a password for the user created in the previous step.

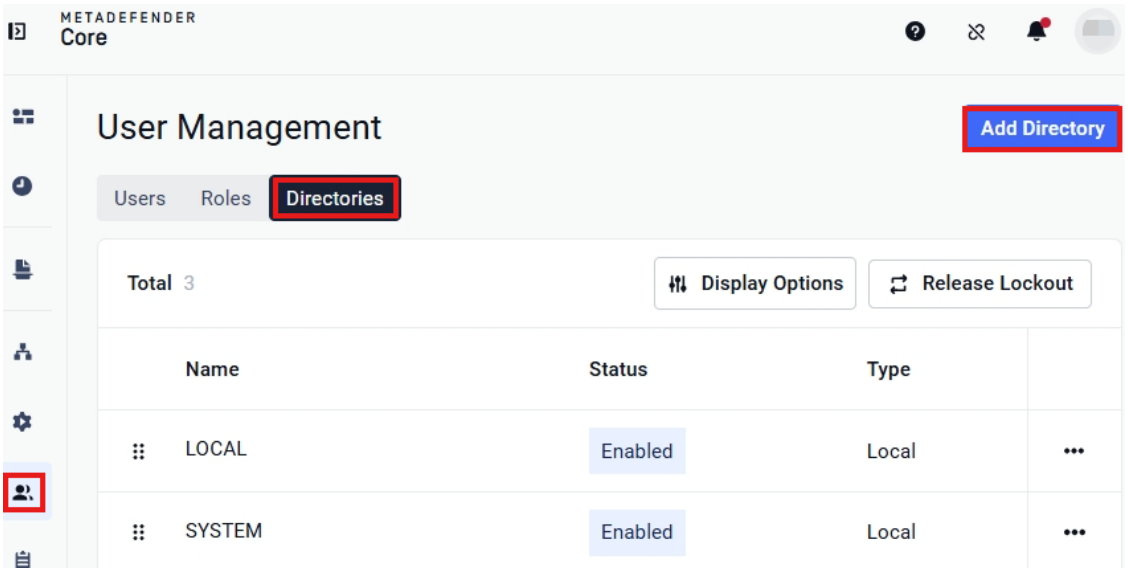


6. Enter the password and toggle `Temporary` to Off, then click `Save`.

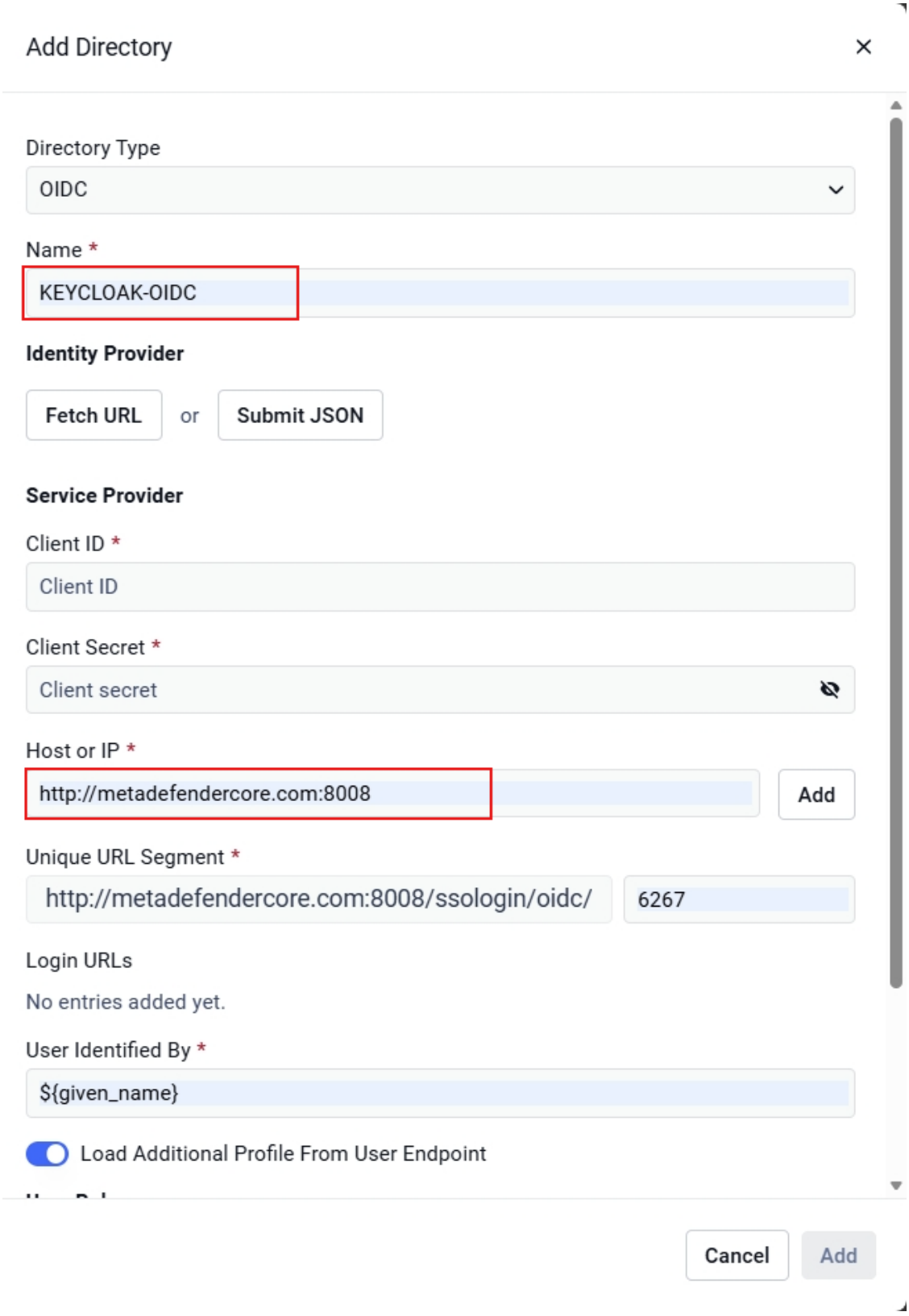


Create OIDC directory in MetaDefender Core

1. Sign in to MetaDefender Core management console.
2. Under `Dashboard`, click `User Management` in the left sidebar.
3. Under `User Management`, select `Directories` tab and hit `Add directory` in the top right.

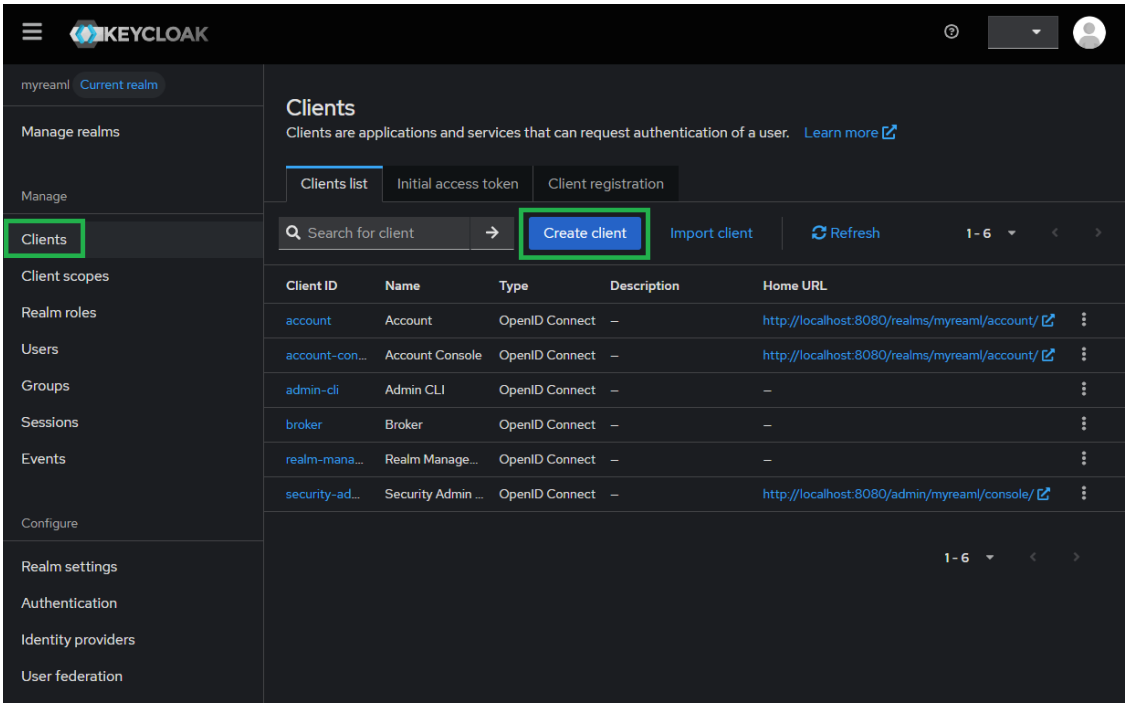


4. On `Add Directory` page, select OIDC in `Directory type`.
5. Fill `Name` of the new directory, such as KEYCLOAK_OIDC.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, using http://metadefendercore.com:8008 as an example.
7. Click `Add`.
8. Copy the value of `Login URLs` and store to *login_url*.

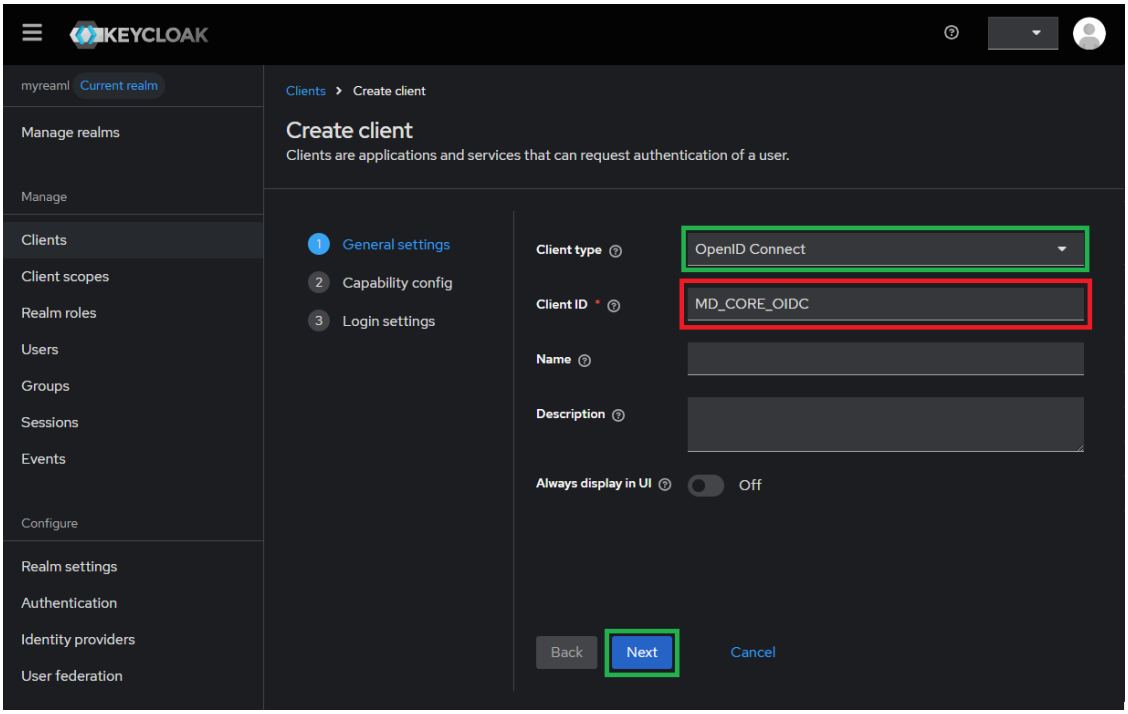


Create Keycloak application

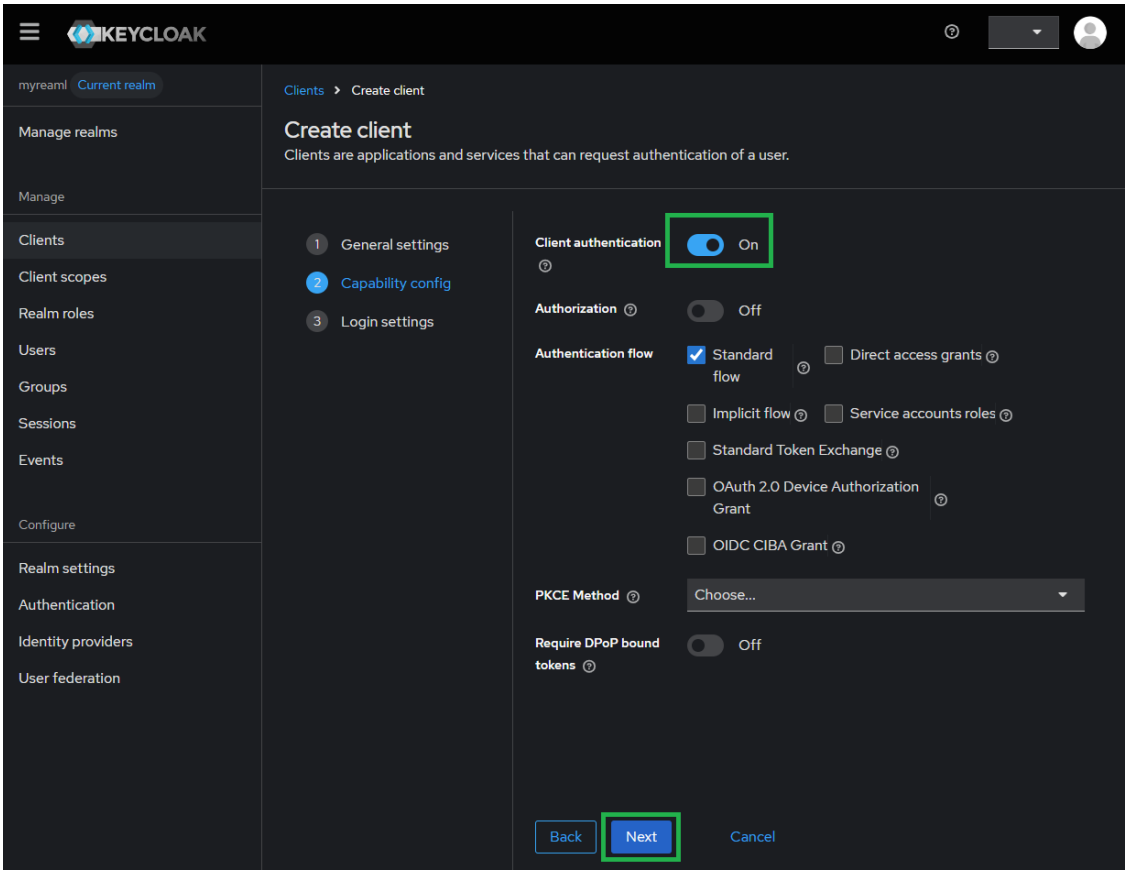
1. On screen `myrealm`, select `Clients` in the sidebar and click `Create client`.



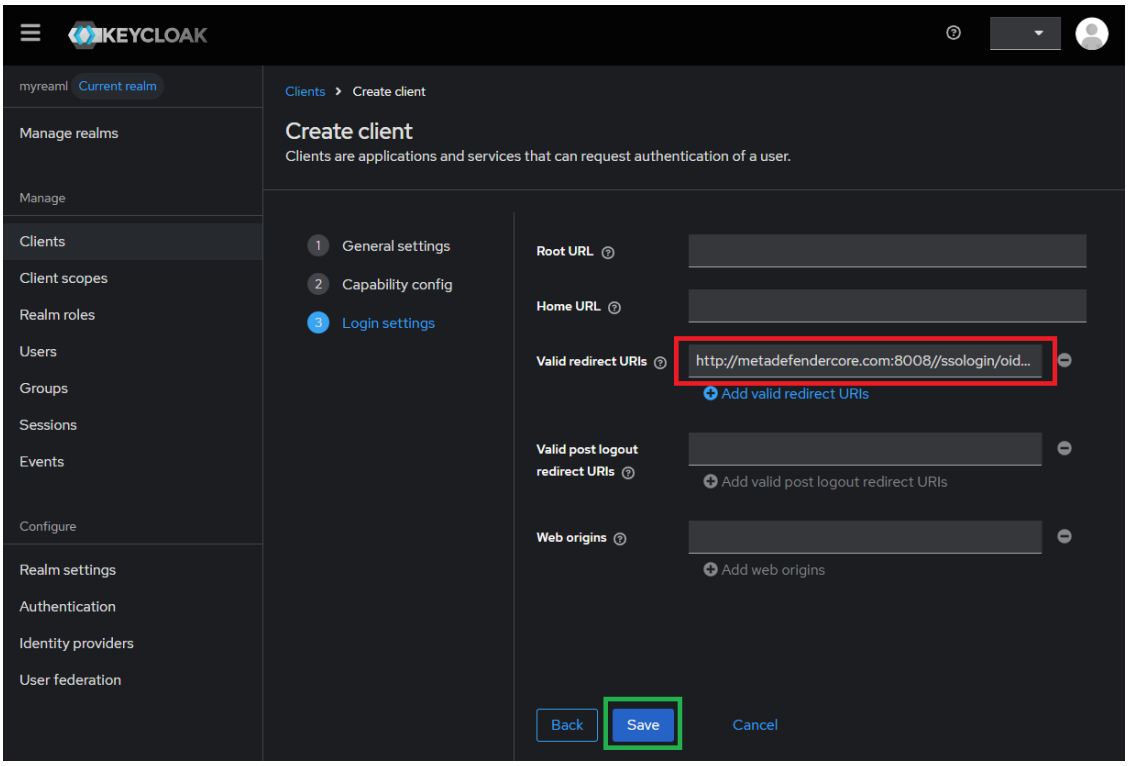
2. Choose `OpenID Connect` for `Client type` and enter `Client ID` (MD_CORE_OIDC for example) and store to *client_id*, then hit `Next`.



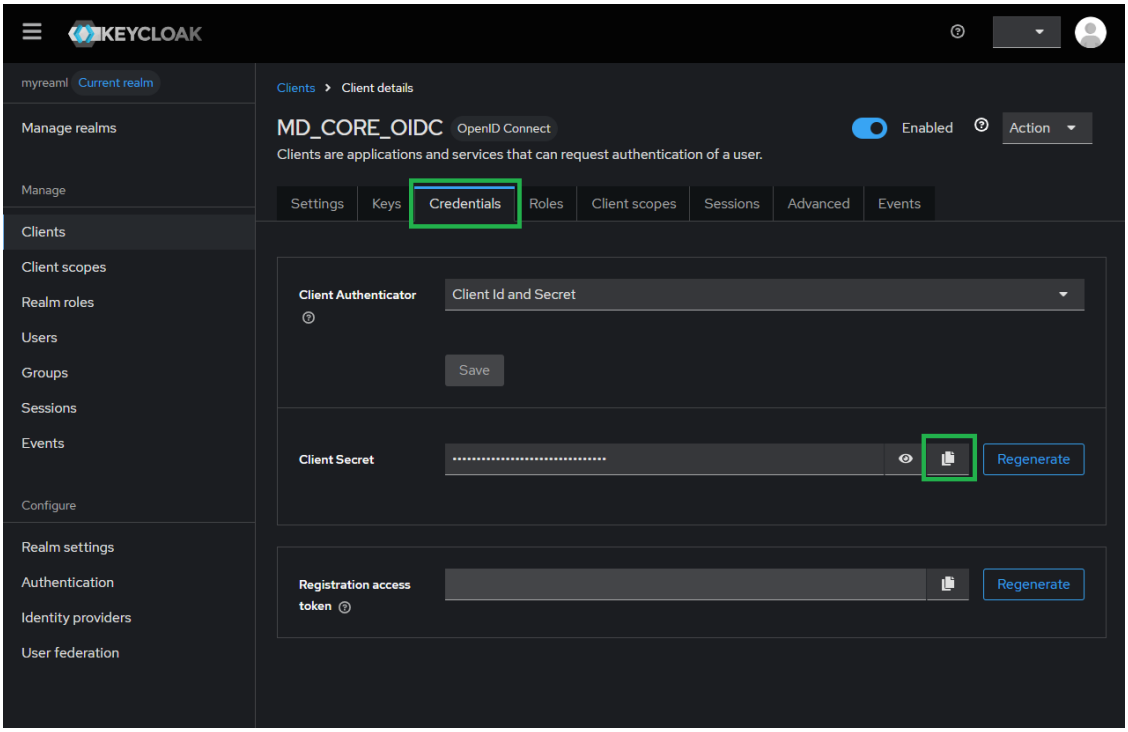
3. Turn on `Client authentication` then click `Next`.



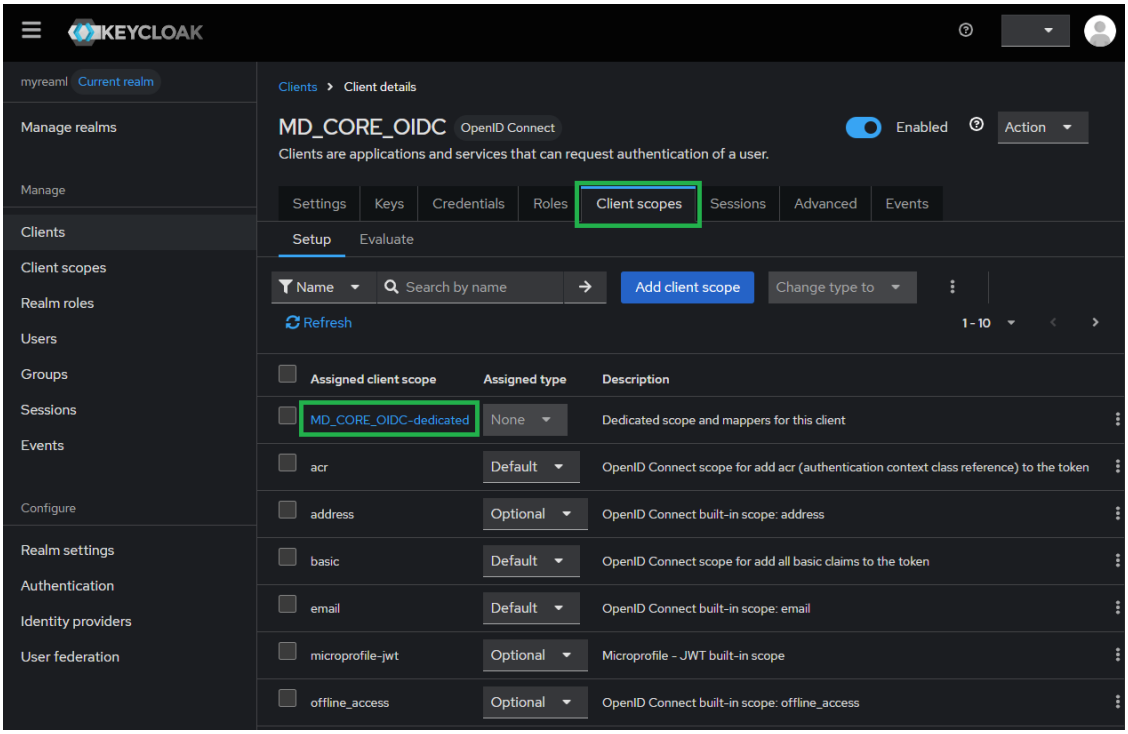
4. Paste the *login_url* into `Valid redirect URIs` and click `Save`.



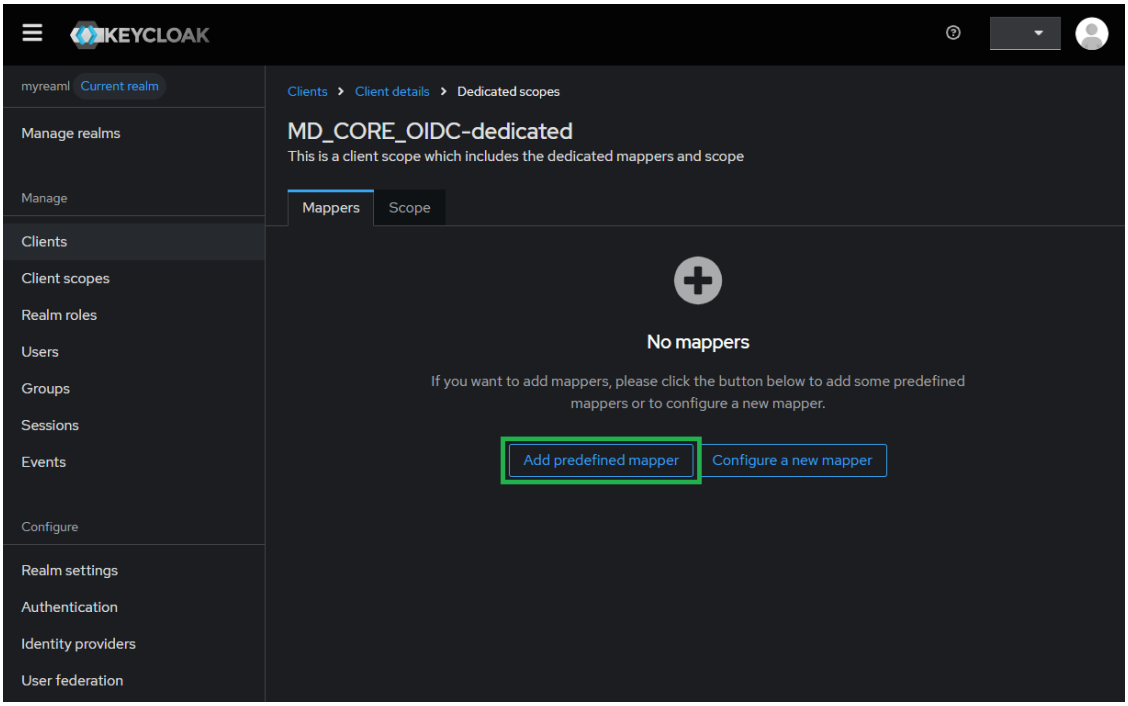
5. Go to tab `Credentials` and copy the `Client Secret` and store to *client_secret*.



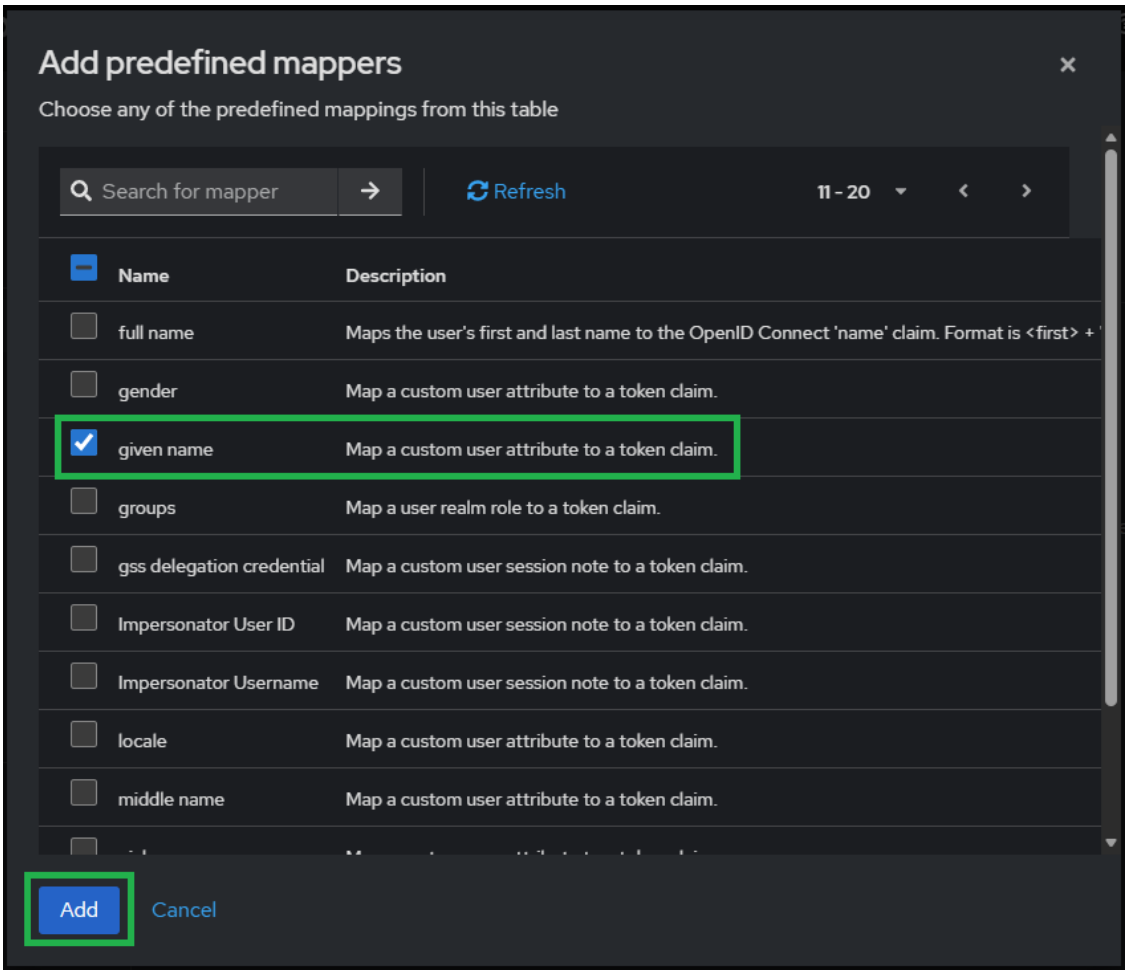
6. At tab `Client scopes`, select `MD_CORE_OIDC-dedicated`.



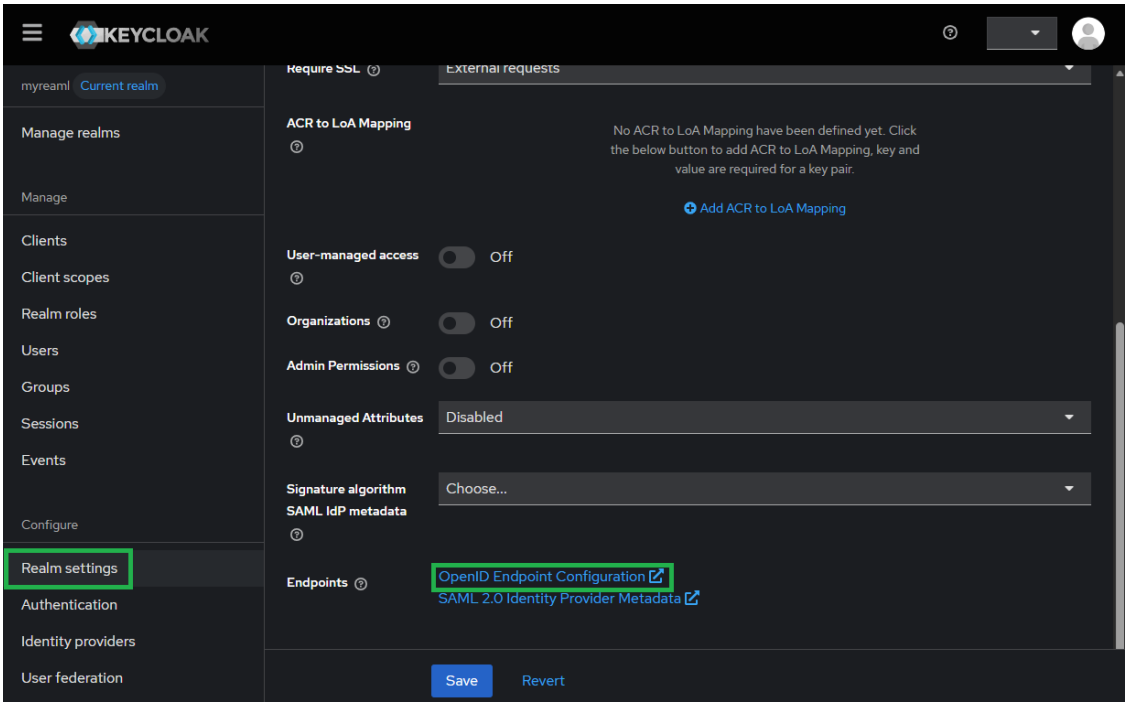
7. Under `Dedicated scopes`, navigate to `Mappers` tab and click `Add predefines mapper`.



8. Search for `given name`, then press `Add`.

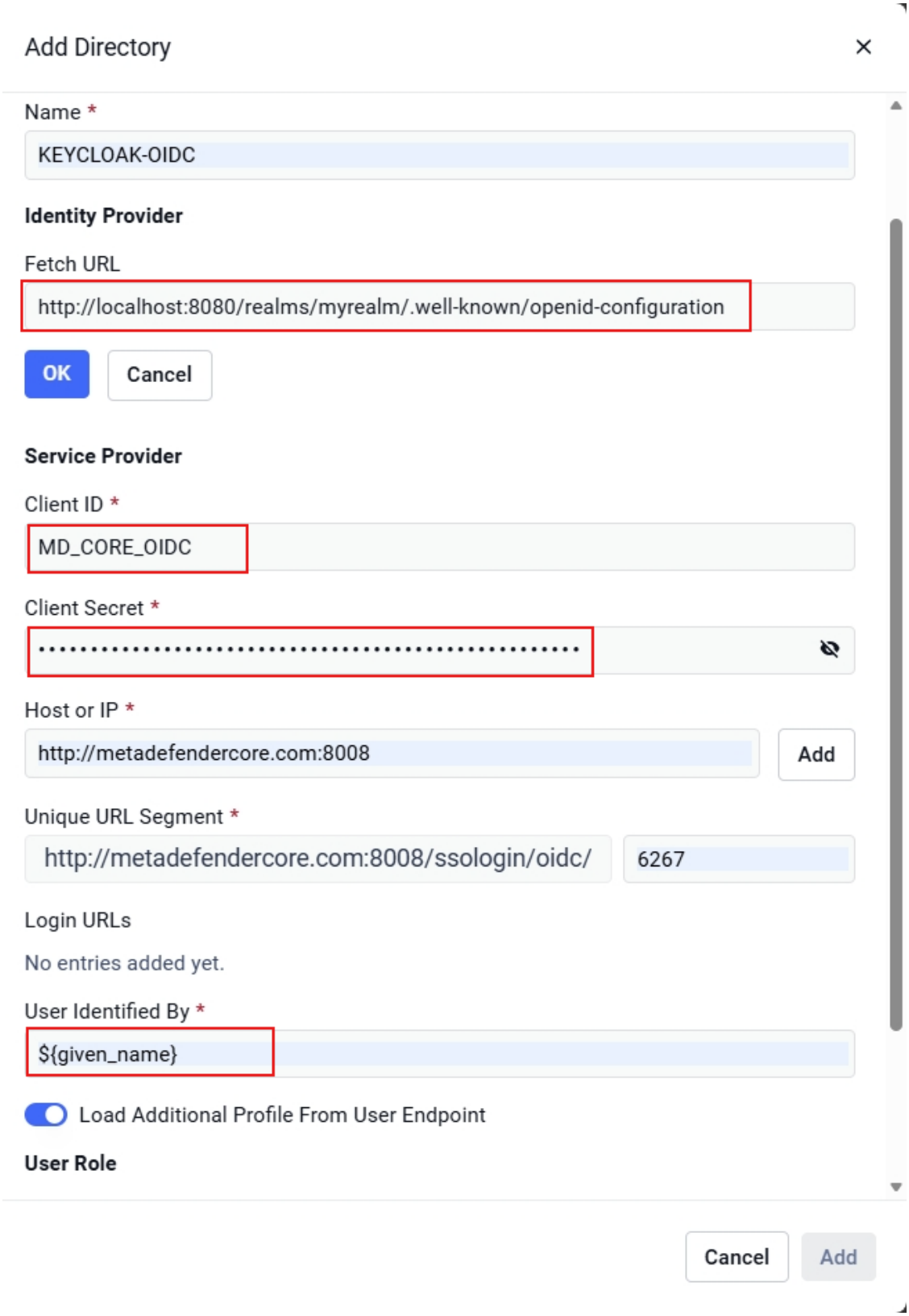


9. Select `Realm settings` in sidebar, navigate to tab `General`, click `OpenID Endpoint Configuration` and store OIDC metadata link to *metadata_url*.

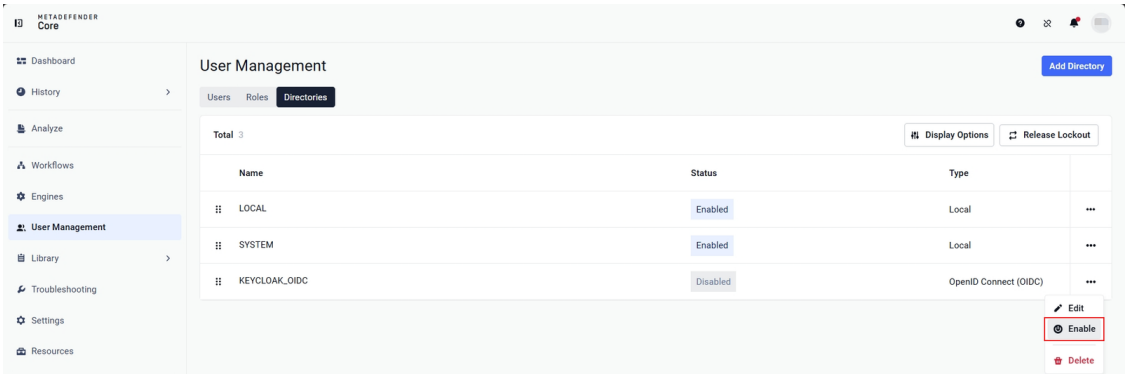


Complete configuration in MetaDefender Core

1. Switch to MetaDefender Core screen, under `Identity Provider`, click `Fetch URL`.
2. Paste the *metadata_url* from Keycloak to the box under `Fetch URL` and click `OK` to ensure MetaDefender Core can set Keycloak as its IdP.
3. Under section `Service Provider`, paste the *client_id* and *client_secret* to boxes under `Client ID` and `Client Secret` respectively.
4. Fill user identity under `User identified` by with `${given_name}`.

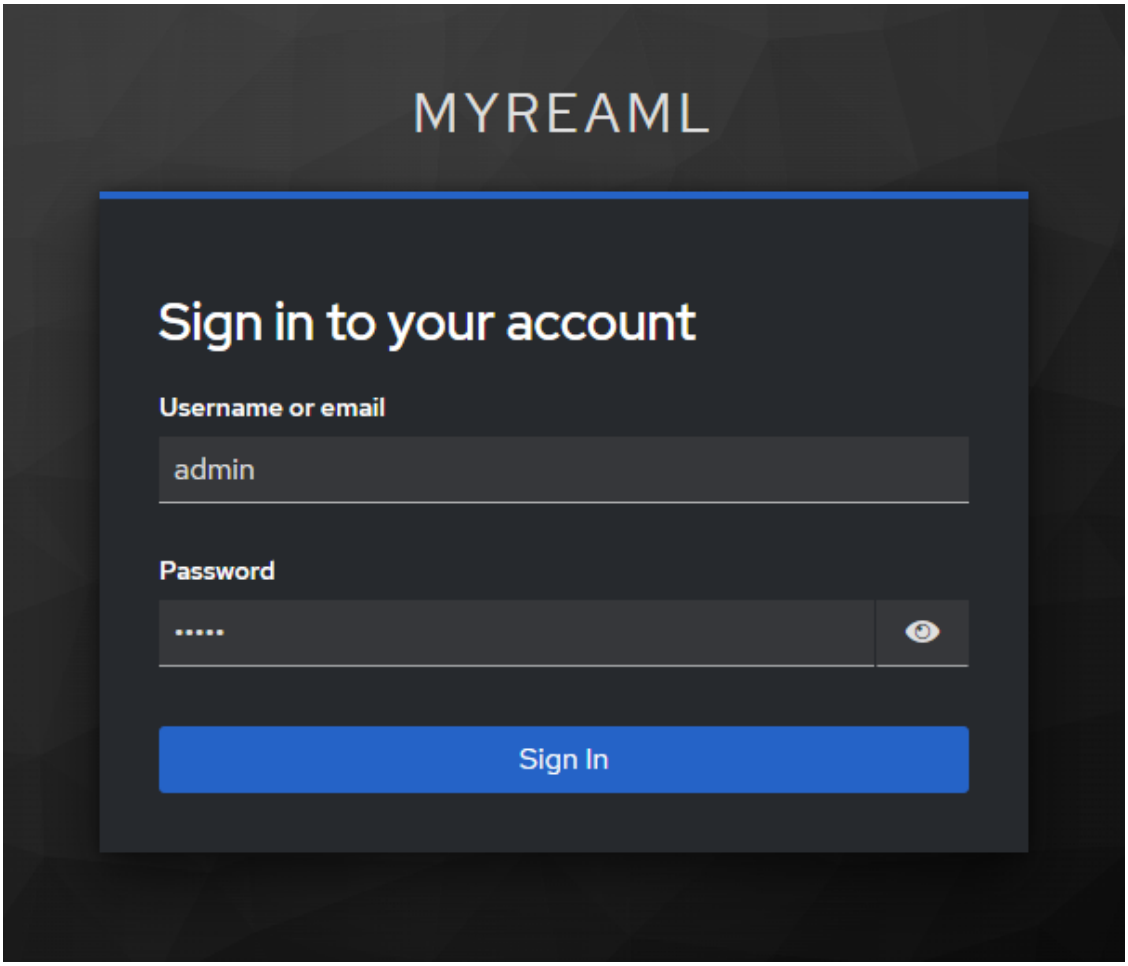


5. Select the appropriate role for the user under `User Role`.
6. Click `Add` to complete the settings.
7. On `User Management` screen, toggle the new directory, *KEYCLOAK-OIDC* in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all existing sessions will expire immediately.

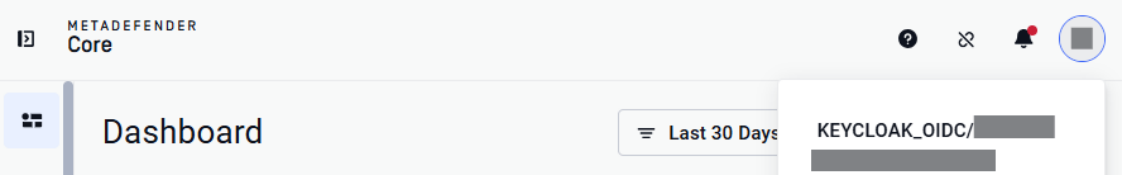


Test the integration

1. Click `Login` from the home screen of MetaDefender Core; the user is redirected to Keycloak page.
2. Sign in with the account registered in Keycloak.



3. If everything goes well, MetaDefender Core dashboard will be displayed with user identity set in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Microsoft Active Directory Federation Services (ADFS)

The instructions guide users on how to configure Microsoft Active Directory Federation Services SAML and OIDC Single Sign-on for MetaDefender Core.

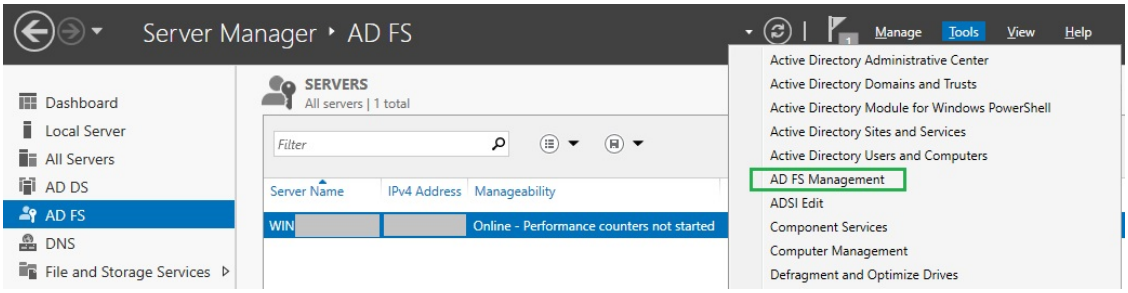
Configure SAML SSO

Info

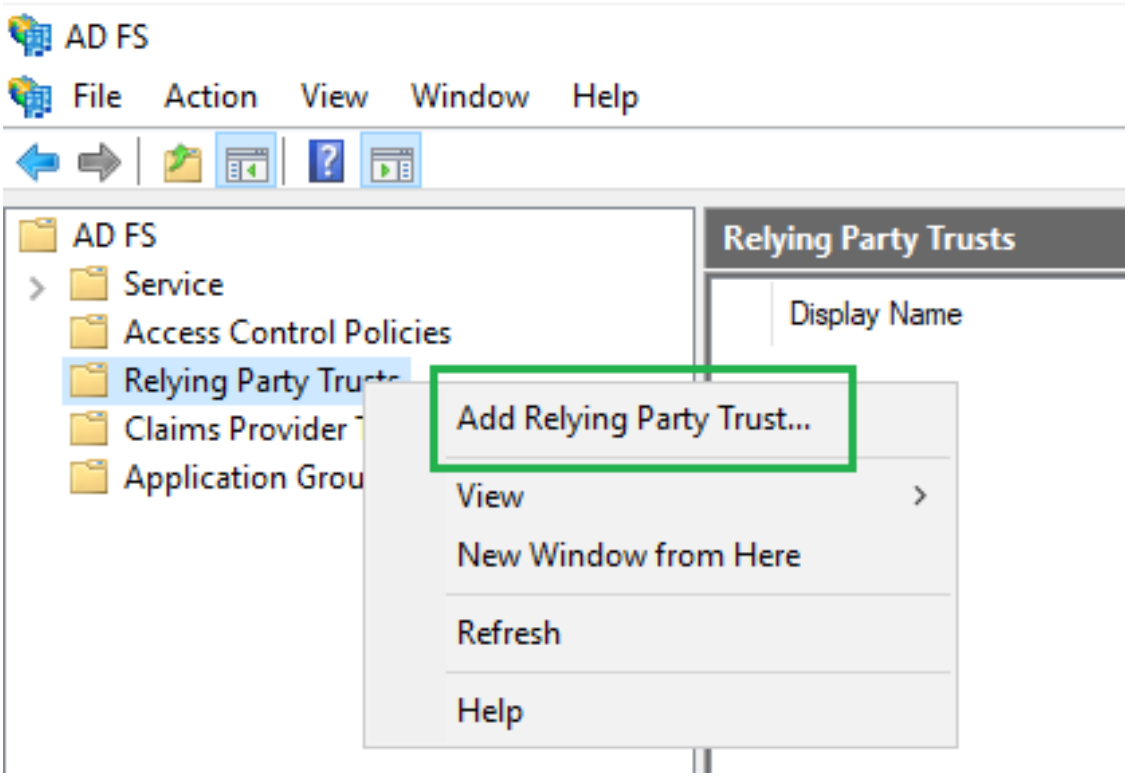
ADFS requires MetaDefender Core to enable HTTPS for connections. Please follow these steps to first enable HTTPS on MetaDefender Core.

Add Relying Party Trust for ADFS

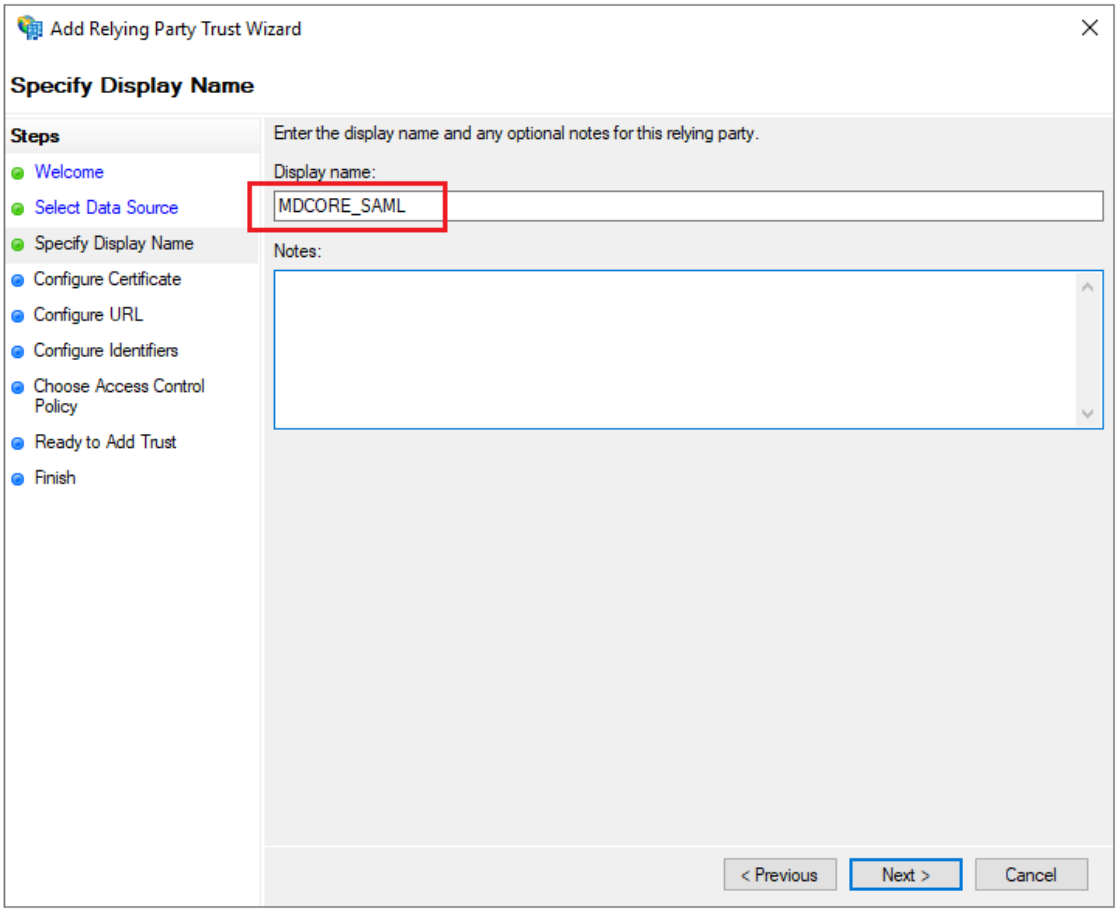
1. In the top right corner of `Server Manager`, click `Tools` and select `AD FS Management` from the drop-down list.



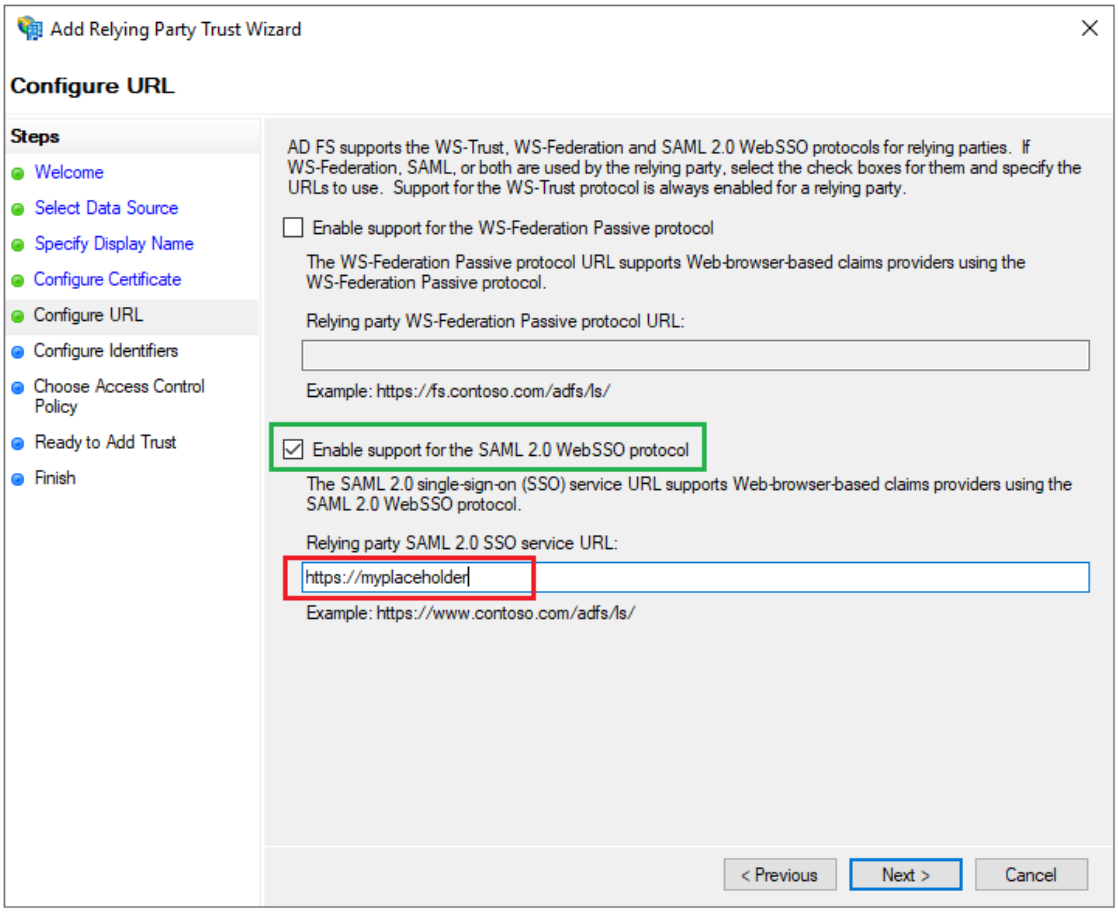
2. In `AD FS` window, right click on `Relying Party Trusts` and select `Add Relying Party Trust` from the drop-down menu.



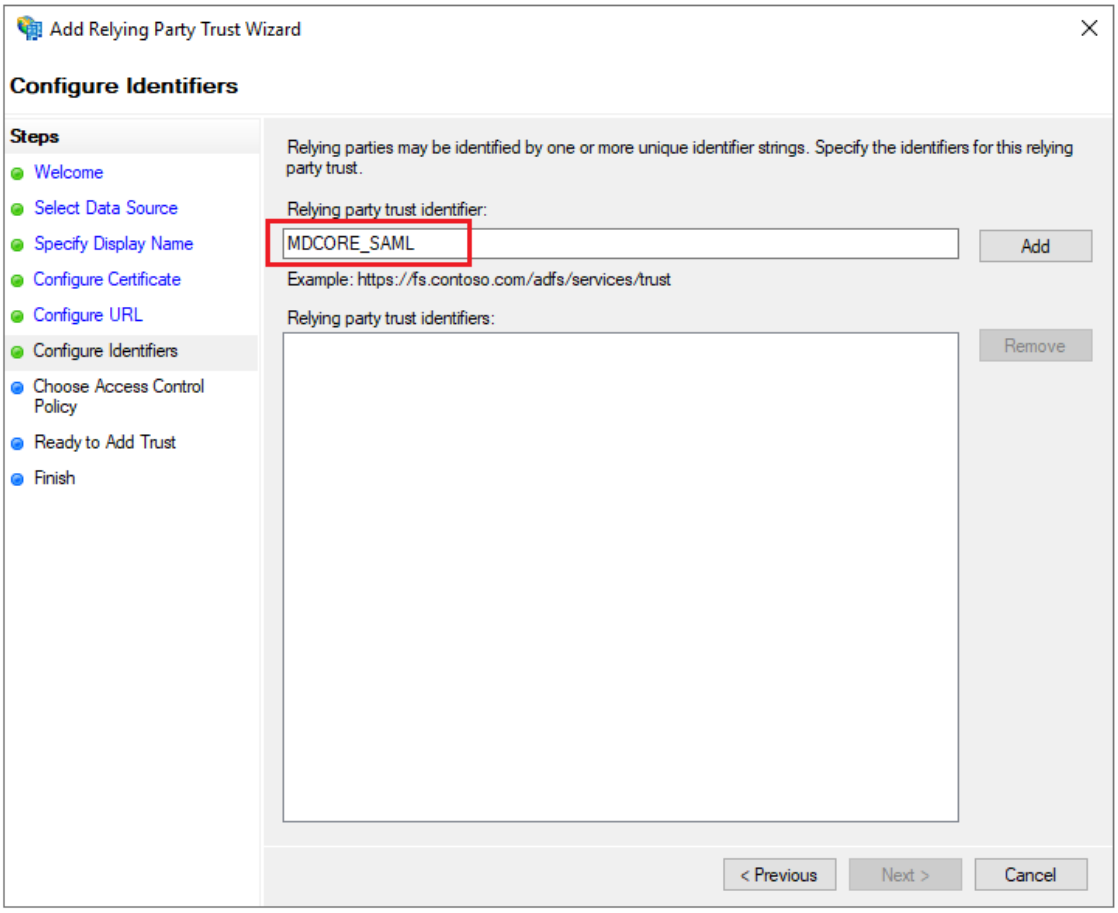
3. In `Add Relying Party Trust Wizard,` select `Claims aware` on the `Welcome` page and click `Start`.
4. On `Select Data Source` page, choose `Enter data about the relying party manually` and click `Next`.
5. On `Specify Display Name` page, provide a descriptive name for your relying party, e.g. MDCORE_SAML, and click `Next`.



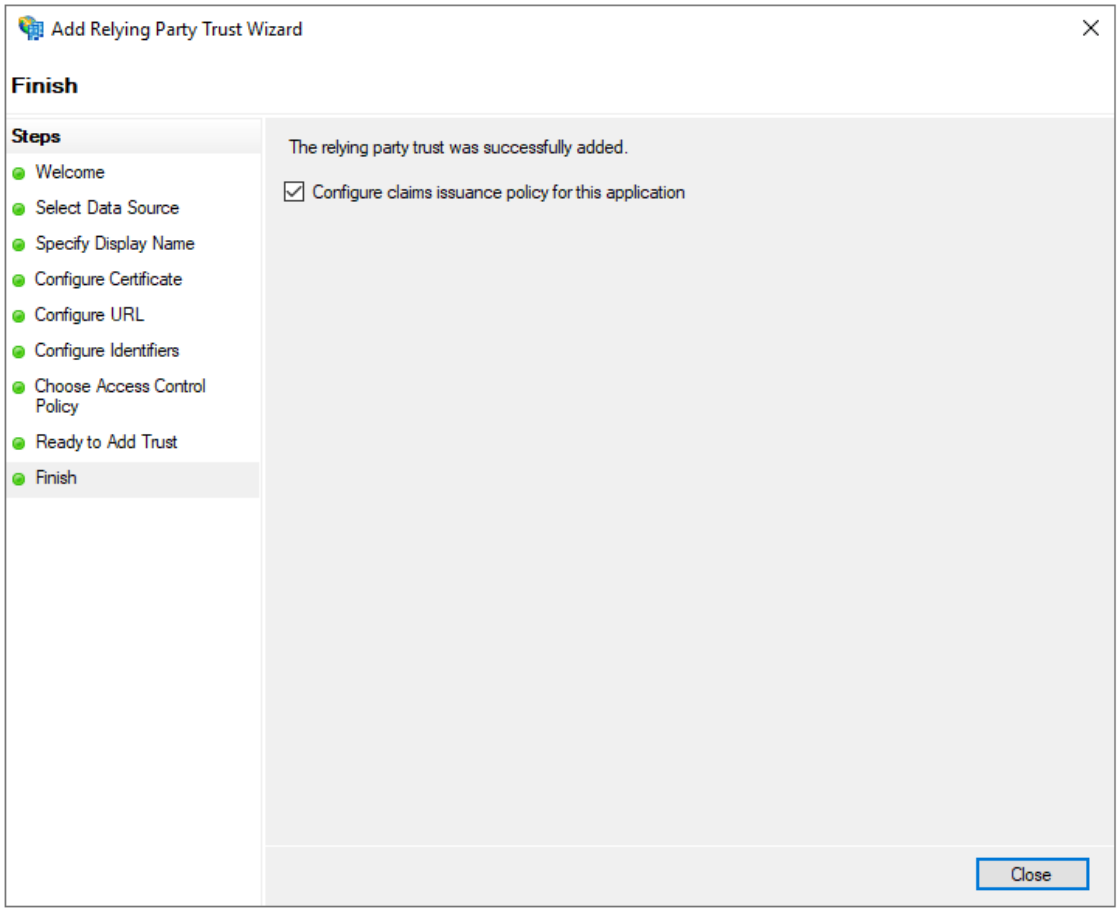
6. Click `Next` on `Configure Certificate` page.
7. On `Configure URL` page, check the box for `Enable support for the SAML 2.0 WebSSO protocol` and set a placeholder, using https://myplaceholder as an example, for `Relying party SAML 2.0 SSO service URL`.



8. On `Configure Identifiers` page, set MDCORE_SAML as `Relying party trust identifier`, then click `Add` and `Next`.

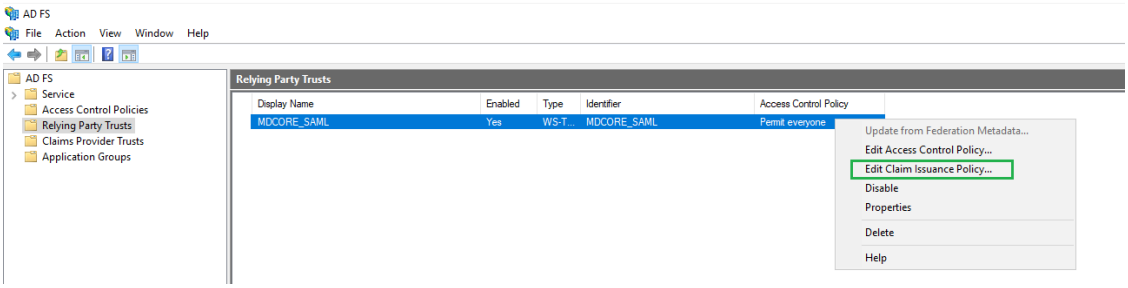


9. On `Choose Access Control Policy` page, select `Permit everyone` and click `Next`.
10. Click `Next` on `Ready to Add Trust` page to save your information.
11. If everything goes well, `Finish` page will be displayed for you to click `Close`.

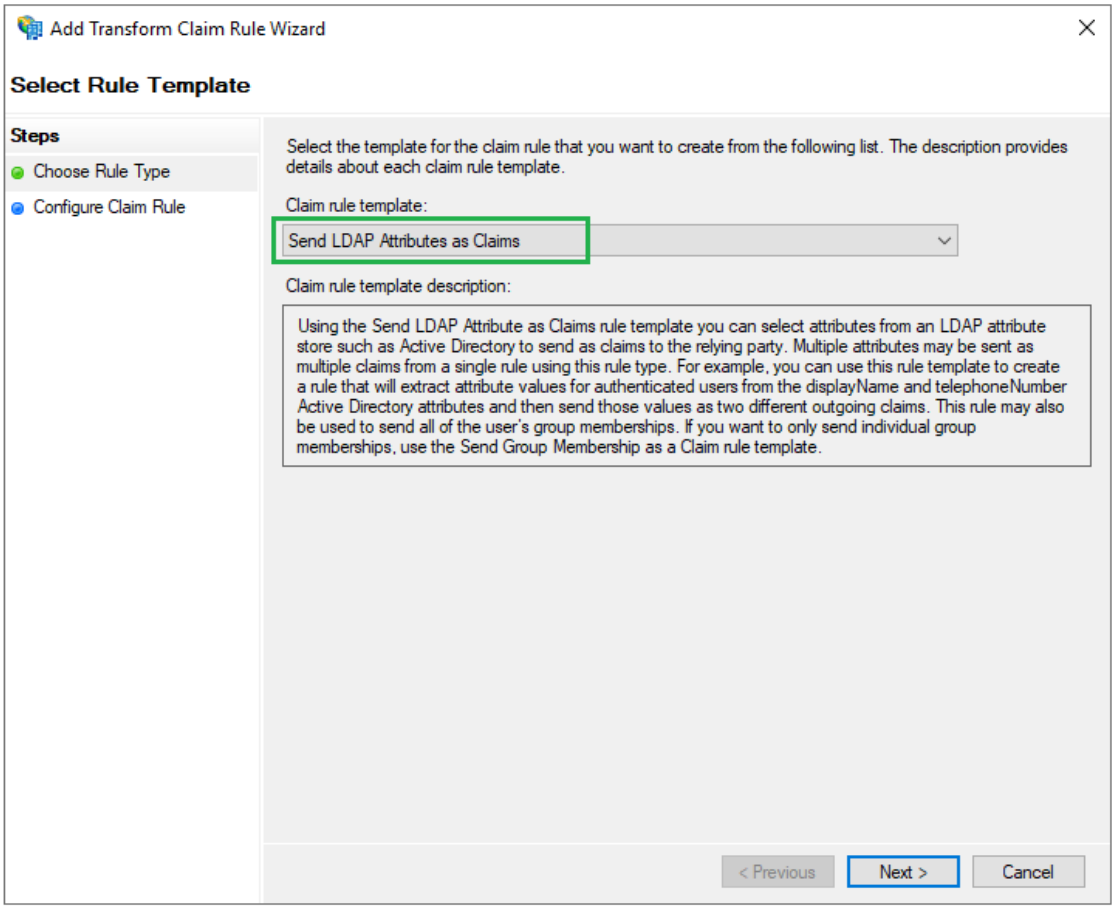


Configure claims Issuance Policy for ADFS

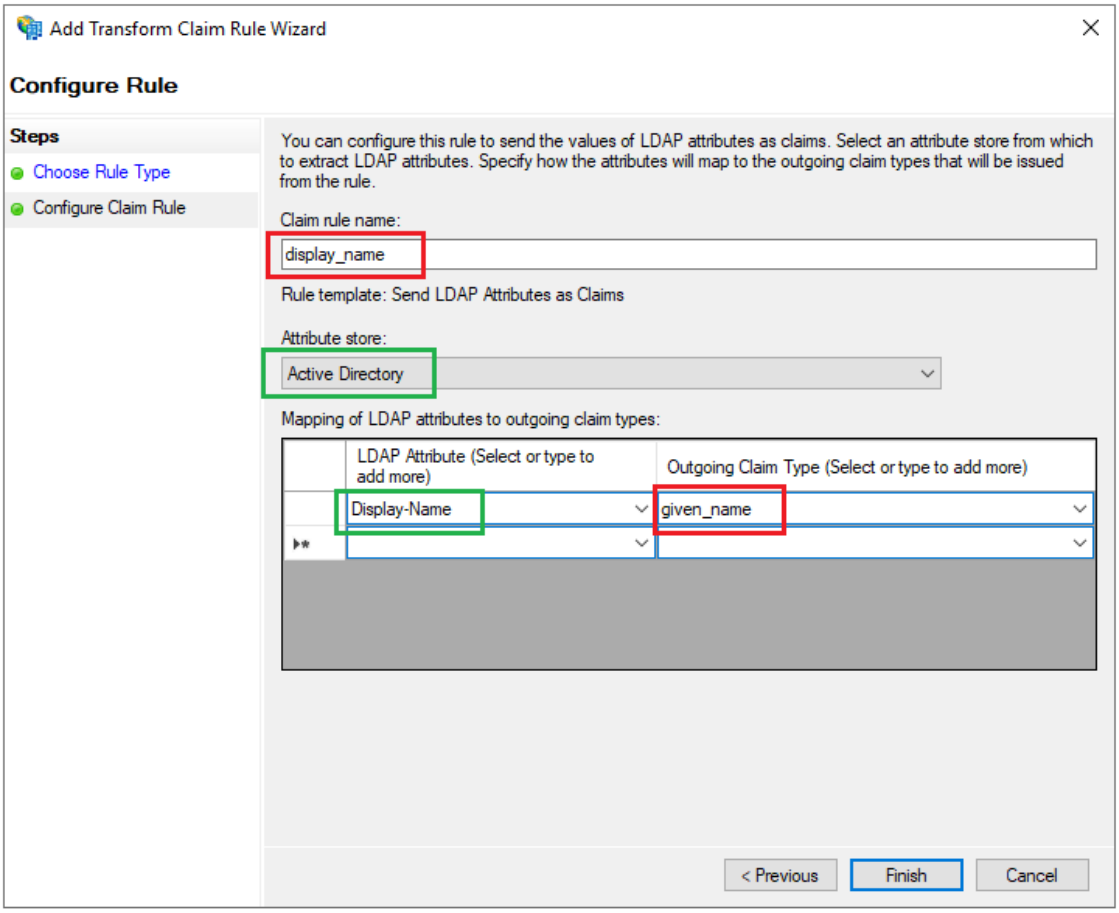
1. In `Relying Party Trusts` table of `AD FS` window, navigate to the item MDCORE_SAML that was added in the previous step, right-click on it, and select `Edit Claim Issuance Policy...` from the drop-down menu.



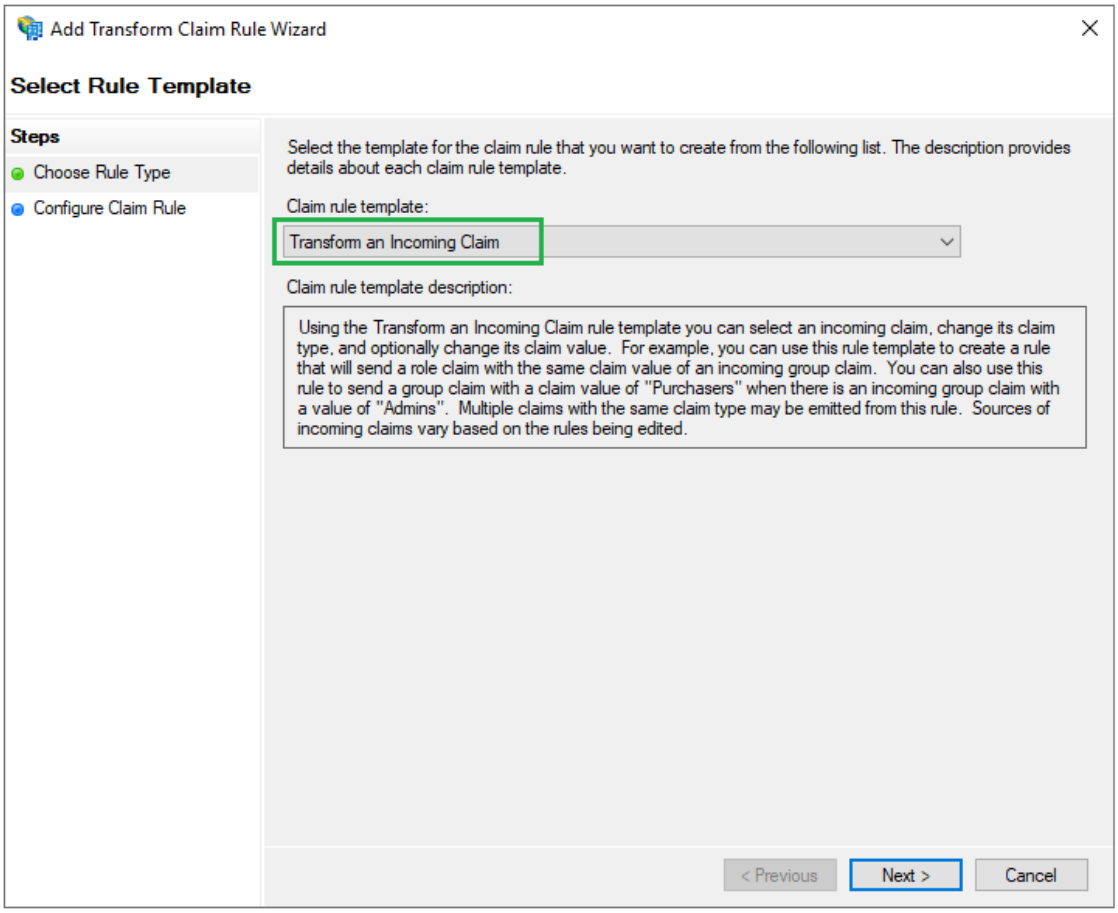
2. Click `Add Rule`.
3. Select `Send LDAP Attributes as Claims` for `Claim rule template`, and click `Next`.



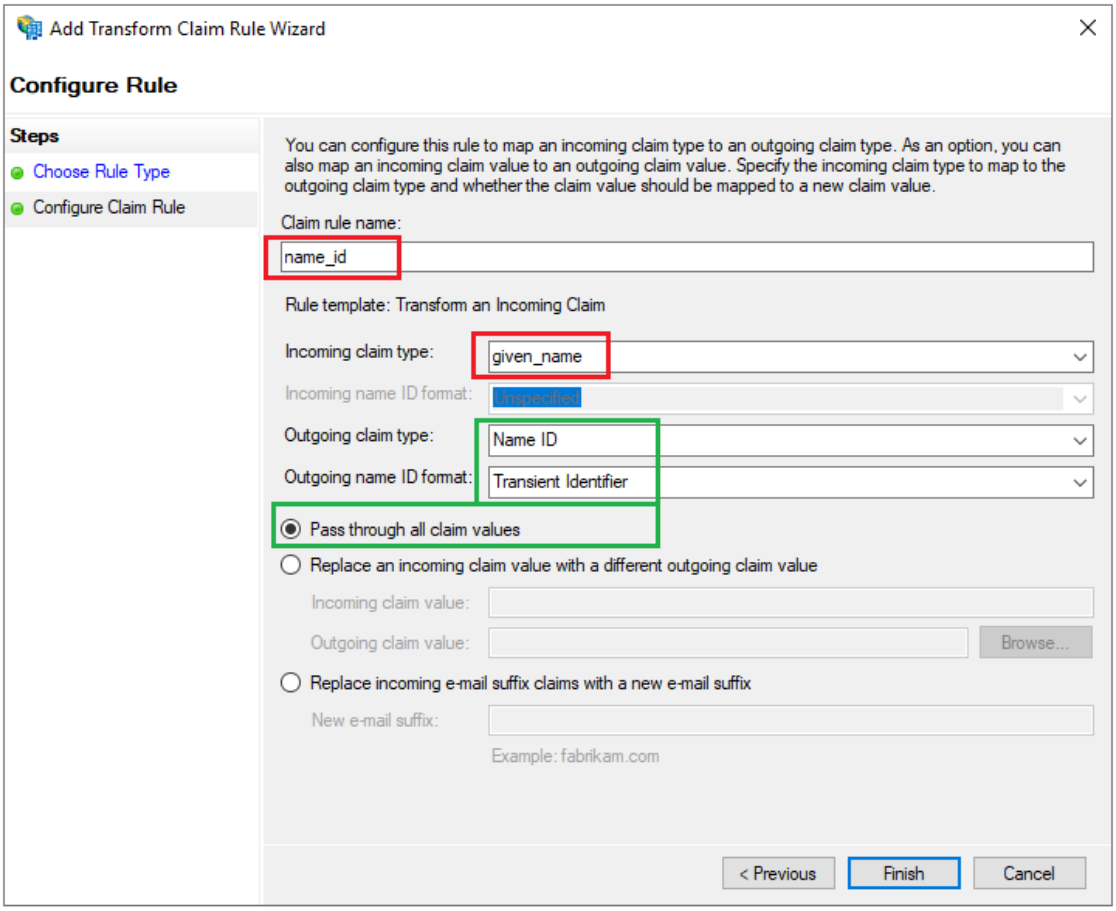
4. Enter a name for `Claim rule name`, for instance, display_name. Select `Active Directory` for `Attribute Store`. Under `LDAP Attribute` column, choose `Display-Name` from the drop-down list, type given_name for the `Outgoing Claim Type` column, and click `Finish`.



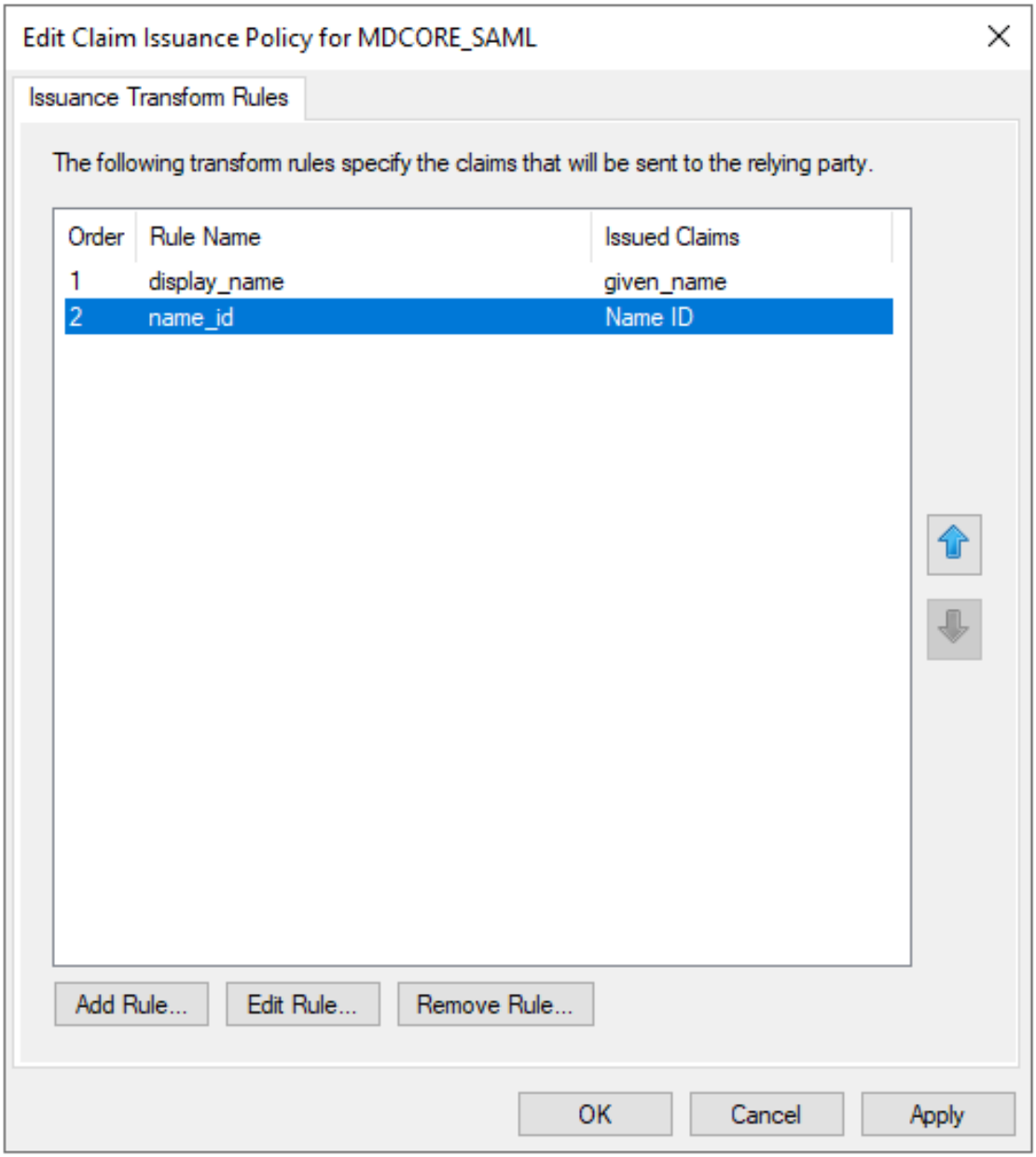
5. Click `Add Rule` once more. At `Choose Rule Type` option, select `Transform an Incoming Claim` for `Claim rule template` and click `Next`.



6. Enter name_id for the `Claim rule name` and given_name from step 4 for `Incoming claim type`. Select `Name ID` for `Outgoing claim type` and `Transient Identifier` for `Outgoing Name ID format` from the respective drop-down lists. Finally, check `Pass through all claim values` and click `Finish`.

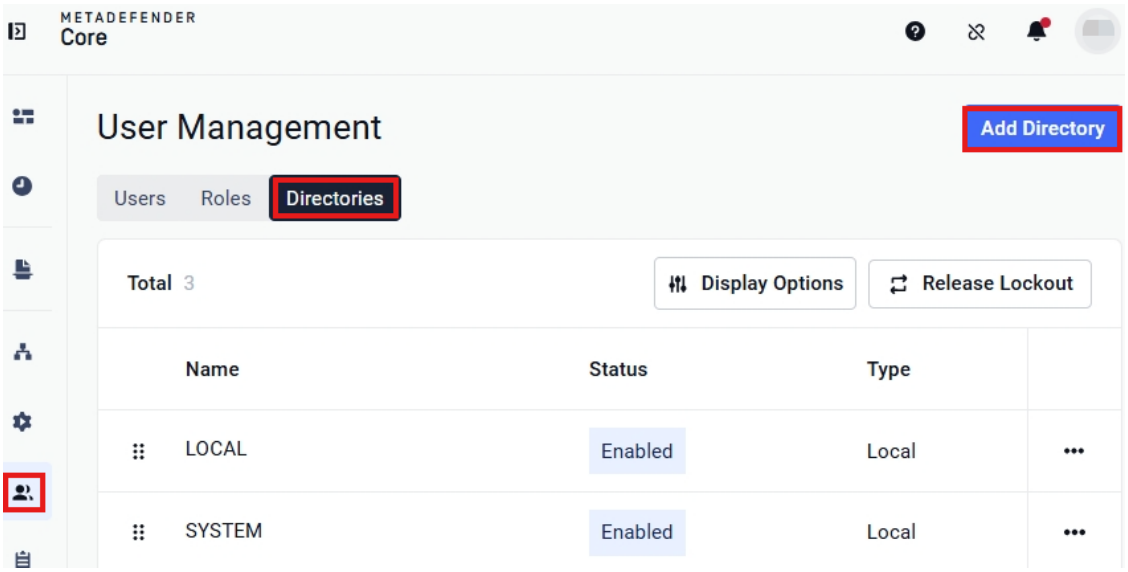


7. Click `Apply` on the next page and `OK` to close the wizard.

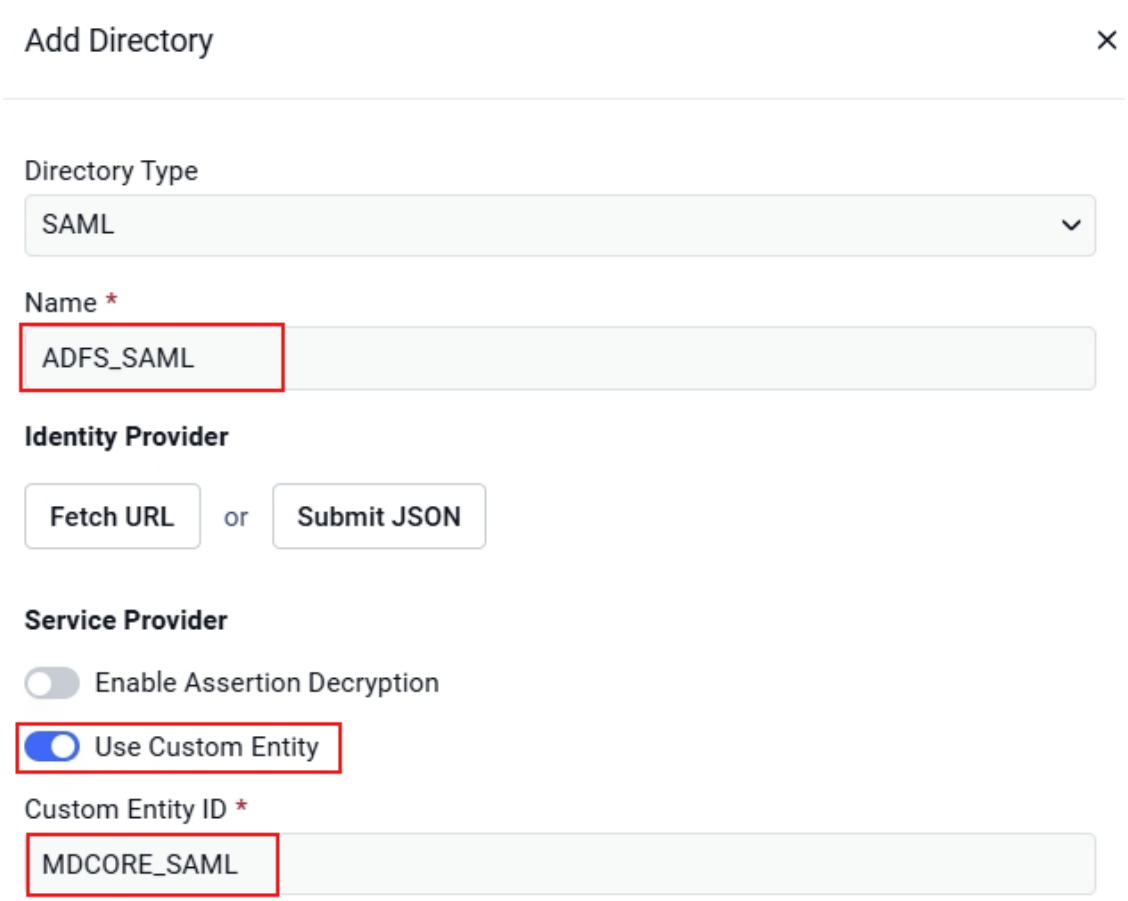


Create a SAML directory in MetaDefender Core

1. Sign in to MetaDefender Core.
2. On the dashboard, click `User Management` in the left sidebar.
3. On `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner .

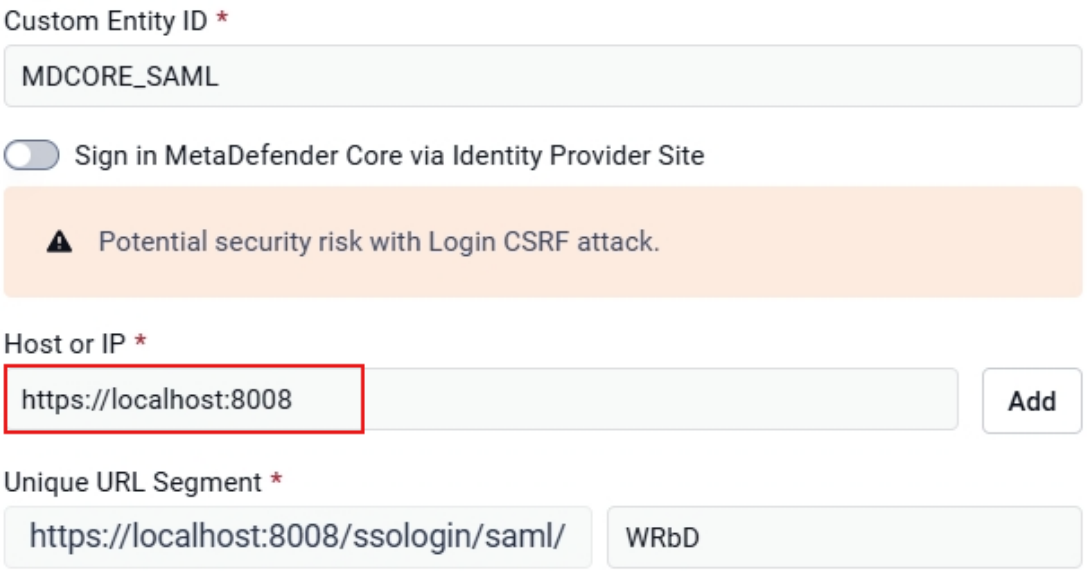


4. On `Add Directory` page, choose SAML as `Directory type`.
5. Enter the name of the new directory, such as ADFS_SAML.
6. In `Service Provider` section, check `Use custom entity ID` and type MDCORE_SAML in `Custom entity ID` text box.



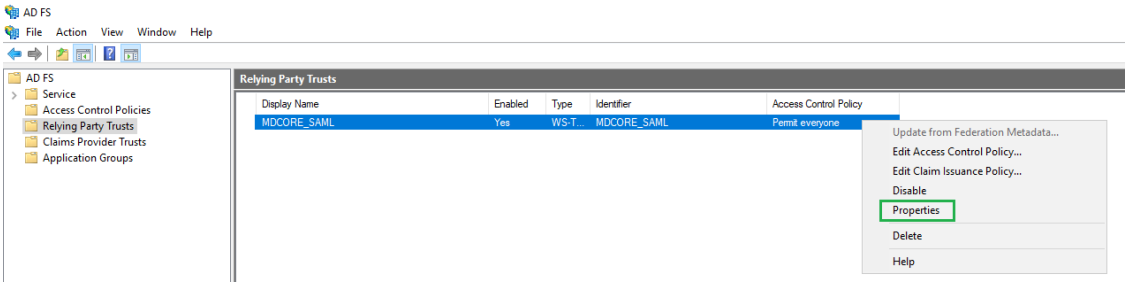
7. In `Service Provider` section, fill in `Host or IP` filed with the address where MetaDefender Core is hosted, using https://localhost:8008 as an example.

8. Copy `Login URL`.

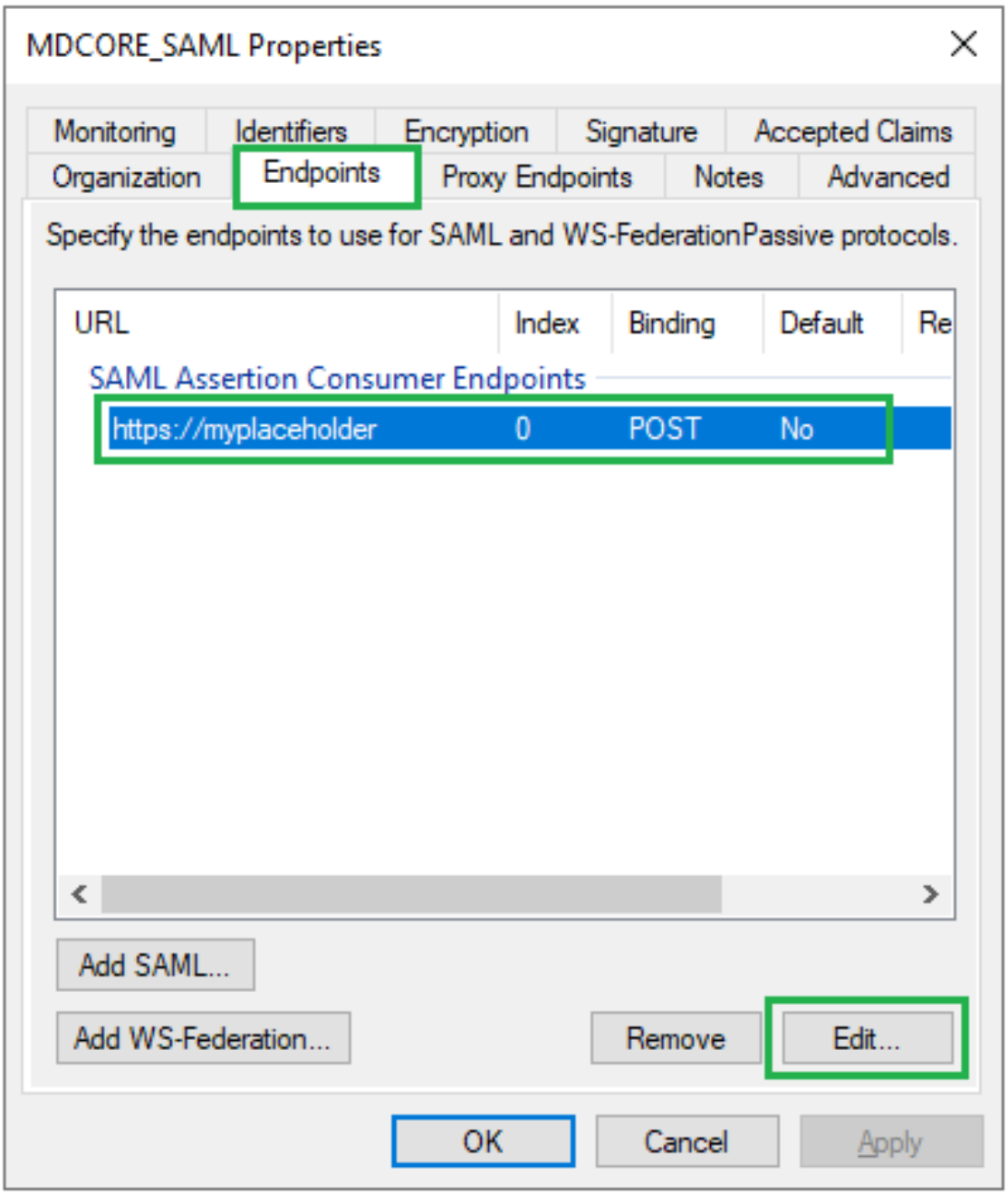


Modify Replying Party Trust on ADFS

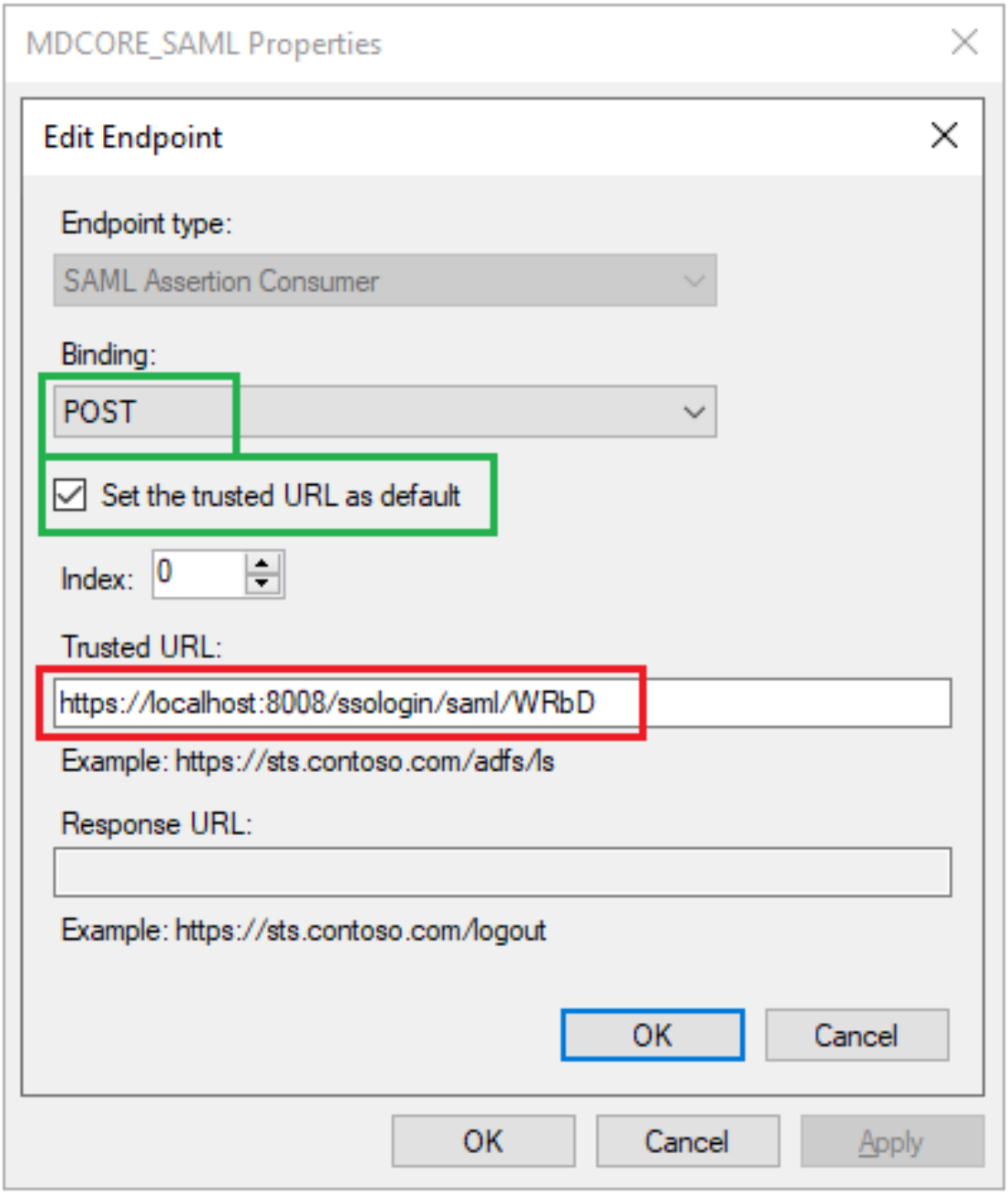
1. Access `AD FS Management` tool from `Server Manager`, select `Relying Party Trusts`, right click on the item MDCORE_SAML, and choose `Properties`.



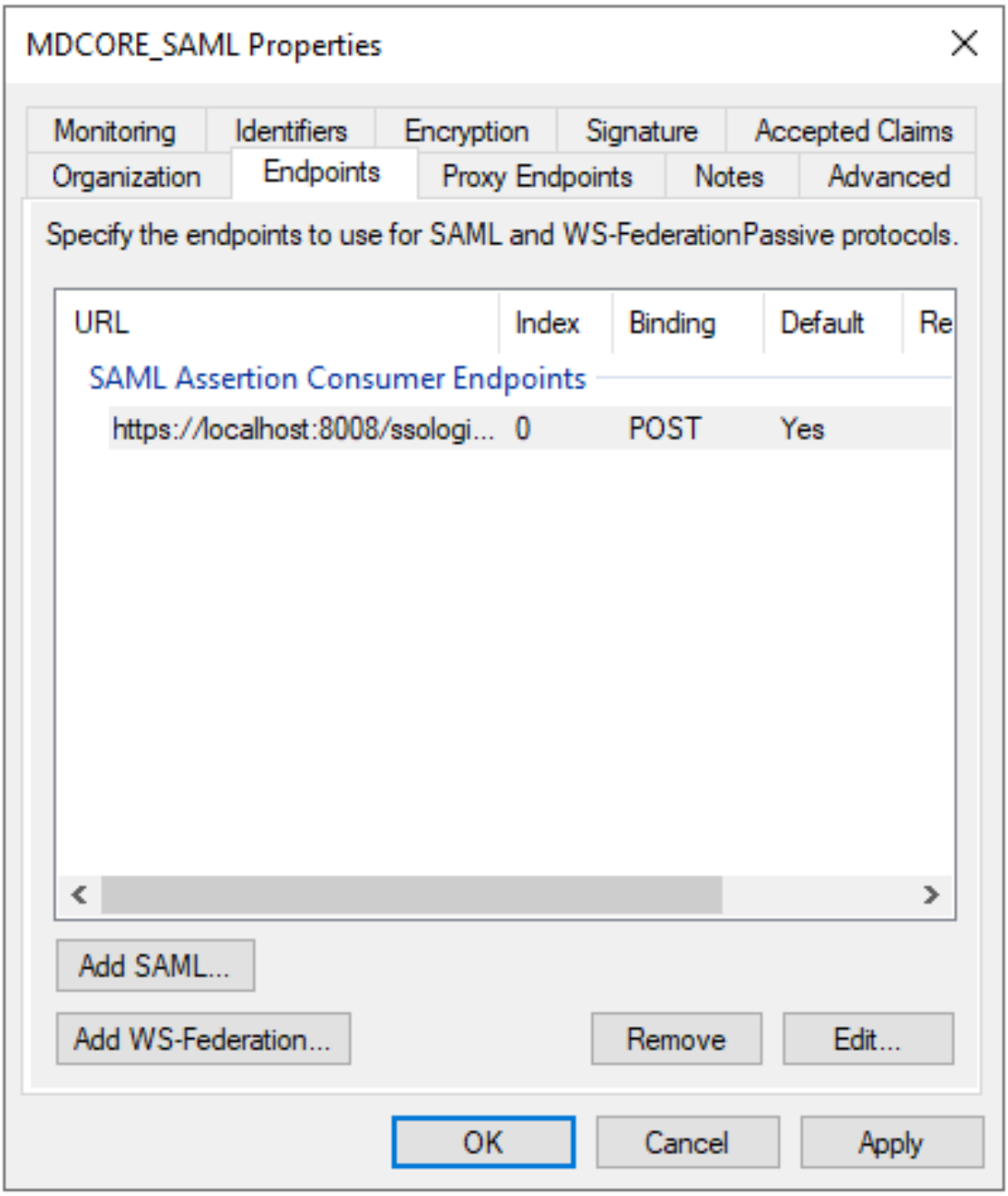
2. Navigate to `Endpoints` tab, select https://myplaceholder, and click `Edit`.



3. Select `POST` from the drop-down list for `Binding`. Check `Set the trusted URL as default`, then paste the `Login URL` from MetaDefender Core into `Trusted URL` and click `OK`.

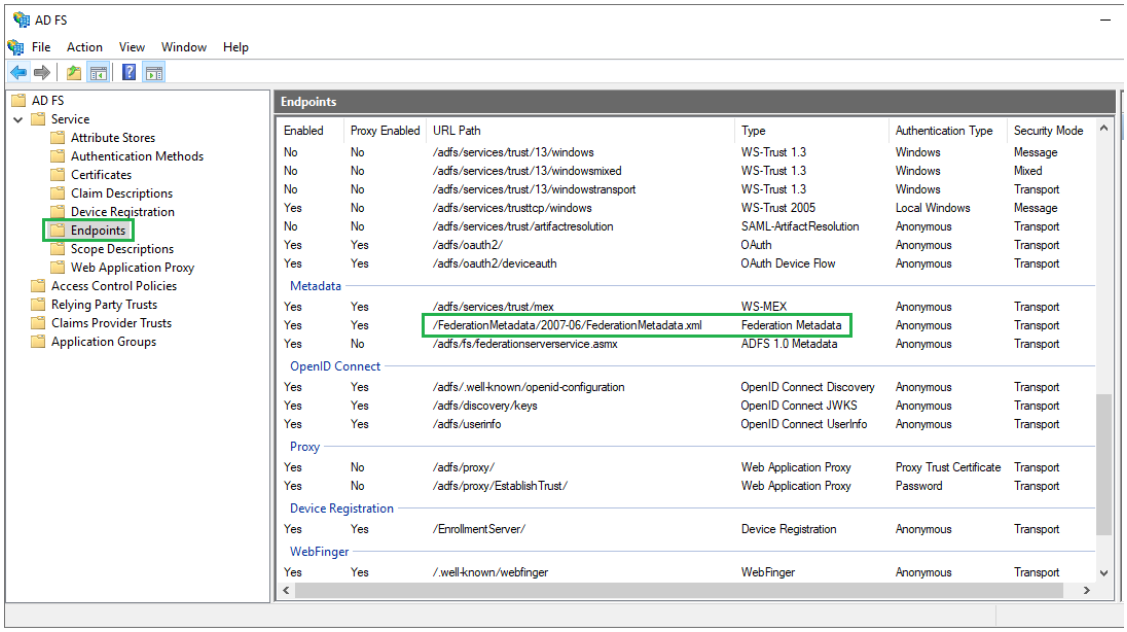


4. Click `Apply` and `OK` to complete the Relying Party Trust modification.



Complete configuration in MetaDefender Core

1. Access `AD FS Management` tool from `Server Manager`, expand `AD FS` --> `Service` --> `Endpoints`. In `Endpoints` panel, navigate to `Metadata` section and copy the URL path of type `Federation Metadata`.



2. Build the complete URL to the metadata endpoint by appending the path from step 1 to the qualified domain name of ADFS.

Code · html

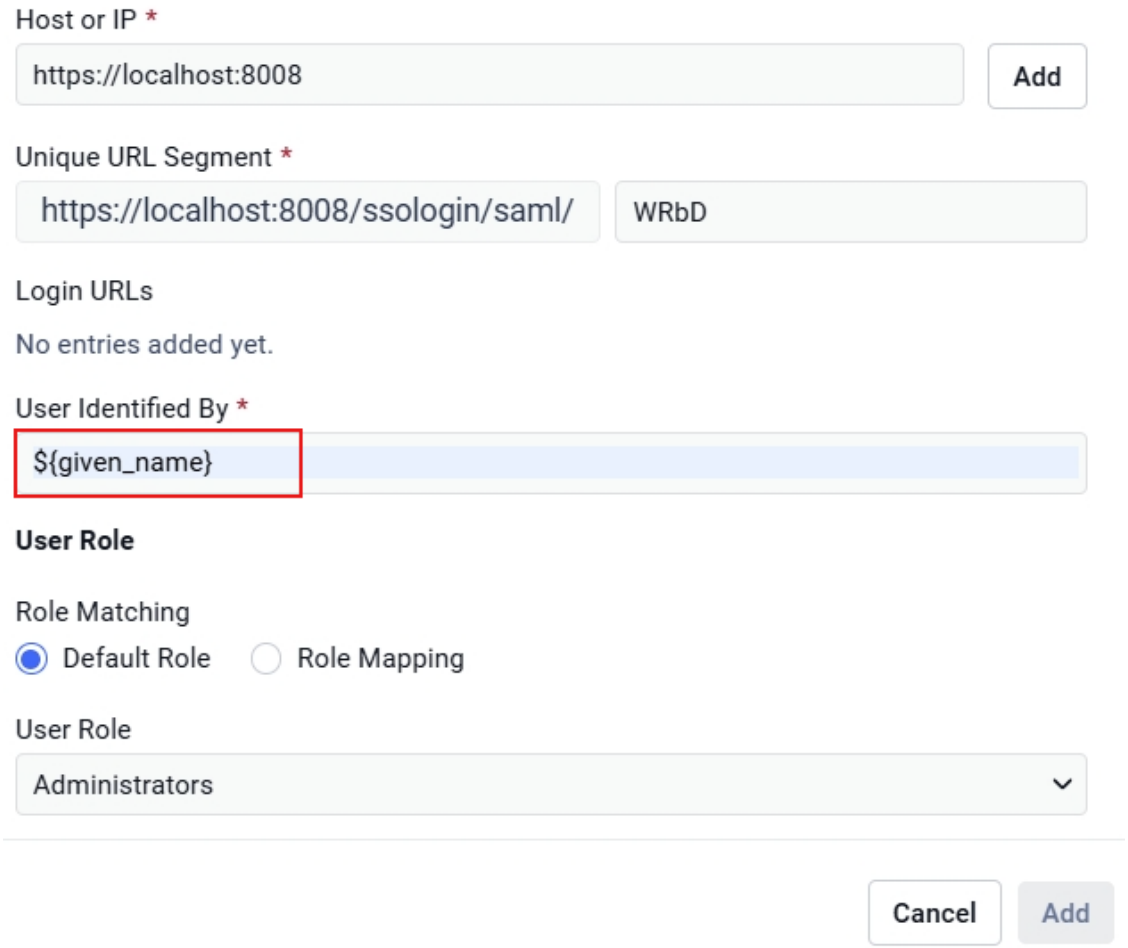
```
https://<adfs_fully_qualified_domain_name>/<metadata_endpoint_path>
```

- For example:

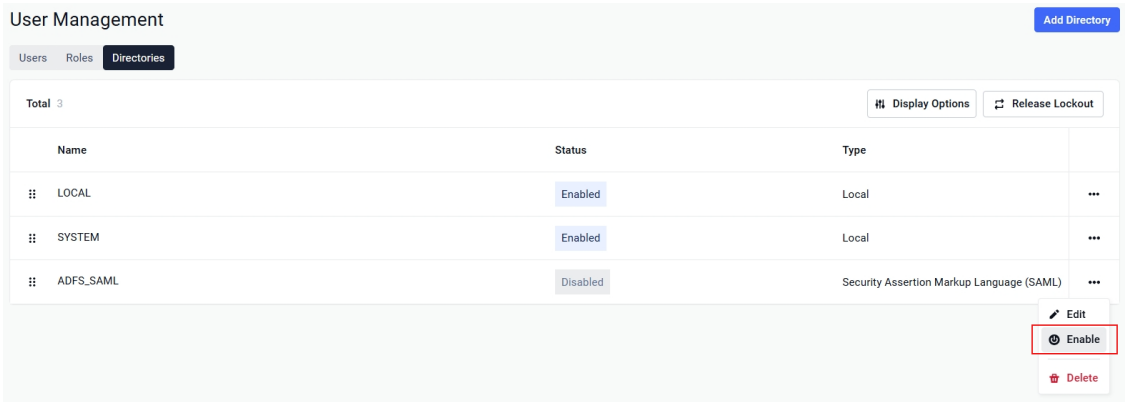
Code · html

```
https://win-tst/FederationMetadata/2007-06/FederationMetadata.xml
```

3. Switch back to MetaDefender Core. Under the section `Identity Provider`, click `Fetch URL` and paste the link above into the box under `Fetch URL`, then click `OK` to ensure MetaDefender Core can set ADFS as its IDP.
4. Under the section `Service Provider`, fill in the user identity under `User identified by` with the configured values of `Outgoing Claim Type`, e.g. ${given_name}.
5. Select the appropriate role for the user under `User Role`.
6. Click `Add` to complete the settings.



7. On `User Management` screen, toggle the new directory `ADFS_SAML` in this example. A dialog box appears to confirm the action. Once `Enable` is clicked, all sessions are immediately expired, and ADFS is used to authenticate the user.

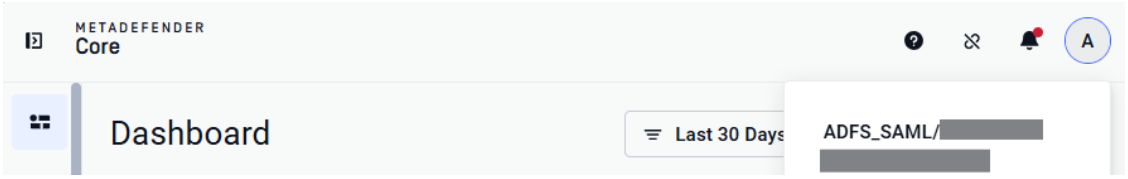


Test the integration

1. Click `Login` from the home screen of MetaDefender Core; the user is redirected to ADFS page.



2. Sign in with the account created in the system domain.
3. If everything goes right, the MetaDefender Core dashboard is displayed with the user identity set in the top right corner.



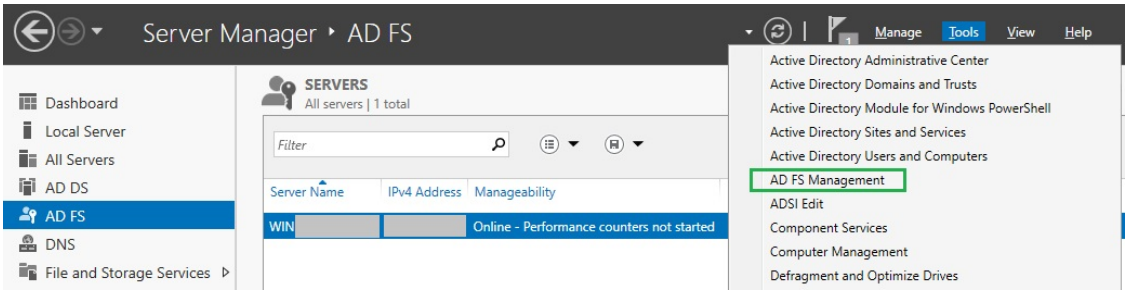
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Configure OIDC SSO

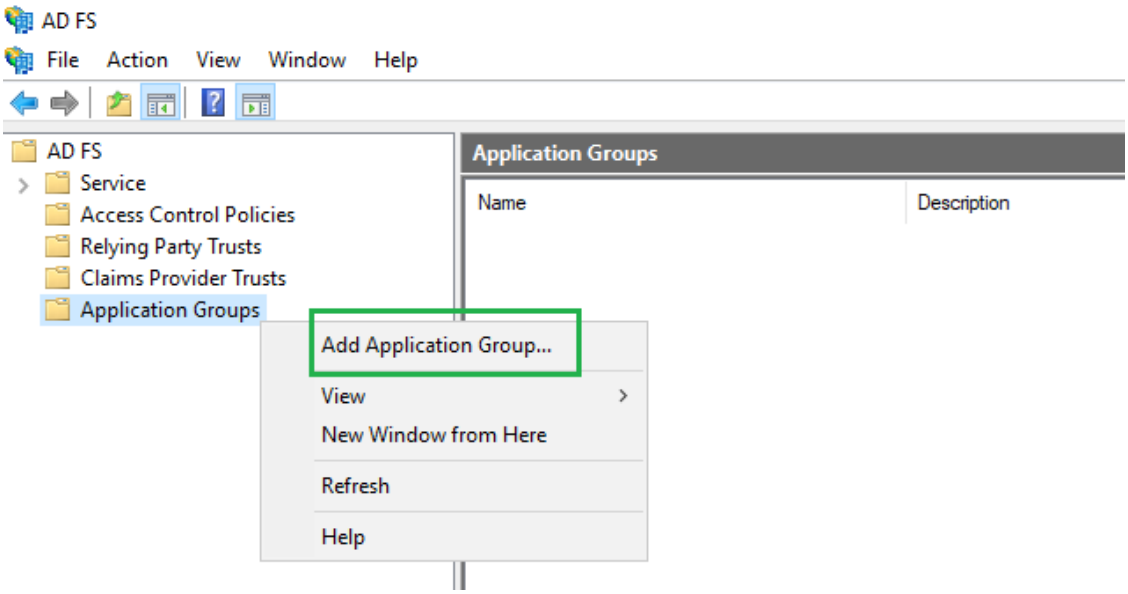
> **Info**
>
> ADFS requires MetaDefender Core to enable HTTPS for connections. Please follow these steps to first enable HTTPS on MetaDefender Core.

Set up OIDC in ADFS

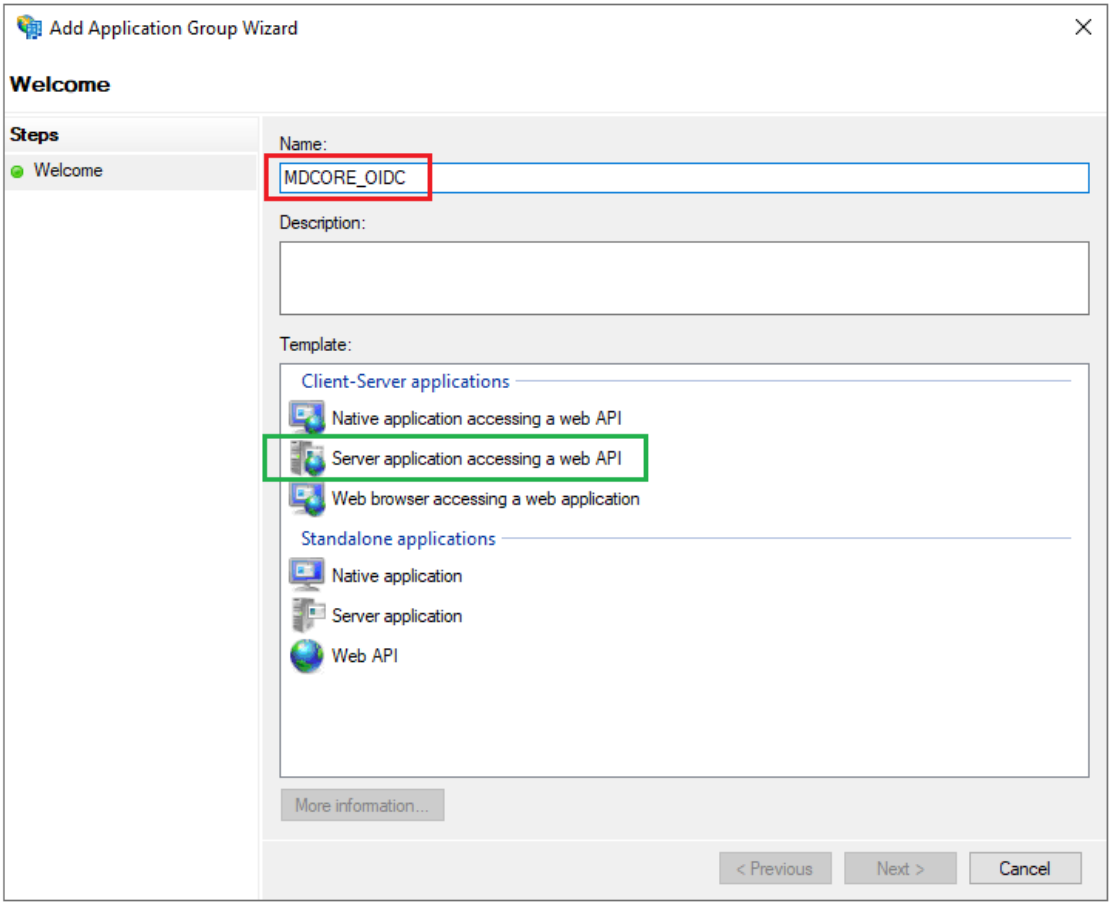
1. In the top right corner of `Server Manager`, click on `Tools` and select `AD FS Management` from the drop-down menu.



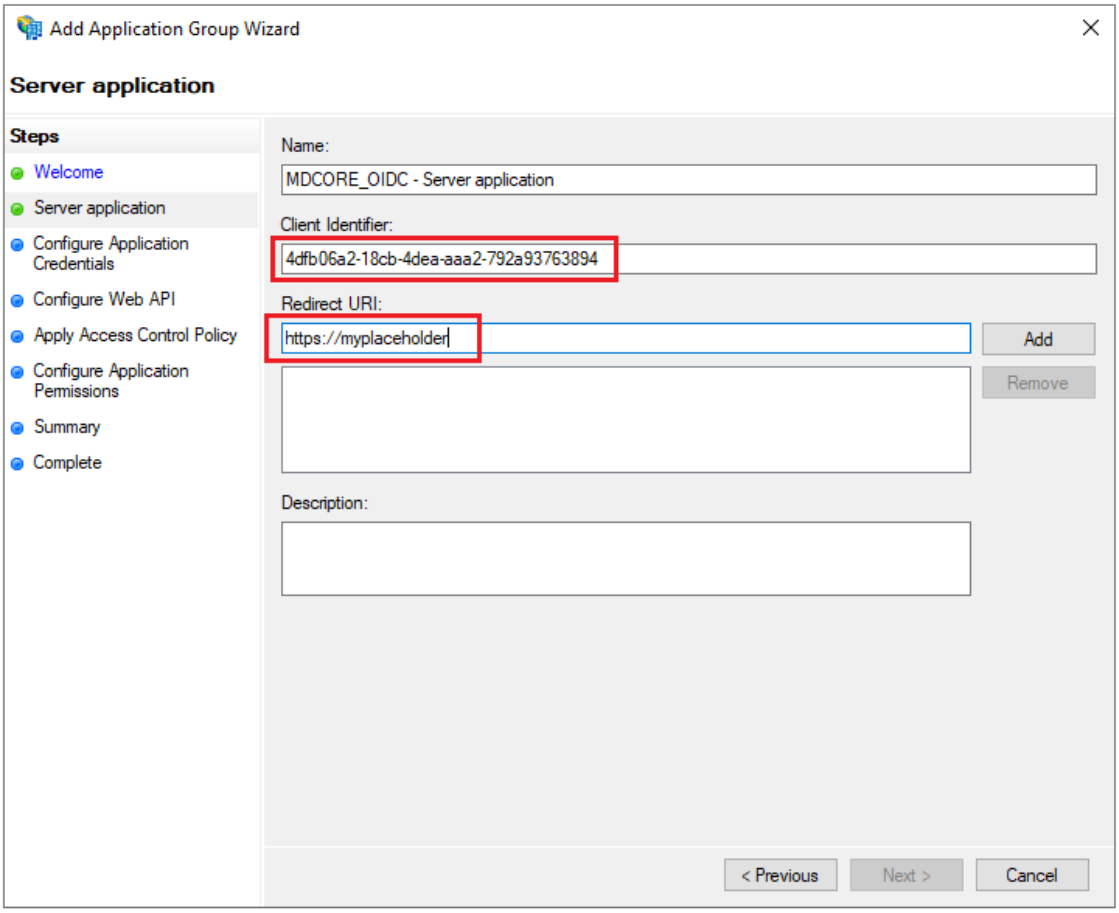
2. In `AD FS` window, right click on `Application Groups` and select `Add Application Group` from the drop-down menu.



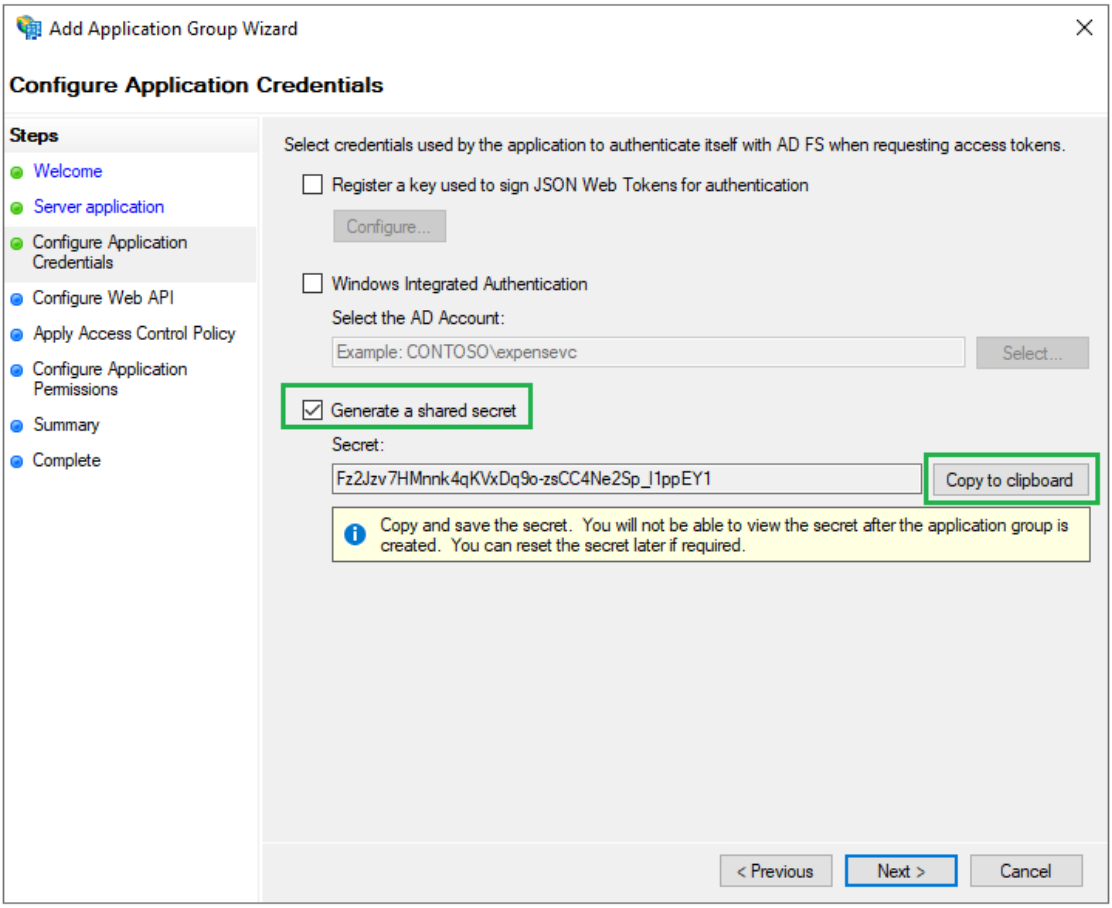
3. On `Welcome` page, provide a descriptive name for your application , e.g. MDCORE_OIDC, select `Server application accessing a web API` under `Template`, and click `Next`.



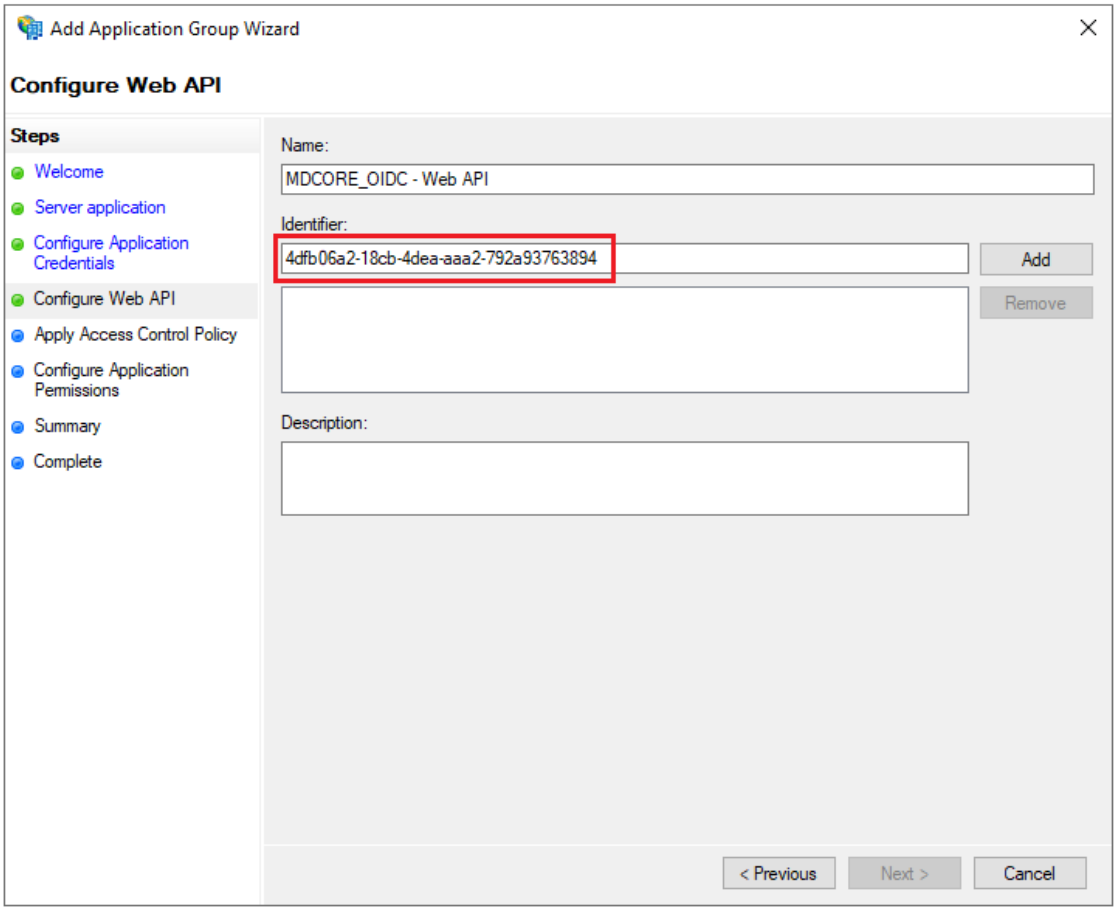
4. On `Server application` page, copy the string under `Client Identifier` for later use, set a placeholder https://myplaceholder for `Redirect URI`, then click `Add` and `Next`.



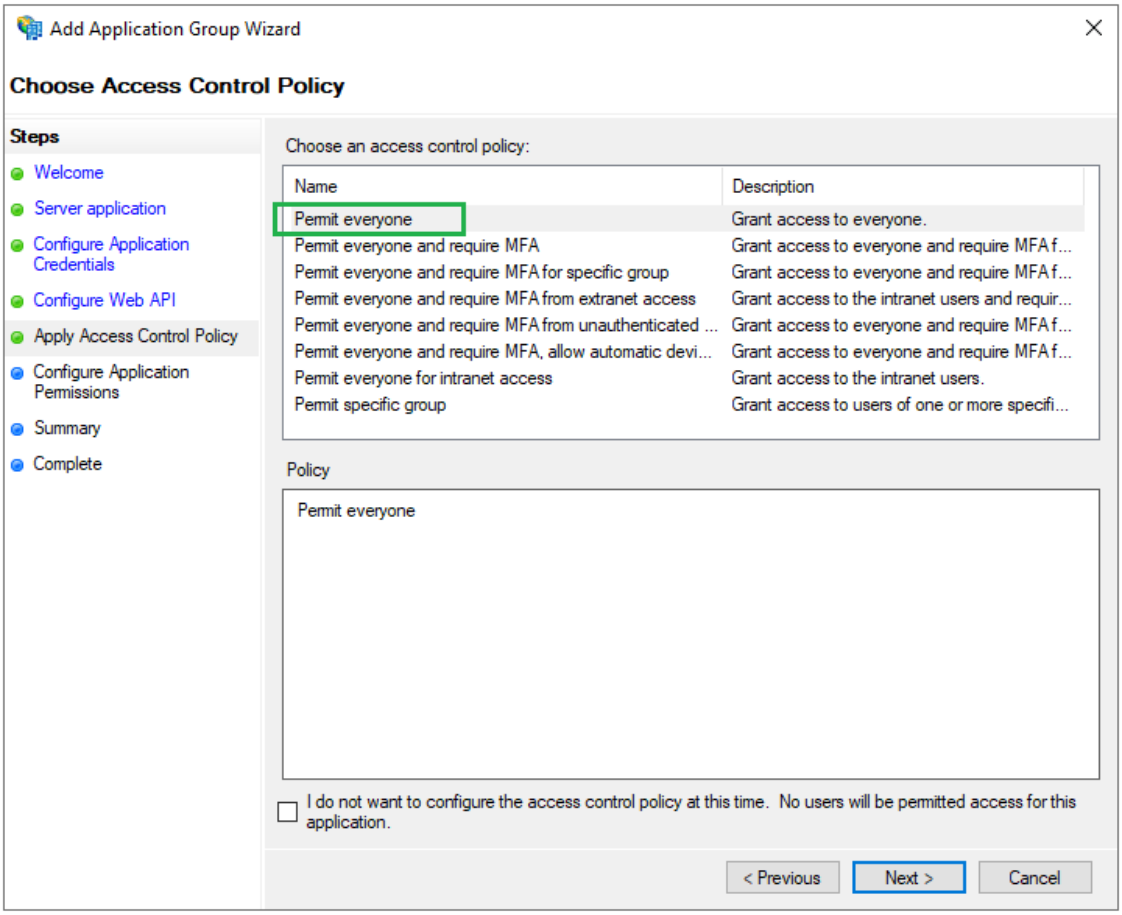
5. On `Configure Application Credentials` page, check `Generate a shared secret`, click `Copy to clipboard`, then save the secret string for later use and click `Next`.



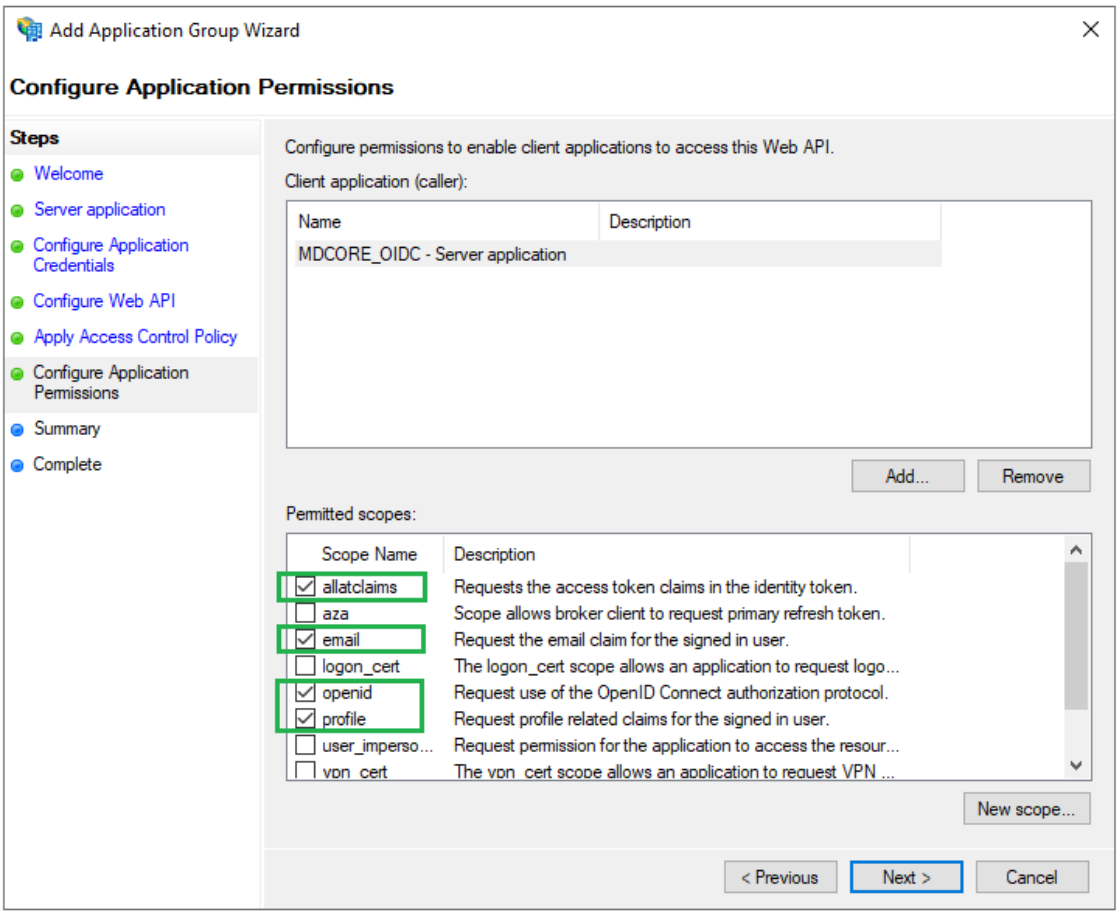
6. On `Configure Web API` page, enter the string of `Client Identifier` from step 4 into the box under `Identifier`, click `Add` , and `Next`.



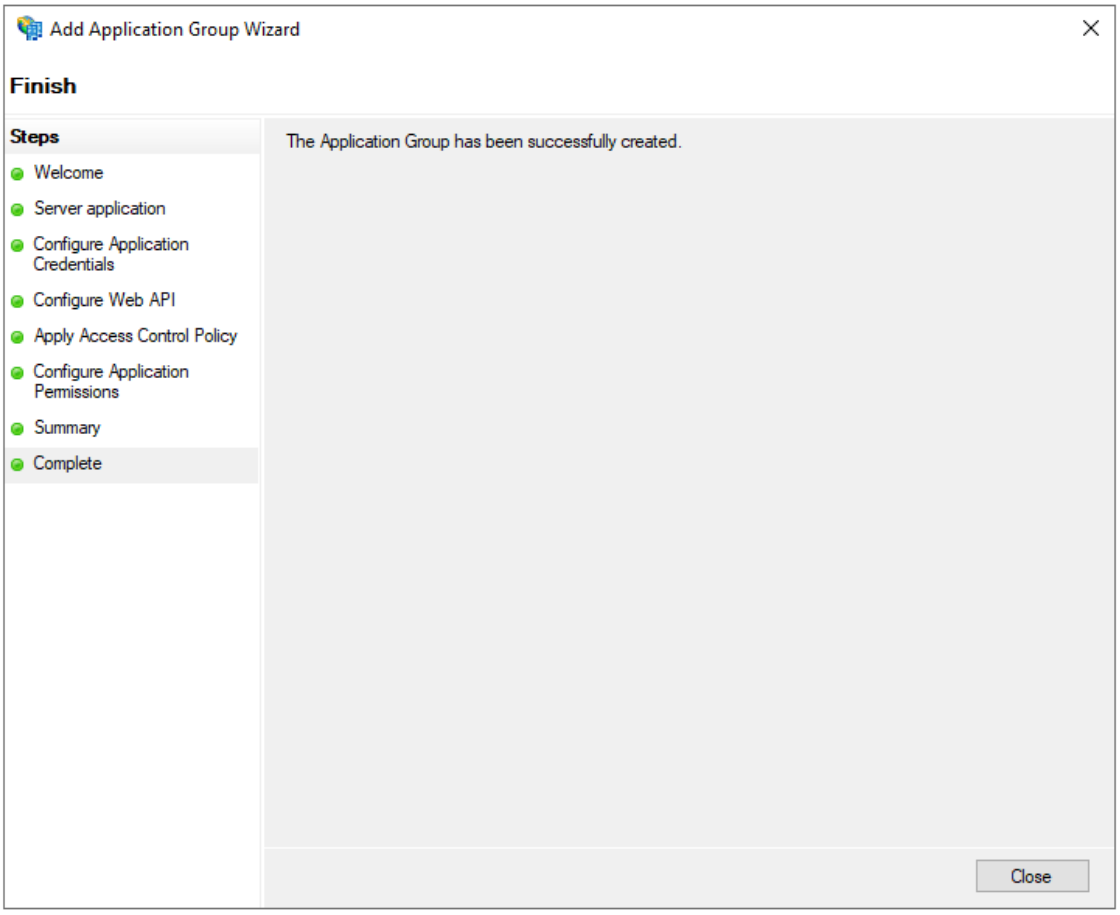
7. On `Choose Access Control Policy` page, select `Permit everyone` and click `Next`.



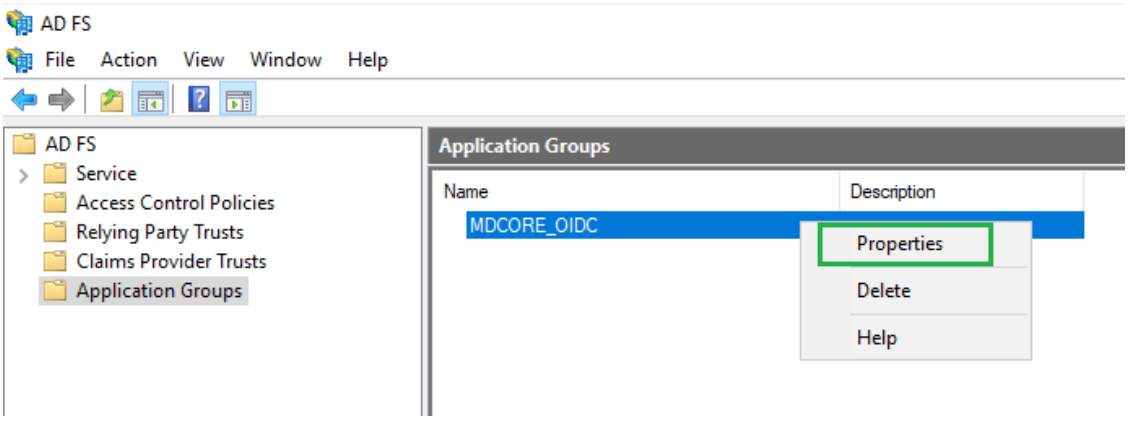
8. On `Configure Application Permissions` page, select the scopes `allatclaims`, `email`, `openid`, and `profile` under `Permitted scopes`, and then click `Next`.



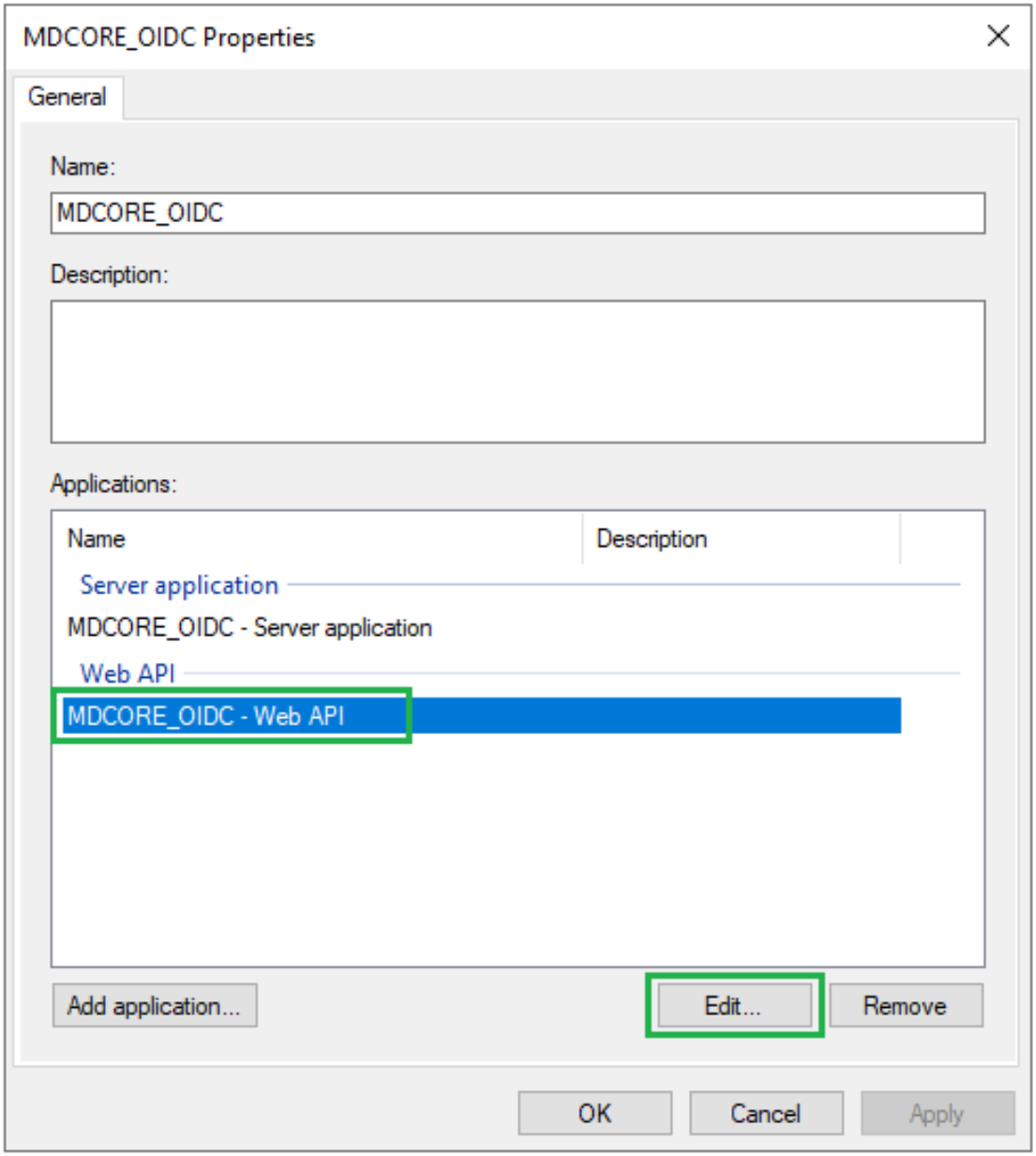
9. Click `Next` on the following page and `Close` to finish.



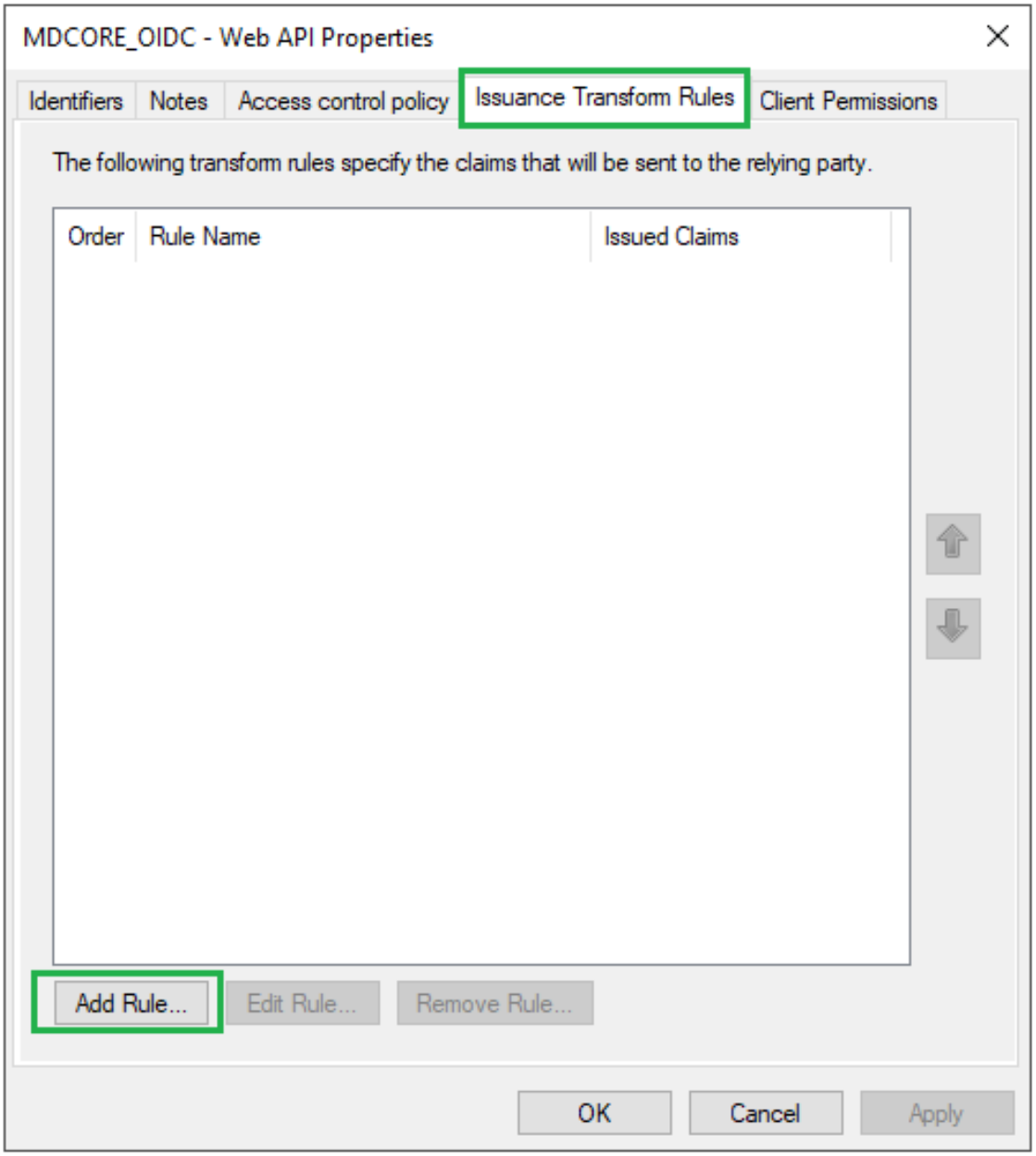
10. Back in `AD FS` window, right click on the item MDCORE_OIDC and select `Properties`.



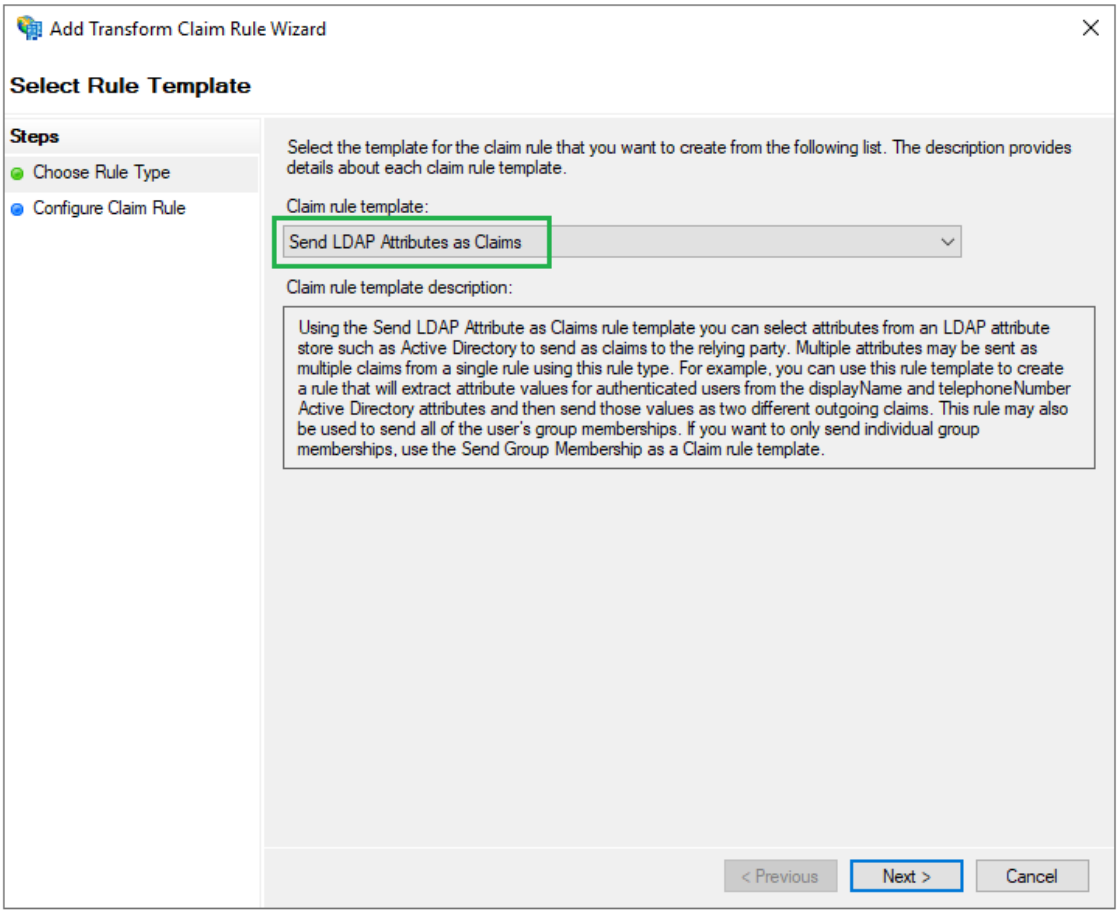
11. Select `MDCORE_OIDC - Web API` and click `Edit`.



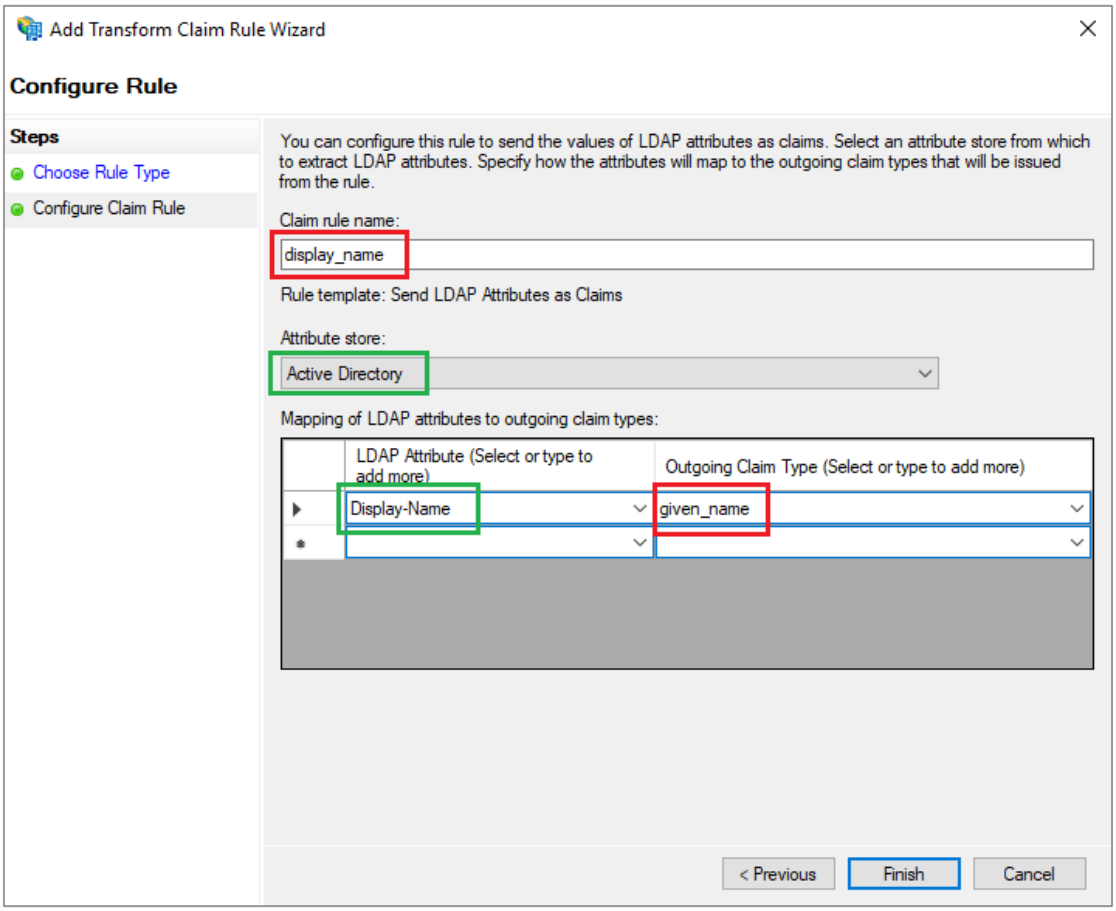
12. Go to `Issuance Transform Rules` tab and click `Add Rule`.



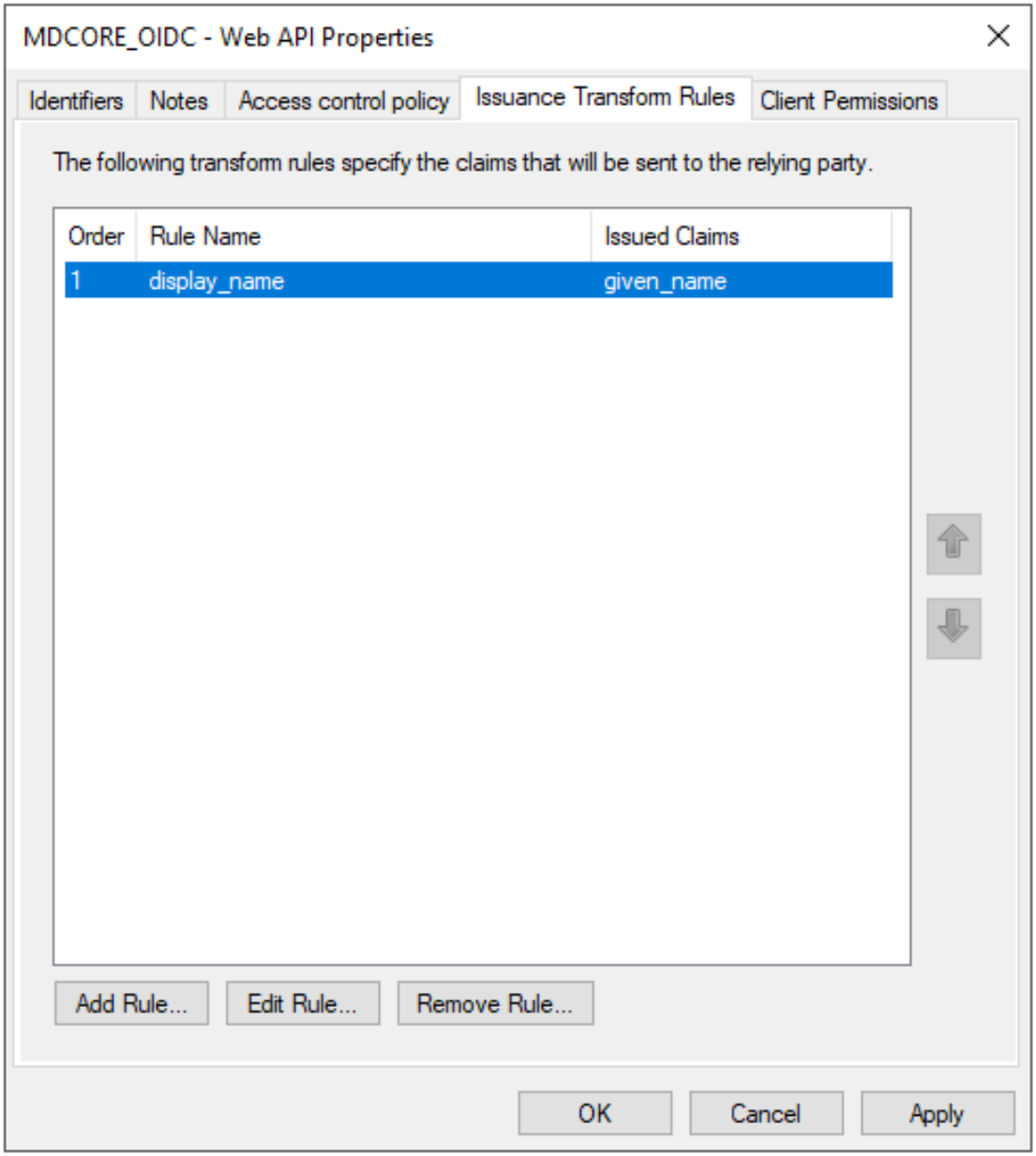
13. Select `Send LDAP Attributes as Claims` for `Claim rule template` and click `Next`.



14. Enter a name for `Claim rule`, e.g. display_name, select `Active Directory` for `Attribute Store`. Under `LDAP Attribute` column, select `Display-Name` from the drop-down list, enter given_name for the `Outgoing Claim Type` column, and click `Finish`.

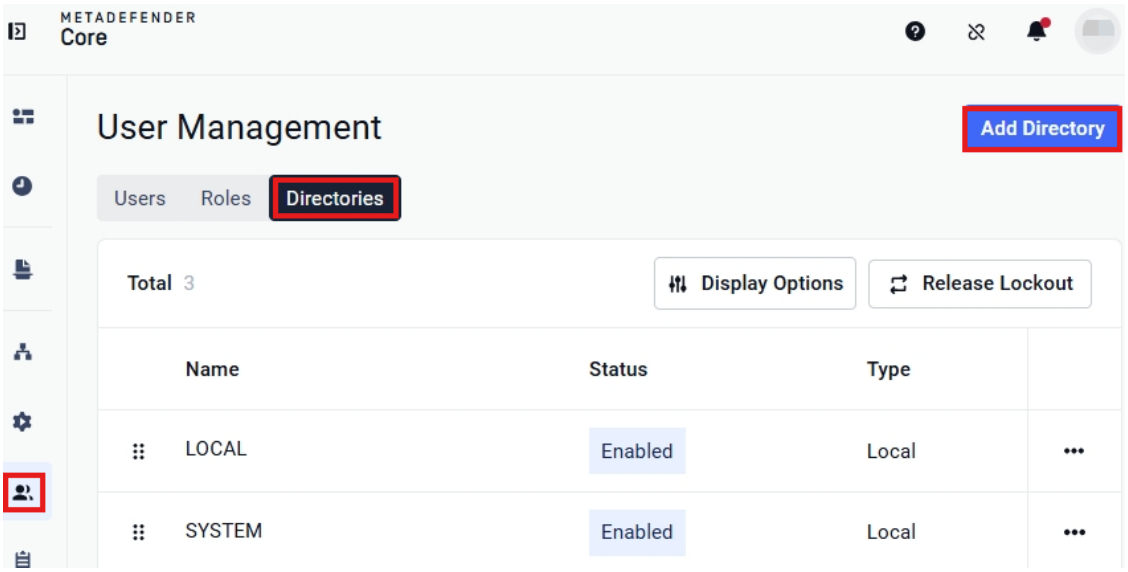


15. Click `Apply` on the next page and `OK` to close the wizard.

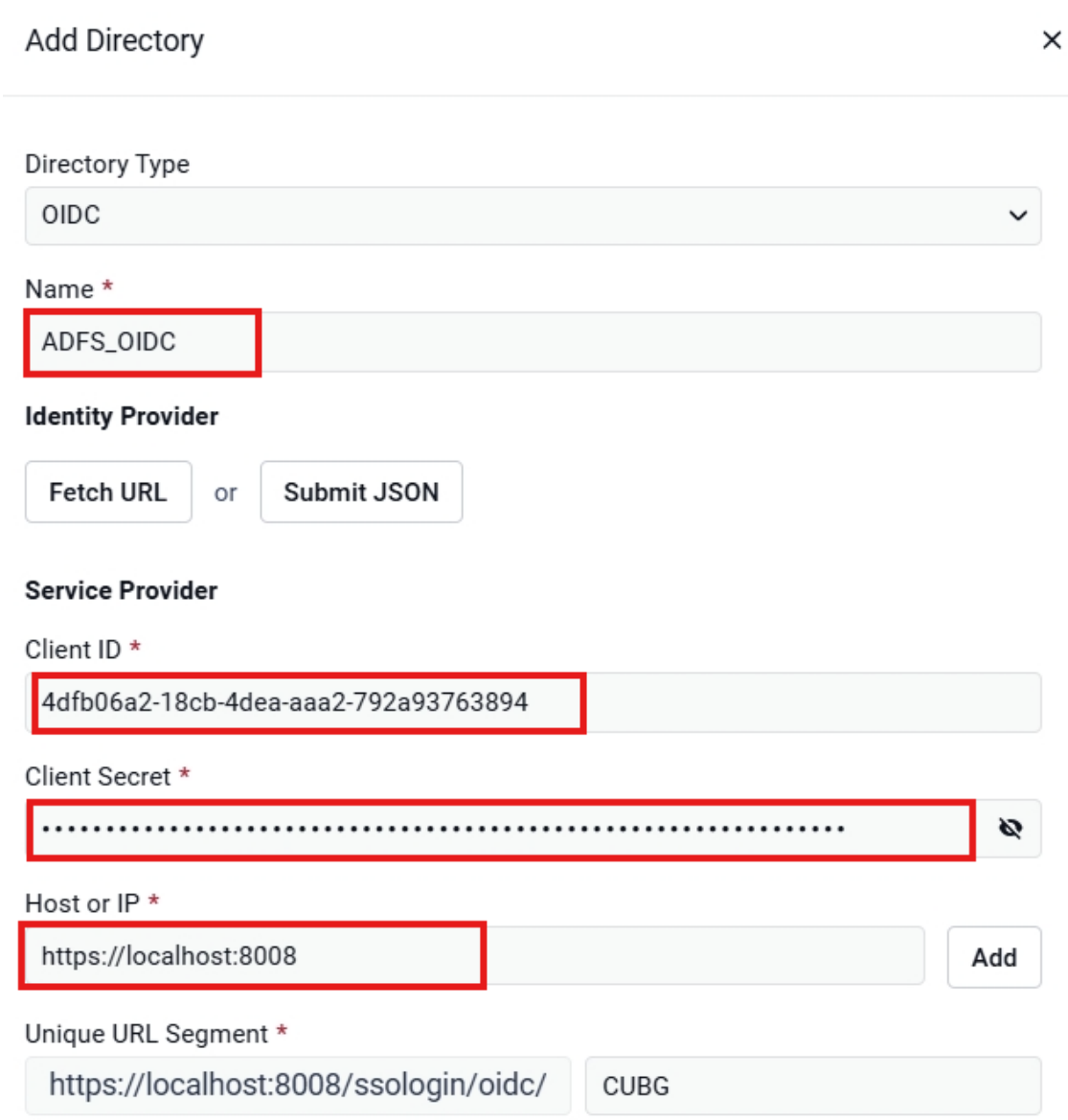


Create OIDC directory in MetaDefender Core

1. Sign in to MetaDefender Core.
2. On the dashboard, click `User Management` in the sidebar.
3. In `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.

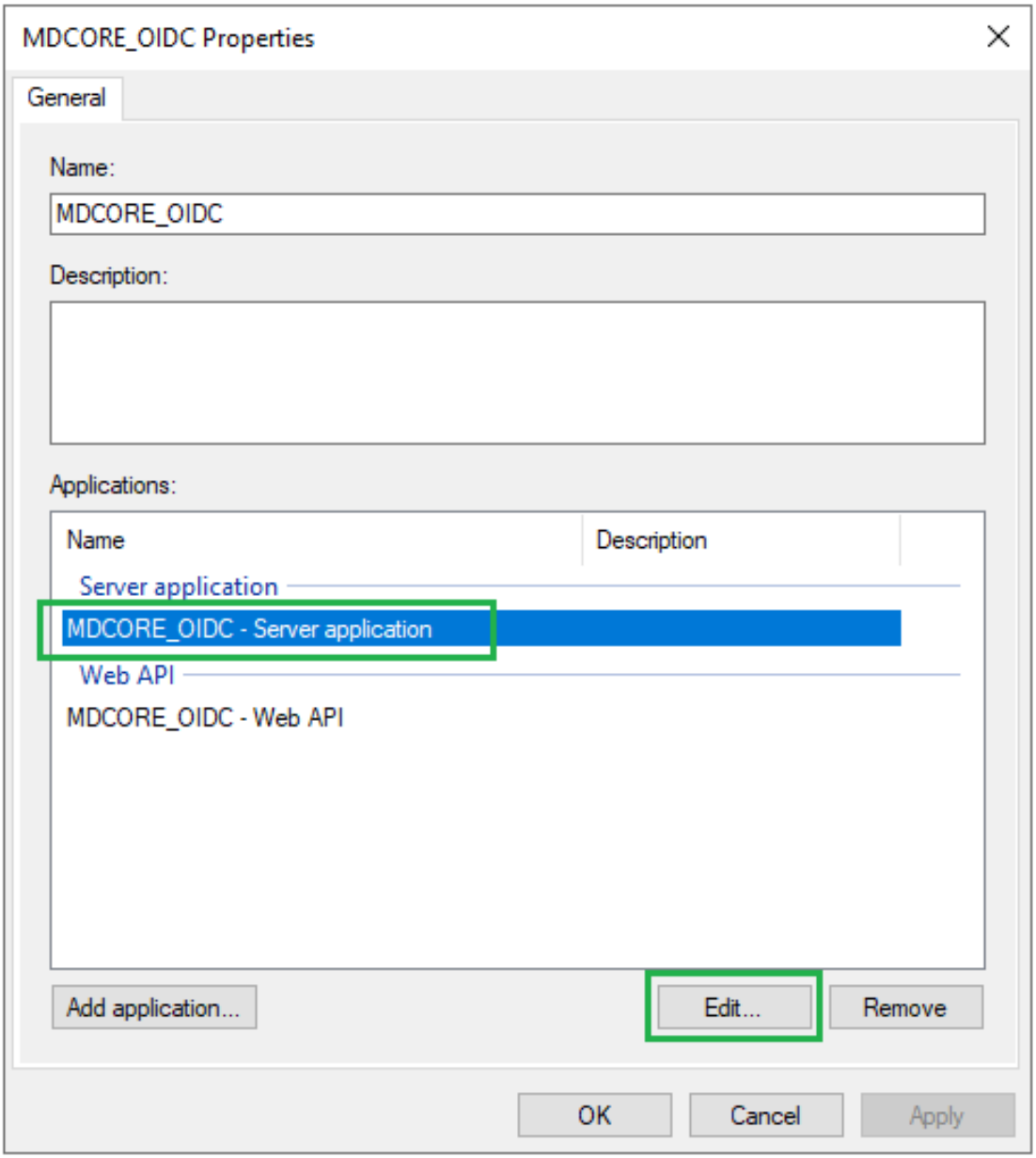


4. In `Add Directory` page, choose OIDC as `Directory type`.
5. Enter the name of the new directory, such as ADFS_OIDC.
6. In `Service Provider` section, paste the values of `Client ID` and `Client Secret` copied from ADFS, respectively.
7. In `Service Provider` section, fill in `Host or IP` with the host or IP address where MetaDefender Core is hosted; for this example, it is https://localhost:8008.
8. Copy `Login URL`.

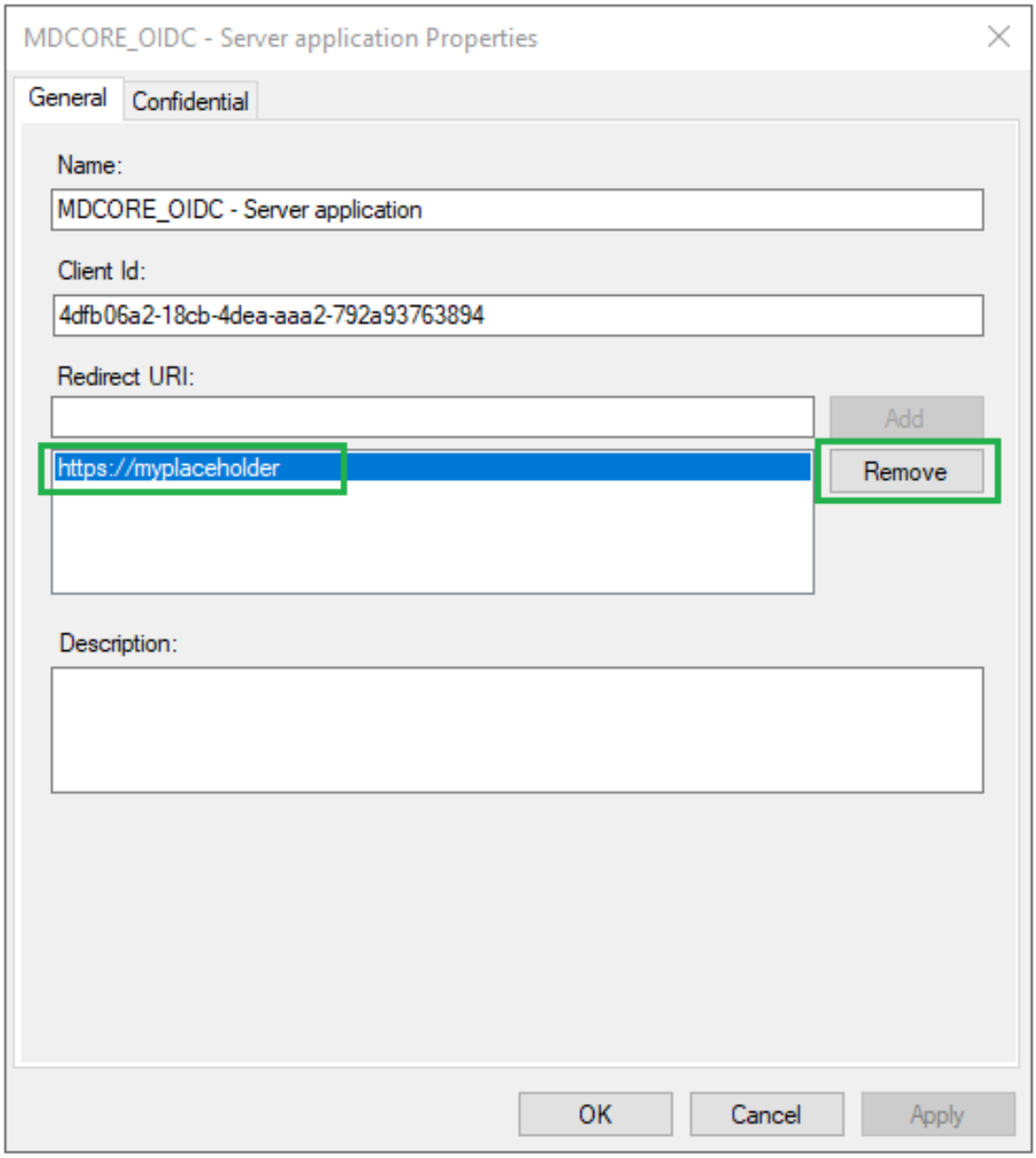


Complete configuration in ADFS

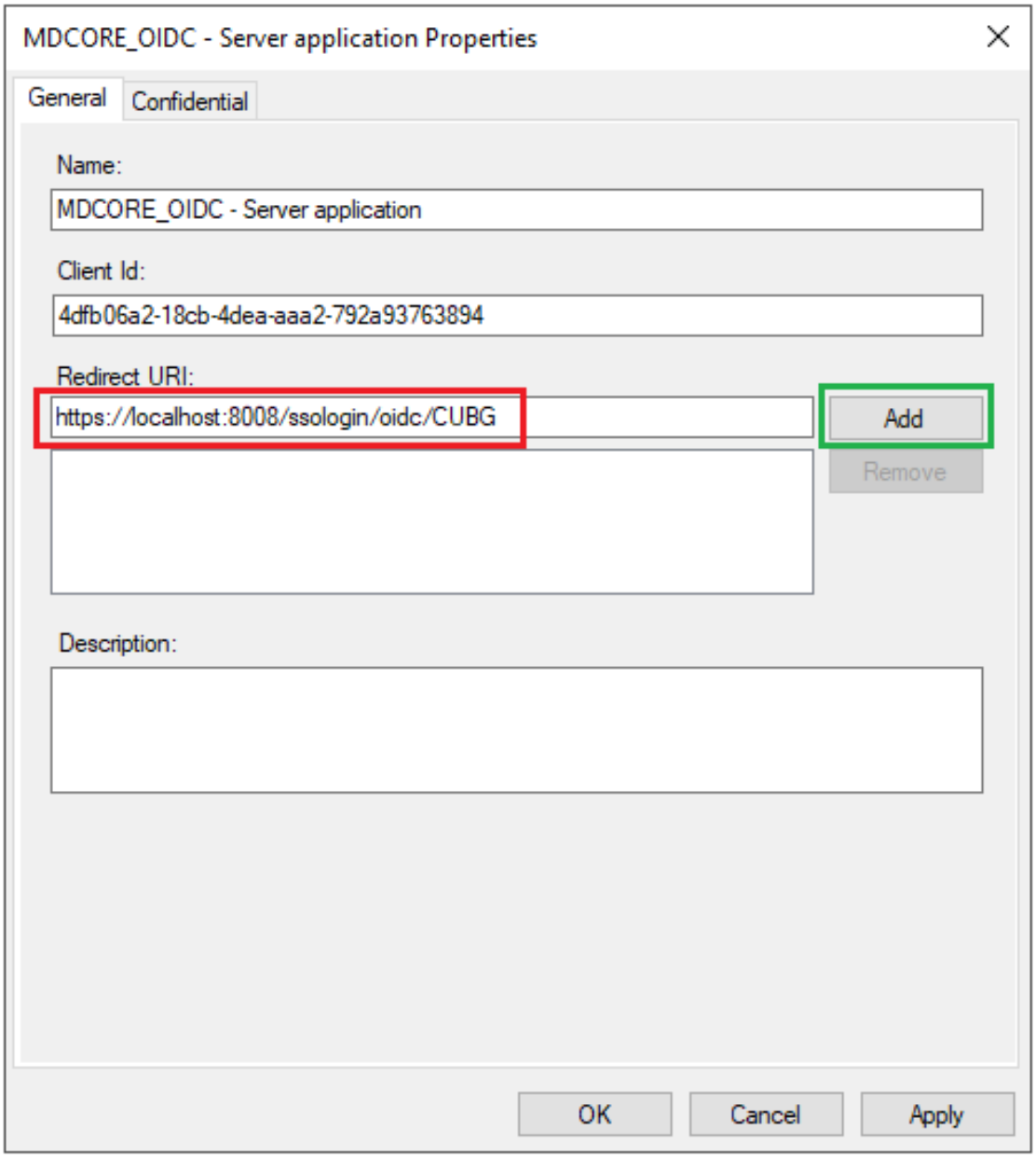
1. Back to `AD FS`, right click on item MDCORE_OIDC and select `Properties`, then choose `MDCORE OIDC - Server application` and click `Edit`.



2. Remove the placeholder https://myplaceholder by selecting it and clicking `Remove`.



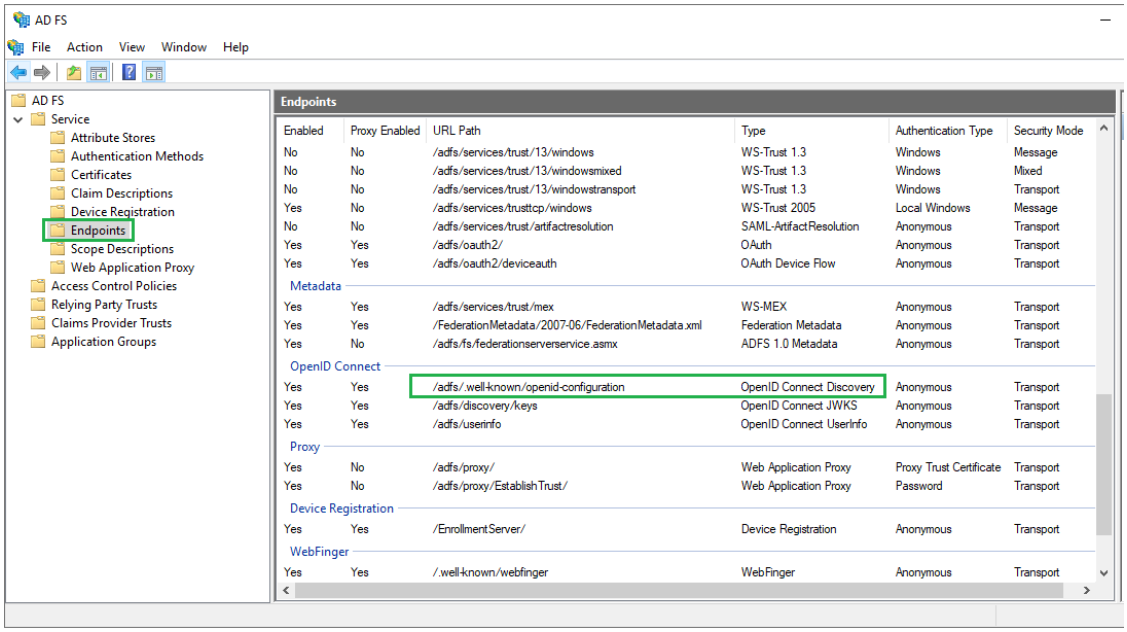
3. Paste the `Login URL` from MetaDefender Core into `Redirect URI`, click `Add`, then `Apply` and `OK`.



4. Click `OK` to complete the setup.

Complete configuration in MetaDefender Core

1. Access `AD FS Management` tool from `Server Manager`, expand `AD FS` --> `Service` --> `Endpoints` on the left panel. In `Endpoints` panel, navigate to `OpenID Connect` section and copy the URL path of type `OpenID Connect Discovery`.



2. Build the full URL to the metadata endpoint by appending the path from step 1 to the qualified domain name of ADFS

Code · html

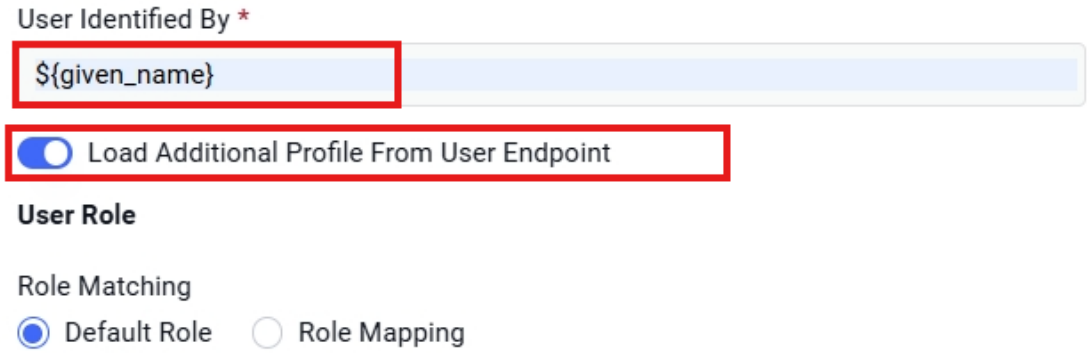
```
https://<adfs_fully_qualified_domain_name>/<metadata_endpoint_path>
```

- For example:

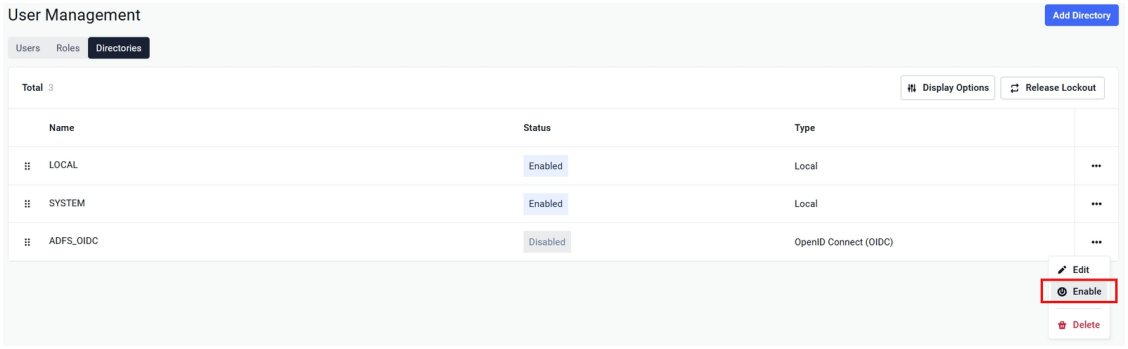
Code · html

```
https://win-tst/adfs/.well-known/openid-configuration
```

3. Switch back to MetaDefender Core. Under `Identity Provider` section, click `Fetch URL` and paste the link above into the box under `Fetch URL`. Click `OK` and wait a moment for MetaDefender Core to set ADFS as its IDP.
4. In `Service Provider` section, fill in the user identity under `User identified by` with the configured values of `Outgoing Claim Type`, e.g. ${given_name} in this instruction.
5. Select the correct role for the user under `User Role`.
6. Disable `Load additional profiles from the user endpoint` option since AD FS 2016 and newer have not yet supported the `/userinfo` endpoint.
7. Click `Add` to complete the settings.



8. On `User Management` screen, toggle ADFS_OIDC directory. A dialog box appears to confirm the action. Once `Enable` is clicked, all sessions are expired immediately, and ADFS is used to authenticate the user.

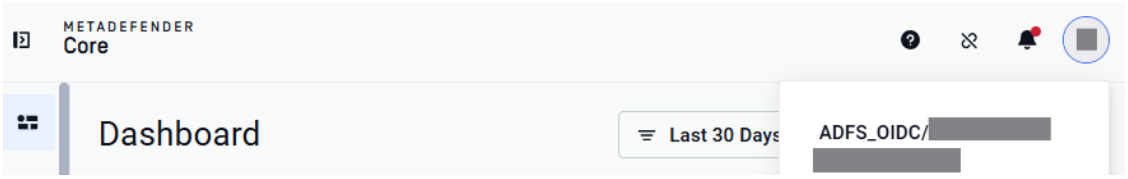


Test the integration

1. Click `Login` from the home screen of MetaDefender Core; the user is redirected to the ADFS page.



2. Sign in with the account created in the system domain.
3. If everything goes right, the MetaDefender Core dashboard is displayed with the user identity set in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Microsoft Entra ID

The instructions guide users on how to configure Microsoft Entra ID SAML and OIDC Single Sign-on for MetaDefender Core.

Configure SAML SSO

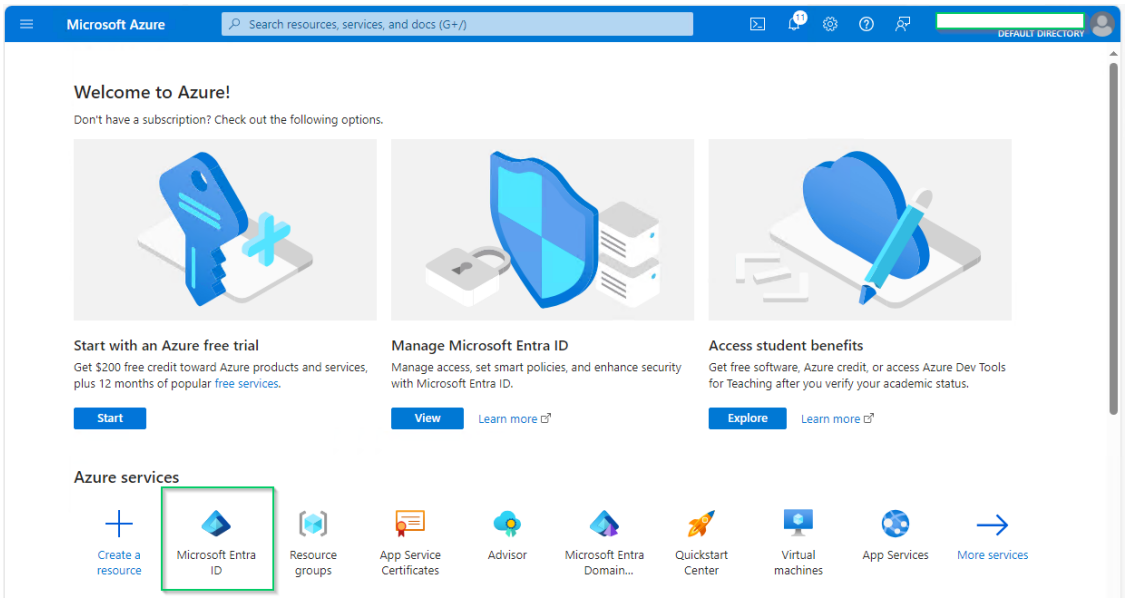
Info

Microsoft Entra ID requires MetaDefender Core to enable HTTPS for connections. Please follow these steps to first enable HTTPS on MetaDefender Core.

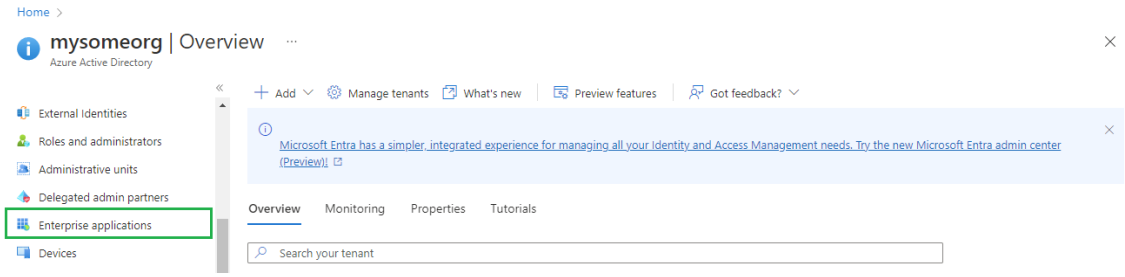
Register a new application in Azure

1. Access Azure portal and sign in.

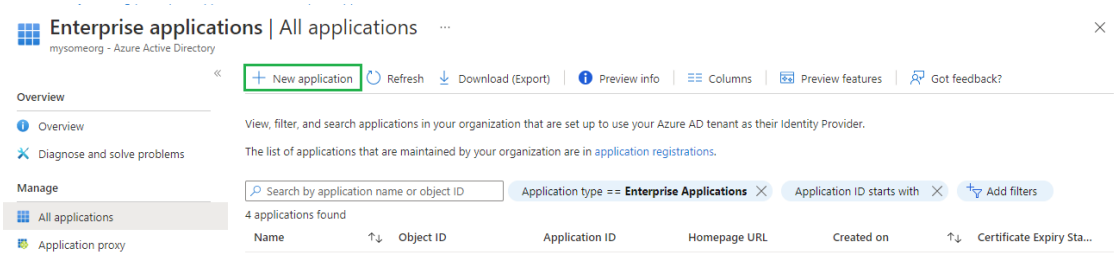
2. On the homepage, select `Microsoft Entra ID` under `Azure services`.



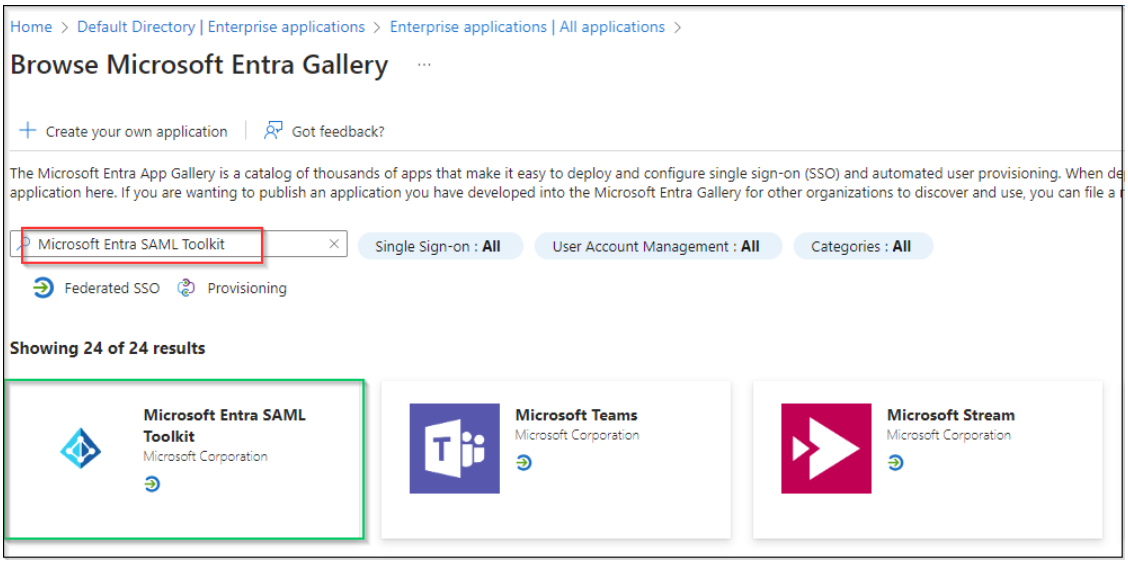
3. Select `Enterprise applications` in the left sidebar.



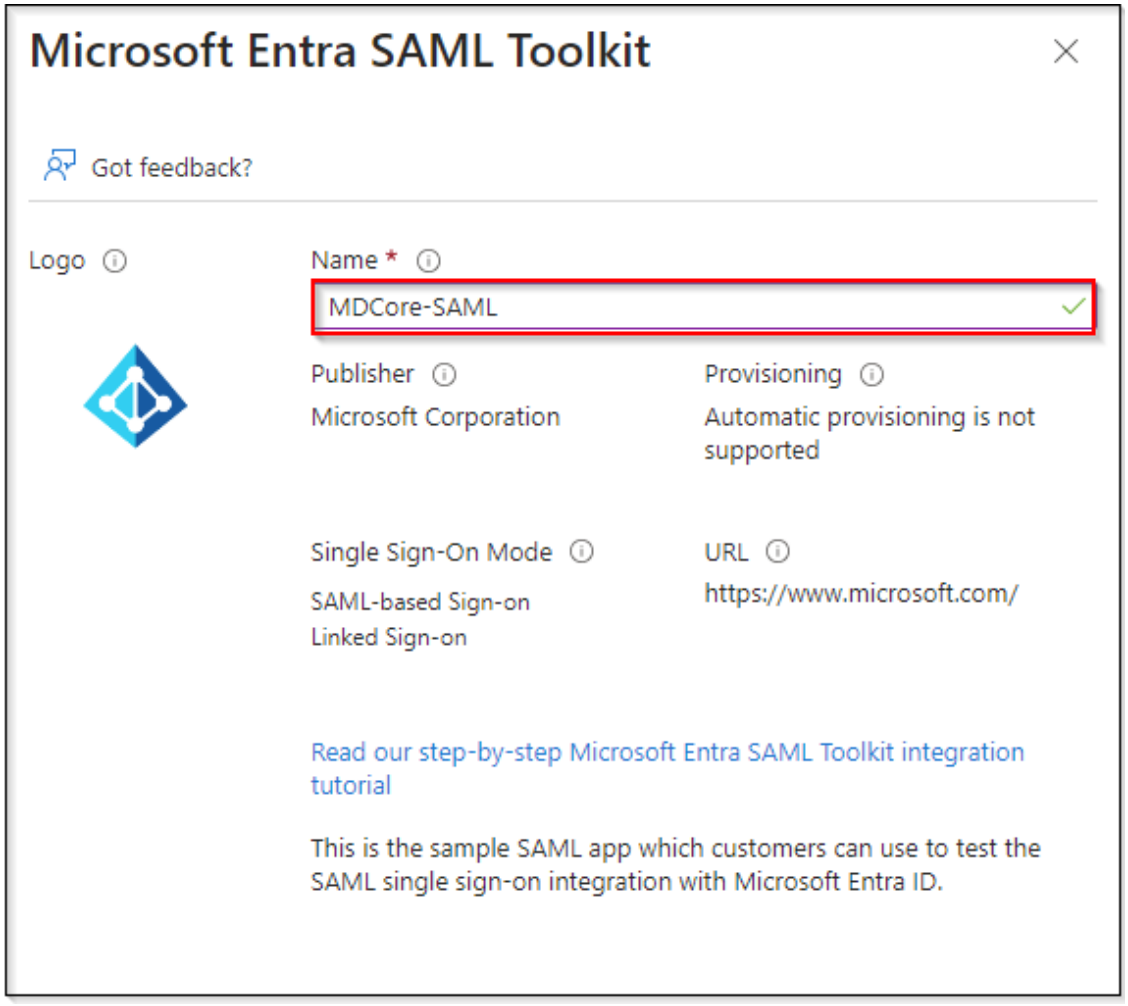
4. On `All applications` page, click `New application`.



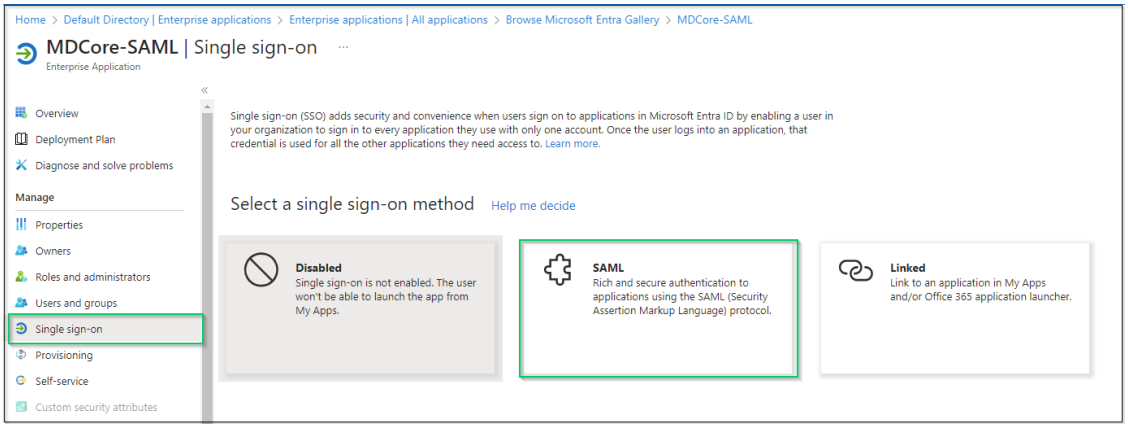
5. On `Browse Azure AD Gallery` page, search for "Microsoft Entra SAML Toolkit" and click on `Microsoft Entra SAML Toolkit` in the result panel.



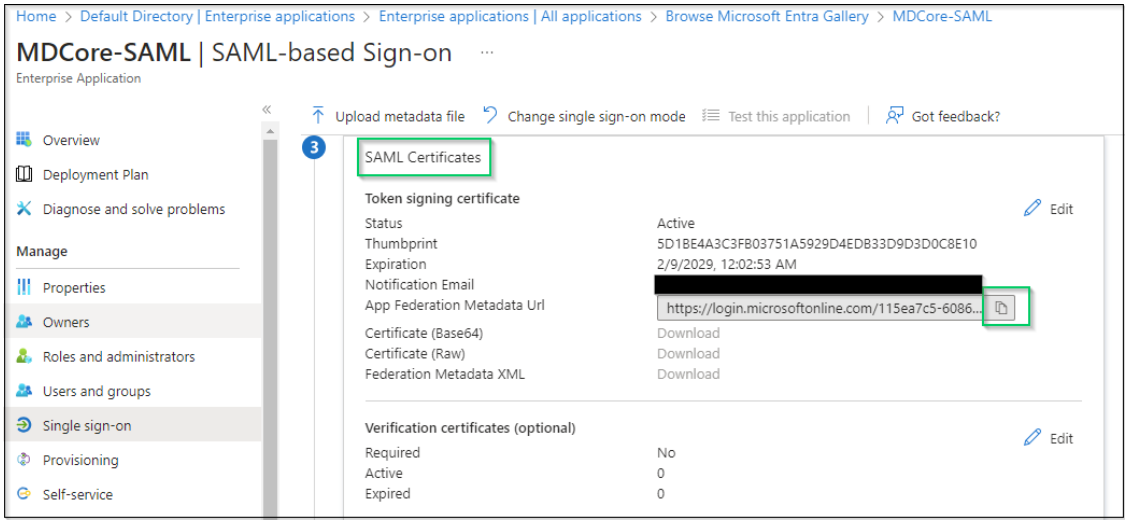
6. Fill in `Name` field with MDCore-SAML**,** for example, and click `Create` in the right sidebar.



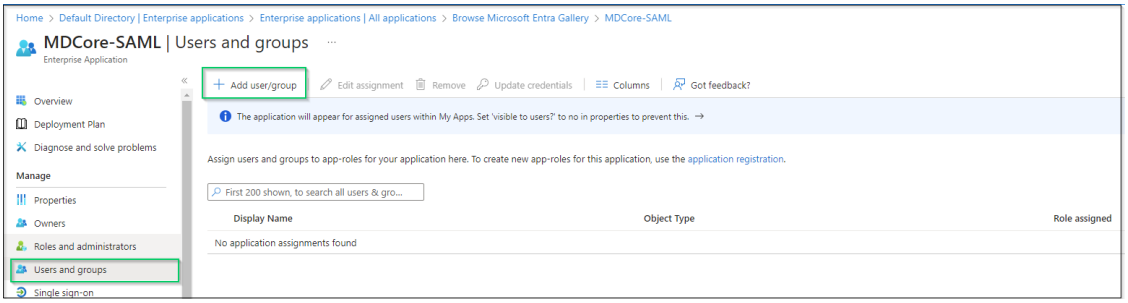
7. Navigate to `Single sign-on` on the left sidebar and click `SAML`.



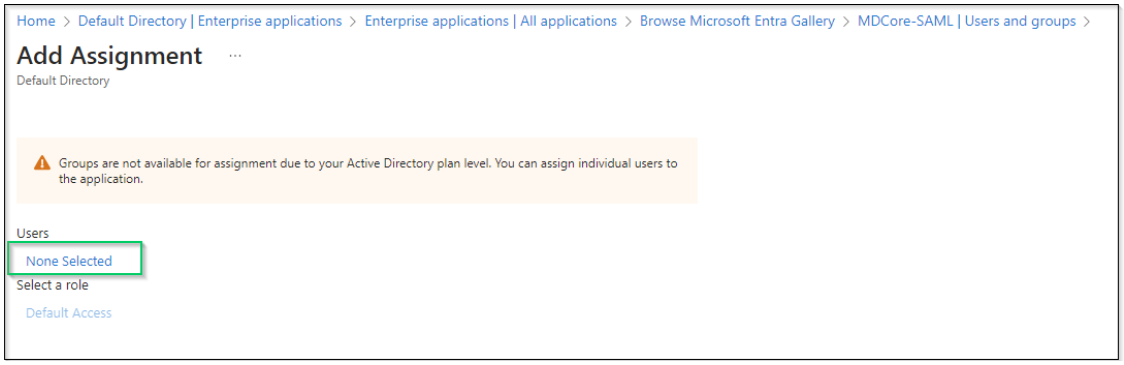
8. Go to `SAML Certificates` section, click the copy button at the far right of `App Federation Metadata Url` and store as *metadata_uri*.



9. Navigate to `Users and groups` in the left sidebar, click `Add user/group`.



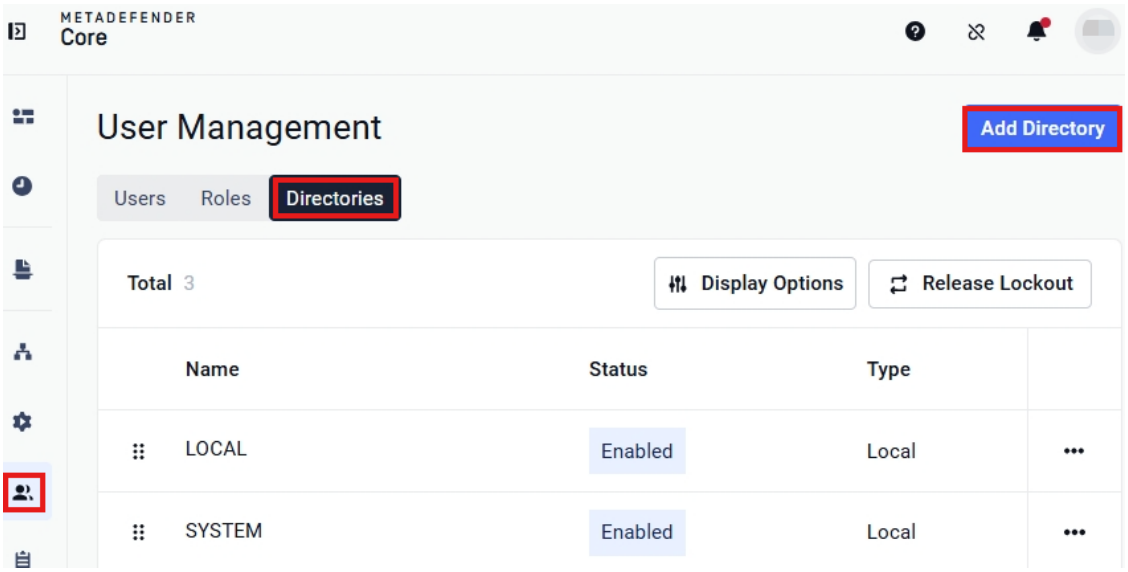
10. On `Add Assignment` screen, click `Non Selected` and add users who are allowed to log in to the app, then click `Select` in the right panel.



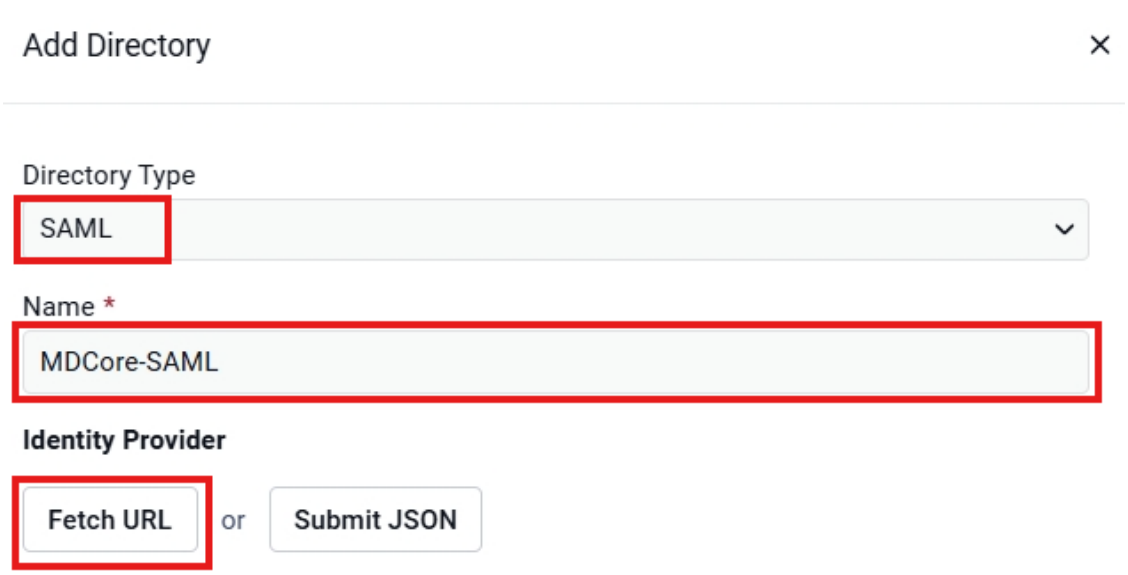
11. Finally, click `Assign` to complete.

Create SAML directory in MetaDefender Core

1. Sign in to MetaDefender Core management console.
2. On the dashboard, click `User Management` in the left sidebar.
3. In `User Management` page, select `Directories` tab and click on `Add directory` in the top right corner.



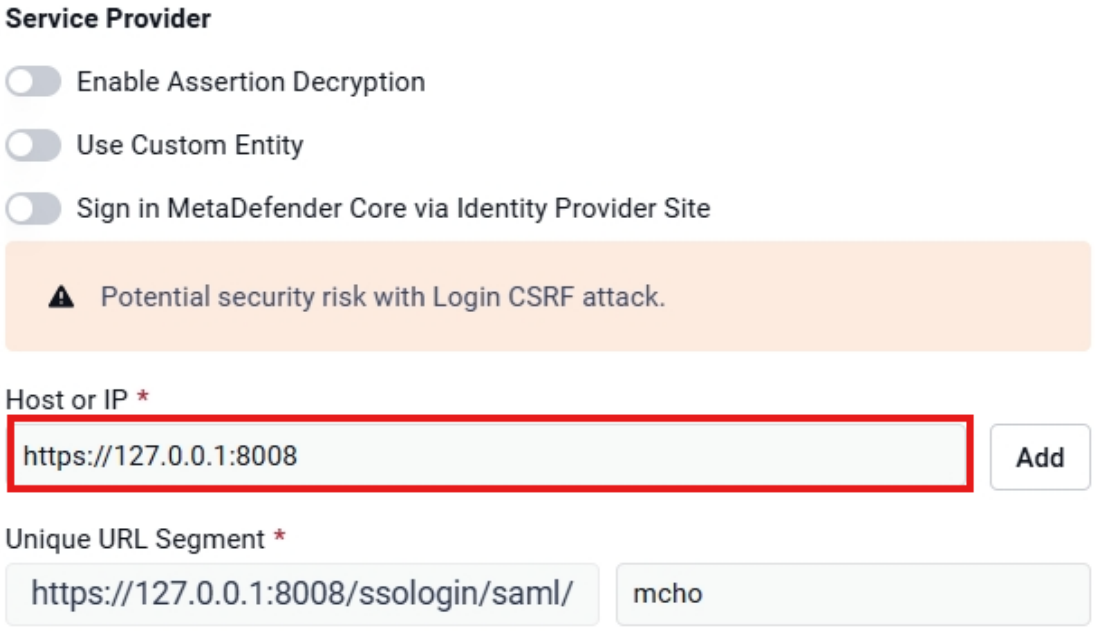
4. In `Add Directory` page, choose SAML in `Directory type`, enter a name for the new directory, such as MDCore-SAML and click `Fetch URL`.



5. Paste the URI stored in *metadata_uri* into the box under `Fetch URL`, then click `OK` and wait a moment for MetaDefender Core to set Microsoft Entra ID as its IDP.

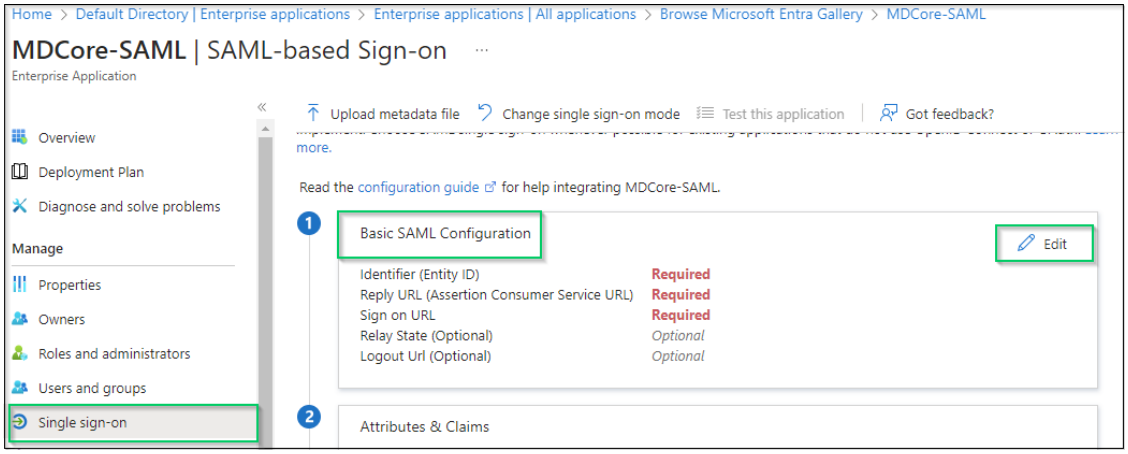


6. In `Service Provider` section, fill in `Host or IP` with the address where MetaDefender Core is hosted, for example https://127.0.0.1:8008.
7. Copy the `Login URL` and store in *reply_uri* for the later steps.

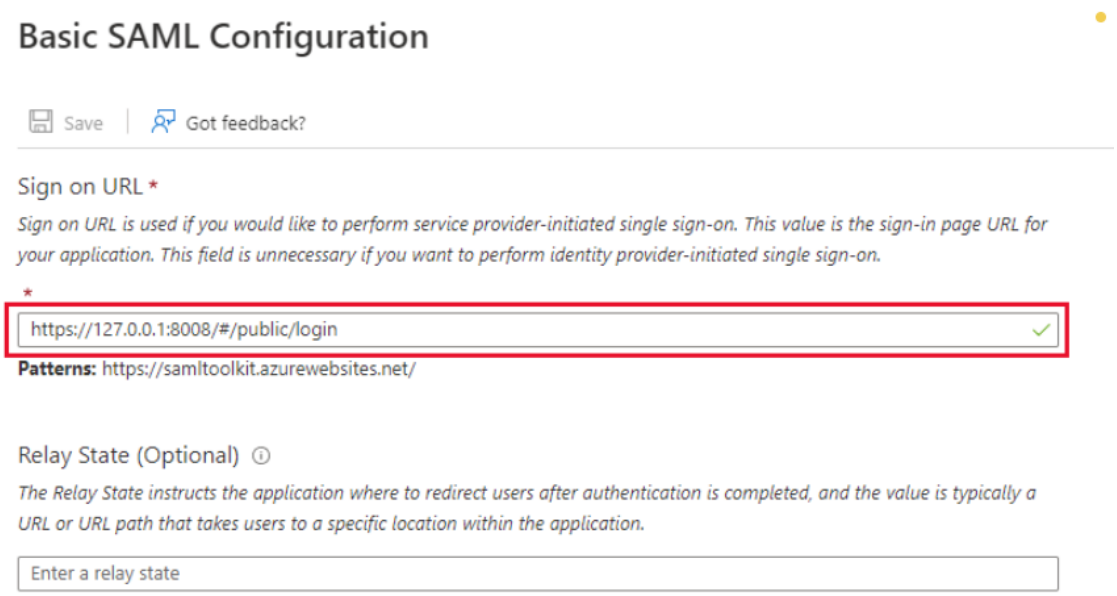


Complete configuration in Entra ID

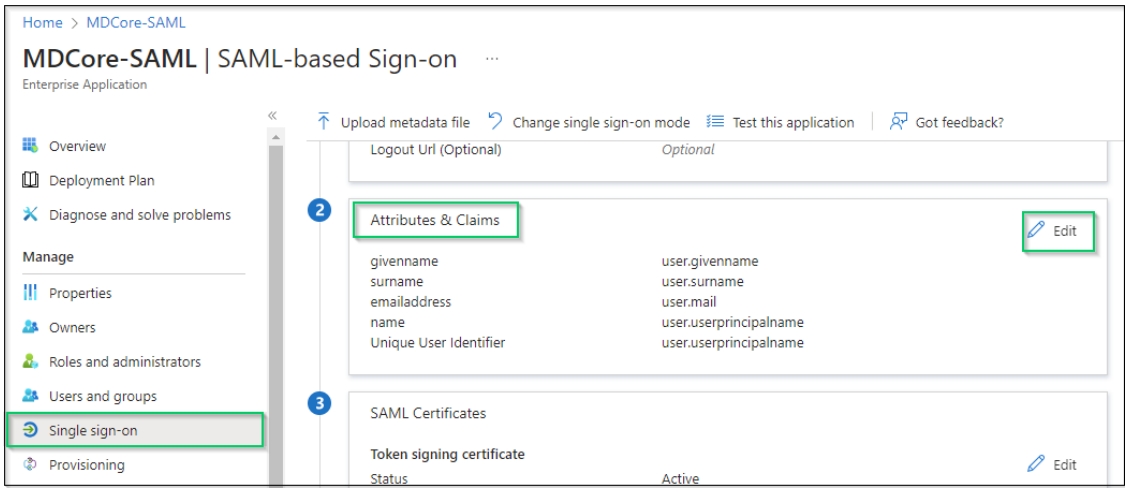
1. Switch back to Microsoft Entra ID, on `SAML-based Sign-on` page, navigate to `Basic SAML Configuration` and click `Edit` in the top right corner.



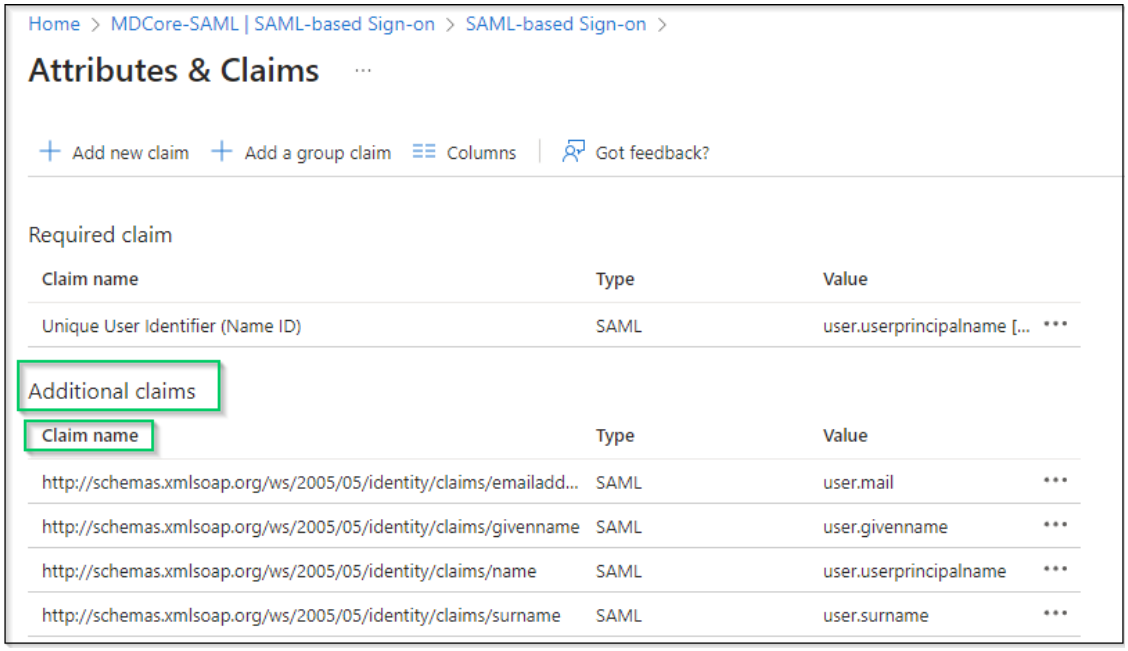
2. Navigate to `Identifier (Entity ID)` in the right sidebar, click `Add identifier`, then enter a unique ID to identify MDCore, such as MDCore-SAML. Store the identifier to *identifier*.
3. Navigate to `Reply URL(Assertion Consumer Service URL)`, click `Add reply URL` and fill in the URI with the value stored in *reply_uri*. Then, in `Sign on URL` (a required field, marked with a red asterisk), enter the MetaDefender Core login URL in the format `https://<FQDN>:8008`, for example `https://mdcore-lab:8008`, and click `Save`.



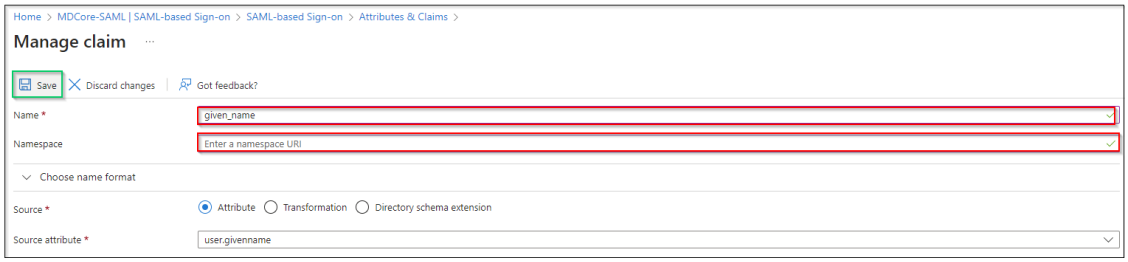
4. Navigate to `Attributes & Claims`, then click `Edit`.

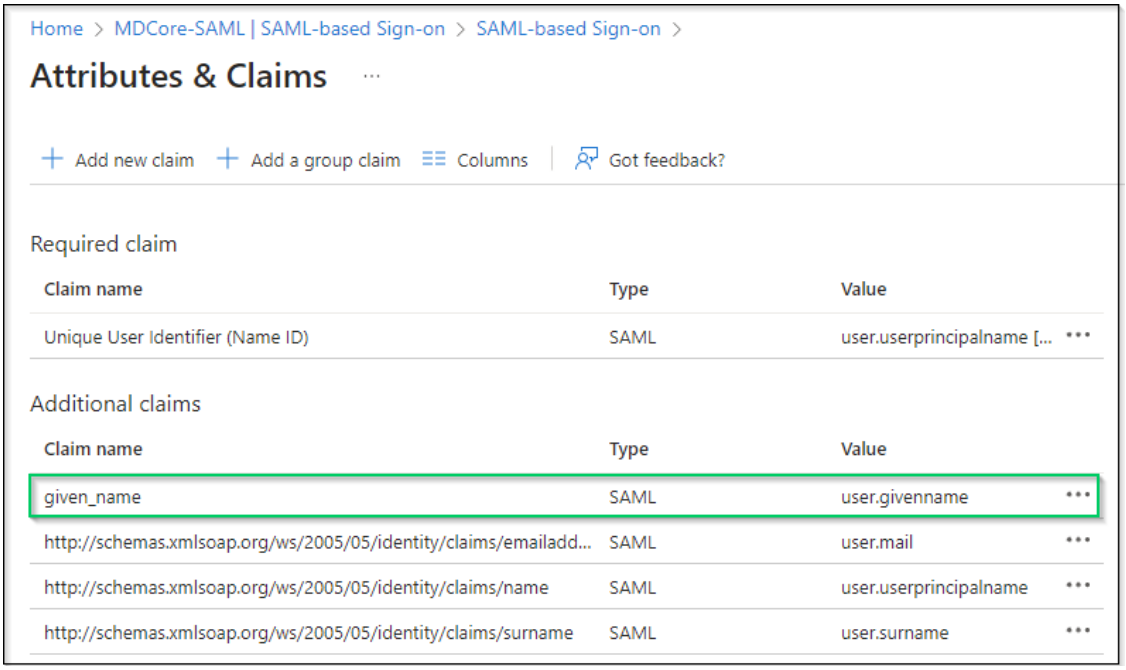


5. In `Attributes & Claims` page, go to `Additional claims`, and click on any item under `Claim name` to change its name.



6. Change value of `Name` field, leave `Namespace` empty and click `Save` to complete. In this instruction, the claim name is changed to given_name, which will be used later to identify the logged-in user on MetaDefender Core.





Info

If the names of attributes and claims provided by Microsoft Entra ID are sufficient, the customers are recommended for direct use in identifying the logged-in user in MetaDefender Core.

Complete configuration in MetaDefender Core

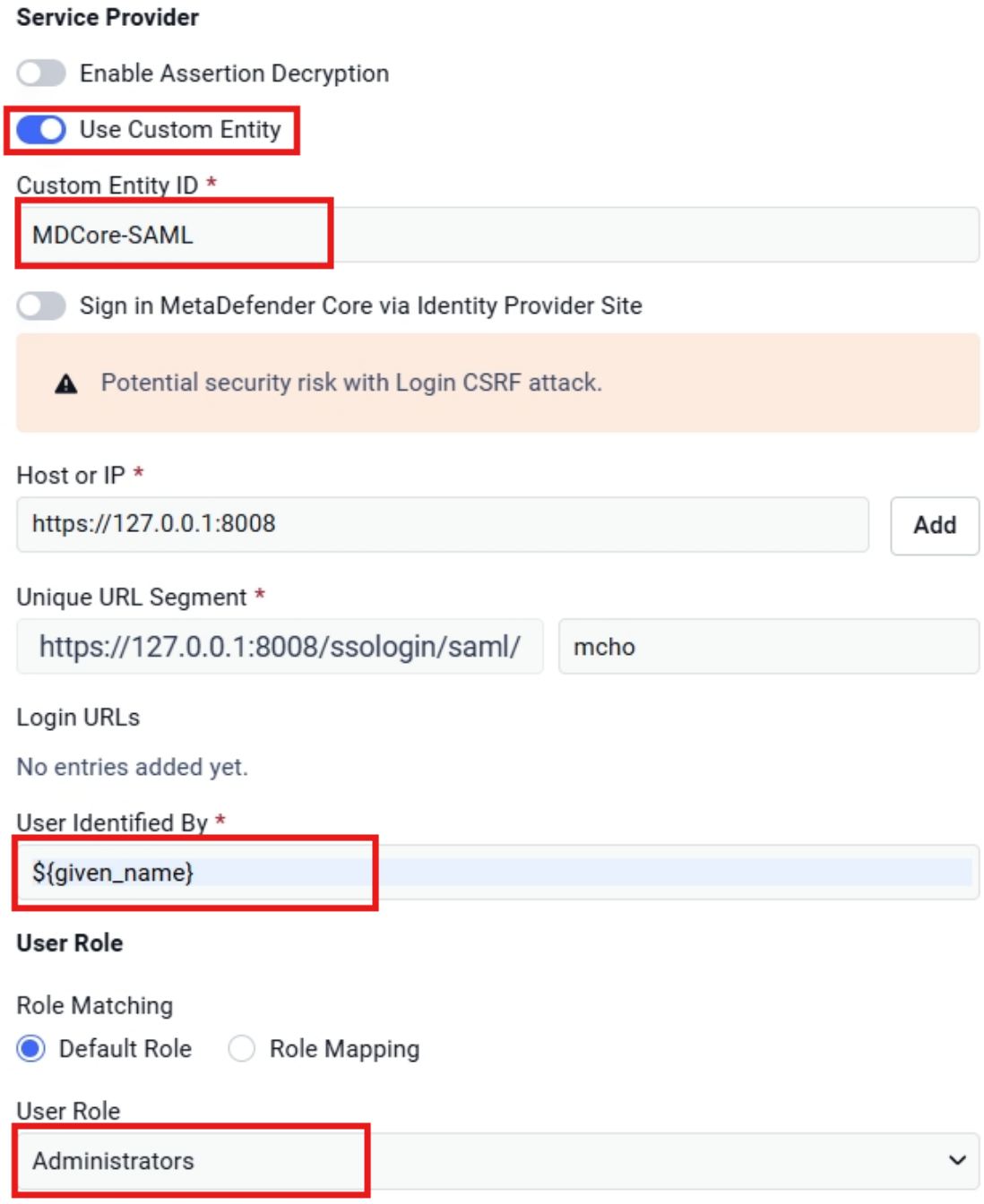
1. Switch back to MetaDefender Core screen. Under `Service Provider`, go to `Use custom entity ID` and fill in it with value stored in *identifier*, MDCore-SAML in this example.
2. Fill in `User identified by` with ${given_name}.

Info

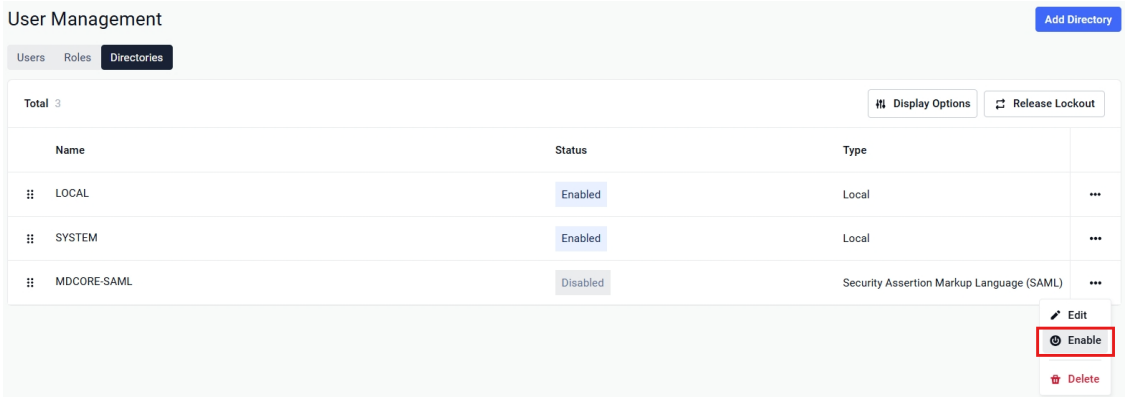
If the namespace is not removed from the claim name in step 6 of the previous section, the full claim name including the namespace must be used here to build user identity.

For example, if the claim of http://schemas.xmlsoap.org/ws/2005/05/identitity/clams/given_name is added to Microsoft Entra ID, then ${http://schemas.xmlsoap.org/ws/2005/05/identitity/clams/given_name} should be used by MetaDefender Core to build user identity.

3. Select appropriate role for the user under `User Role`.
4. Click `Add` to complete the settings.

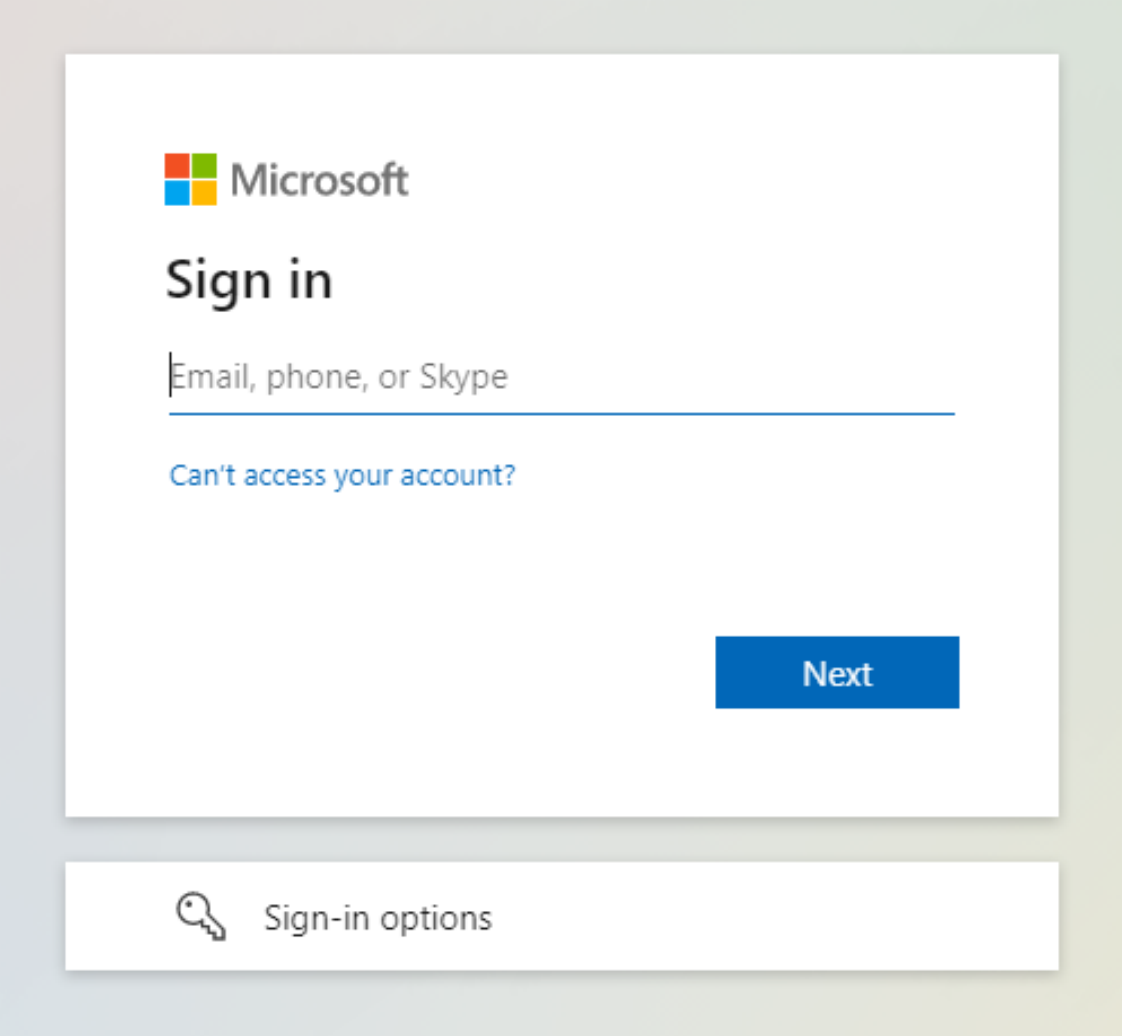


5. In `User Management` screen of MetaDefender Core, toggle the new directory named MDCORE-SAML. A dialog box appears to confirm the action. Once `Enable` is clicked, all sessions are expired immediately.



Test the integration

1. At the home screen of MetaDefender Core, click `Login`; the user is redirected to Microsoft Entra ID sign-in page.

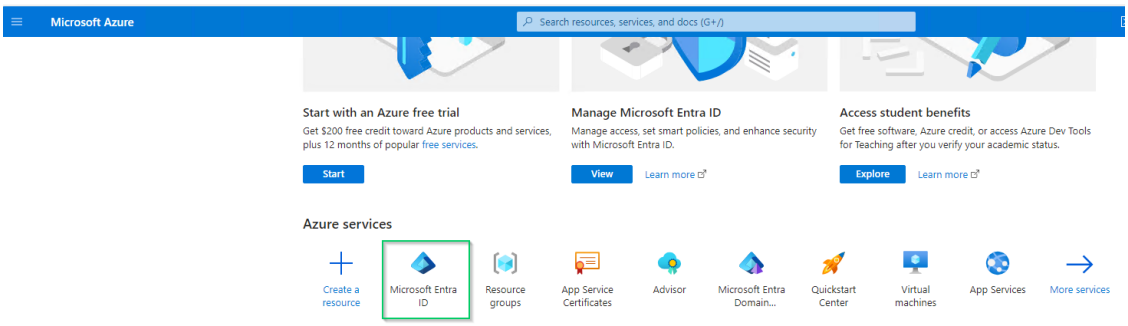


2. Sign in using the account registered with Microsoft Entra ID.
3. If everything goes correctly, MetaDefender Core dashboard is displayed with user's identity shown in the top right corner.
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

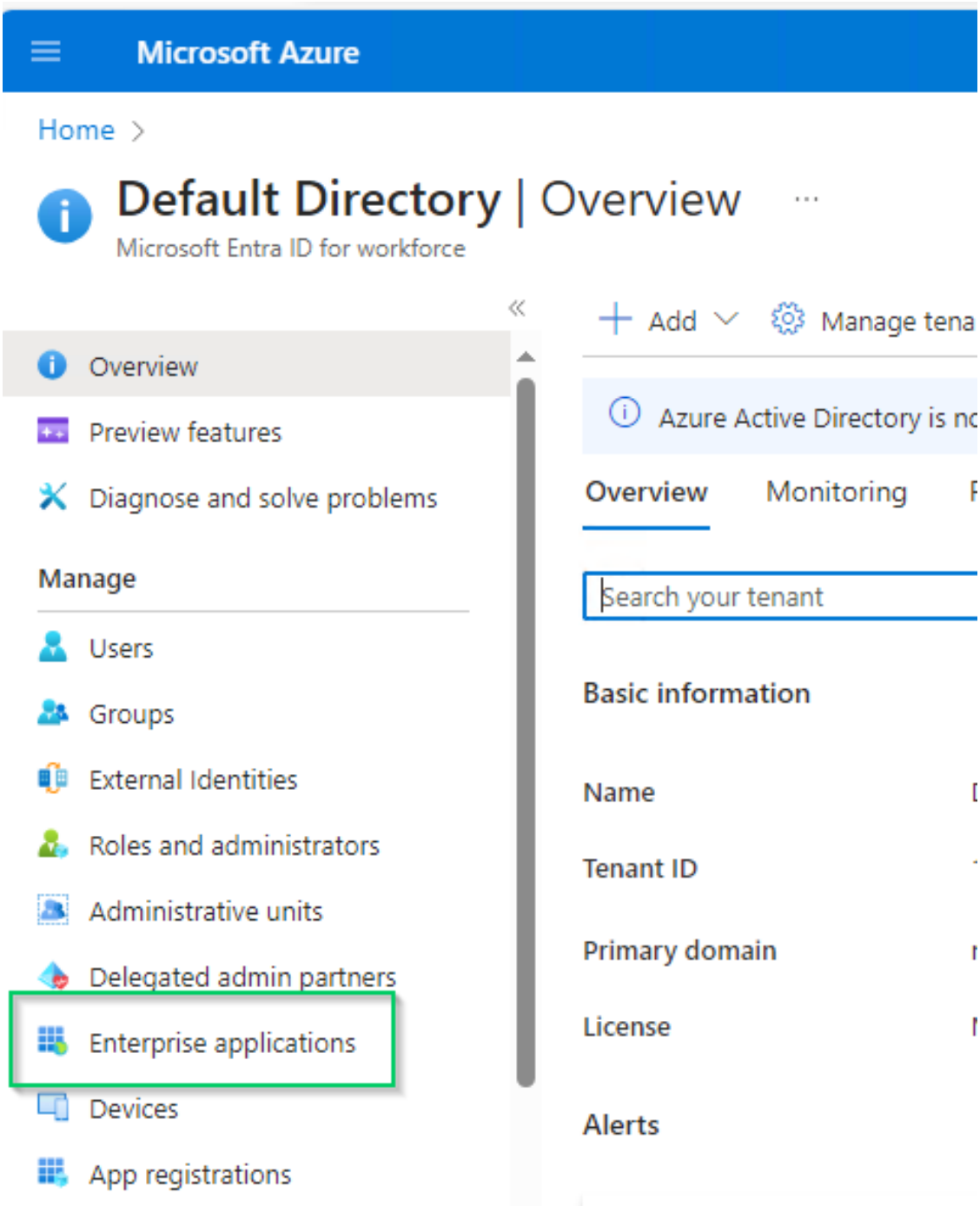
Setup pseudo IdP-initiated SSO

> **Info**
> Microsoft Entra ID does not support IdP-initiated SSO in the same way as other Identity Providers. It actually accesses the login page of its SP and does initiate Single Sign-in from there; so basically, it is SP-initiated.

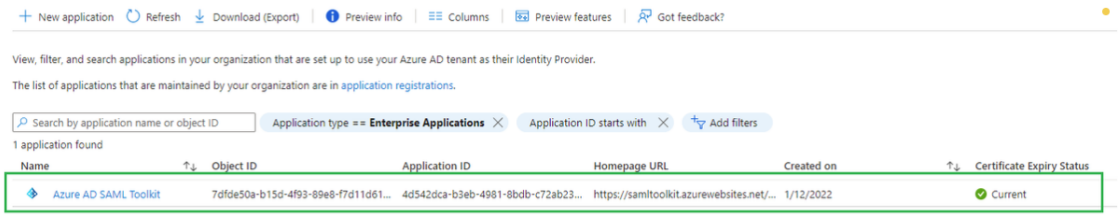
1. Sign in to Azure management page.
2. Navigate to `Azure services` section, click on `Microsoft Entra ID`.



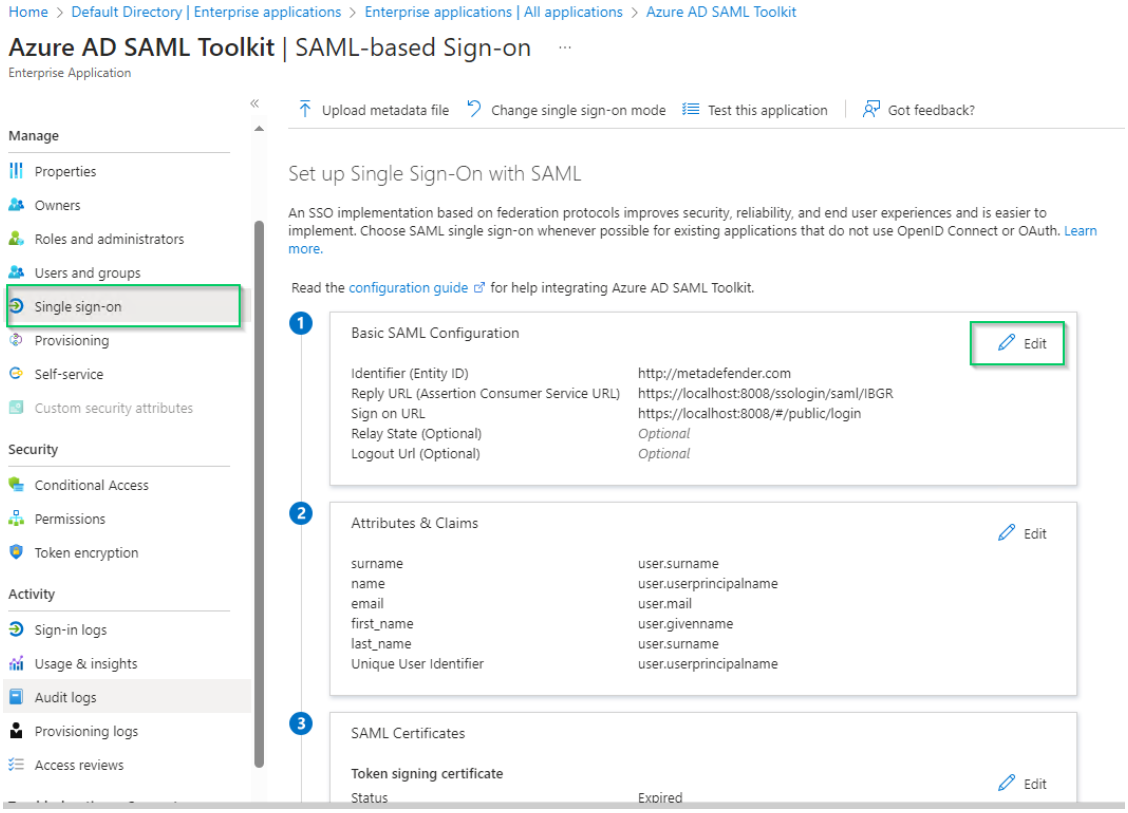
3. In the left sidebar, click on `Enterprise applications`.



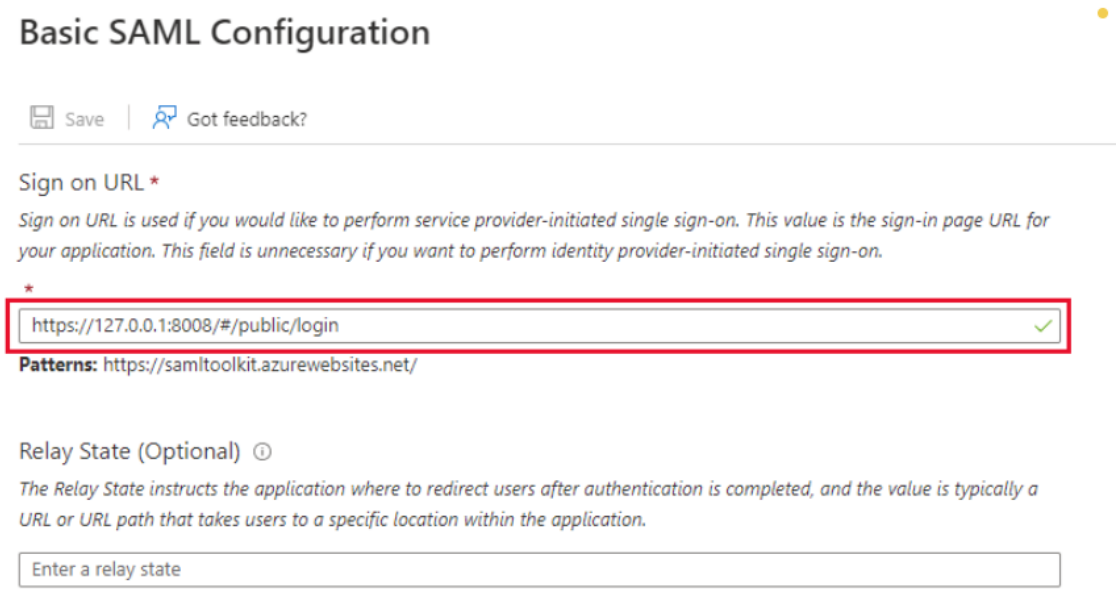
4. Select `Azure AD SAML Toolkit` from the list of enterprise applications.



5. Choose `Single sign-on` in the left sidebar, go to `Basic SAML Configuration` section, and click `Edit` at the top right of the section.



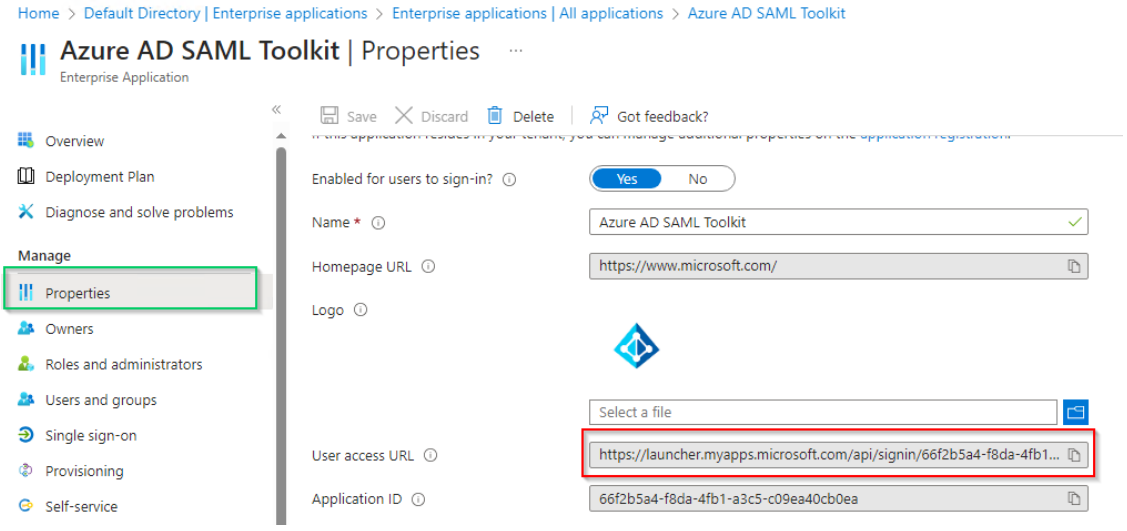
6. In `Basic SAML Configuration` right sidebar, enter Core login URL in the field under `Sign on URL`.



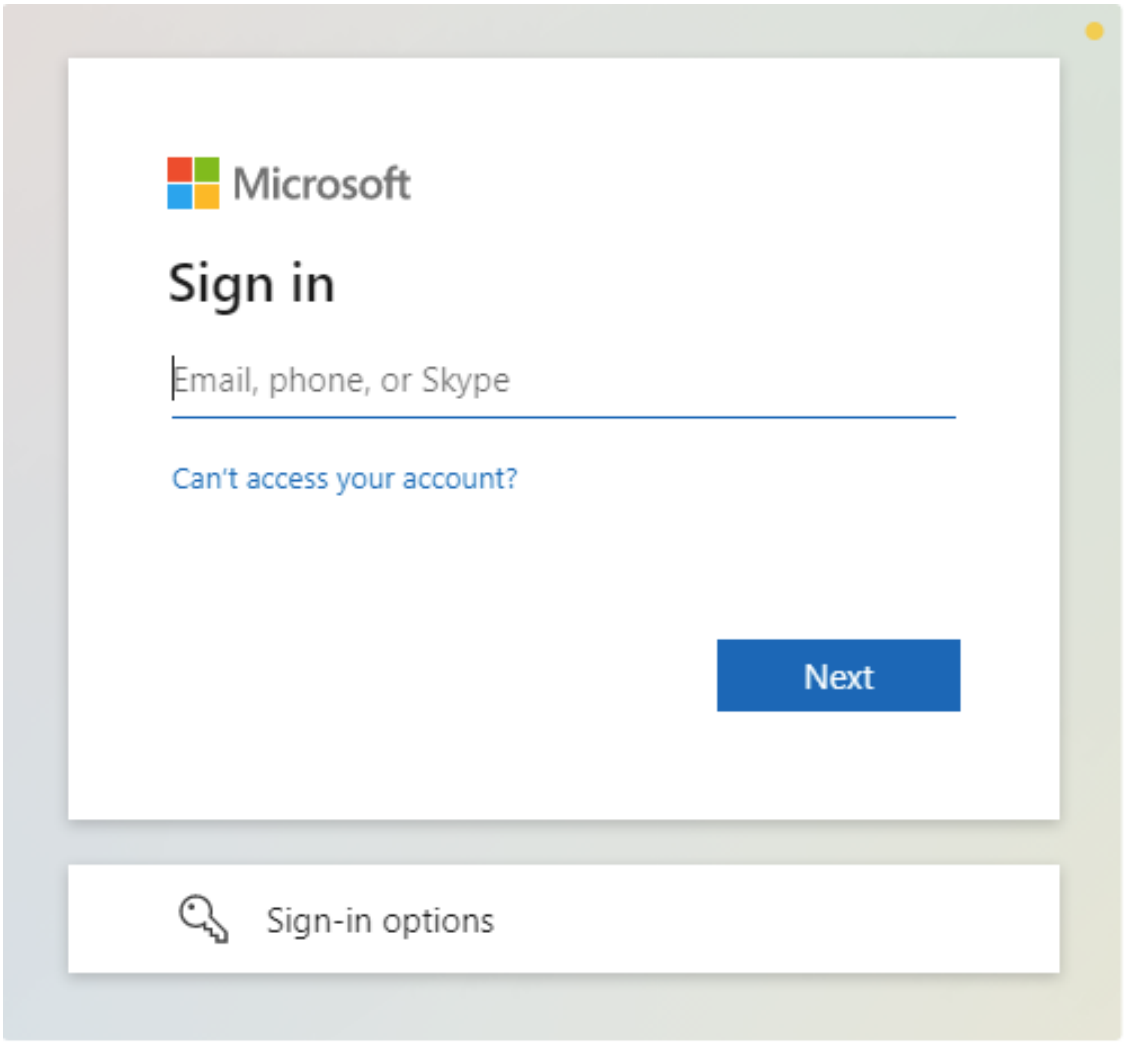
7. Click `Save` to complete.

Test IdP-initiated SSO

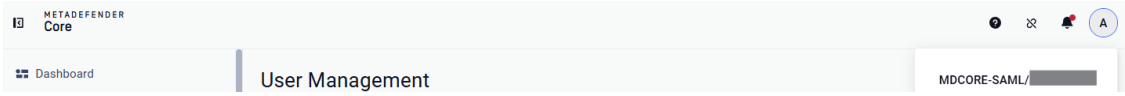
1. Sign in to Azure management page.
2. Navigate to `Azure services` section, and select `Microsoft Entra ID`.
3. In the left sidebar, click on `Enterprise applications`.
4. Choose `Azure AD SAML Toolkit` from the list of enterprise applications.
5. Go to `Properties` tab.



6. Copy the URL next to `User access URL`.
7. Paste the copied URL into your browser and sign in.



8. If everything goes well, MetaDefender Core dashboard will appear with the user identity displayed in the top right corner.

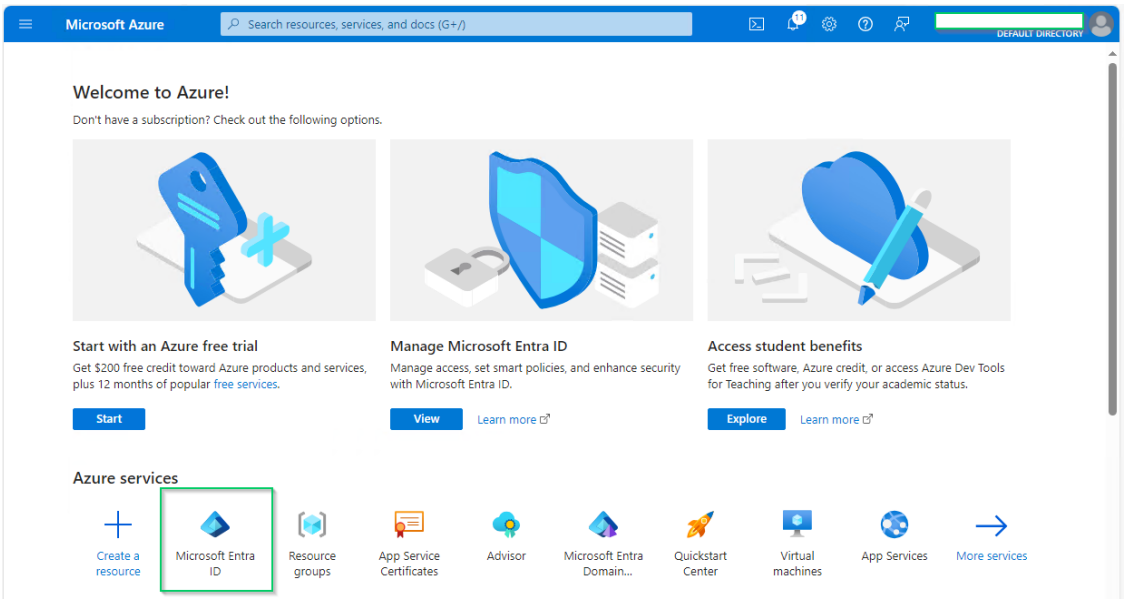


9. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

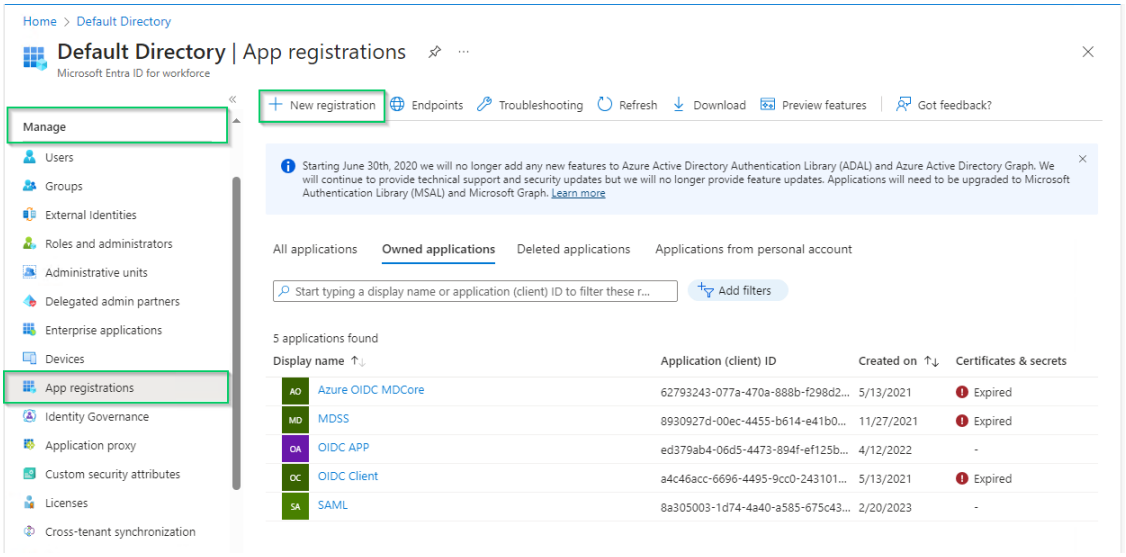
Configure OIDC SSO

Register a new application in Azure

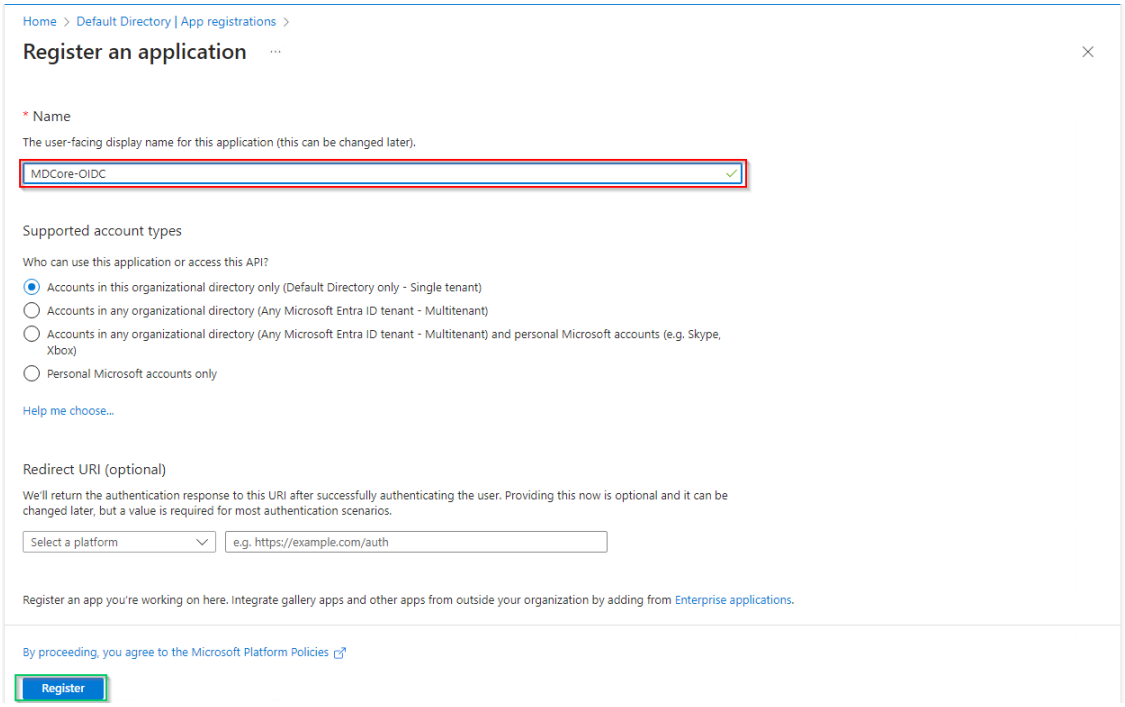
1. Access Azure portal and sign in.
2. On the home page, select `Microsoft Entra ID` under `Azure services`.



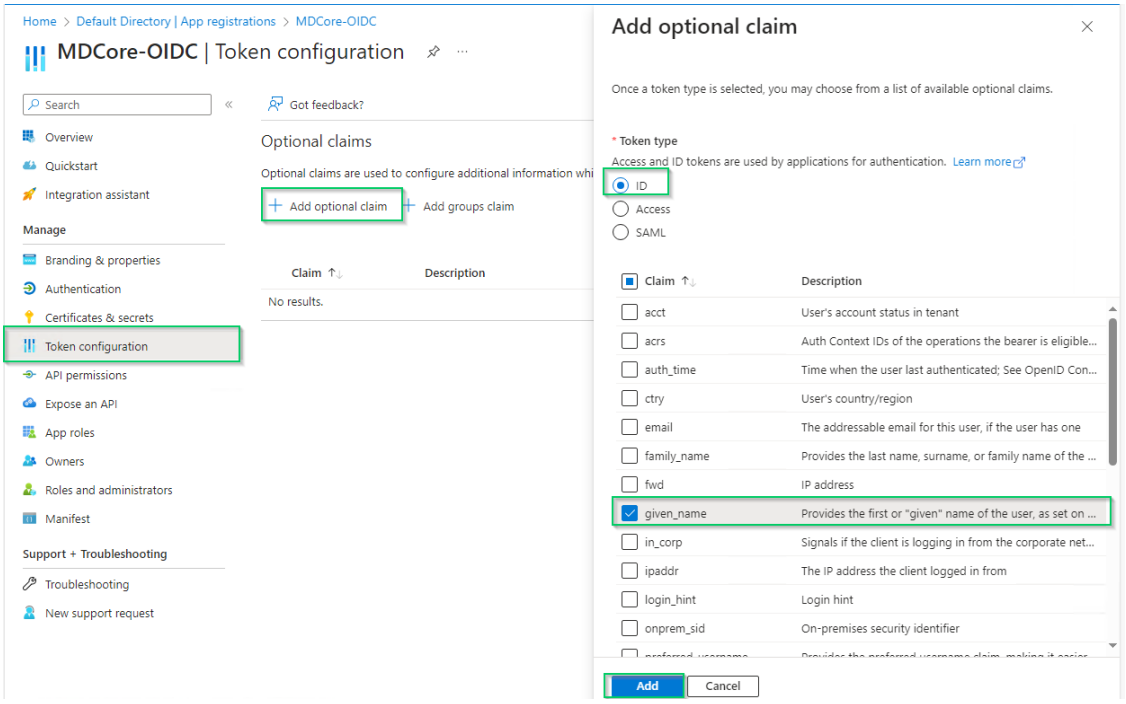
3. Select `App registrations` under `Manage` section on the left sidebar, then click on `New registration` on the top menu bar.



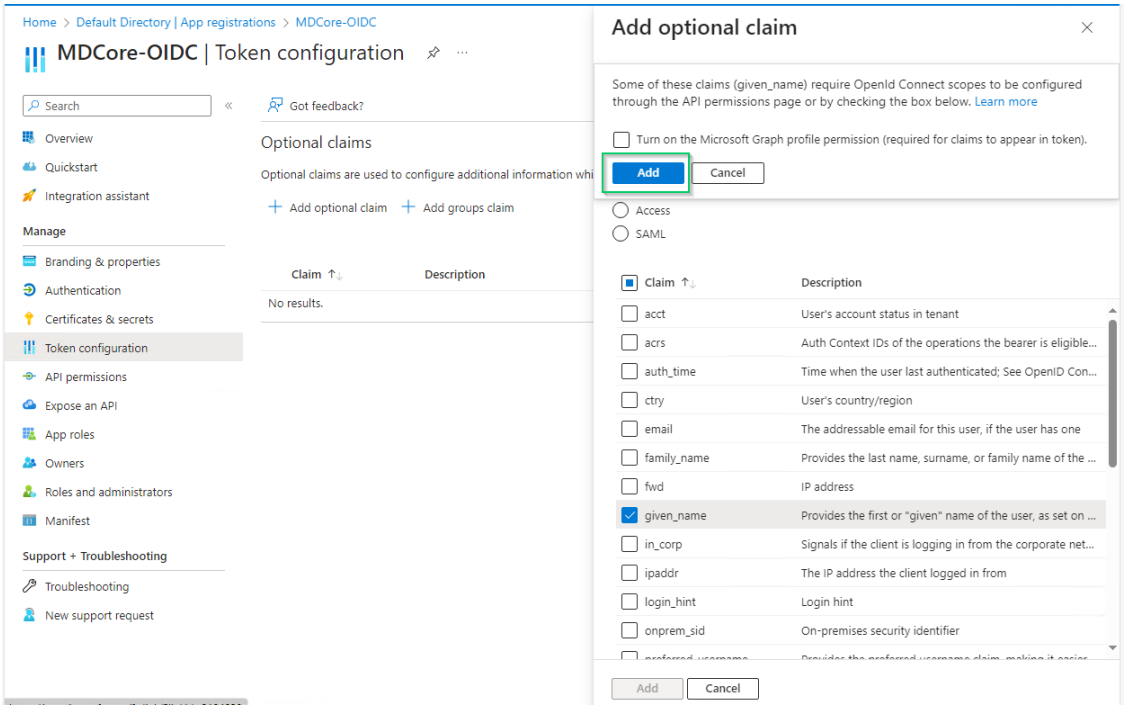
4. In `Register an application` page, fill in the name for the new application, using MDCore-OIDC as an example, and click `Register` to complete.



5. Select `Token configuration` in the left sidebar, hit `Add optional claim`, choose `ID` on the right sidebar, enable essential claims (given_name in this example) and click `Add` at the bottom.



6. Click `Add` on the popup to finish.

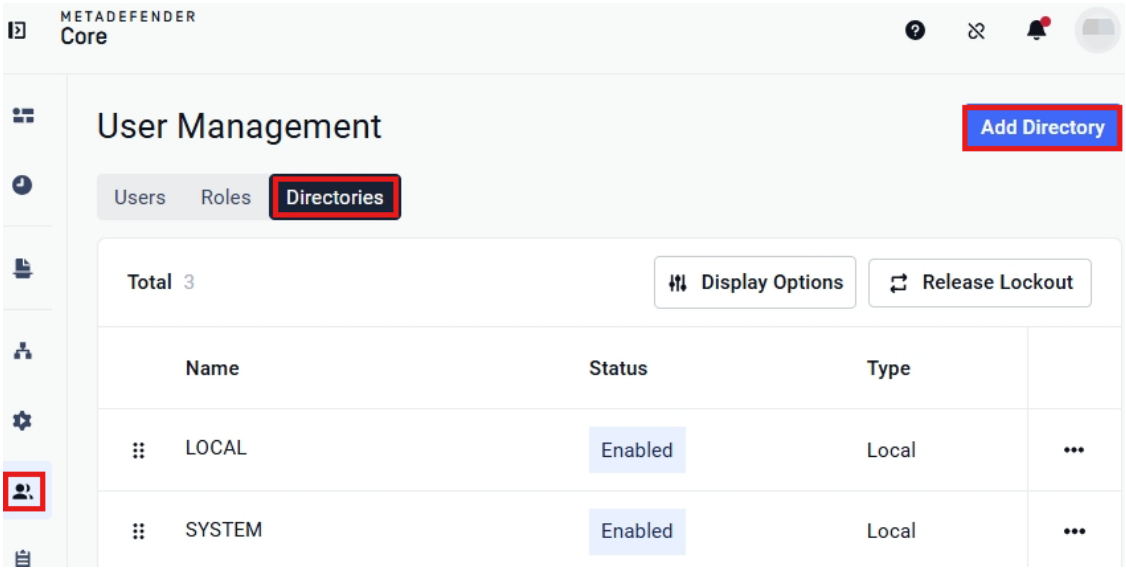


Info

Enabled claims will be used later by MetaDefender Core to identify logged-in user.

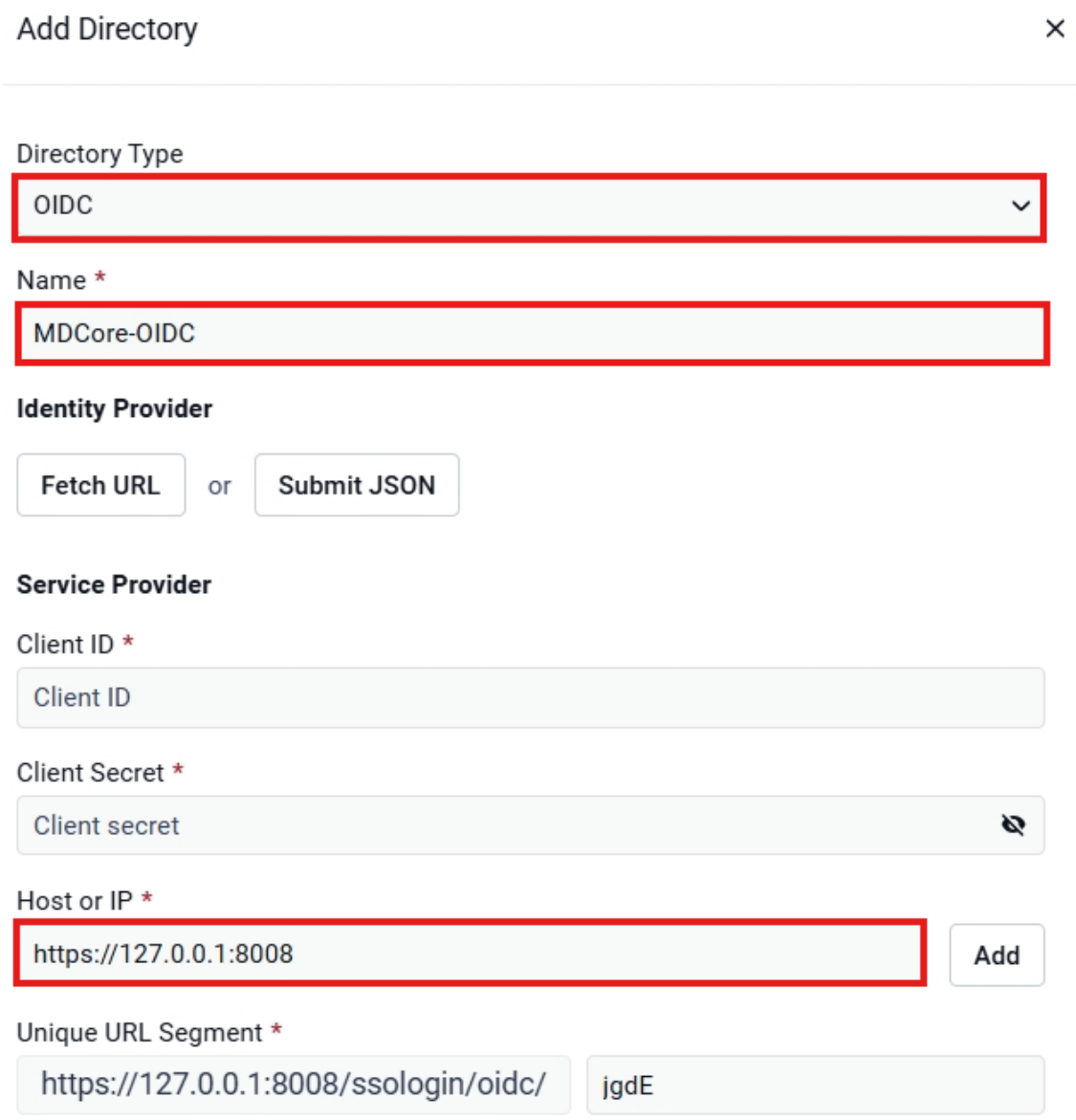
Create OIDC directory in MetaDefender Core

1. Sign in to MetaDefender Core.
2. At the dashboard, click `User Management` in the left sidebar.
3. In `User Management` page, choose `Directories` tab and click `Add directory` on the top right corner.



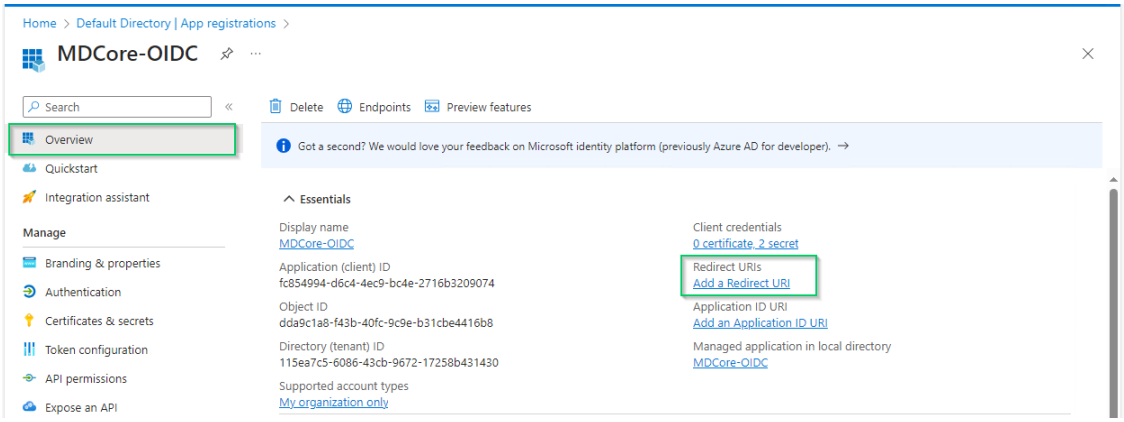
4. In `Add Directory` page, choose OIDC in `Directory type`.
5. Fill in the name the new directory, such as MDCore-OIDC.

6. In `Service Provider` section, fill in `Host or IP` with the host or IP address where MetaDefender Core is hosted; for this example, it is https://127.0.0.1:8008.
7. Copy the `Login URL` and store it somewhere for later use.

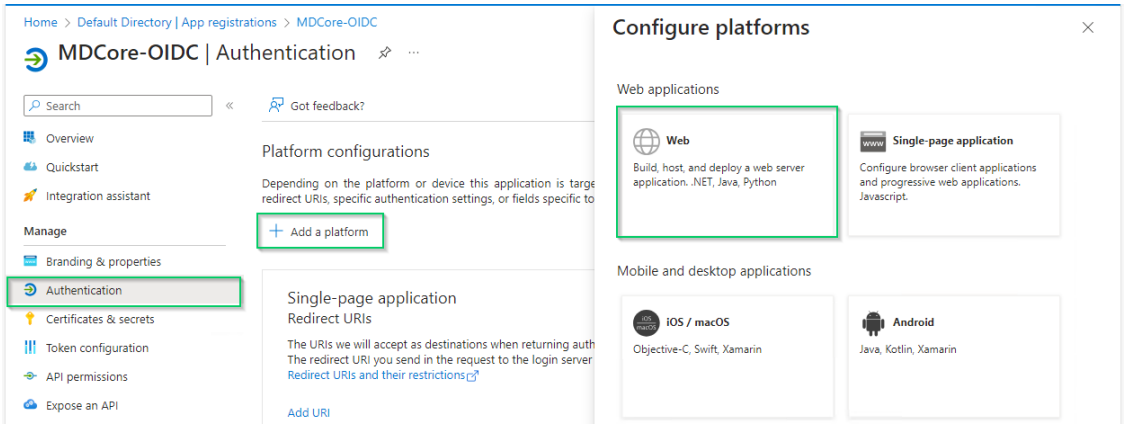


Complete configuration in Entra ID

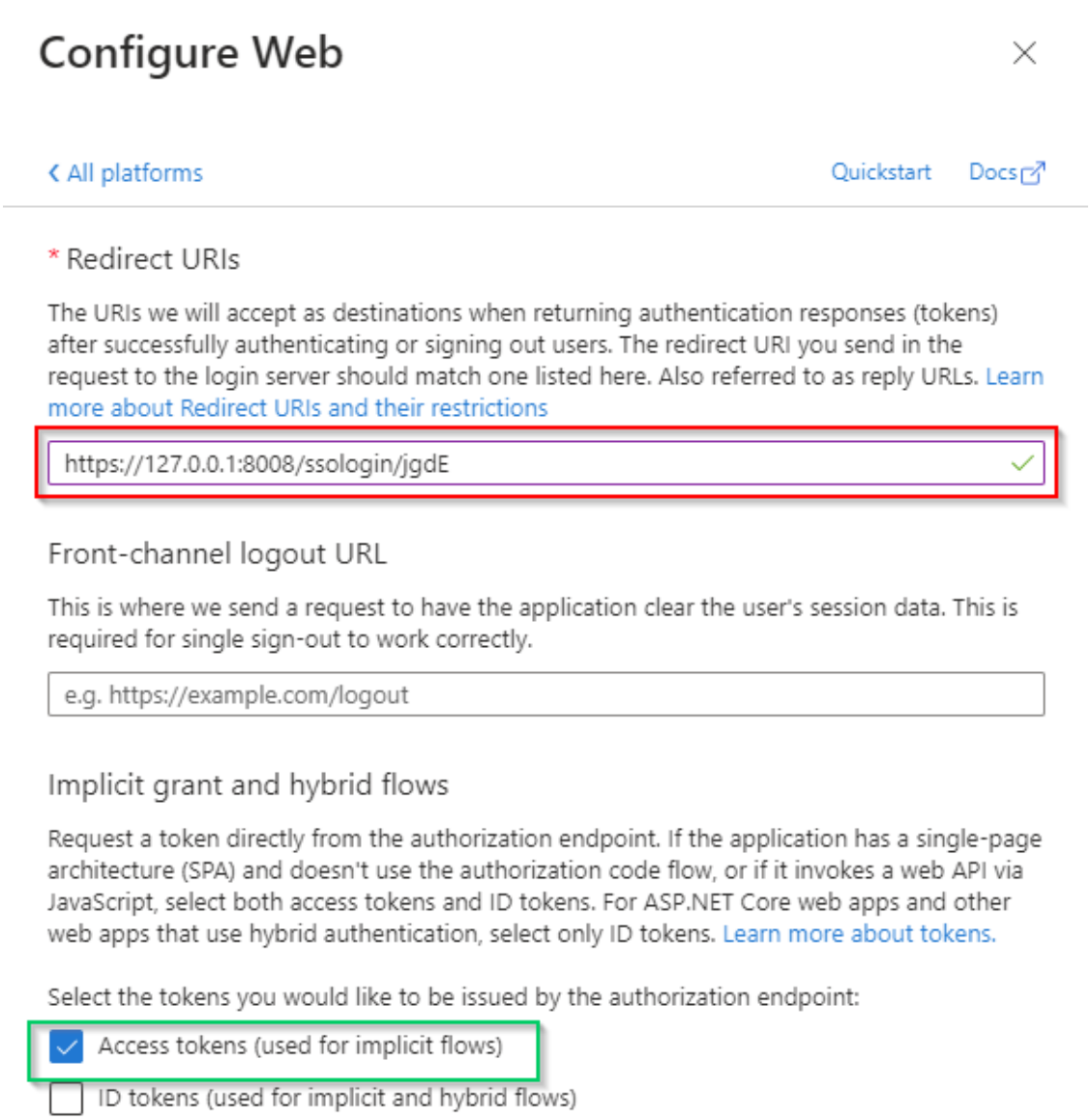
1. Back to Microsoft Entra ID, on `MDCore-OIDC` page, select `Overview` section in the left sidebar and click on `Add a Redirect URI` link to switch to `Authentication` page.



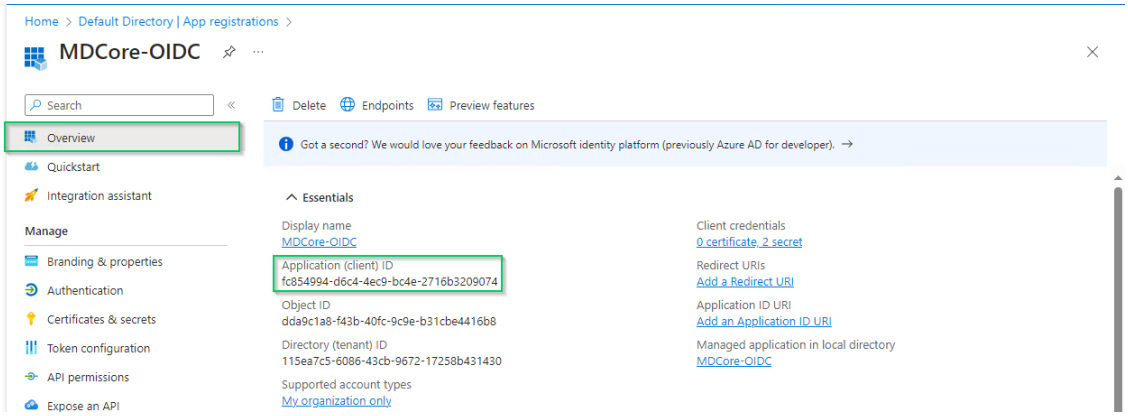
2. In `Authentication` page, click `Add a platform` and choose `Web` option in the right sidebar.



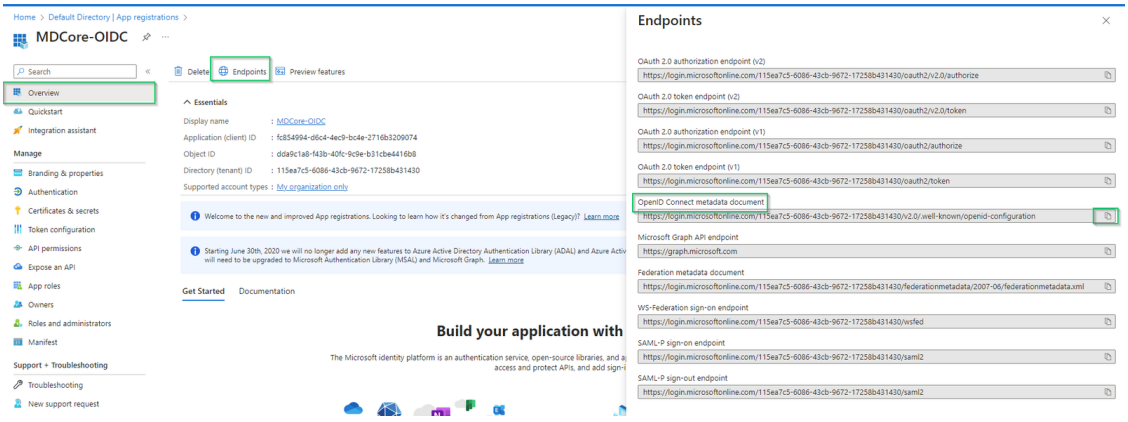
3. Fill in `Redirect URIs` field with the `Login URL` copied from MDCore, enable `Access tokens` option and click `Configure`.



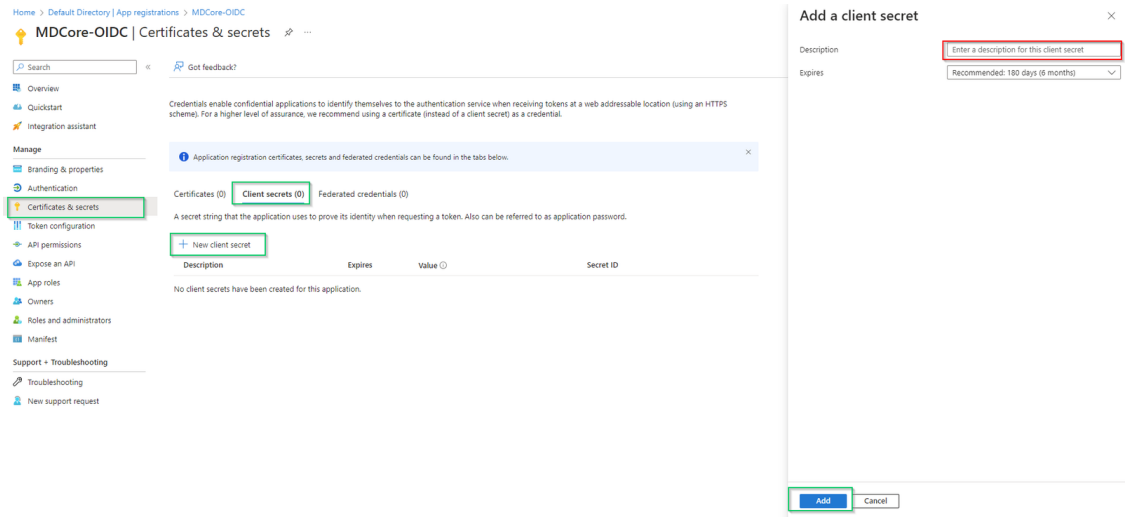
4. Select `Overview` in the left sidebar again, copy the string next to `Application (client) ID` and store as *client_id*.



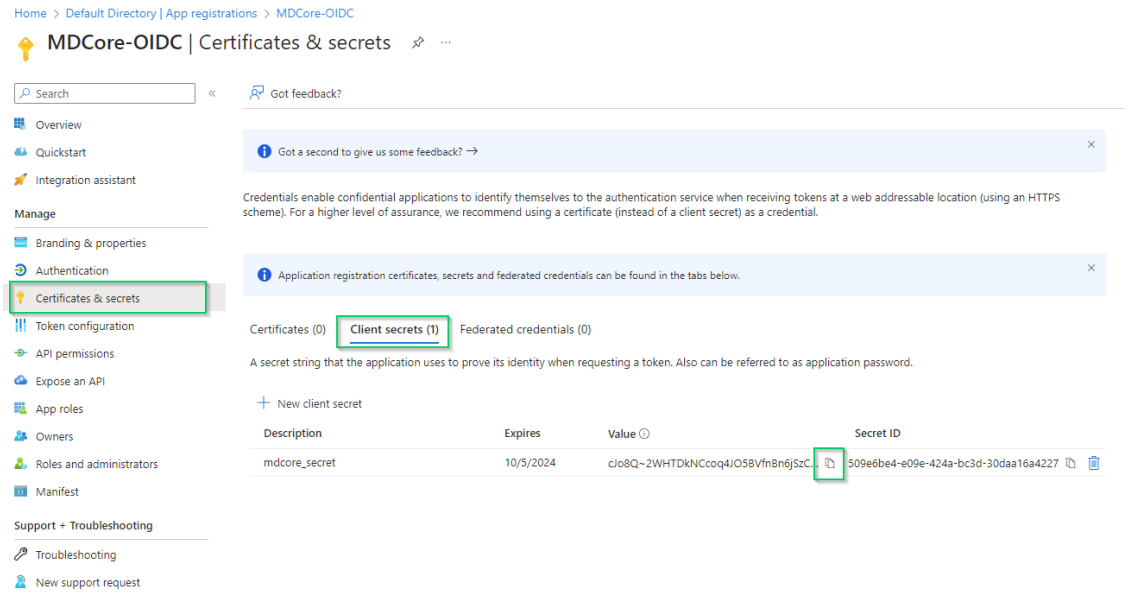
5. Click `Endpoints` in the top bar, copy the URI under `OpenID Connect metadata document` in the right sidebar and store it as *metadata_uri*.



6. Select `Certificates & secrets` in the left sidebar, go to `Client secrets (0)` tab, click `New client secret`, fill in the secret description and set expiration and, then click `Add` to complete.



7. A new client secret is created under `Client secret (1)` tab; copy the string under `Value` column and store it as *client_secret*.



Info

Entra ID will hide the client secret completely when page `Certificate & secrets` is left. Administrator is recommended to store the client secret in a safe place, otherwise they have to create a new one.

Complete configuration in MetaDefender Core

1. Switch back to MetaDefender Core. Under `Identity Provider` section, click `Fetch URL` and paste *metadata_uri* gathered in **Configure Azure** stage into the box under `Fetch URL`. Click `OK` and wait a moment for MetaDefender Core to set Microsoft Entra ID as its IDP.

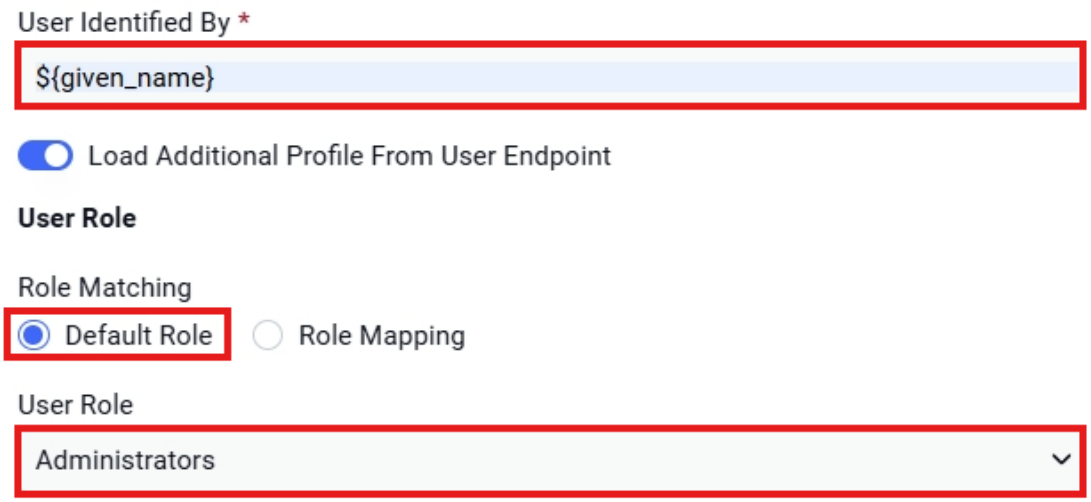


2. Under `Service Provider` section, paste *client_id* and *client_secret* gathered in **Configure Azure** stage to `Client ID` and `Client secret` respectively.

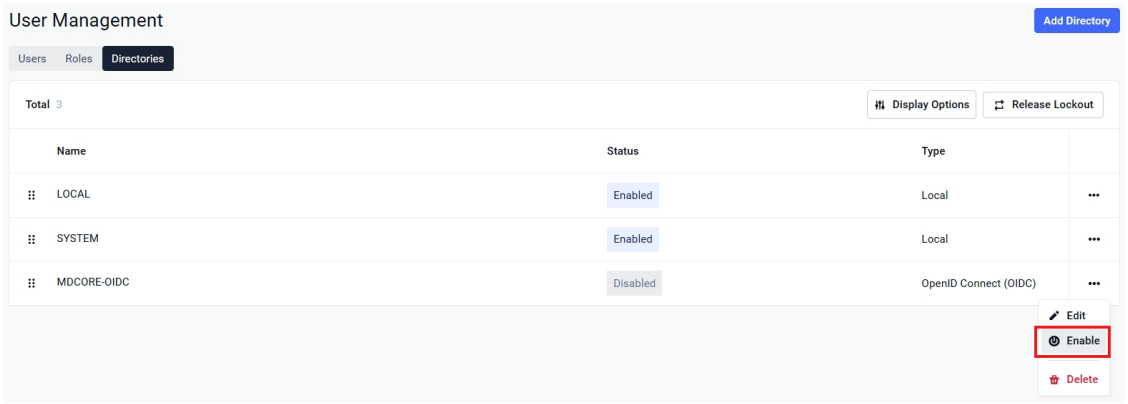


3. Fill user identity under `User identified by`, ${given_name} is used in this example.
4. Select `Default role` option and select correct role for the user under `User Role`

5. Click `Add` to complete the settings.

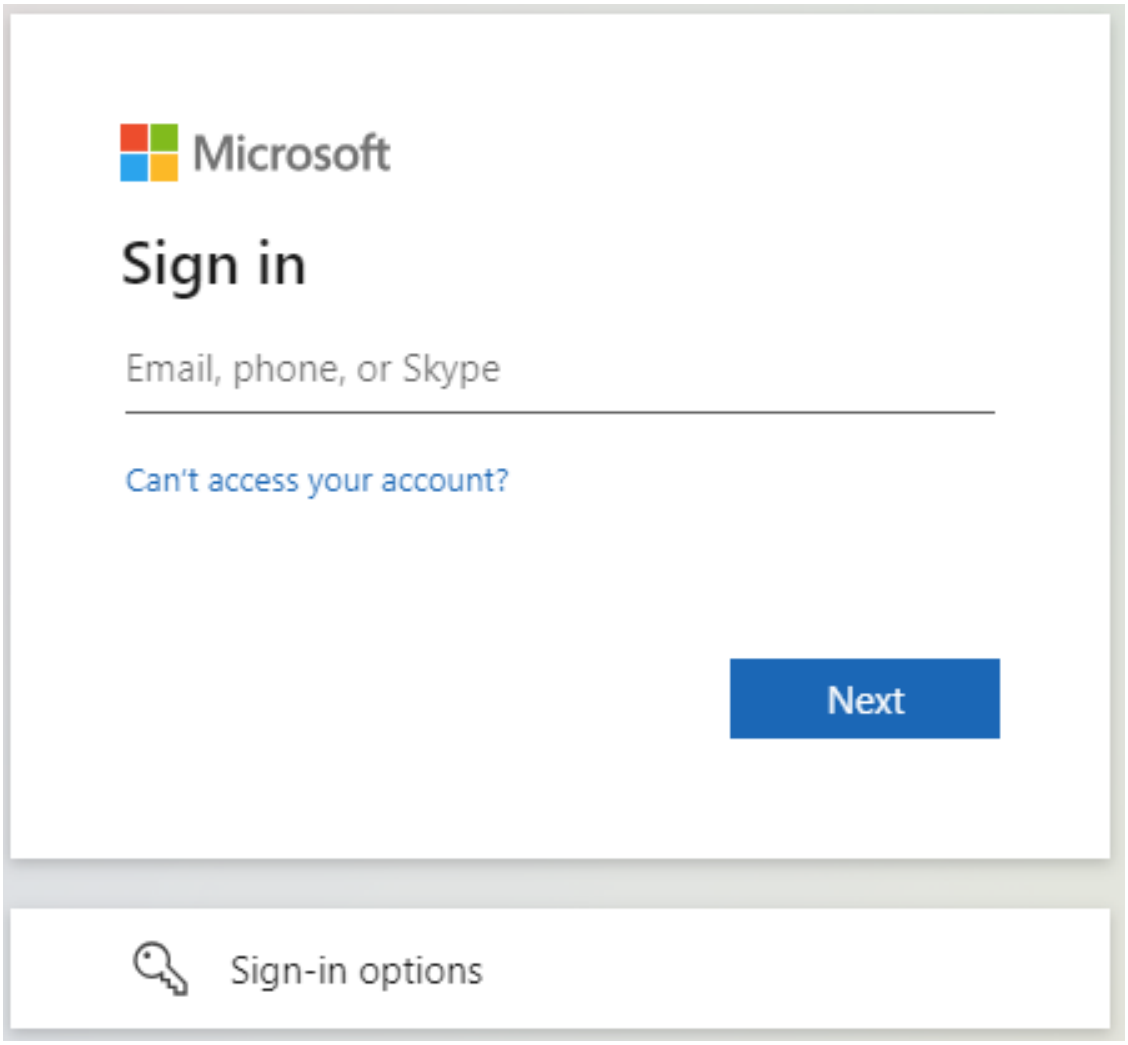


6. In `User Management` screen of MetaDefender Core, toggle MDCORE-OIDC directory. A dialog box appears to confirm the action. Once `Enable` is clicked, all sessions expire immediately.

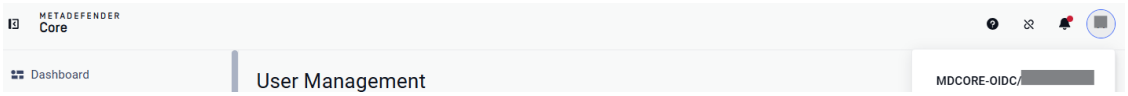


Test the integration

1. At the home screen of MetaDefender Core, click `Login`; the user is redirected to Microsoft Entra ID sign-in page.



2. Sign in using the account registered with Microsoft Entra ID.
3. If everything goes correctly, MetaDefender Core dashboard is displayed with user's identity shown in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Okta

The instructions guide users on how to configure Okta SAML and OIDC Single Sign-on for MetaDefender Core.

Configure SAML SSO

Warning

Since 5.15.0, MetaDefender Core has stopped supporting weak digest algorithms SHA1 and MD5. Customers

who integrate MD Core with Okta SAML 2.0 are advised to update their Digest Algorithm option to a stronger algorithm if the option is configured in the current settings.

Please follow these steps to update your digest algorithm in Okta:

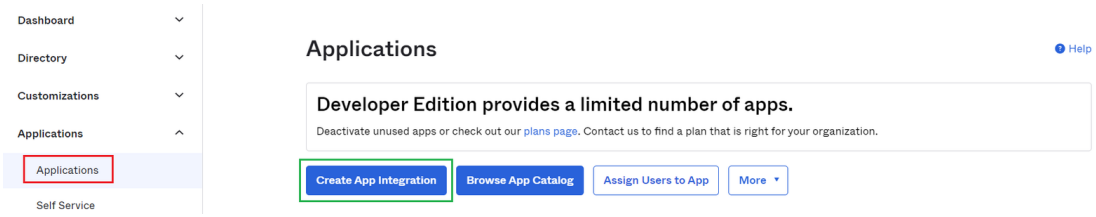
1. Sign to Okta console with your account.
2. In dashboard, expand `Applications` in the left sidebar and click `Applications`.
3. In your SSO integration, select `General` tab.
4. Navigate to `SAML Settings` and click `Edit`.
5. On `Edit SAML Integration` page, select `Configure SAML` tab.
6. Expand Advanced Settings, navigate to `Digest Algorithm` and select SHA256 or higher.

From MetaDefender Core,

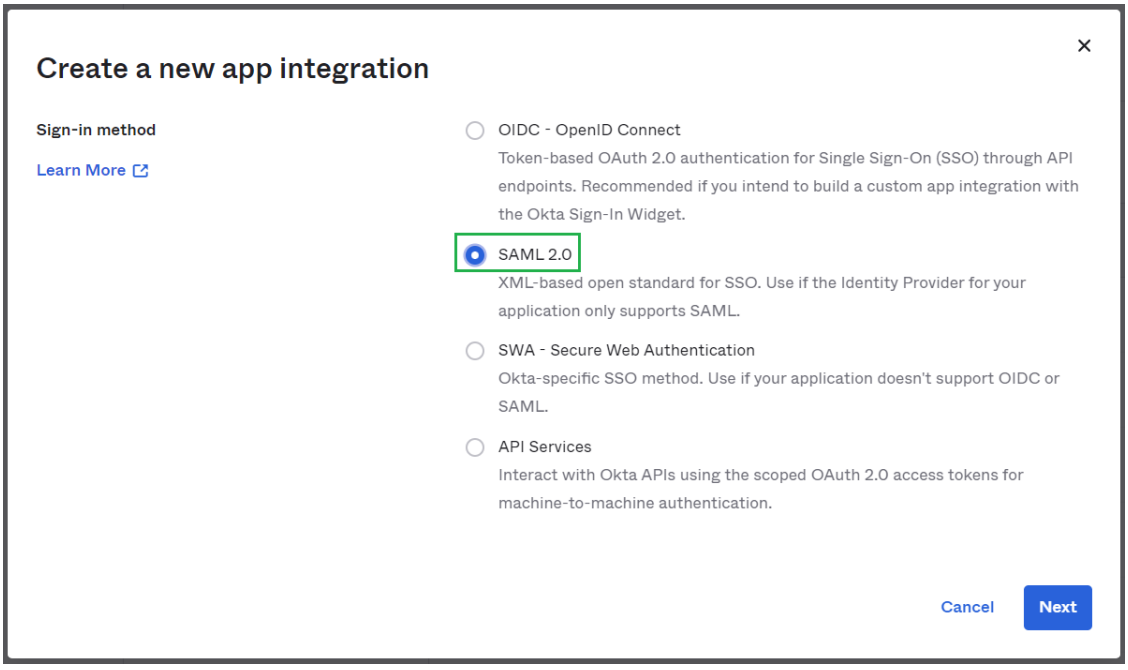
1. Sign in to MetaDefender Core with your account.
2. Select your SSO directory and click `Edit`.
3. Under `Identity Provider` , click `Fetch URL`.
4. Fill the metadata URL with SAML metadata link from Okta.
5. Click `Save Changes`.

Create Okta application

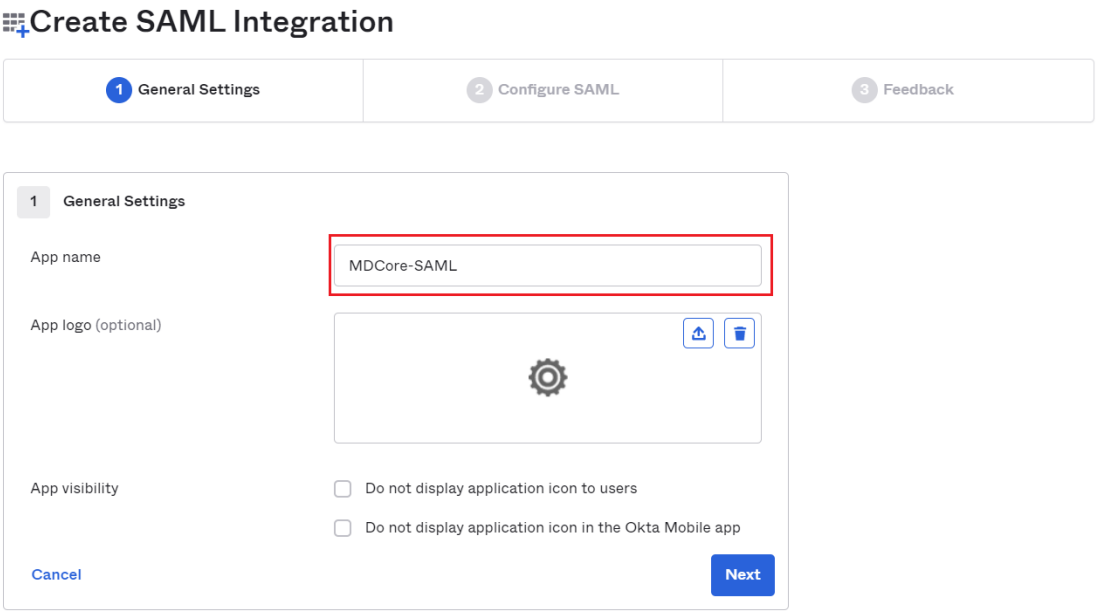
1. Access https://www.okta.com/login/ and sign in.
2. In sidebar of dashboard, hit `Applications` to drop sub items down, then select `Applications`.
3. Click `Create App Integration`.



4. In `Sign-in method`, choose `SAML 2.0` and click `Next`.

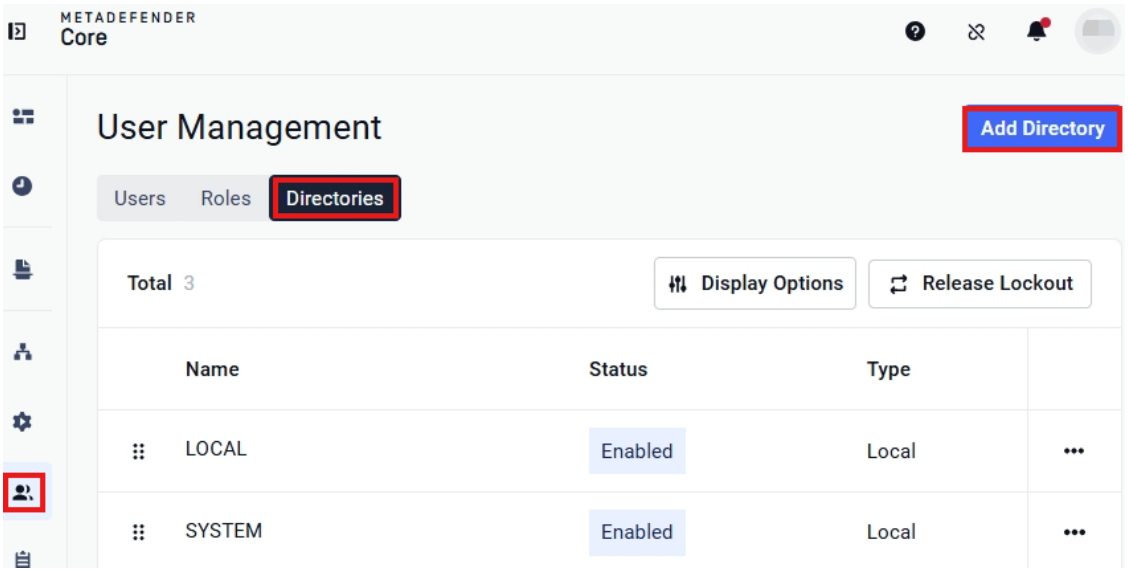


5. Fill `App name`, MDCore-SAML for example and click `Next`.

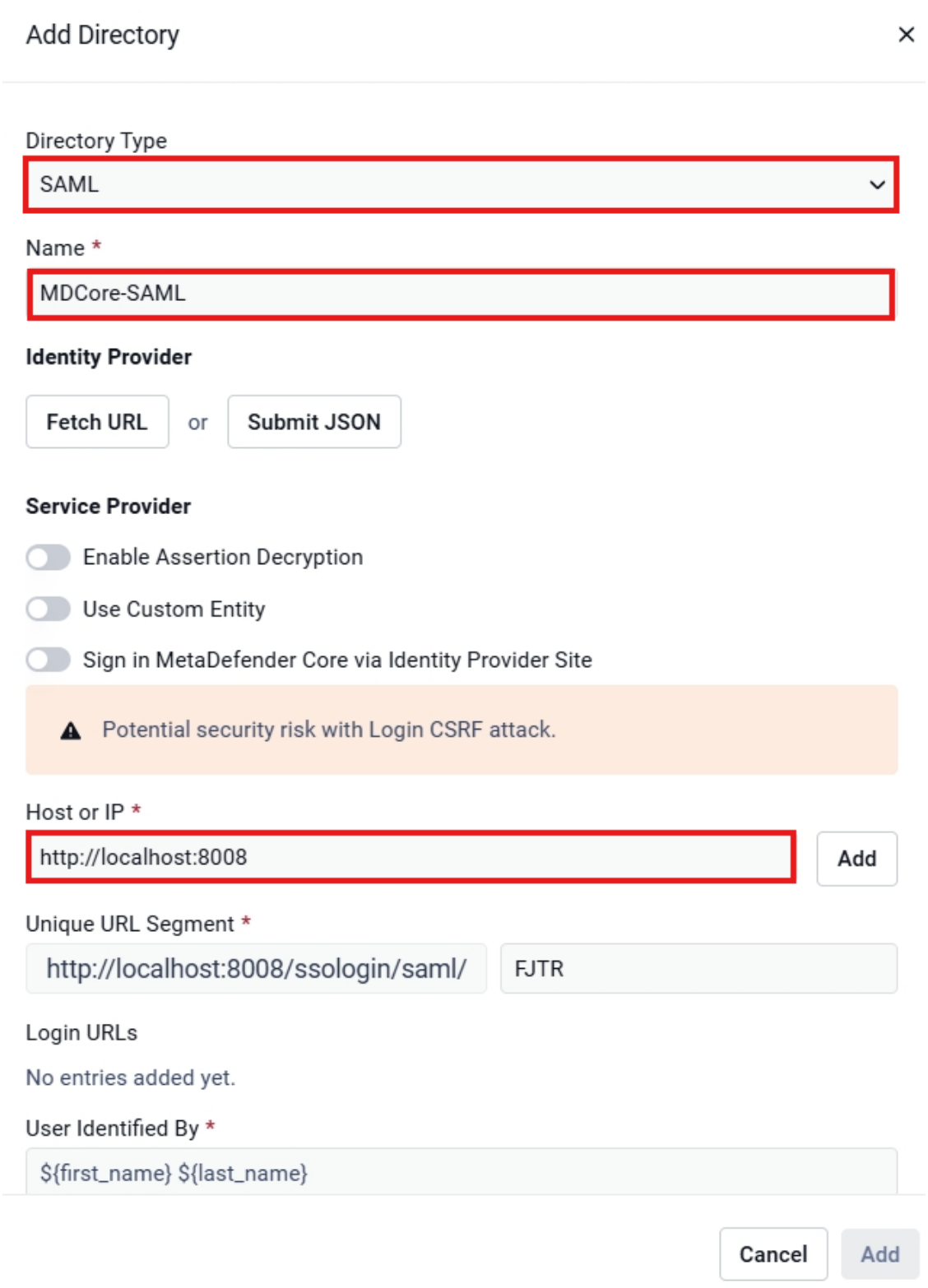


Create SAML directory in MetaDefender Core

1. Sign in to MetaDefender Core management console.
2. Under `Dashboard`, hit `User Management` in the left sidebar.
3. Under `User Management`, choose `Directories` tab and click `Add directory` in the top right.

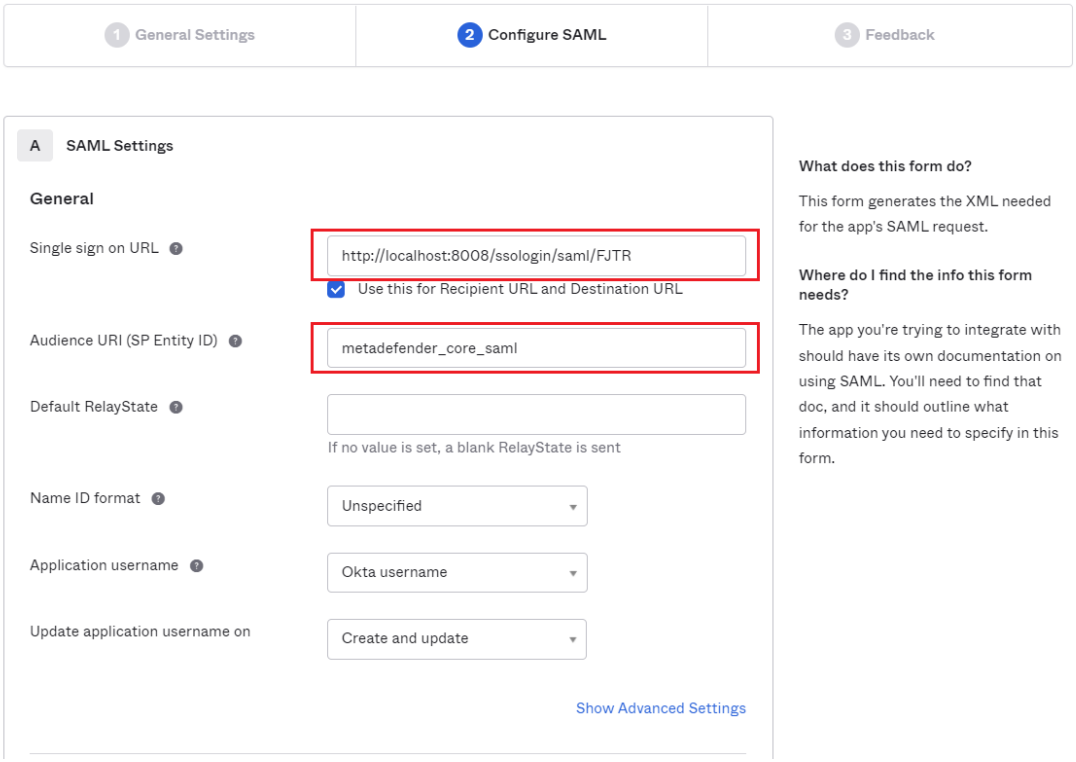


4. In `Add Directory` page, select SAML for `Directory Type`.
5. Fill `Name` of the new directory, MDCore-SAML for example.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, *http://localhost:8008* for this example.

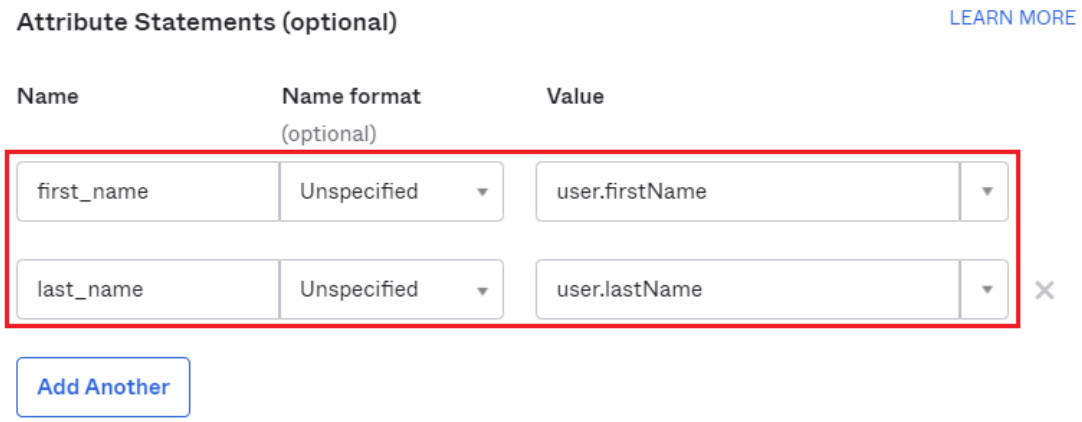


Configure Okta

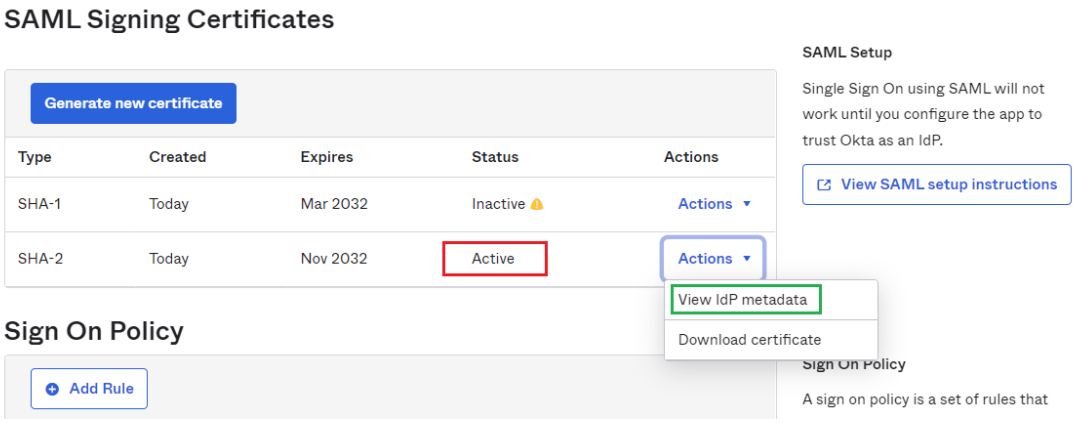
1. Switch to `Applications` screen in Okta home page.
2. Paste the `Login URL` from MetaDefender Core into `Single sign on URL` and set ID for `Audience URI`, metadefender_core-saml for example.



3. Scroll down, navigate to `Attribute Statements` and set mapping from Okta key name to the one set at `User identified by` in MetaDefender Core, then hit `Next`.



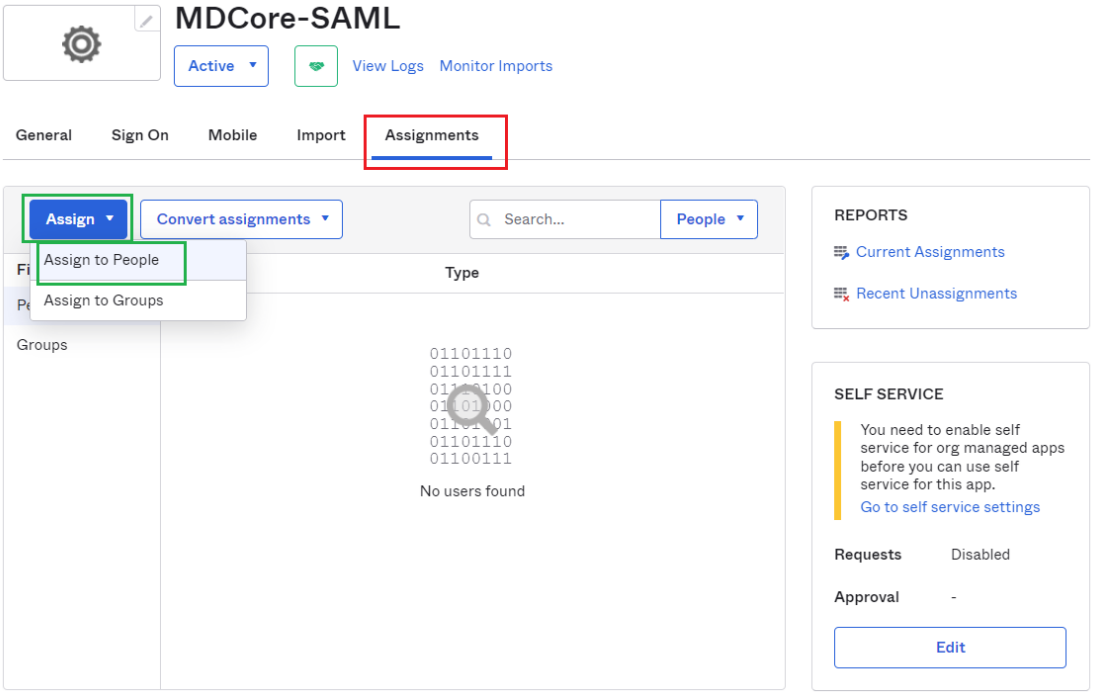
4. Choose your answer at `Feedback` screen and click `Finish`.
5. On the next screen, navigate to `SAML Signing Certificates`, choose the row that has value of column `Status` is "Active" then drop `Actions` down and click `View IdP metadata`.



6. Copy SAML Metadata link.

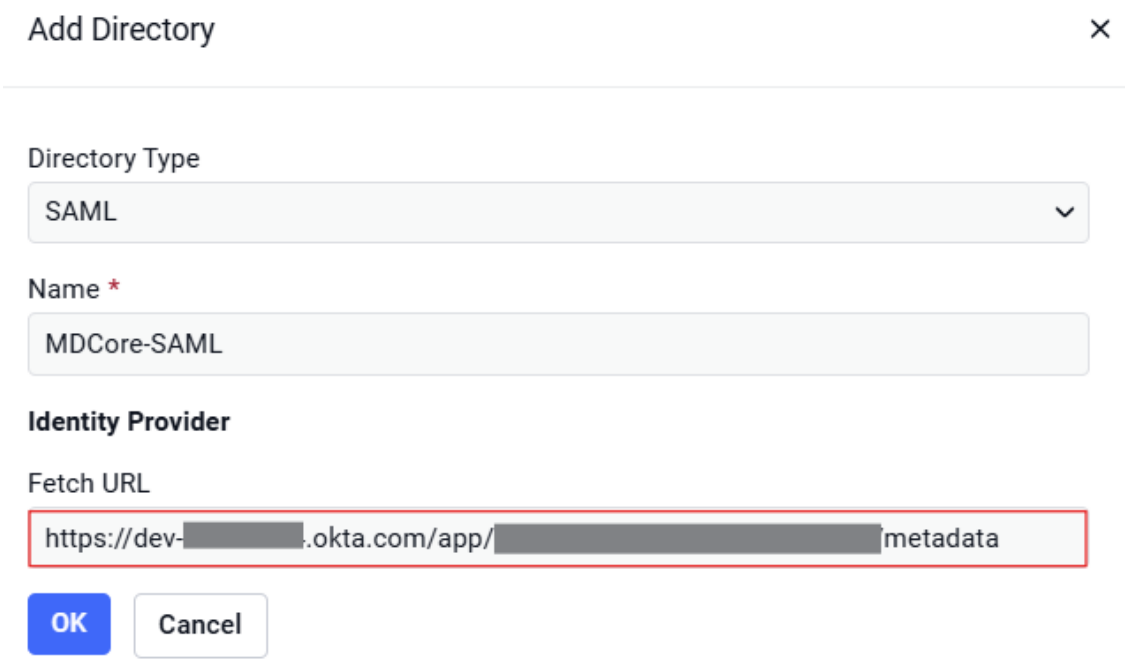


7. Switch to `Applications` screen in Okta, navigate to `Assignments` tab, and assign users for the application.

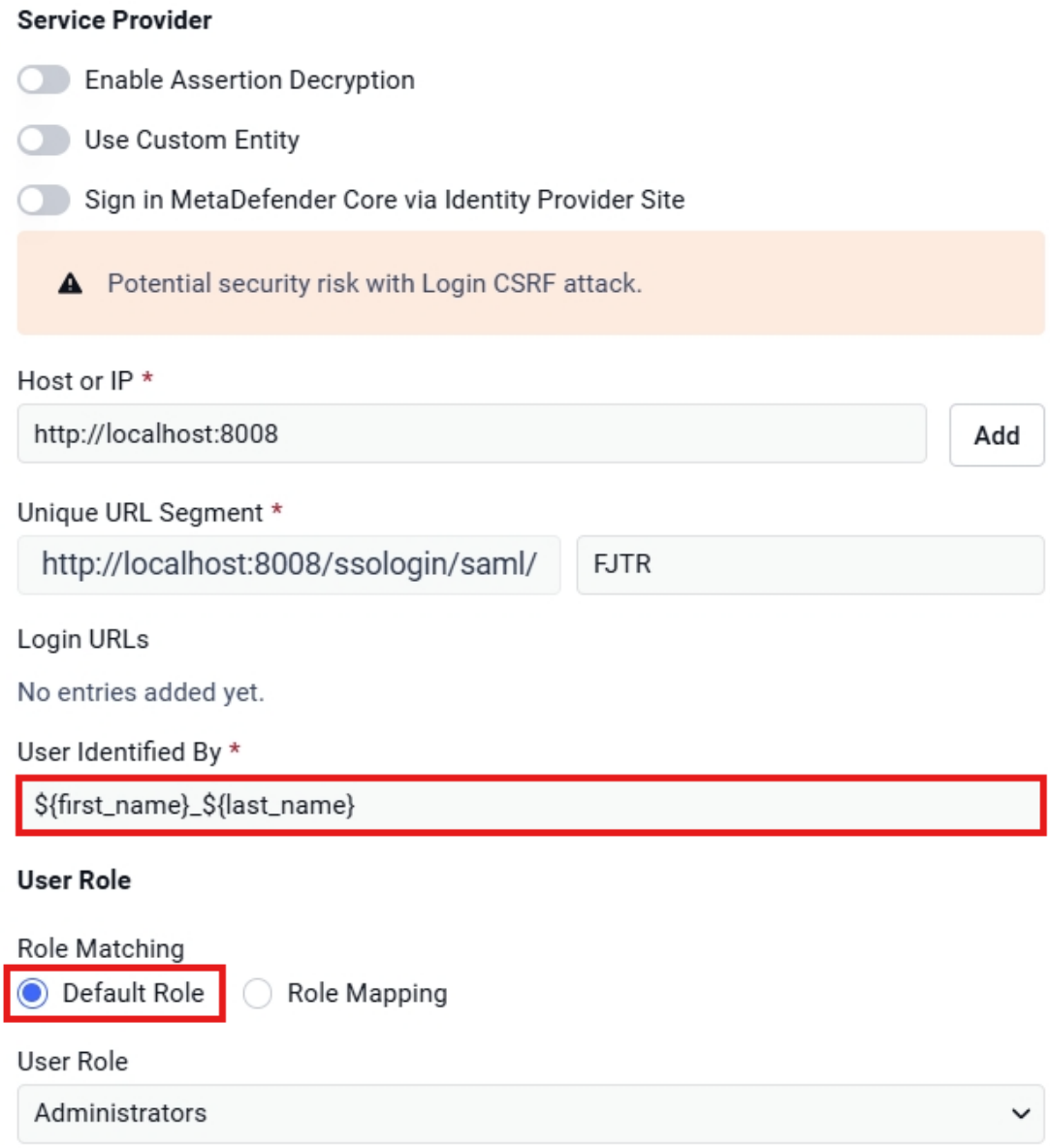


Complete configuration in MetaDefender Core

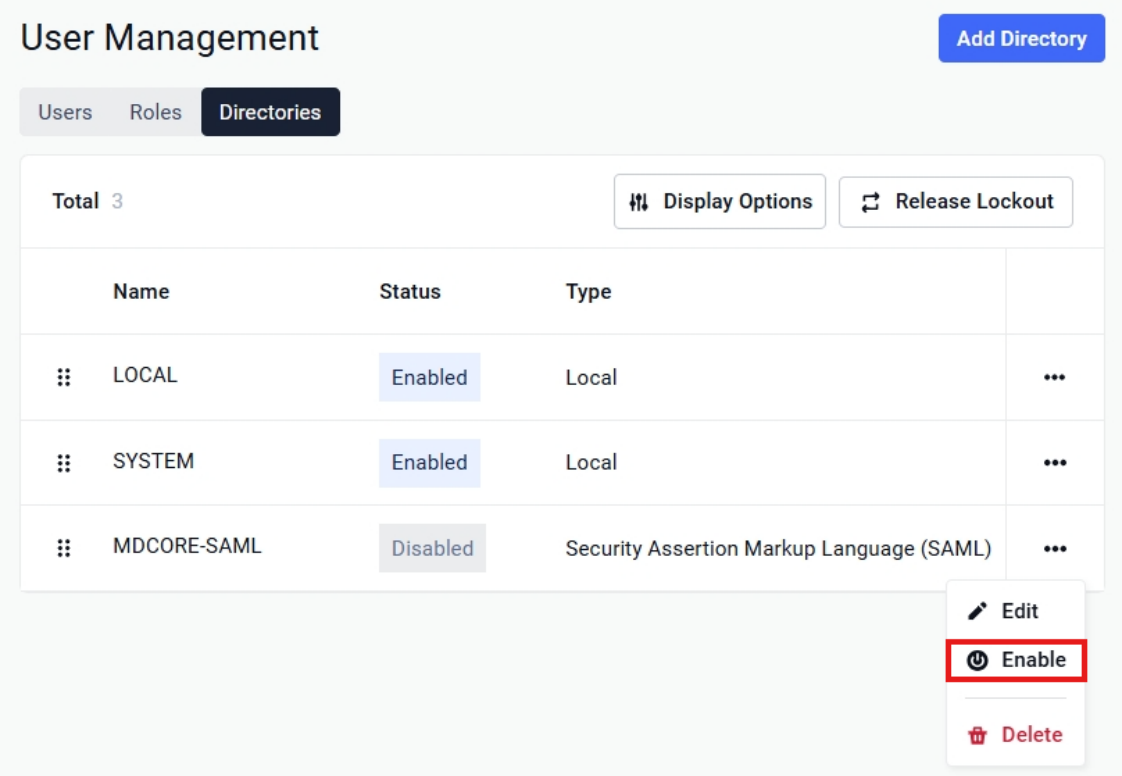
1. Switch to MetaDefender Core screen, under `Identity Provider`, click on `Fetch URL`.
2. Paste "SAML Metadata link" from Okta to the box under `Fetch URL` and click `OK` to ensure MetaDefender Core can set Okta as its IdP.



3. Under `Service Provider`, fill in user identity under `User identified by` with ${first_name}_${last_name} for example.
4. Select the appropriate role for the user under `User Role`.
5. Click `Add` to complete the settings.

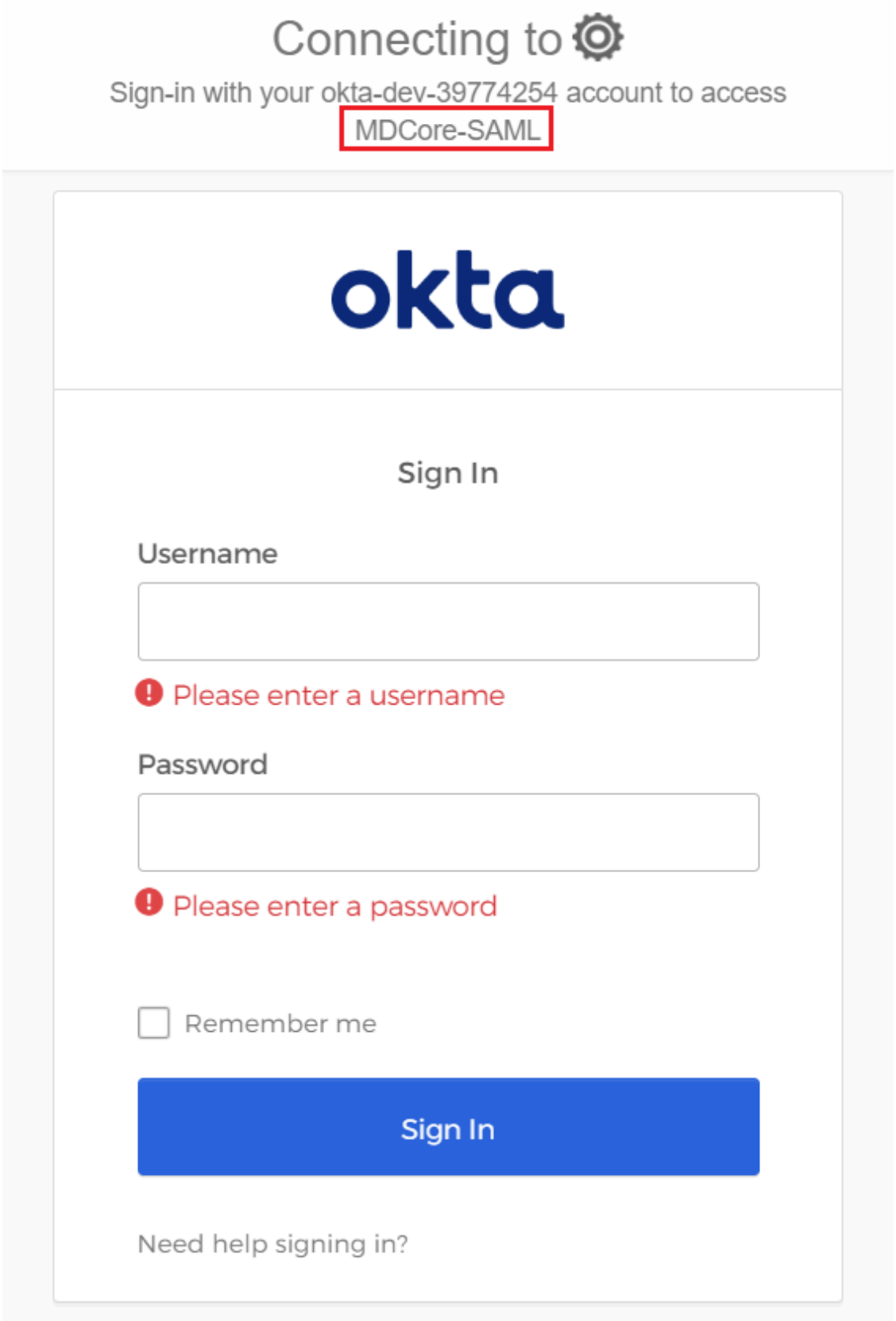


6. In `User Management` screen, toggle the new directory, MDCORE-SAML in this example. A dialog box is shown to confirm the action. Once `Enable` is hit, all existing sessions will be expired immediately and Okta will be used to authenticate users going forward.



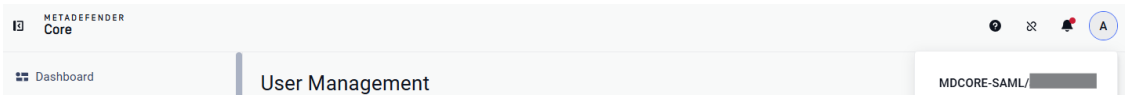
Test the integration

1. Browse MetaDefender Core, hit `Login`, user is redirected to Okta's sign-in page.



2. Sign in by the account registered in Okta.

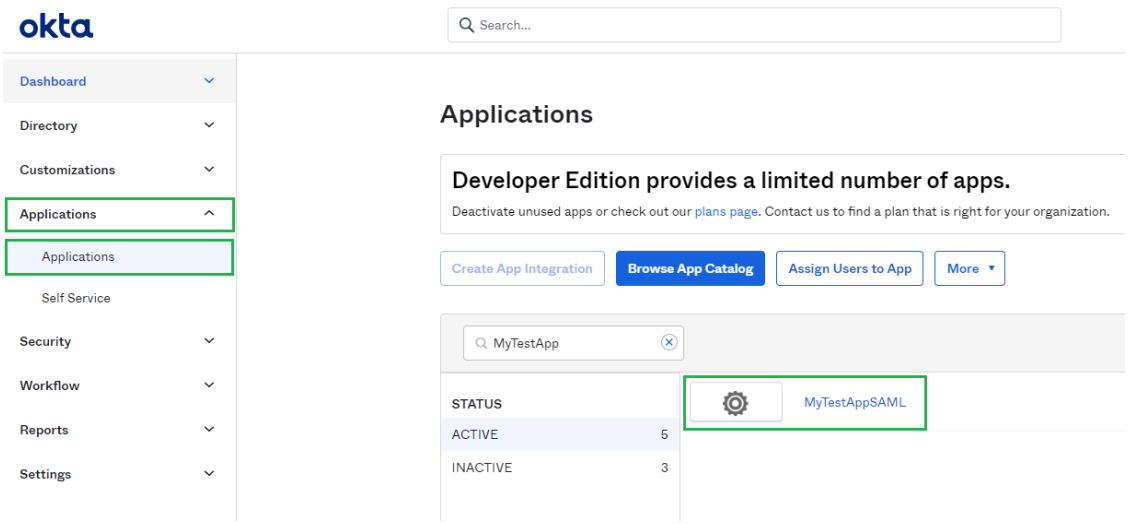
3. If everything goes right, MetaDefender Core dashboard is shown with user identity set at the top right corner.



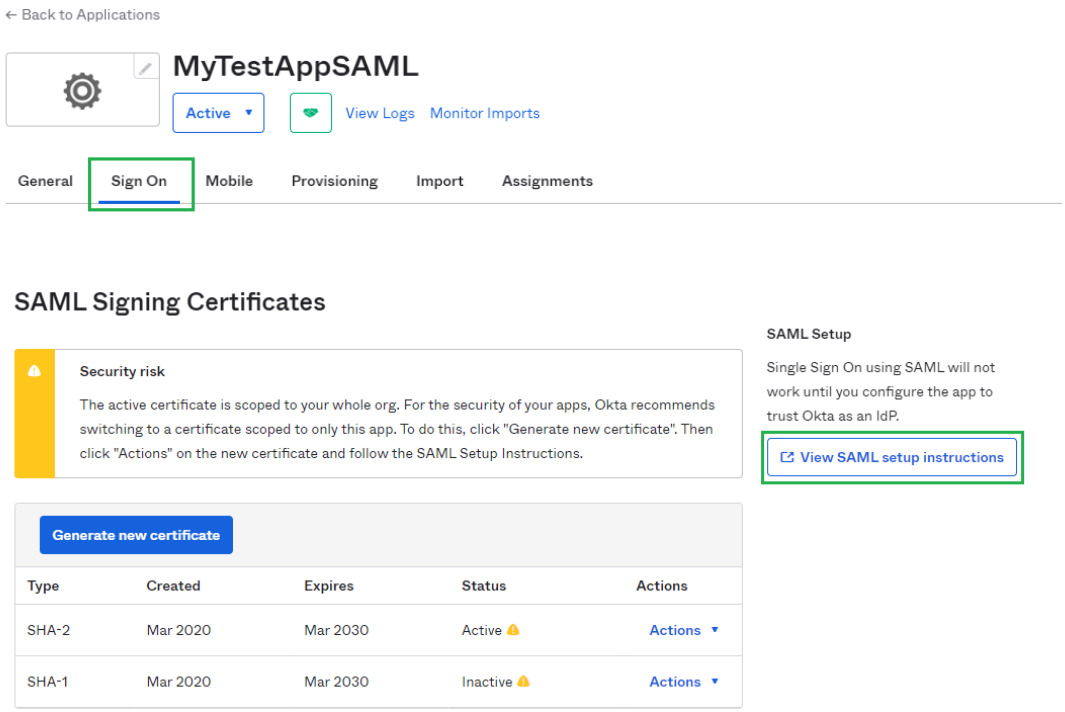
4. For troubleshooting, browse `<mdcore-host>#/public/backuplogin`.

Test IdP-initiated SSO

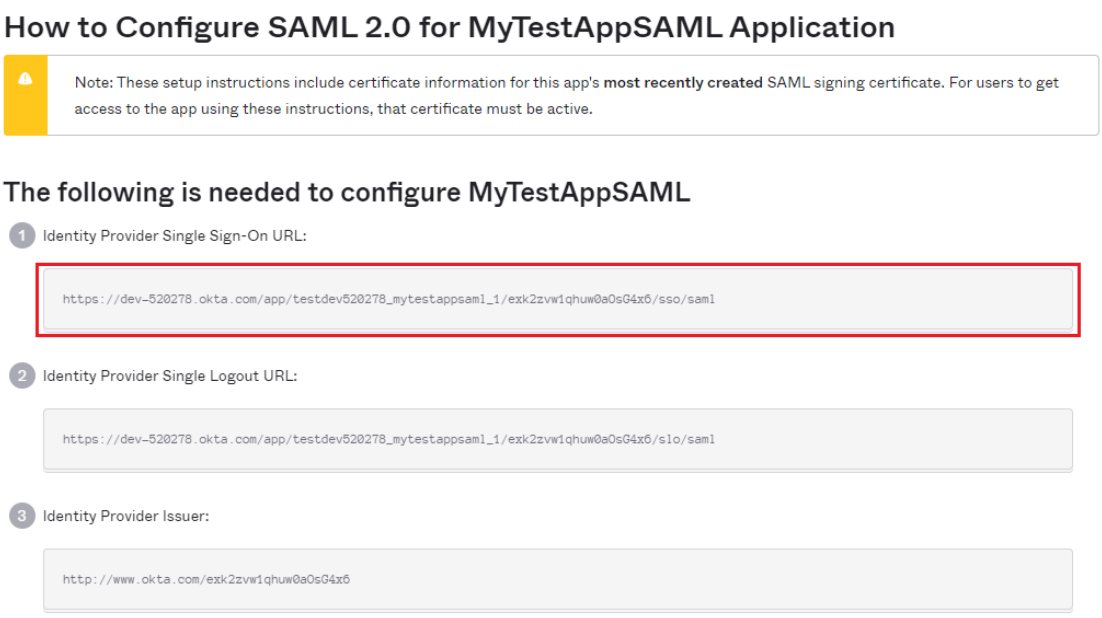
1. Sign in to Okta management page.
2. Navigate to `Applications`/ `Applications` on the sidebar.
3. Select the active SAML application.



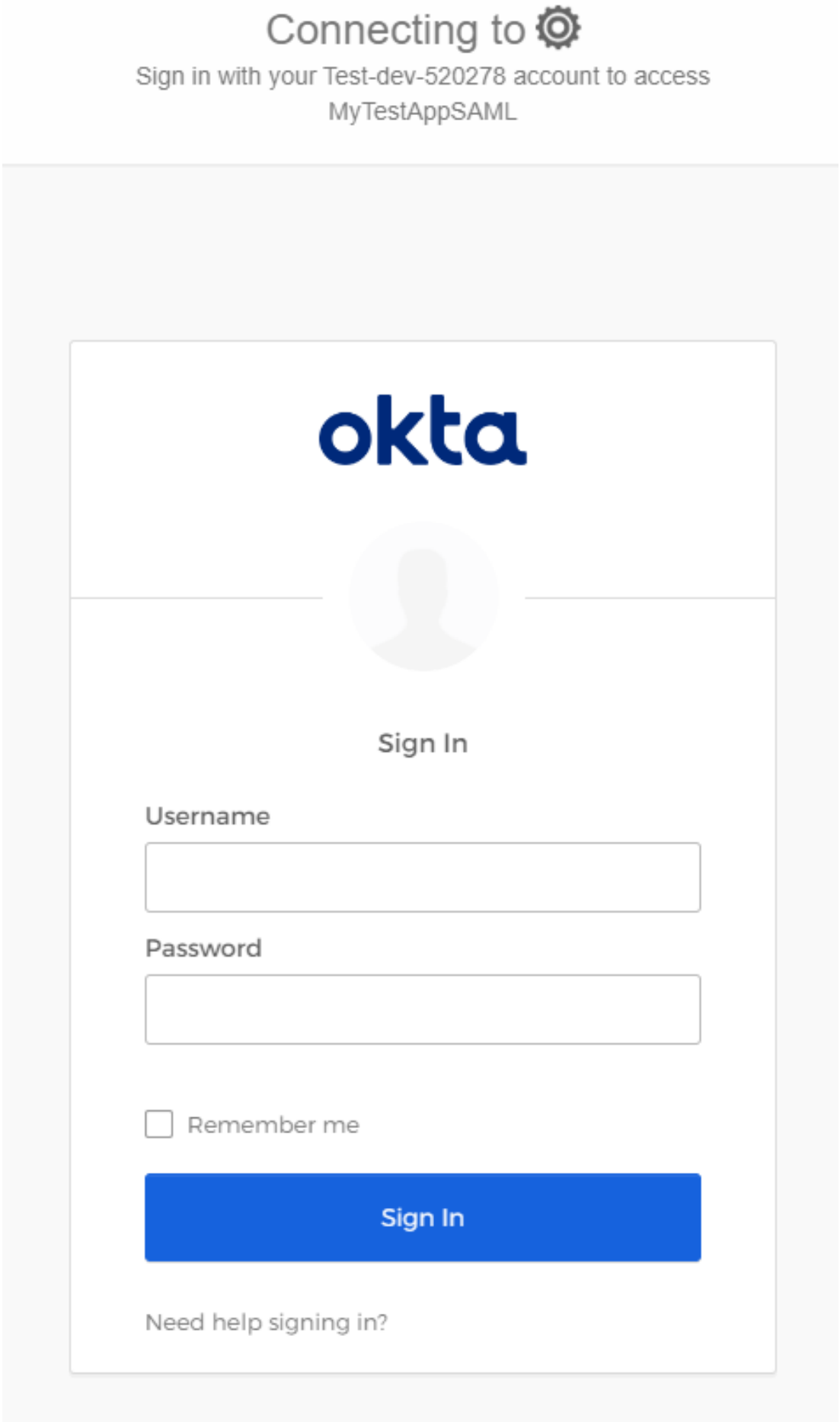
4. In `Sign On` tab, find and hit `View SAML setup instructions` button.



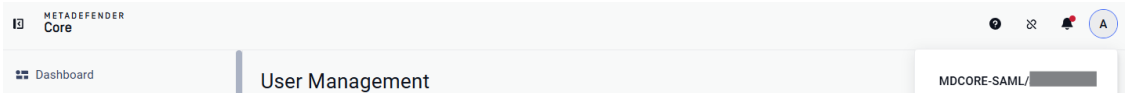
5. In the new page, copy the URL under `Identity Provider Single Sign-On URL` section.



6. Paste the copied URL to a web browser and sign in.


Connecting to
Sign in with your Test-dev-520278 account to access
MyTestAppSAML
okta
Sign In
Username
Password
Remember me
Sign In
Need help signing in?

3. If everything goes right, MetaDefender Core dashboard is shown with user identity set at the top right corner.



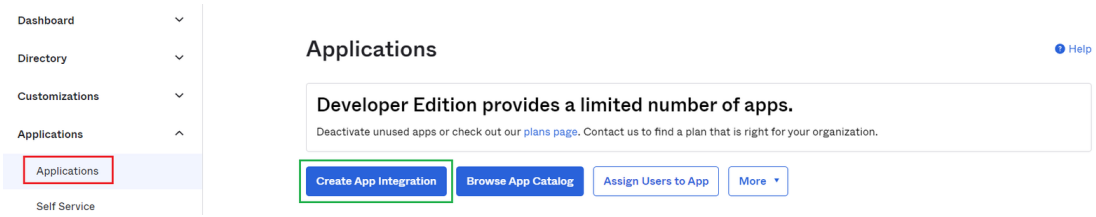
4. For troubleshooting, browse `<mdcore-host>#/public/backuplogin`.

Configure OIDC SSO

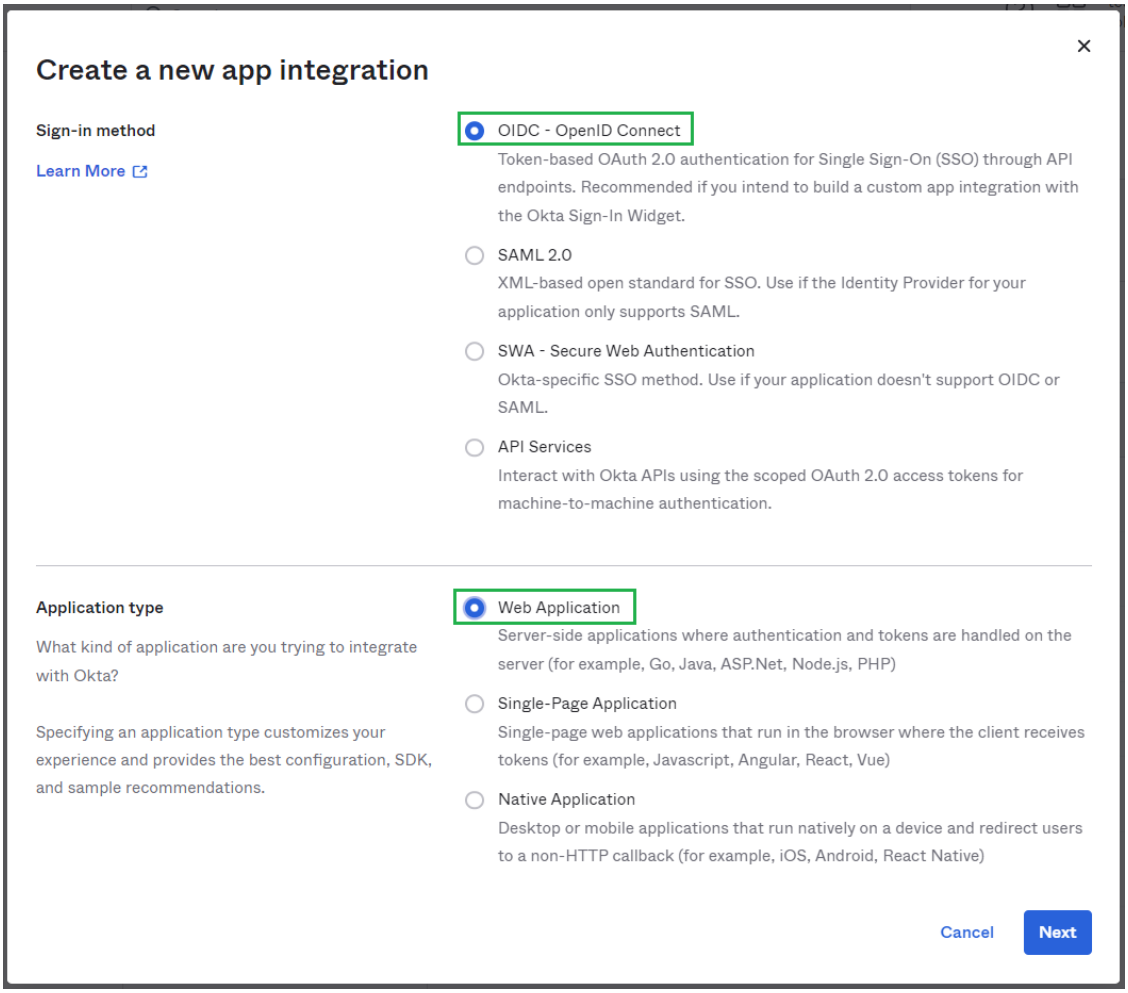
> **Warning**
>
> After the integration is successfully set up, any modifications on Okta application could lead to the regeneration of the client secret, causing Okta to reject requests from MetaDefender Core due to a mismatch in the client secret. In this instance, the customer should copy the new secret from Okta and update `Client secret` field within `Service Provider` of MetaDefender Core with the new one.

Create Okta application

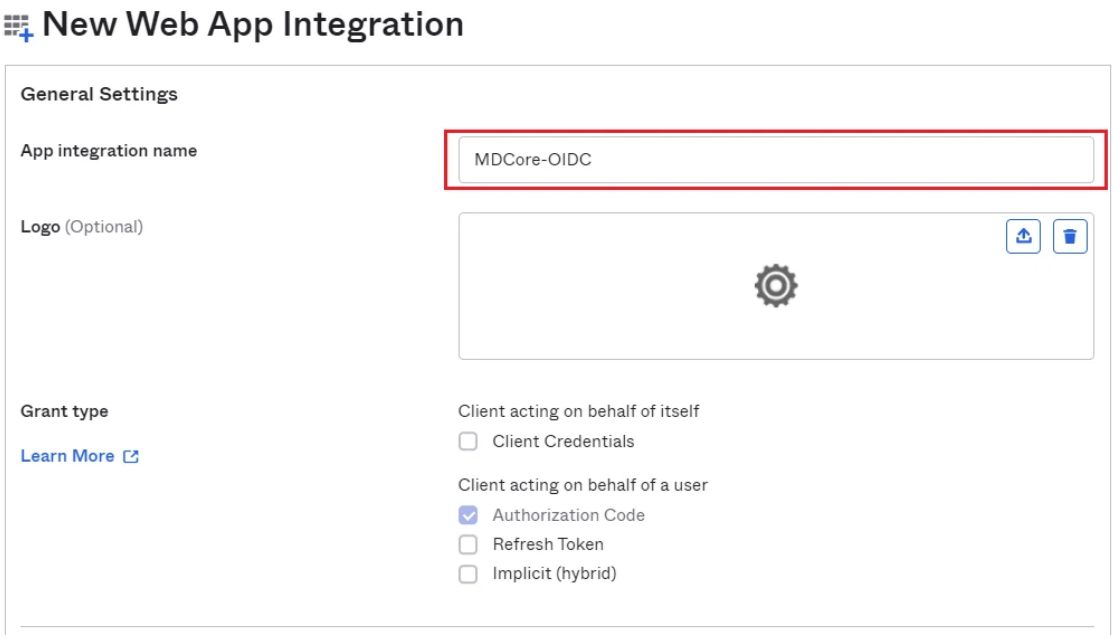
1. Access https://www.okta.com/login/ and sign in.
2. In the sidebar of dashboard, hit `Applications` to drop sub items down, then select `Applications`.
3. Click `Create App Integration`.



4. In `Sign-in method`, select `OIDC - OpenID Connect`. In `Application type` choose `Web Application` and click `Next`.



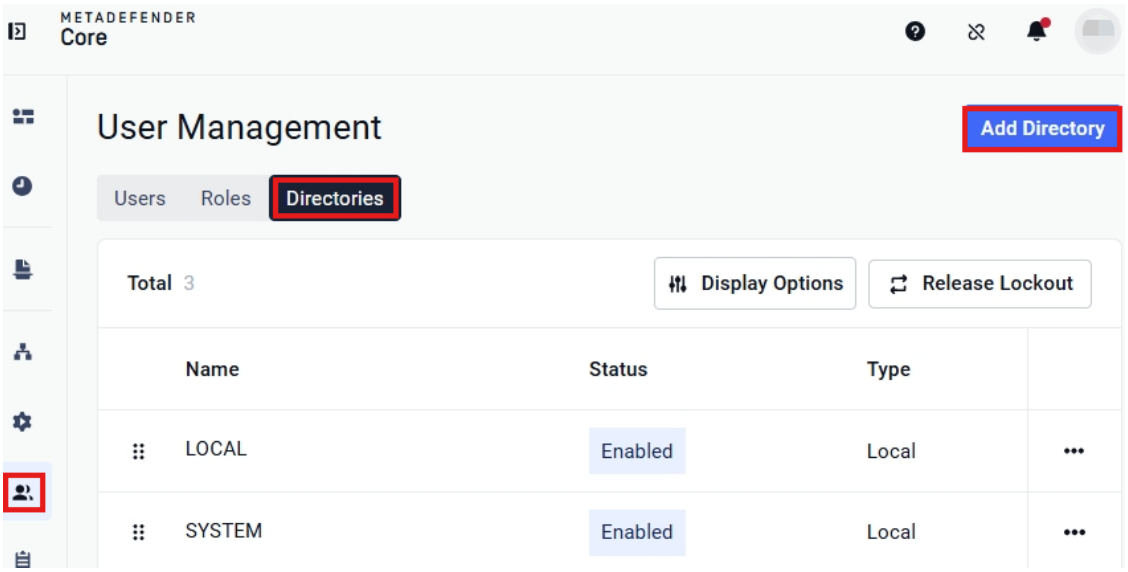
5. Fill `App name`, MDCore-OIDC for example and click `Next`.



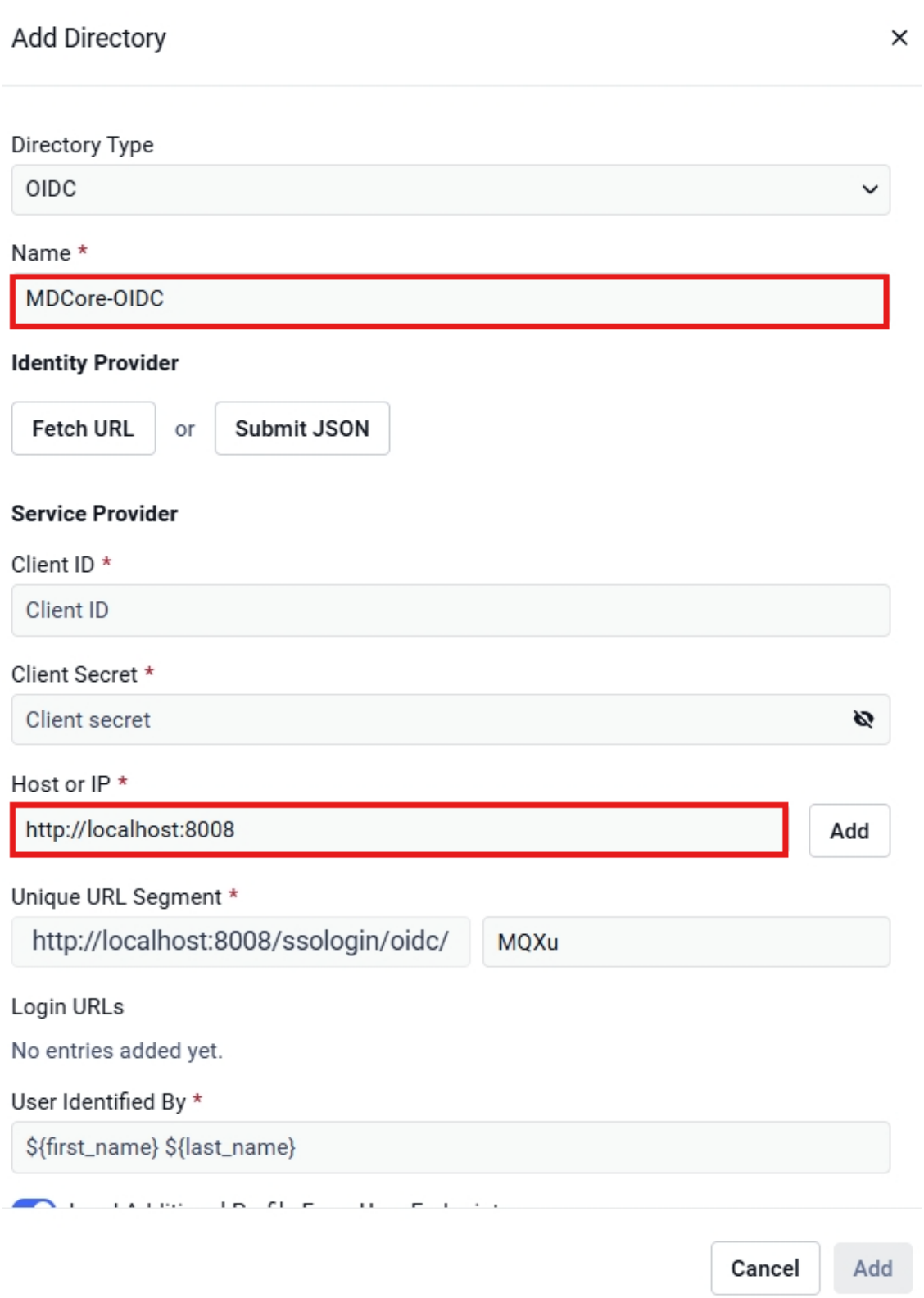
Create OIDC directory in MetaDefender Core

1. Sign in to MetaDefender Core management console.

2. Under `Dashboard`, hit `User Management` in the sidebar.
3. Under `User Management`, select `Directories` tab and click `Add directory` in the top right.

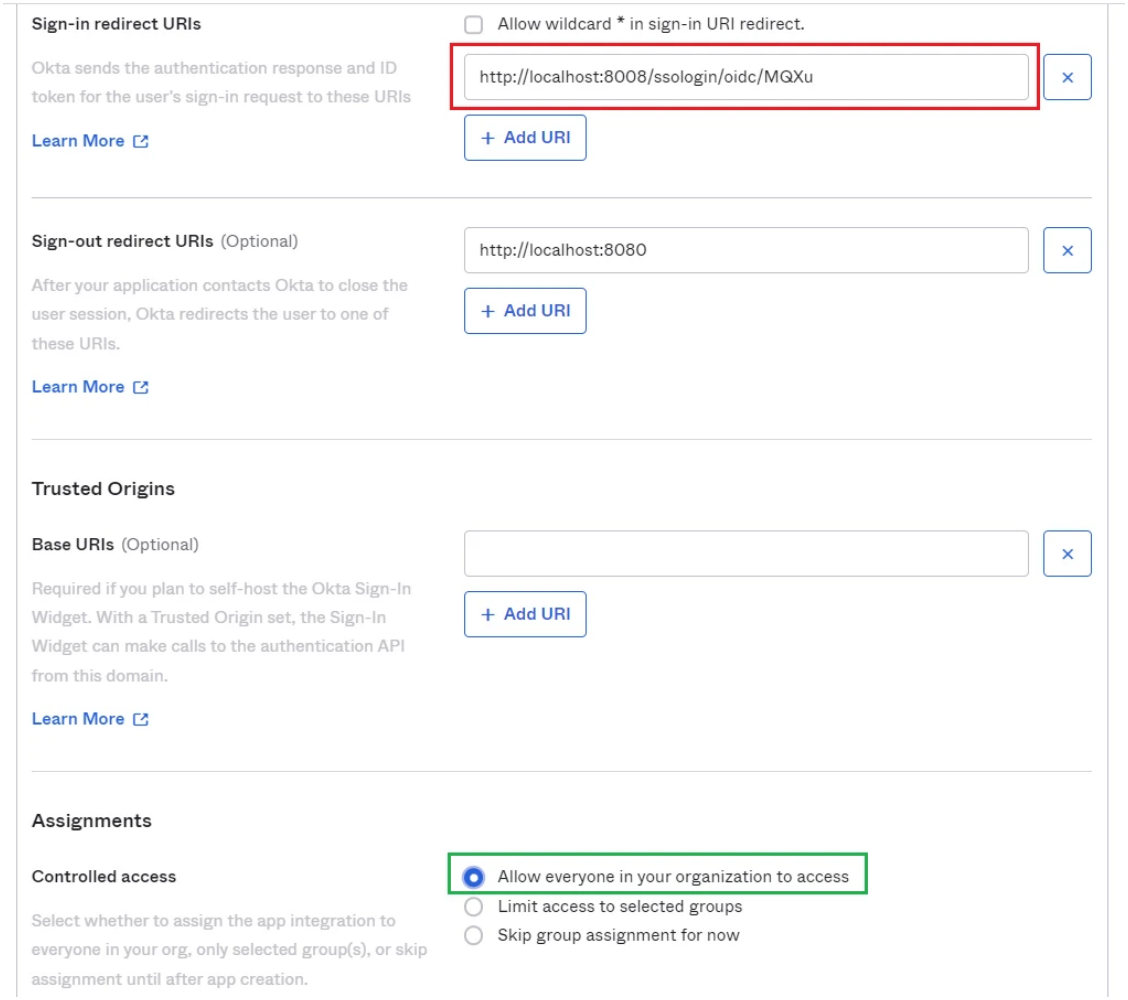


4. In `Add Directory` page, choose OIDC for `Directory Type`.
5. Fill `Name` of the new directory, MDCore-OIDC for example.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, *http://localhost:8008* for this example.

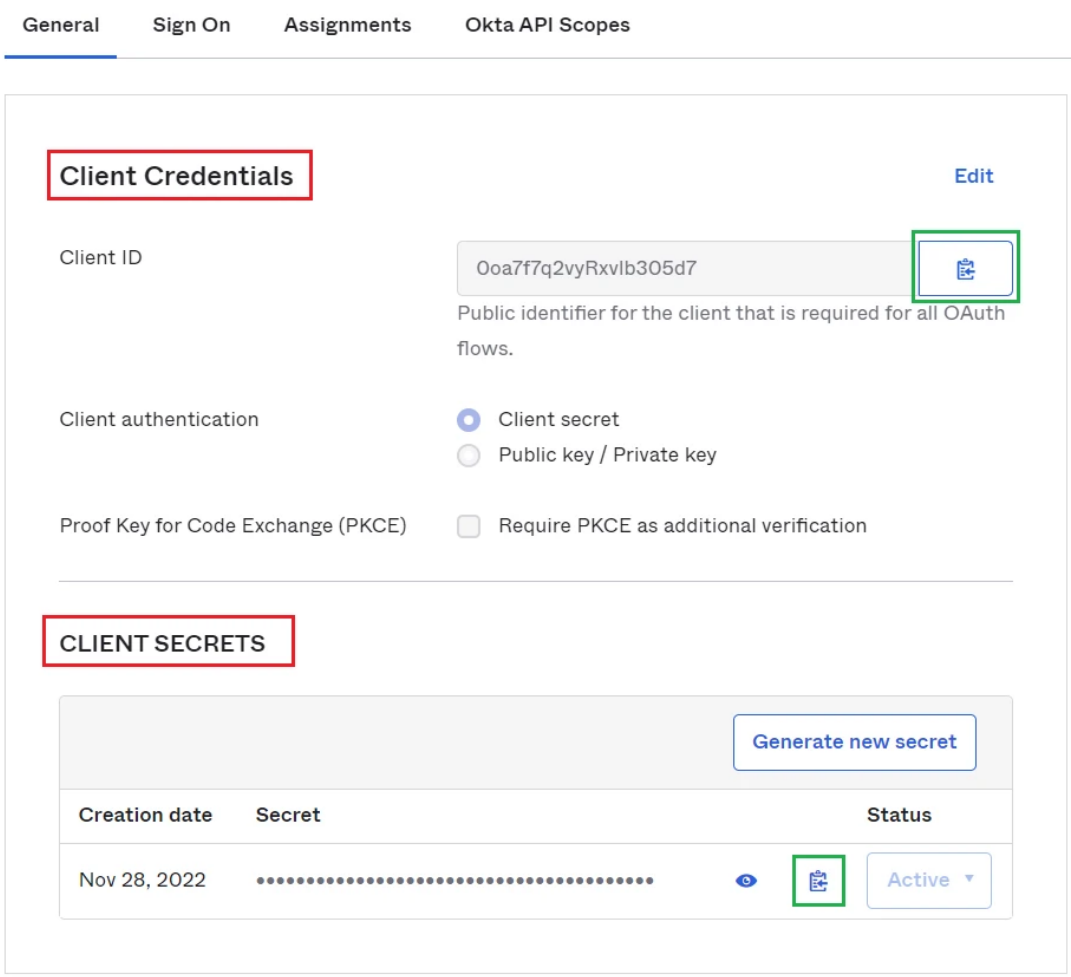


Complete configuration in Okta

1. Switch to `Applications` screen in Okta home page.
2. Paste the `Login URL` from MetaDefender Core into `Sign-in redirect URIs` and grant permission for everyone to access our application in `Assignments` then click `Save`.

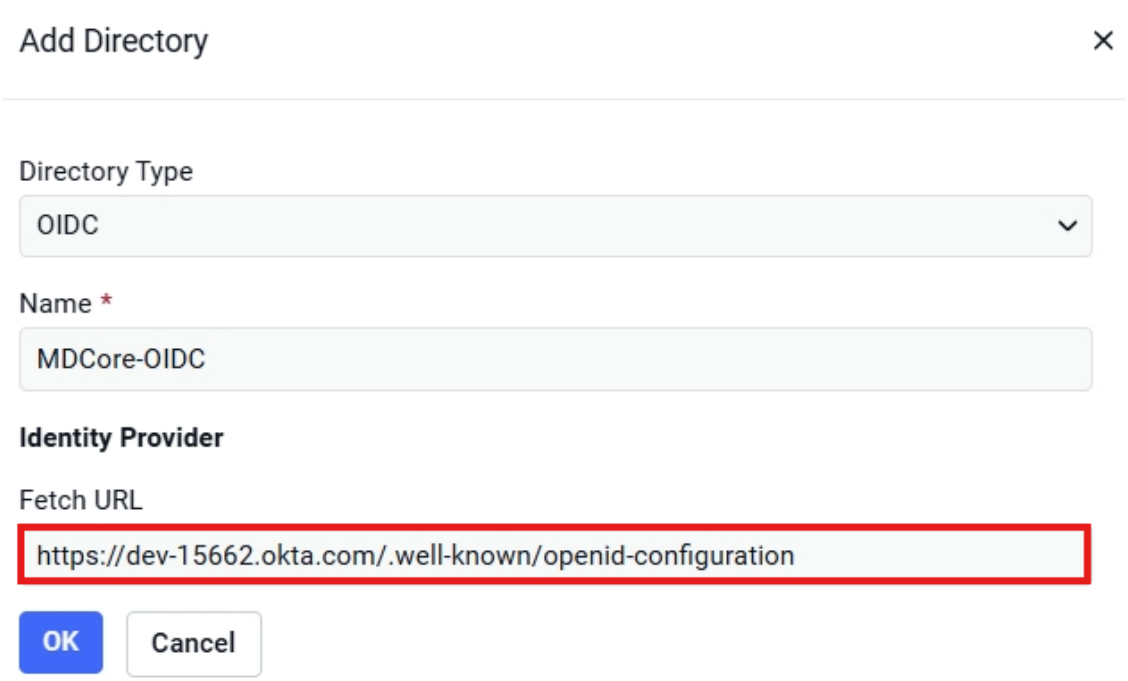


3. In `General` tab, copy `Client ID` and `Client Secret`.



Complete configuration in MetaDefender Core

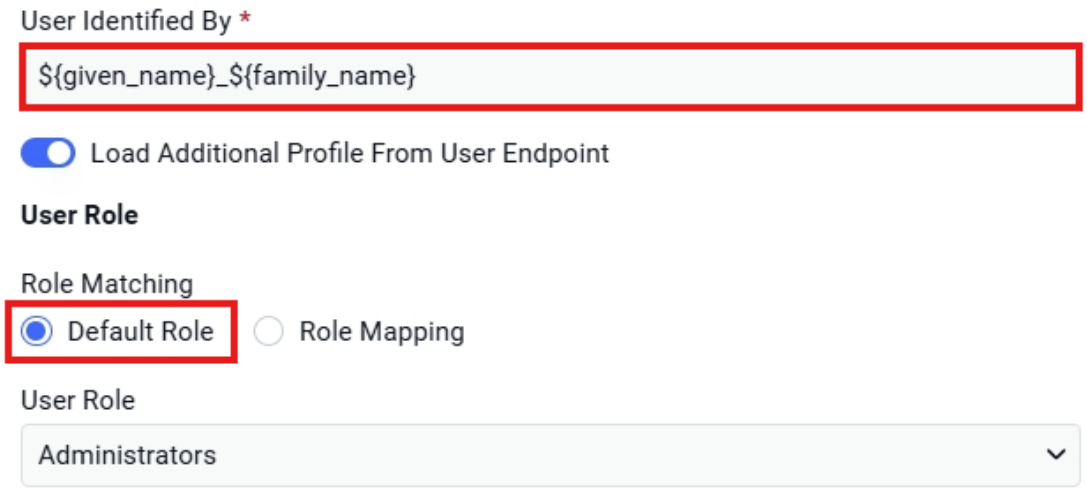
1. Switch to MetaDefender Core screen, under `Identity Provider`, click on `Fetch URL`.
2. OIDC link will be built by appending */.well-known/openid-configuration* to your Okta domain url (Referred Okta document). For example `dev-115662.okta.com/.well-known/openid-configuration`.
3. Paste OIDC configuration link above to the box under `Fetch URL` and hit `OK` to ensure MetaDefender Core can set Okta as its IdP.



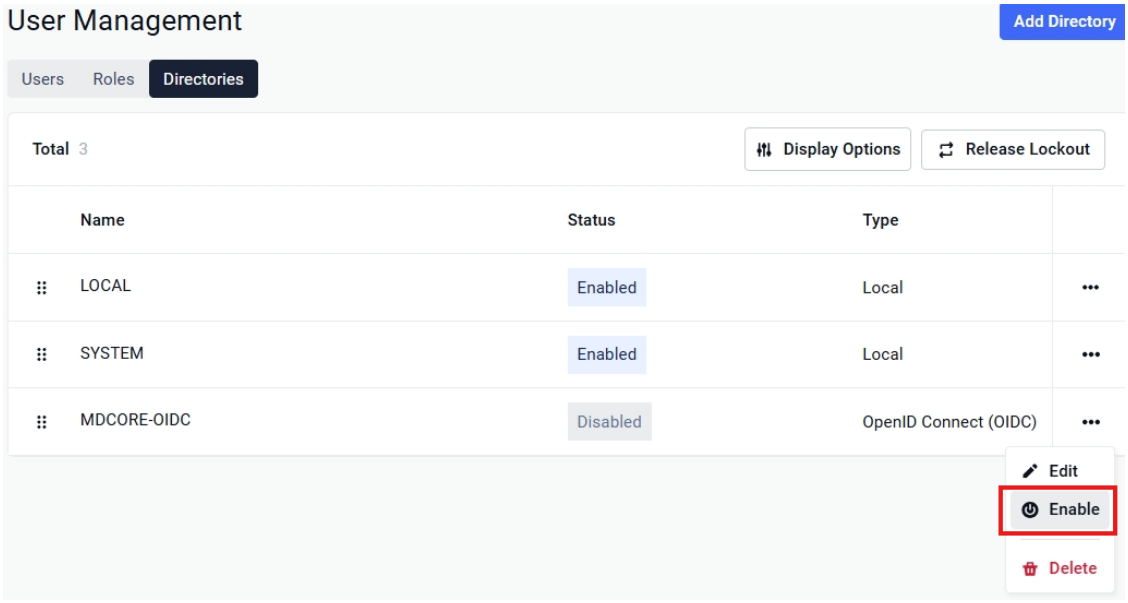
4. Under `Service Provider`, paste the value of `Client ID` and `Client secret` that you copied from Okta respectively.



4. Under `Service Provider`, fill in `User identified by` with ${given_name}_${family_name} for example.
5. Select appropriate role for the user under `User Role`.
6. Click `Add` to complete the settings.

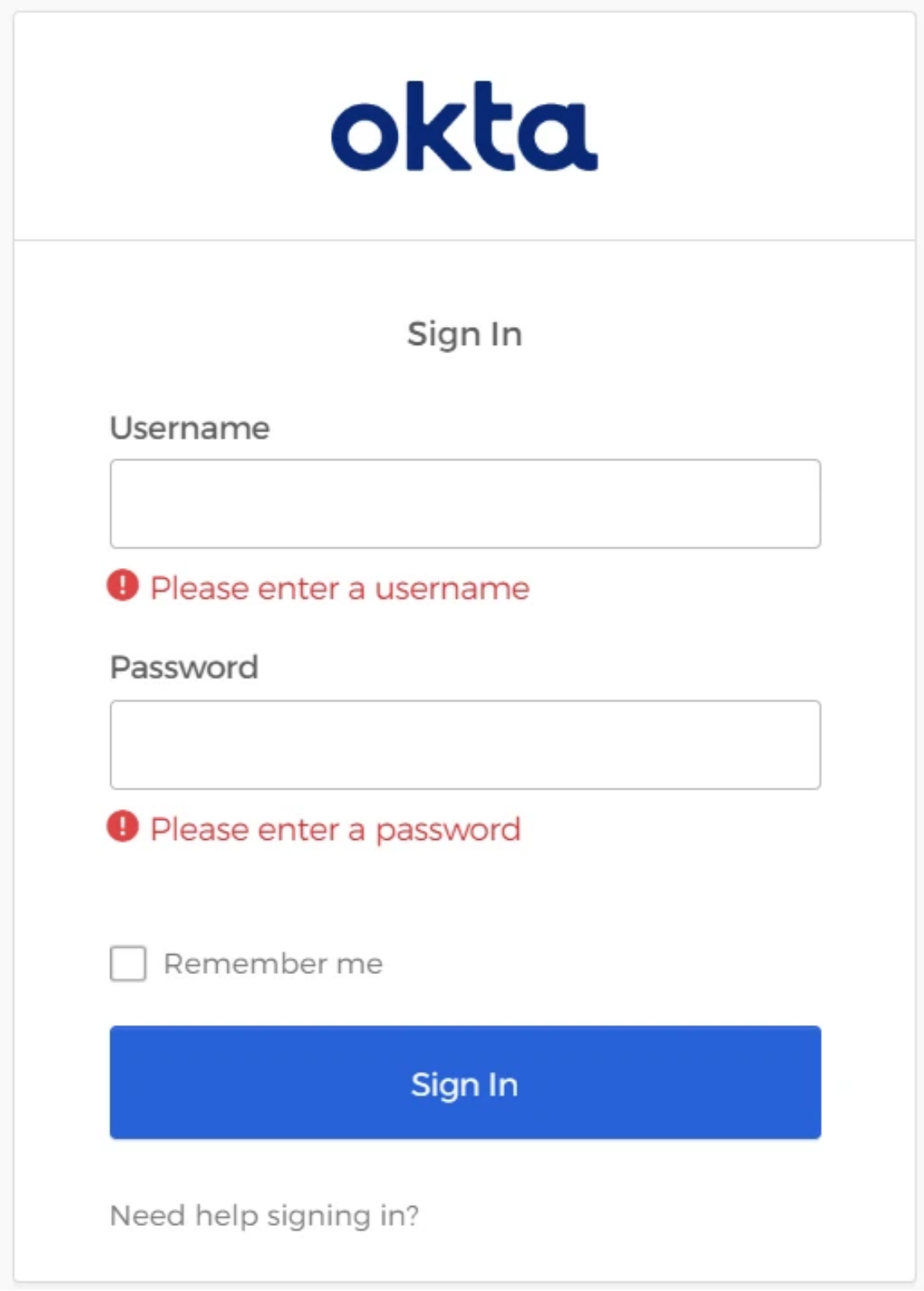


7. On `User Management` screen, toggle the new directory, MDCORE-OIDC in this example. A dialog box is shown to confirm the action. Once `Enable` is hit, all existing sessions will be expired immediately and Okta will be used to authenticate users going forward.



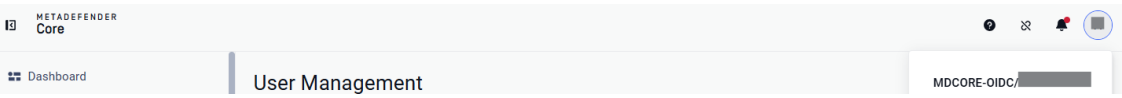
Test the integration

1. Browse MetaDefender Core, hit `Login`, user is redirected to Okta's sign-in page.



2. Sign in by the account registered in Okta.

3. If everything goes right, MetaDefender Core dashboard is shown with user identity set at the top right corner.



4. For troubleshooting, browse `<mdcore-host>#/public/backuplogin`.

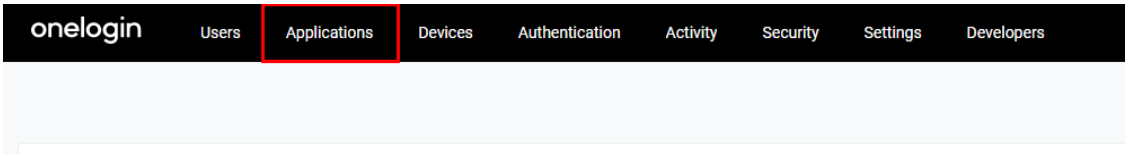
OneLogin

The instructions guide users on how to configure OneLogin SAML and OIDC Single Sign-on for MetaDefender Core.

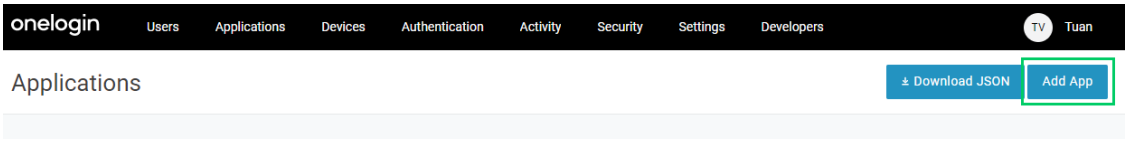
Configure SAML SSO

Create OneLogin application

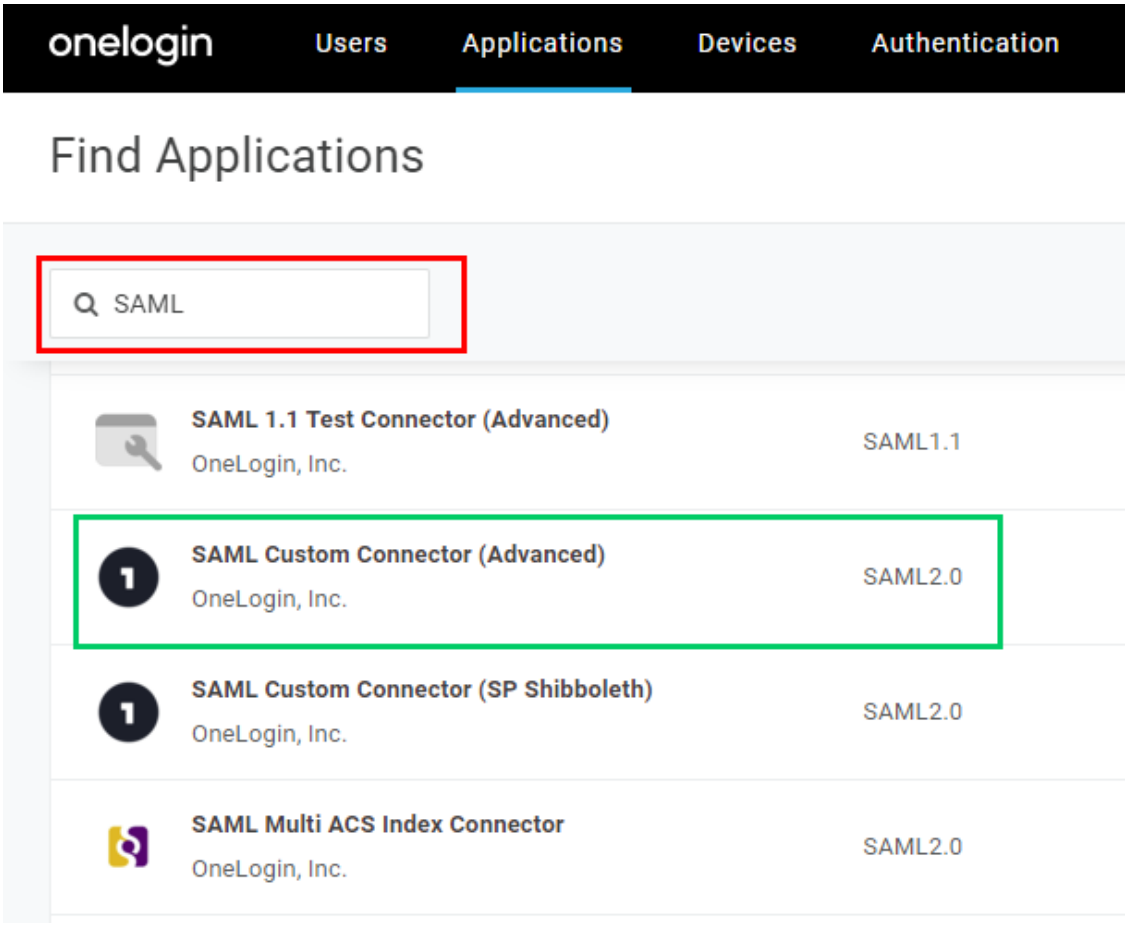
1. Access OneLogin and sign in.
2. On the navigation bar, click `Administration` to access administrator mode, then click on `Applications` to add a new application for integration.



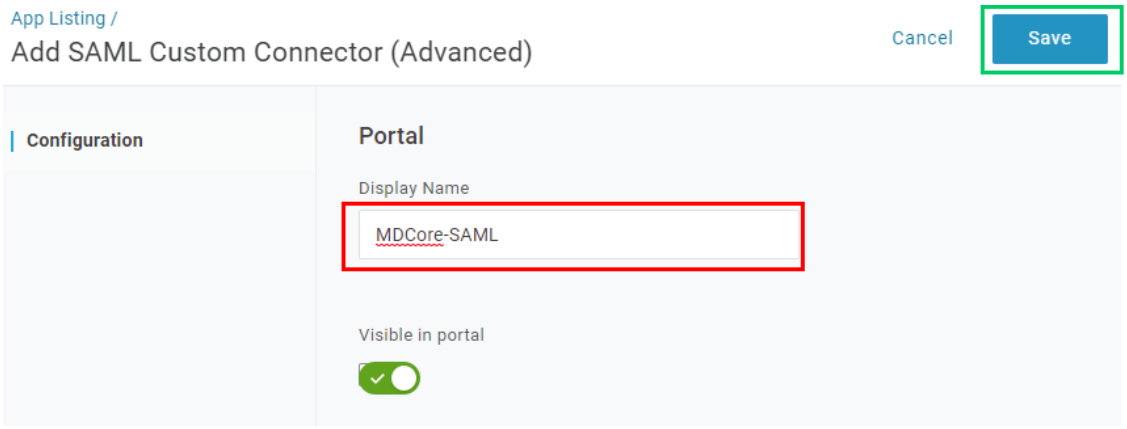
3. Click `Add App` to add a new application.



4. Enter "SAML" in the search box and select `SAML Custom Connector (Advanced)` version SAML2.0 in the results panel.

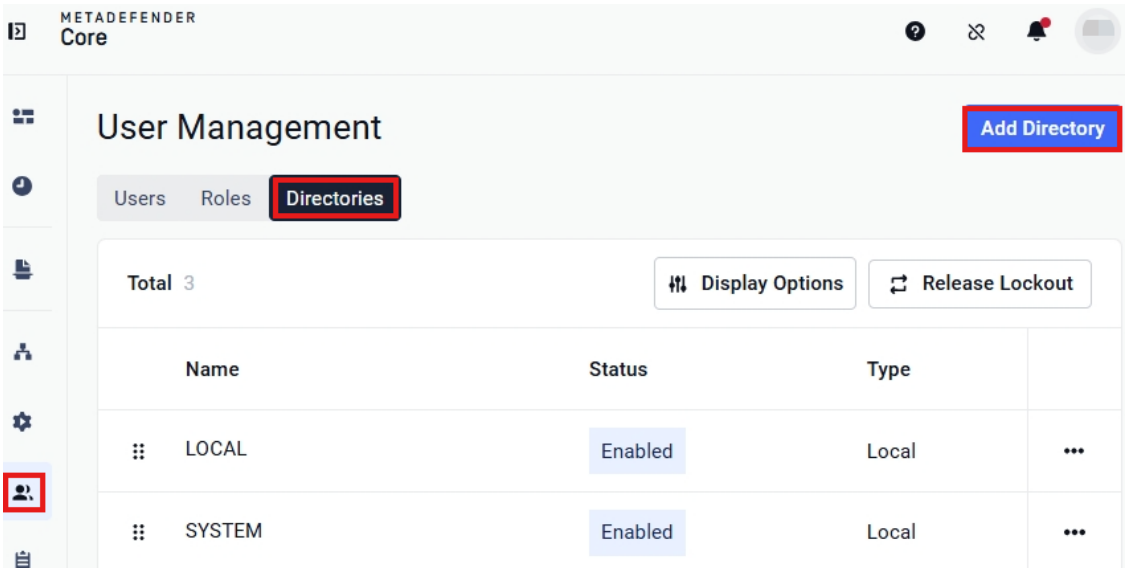


5. In the new page, fill in `Display Name` with MDCore-SAML, for example, and click `Save`.

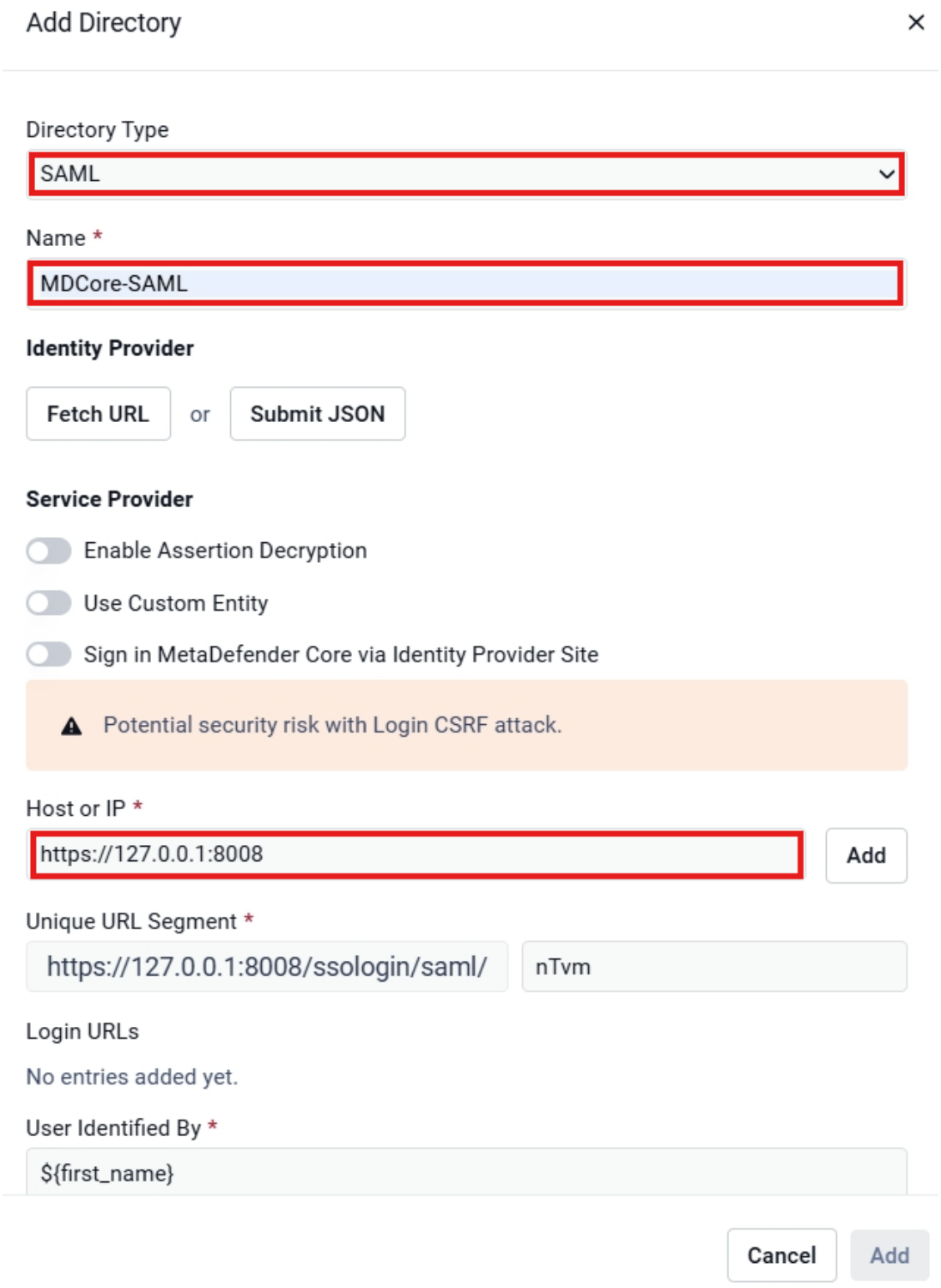


Create SAML directory in MetaDefender Core

1. Sign in to MetaDefender Core.
2. In the dashboard, click on `User Management` in the left sidebar.
3. On `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.

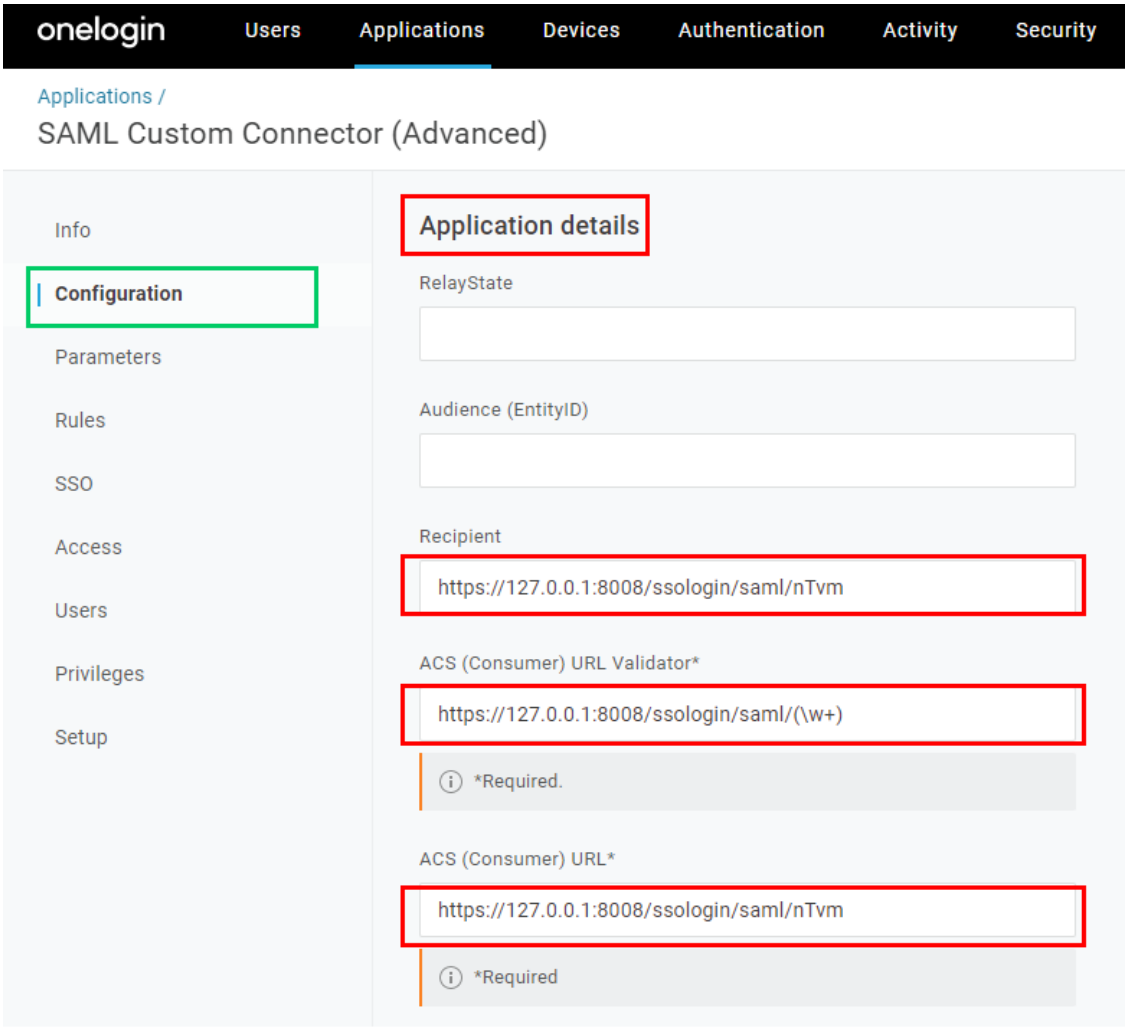


4. In `Add Directory` page, select SAML as `Directory type`, and enter a name for the new directory, such as MDCore-SAML.
5. In `Service Provider` section, enter `Host or IP` where MetaDefender Core is hosted, for this example use https://127.0.0.1:8008.
6. Copy the string generated under `Login URL` and store it as *reply_uri*.

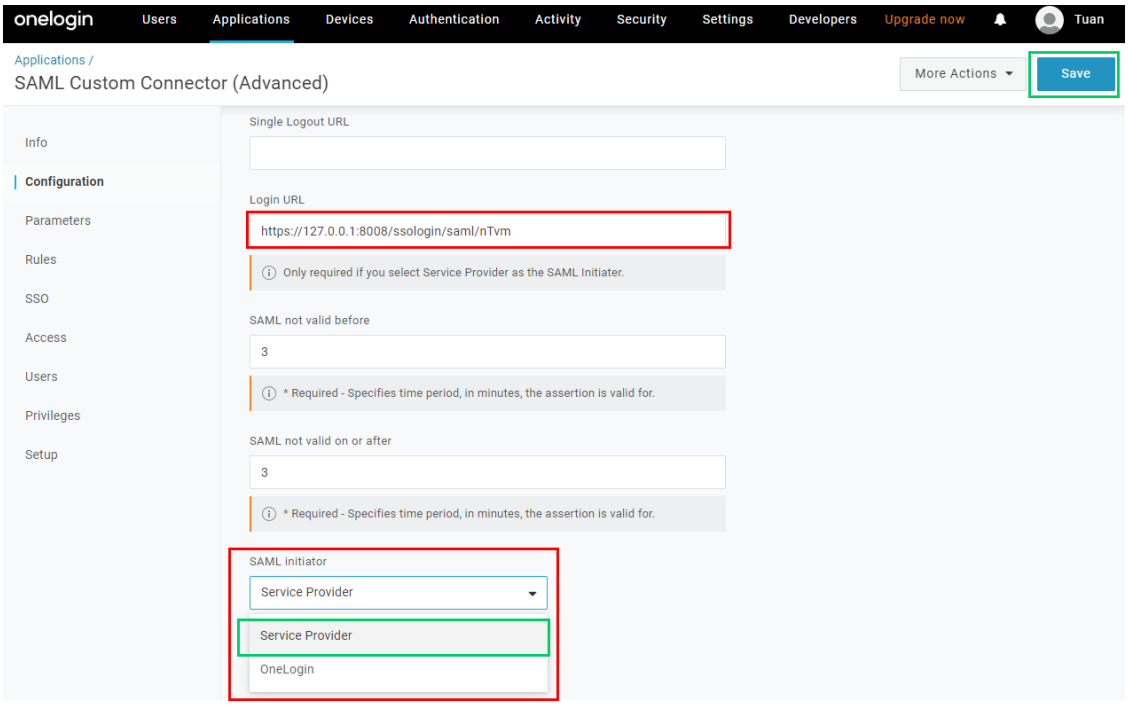


Complete configuration in OneLogin

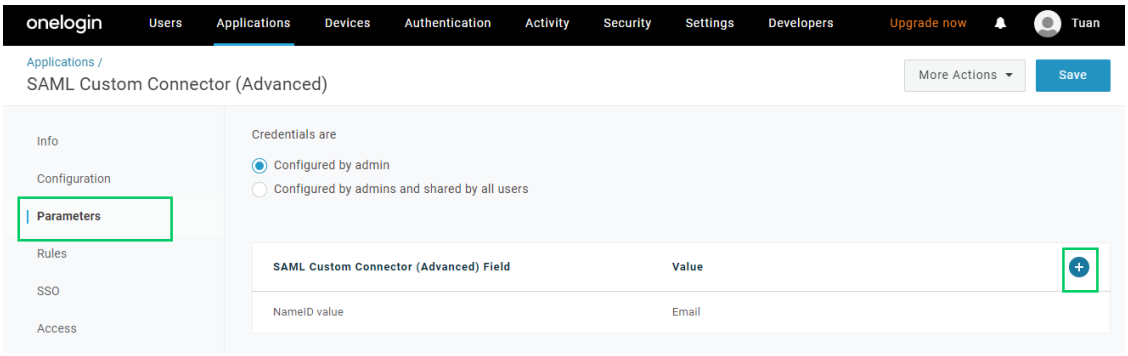
1. In `Applications` screen on OneLogin, select `Configuration` tab.
2. In `Application Details` section, fill in `Recipient` and `ACS (Consumer) URL*` with the `Login URL` copied from MetaDefender Core. Fill in `ACS (Consumer) URL Validator*` with a regular expression version of `Login URL`.



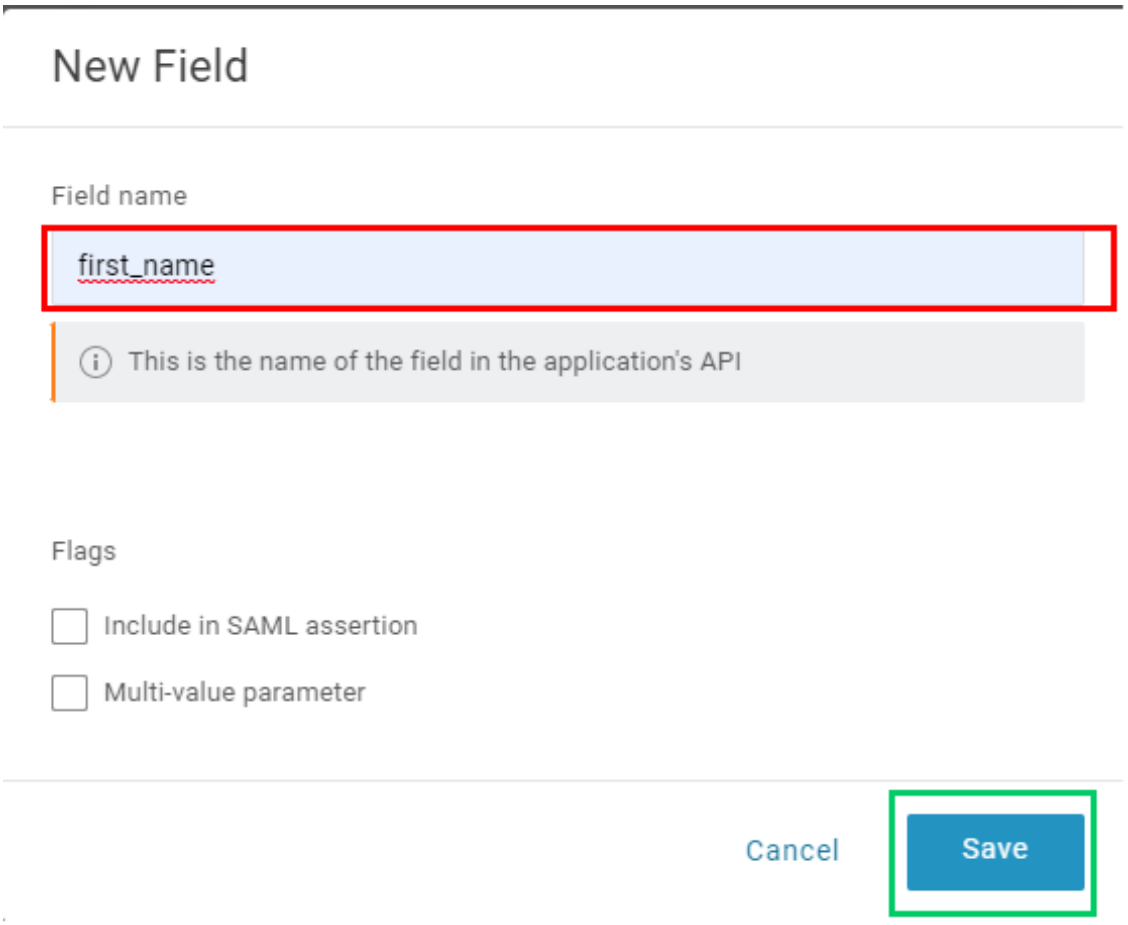
3. Expand `SAML Initiator` section, select `Service Provider`, enter the `Login URL` copied from MetaDefender Core and click `Save`.



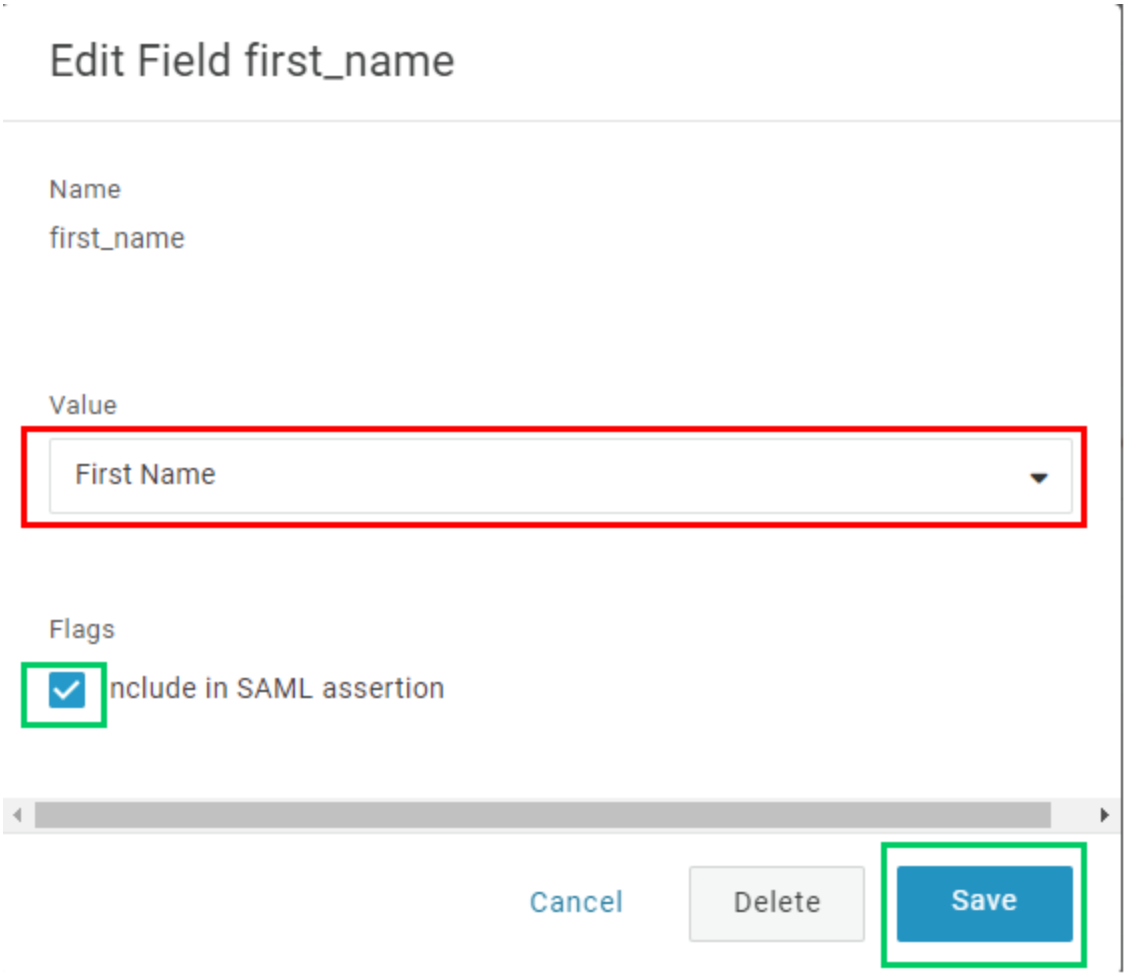
4. Select `Parameters` tab and click the plus button to add more fields that will be used to identify the signed-in users in MetaDefender Core.



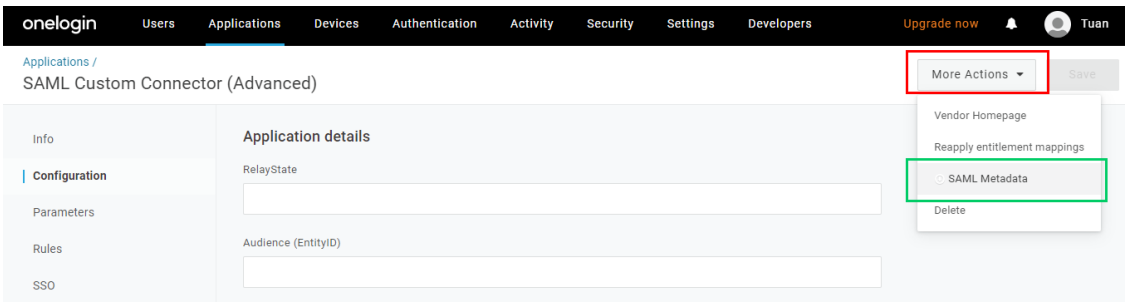
5. Fill in `Field name` with first_name and click `Save`.



6. Expand `Value` section, choose an appropriate attribute, First Name is an example. Then toggle `Include in SAML assertion` and click `Save`.

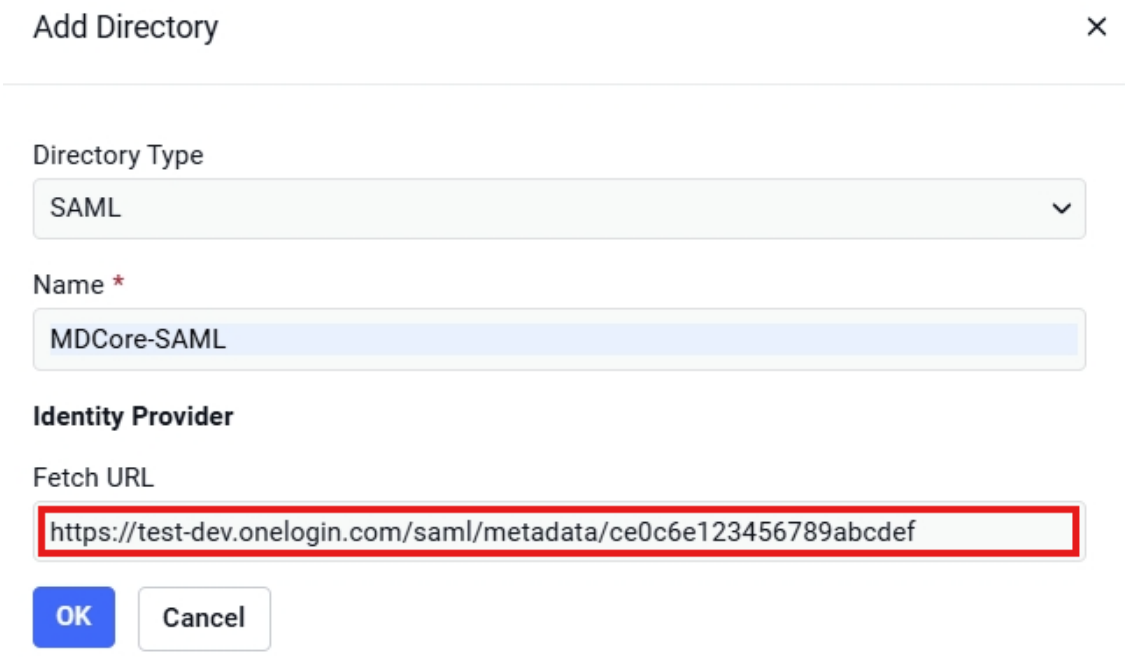


7. Click `Save` in the top right corner.
8. Drop `More Actions` down and click on `SAML Metadata` to copy *metadata_url*.

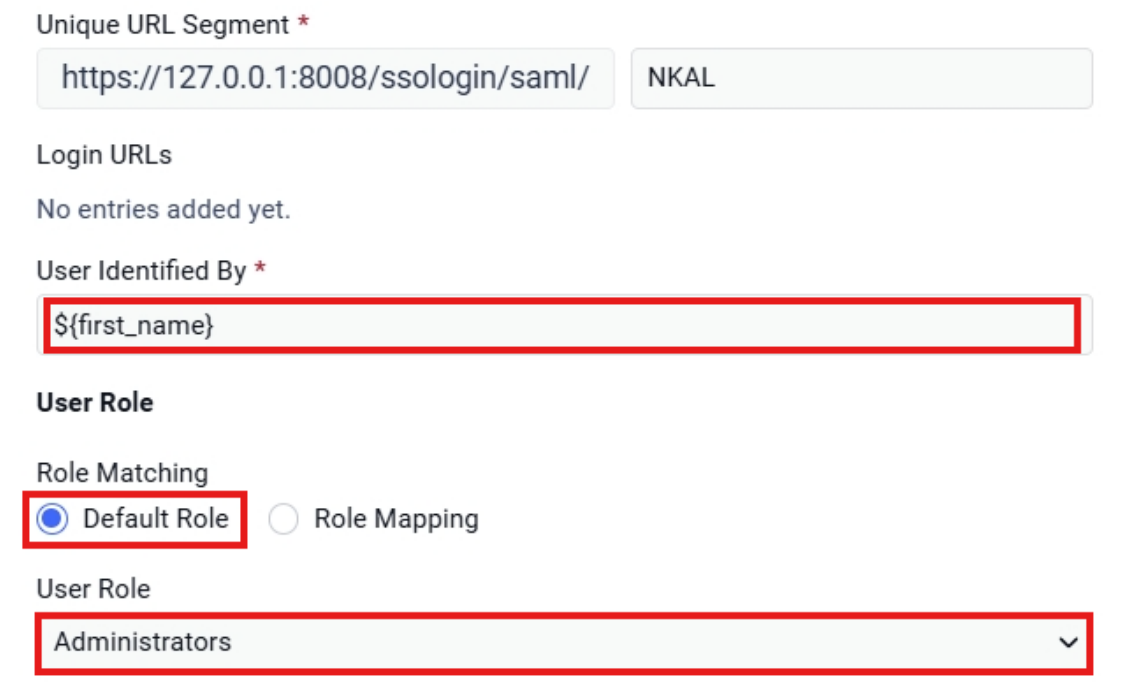


Complete configuration in MetaDefender Core

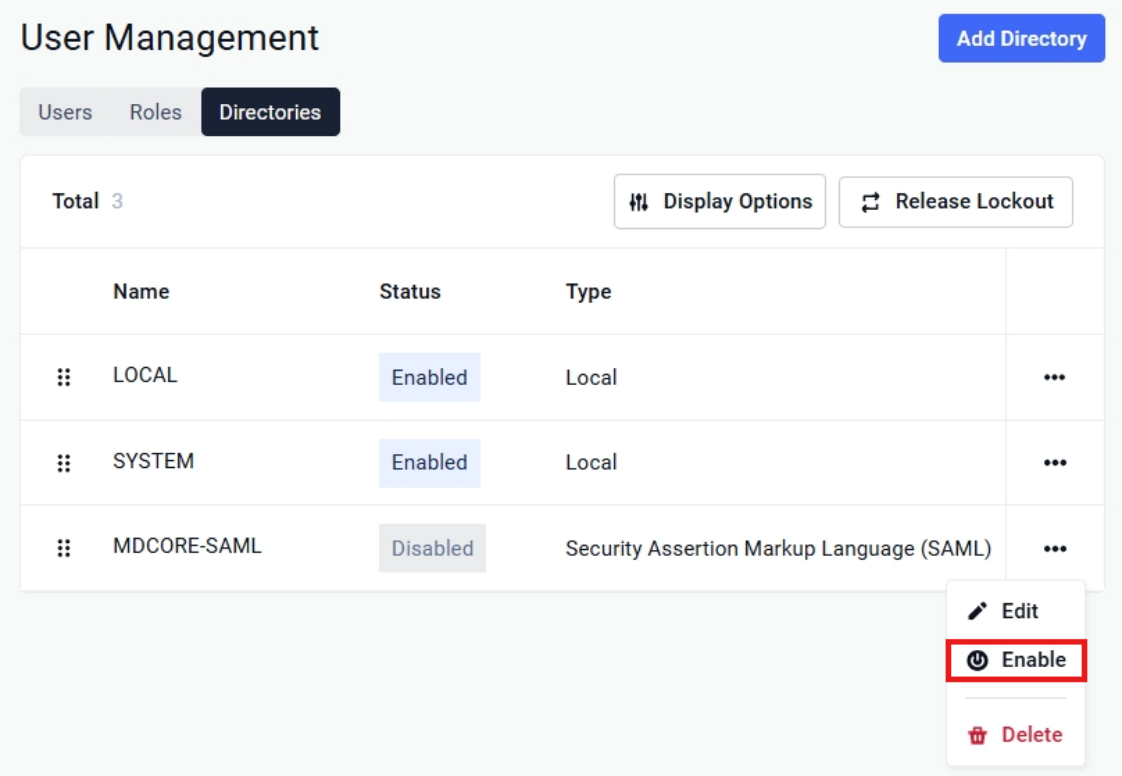
1. Switch back to MetaDefender Core. Under `Identity Provider`, click `Fetch URL`, paste the value of *metadata_url* into the box below, click `OK`, and wait a moment for MetaDefender Core to check and set OneLogin as its IdP.



2. Under `Service Provider`, fill in ${first_name} in the box labeled `User identified by`.
3. Select `Default role` option, choose the appropriate role to assign to users under `User Role`, and click `Add` to complete the settings in MetaDefender Core.

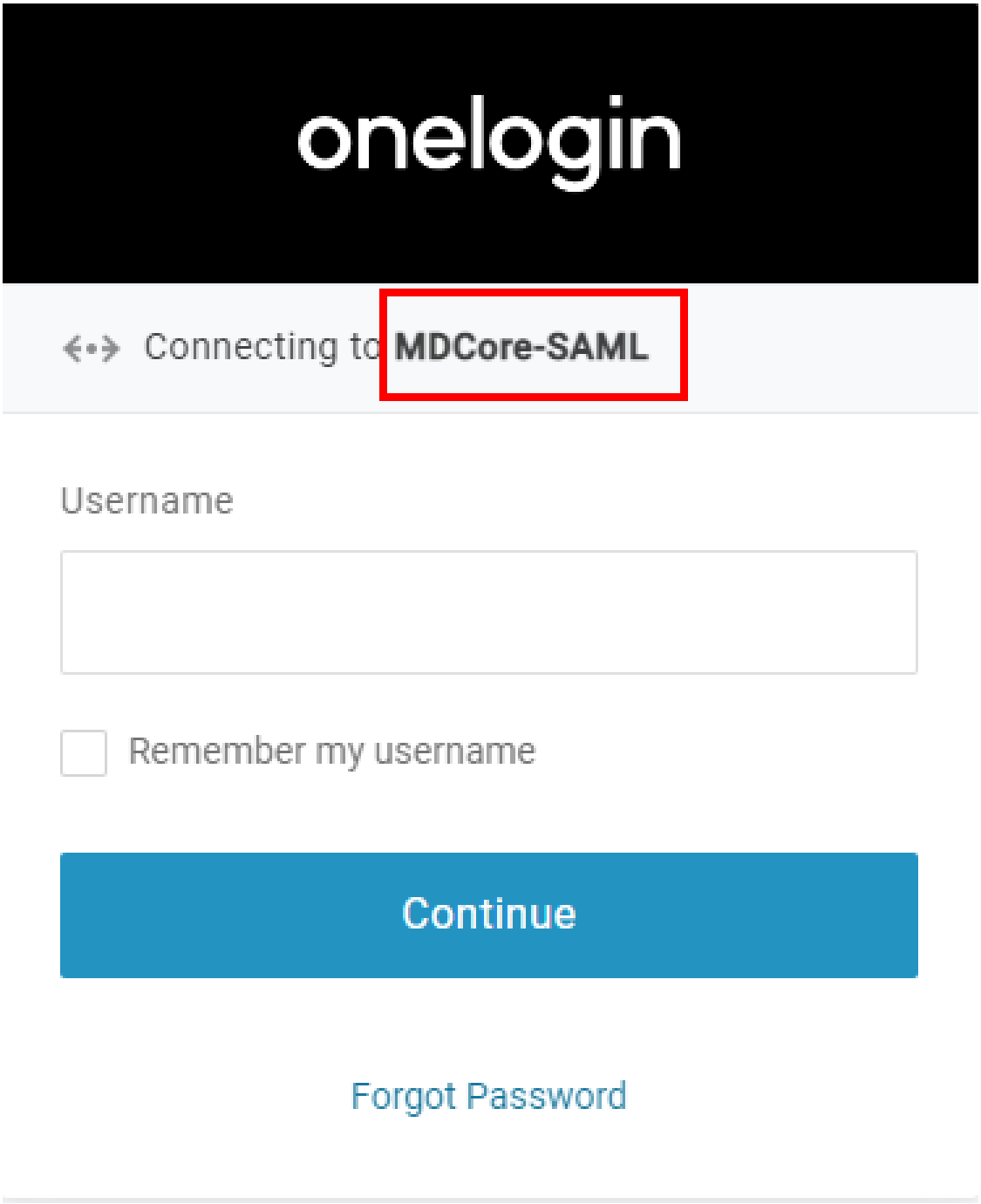


4. On `User Management` page, toggle the new directory, MDCORE-SAML in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked , all sessions will expire immediately.



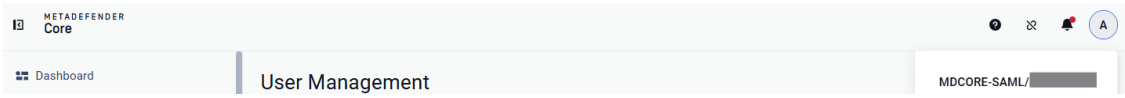
Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to OneLogin page.



2. Sign in using the account registered with OneLogin.

3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.

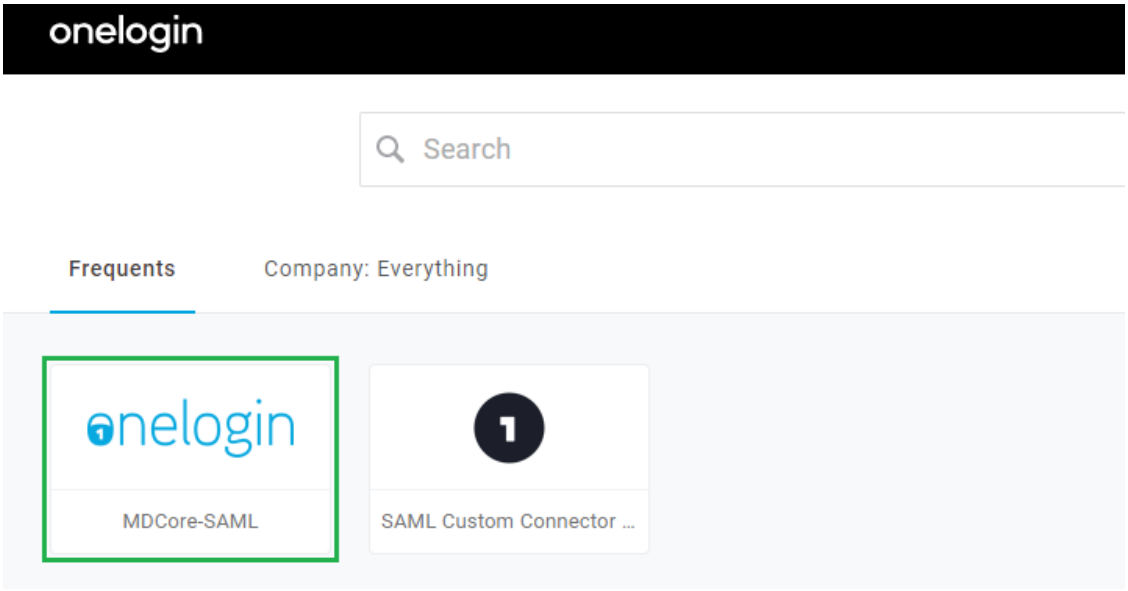


4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

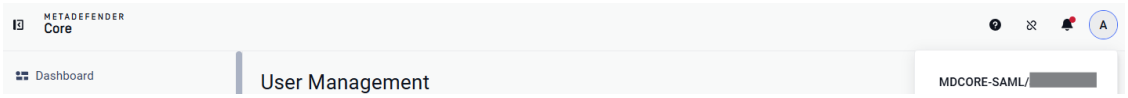
Test IdP-initiated login

1. Sign in to OneLogin management page.

2. From the portal menu, navigate to MDCore-SAML and click on the icon.



3. If everything goes well, the user is redirected to MetaDefender Core dashboard with the user identity in the top right corner.

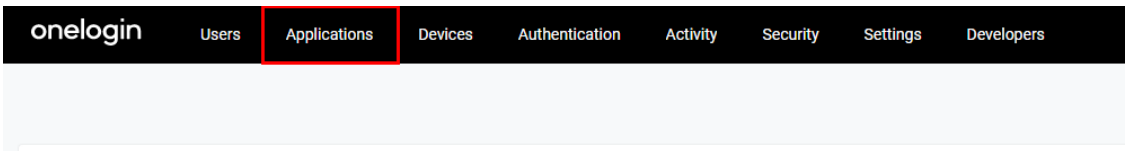


4. Otherwise, access backup login page at <mdcore-host>#/public/backuplogin for trouble shooting.

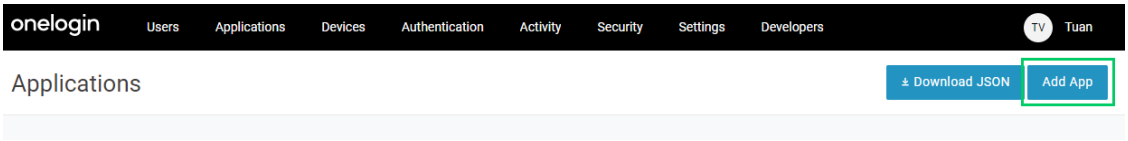
Configure OIDC SSO

Create OneLogin application

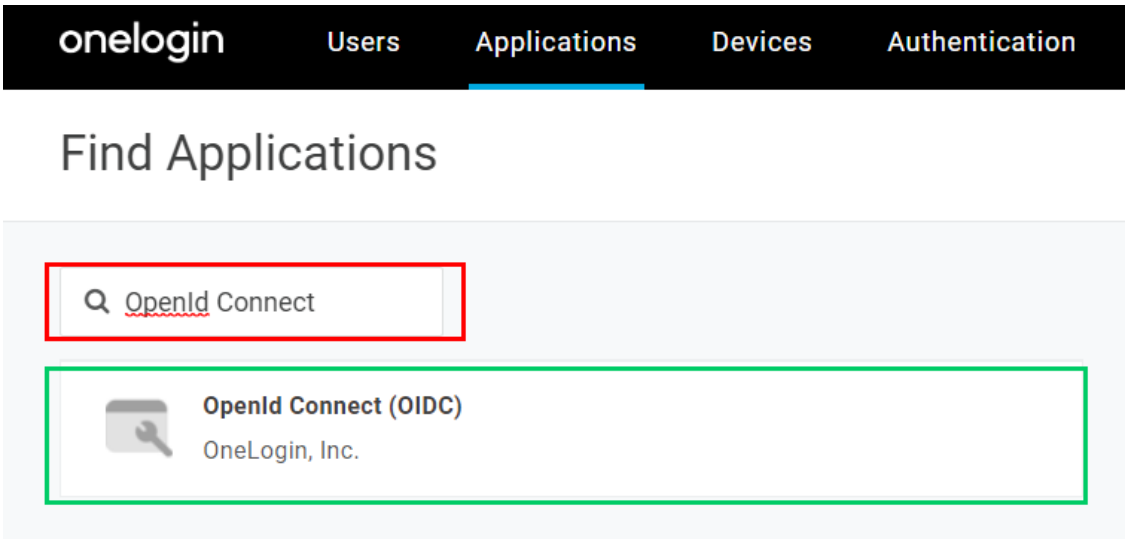
1. Access OneLogin and sign in.
2. On the navigation bar, click `Administration` to access administrator mode, then click on `Applications` to add a new application for integration.



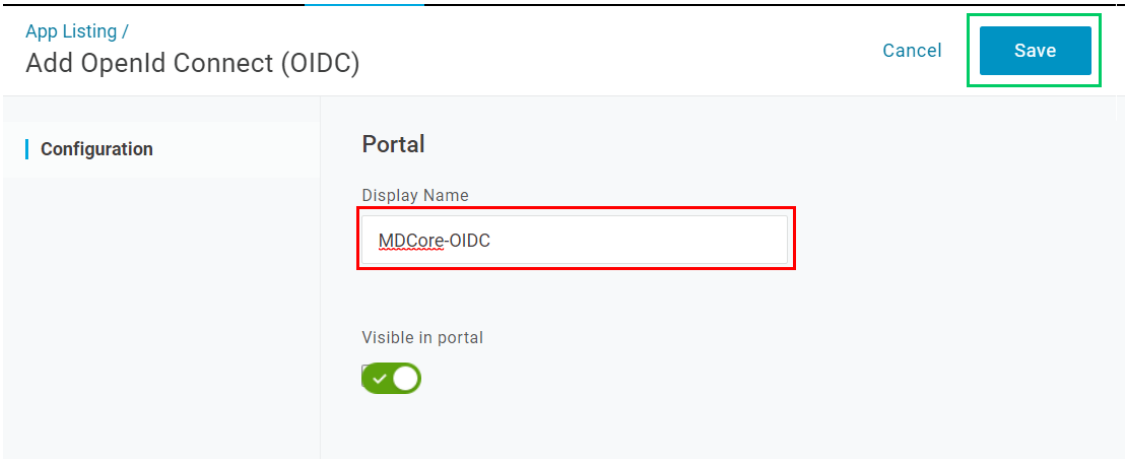
3. Click `Add App` to add a new application.



4. Enter "OpenId Connect" in the search box and select `OpenId Connect (OIDC)` in the results panel.

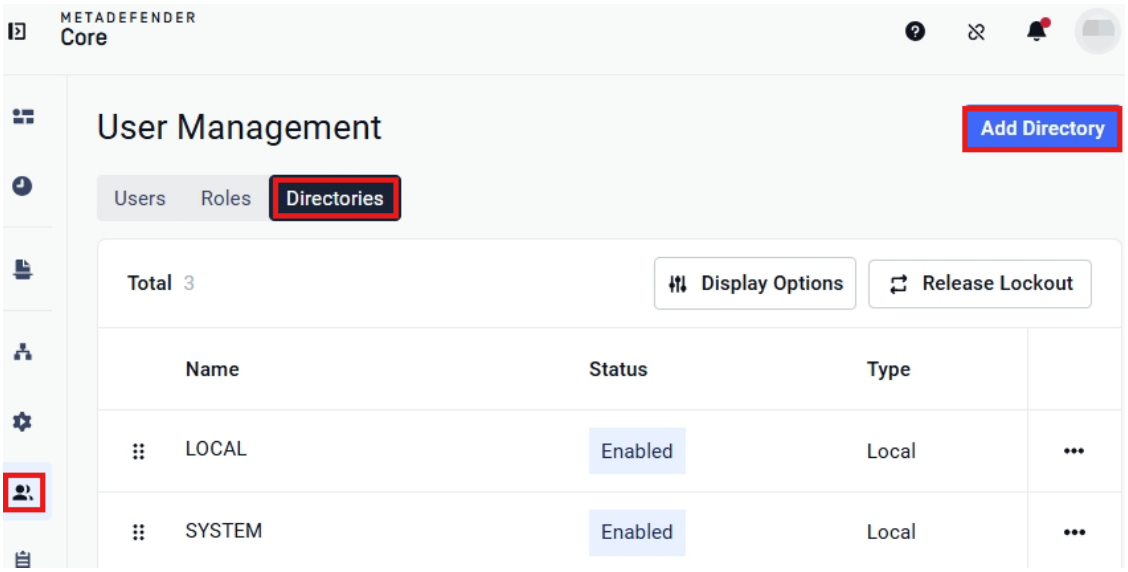


5. In the new page, fill in `Display Name` with MDCore-OIDC, for example, and click `Save`.

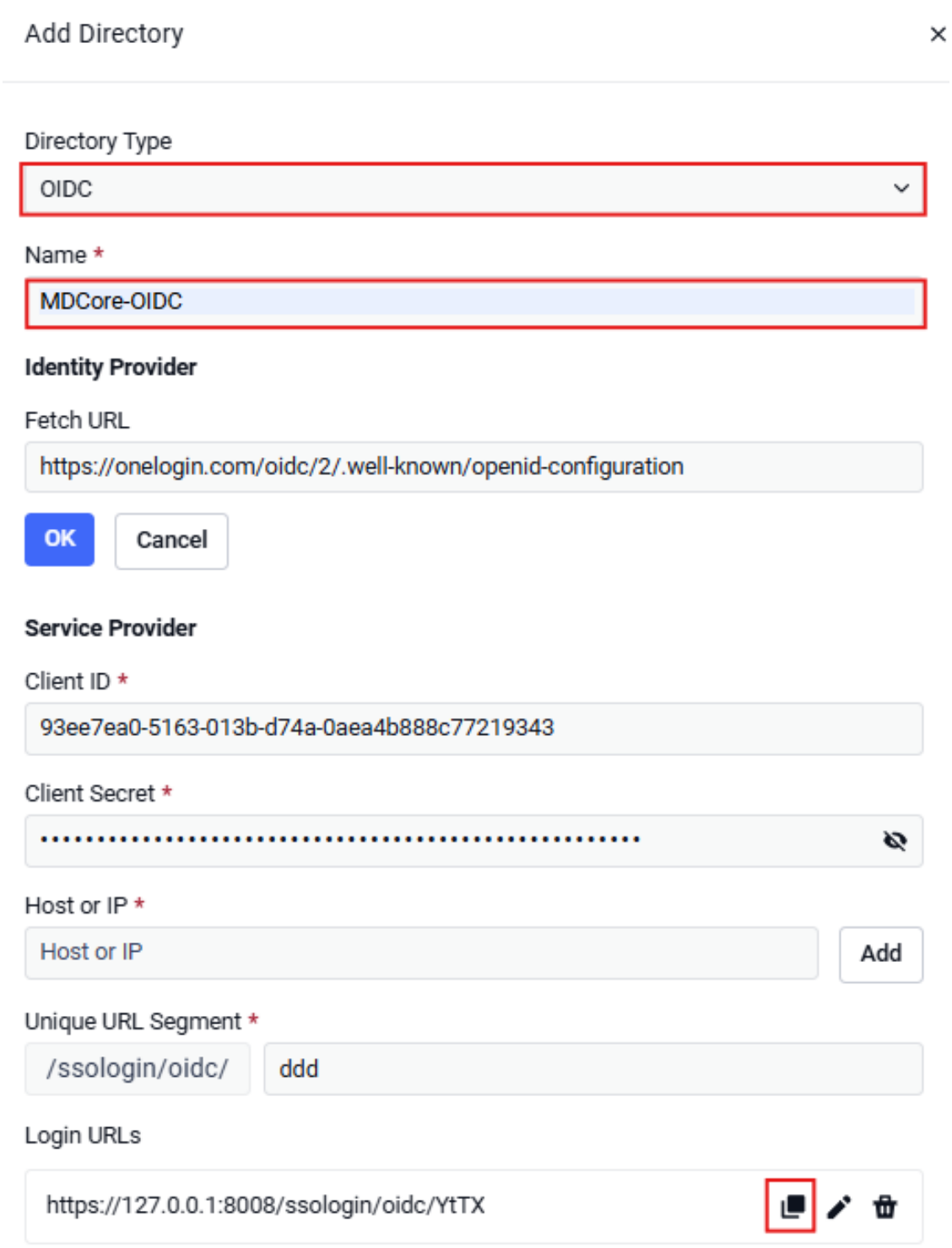


Create OIDC directory in MetaDefender Core

1. Sign in to MetaDefender Core.
2. In the dashboard, click on `User Management` in the left sidebar.
3. On `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.

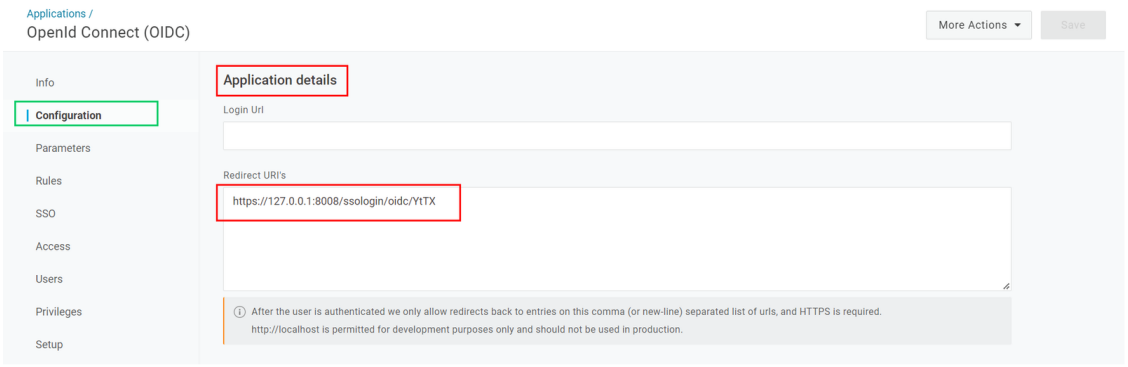


4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCore-OIDC.
5. Enter `Host or IP` where MetaDefender Core is hosted, for this example use https://127.0.0.1:8008.
6. Copy the string generated under `Login URL`.



Complete configuration in OneLogin

1. Back to `Applications` screen in OneLogin.
2. Choose `Configuration` in the left side bar and paste value of `Login URL` from MetaDefender Core into `Redirect URI's` field.



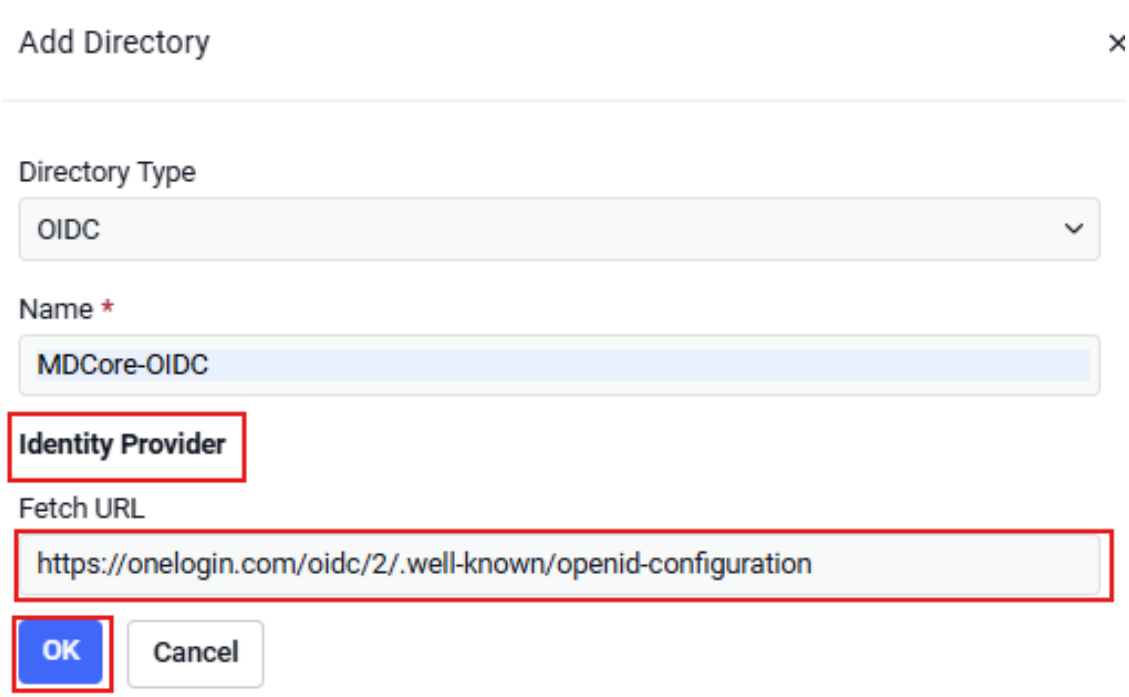
3. In `SSO` of the left side bar, under `Enable OpenID Connect` section, copy `Client ID`, `Client Secret`, and `Issuer URL`.



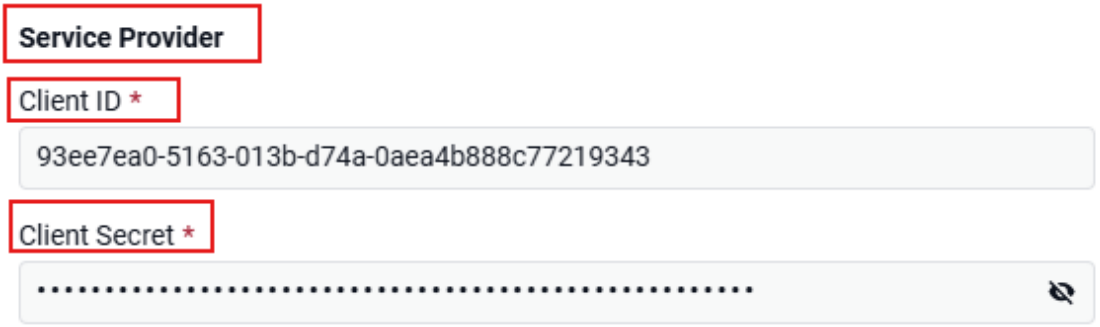
4. Click `Save` to complete.

Complete configuration in MetaDefender Core

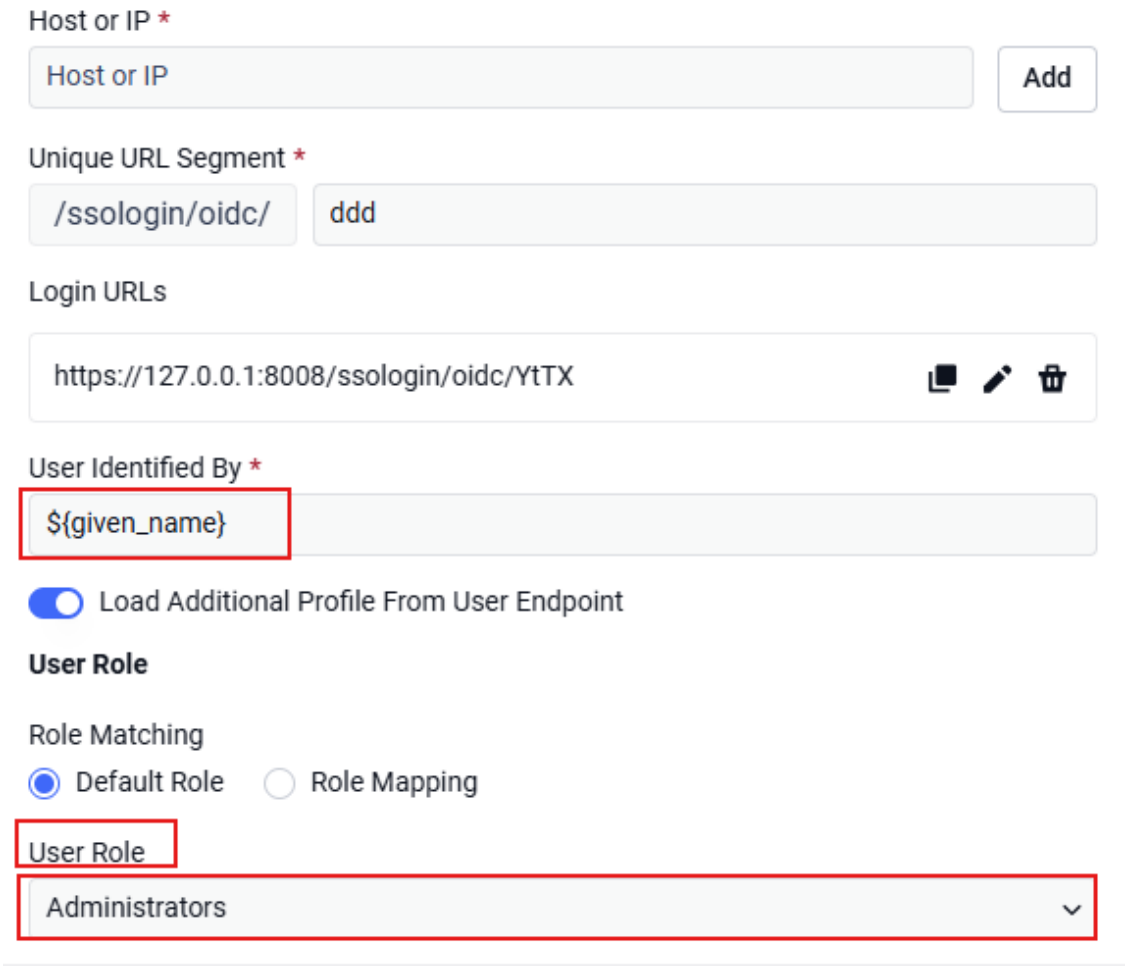
1. Switch back to MetaDefender Core. Under `Identity Provider`, click `Fetch URL`, paste value of `Issuer URL` from OneLogin into the box below, click `OK`, and wait a moment for MetaDefender Core to check and set OneLogin as its IdP.



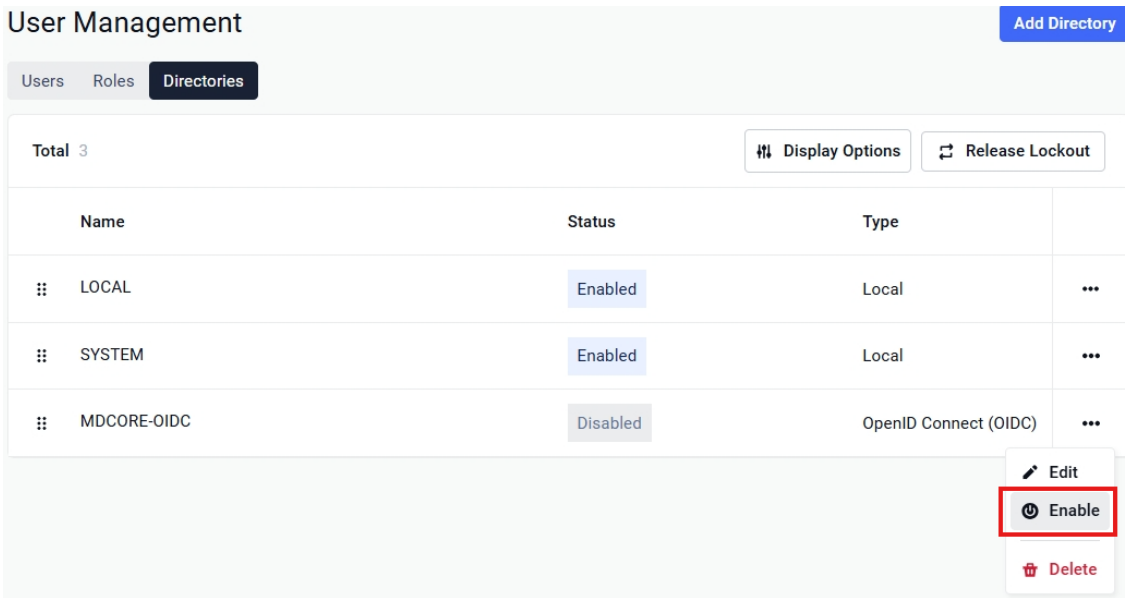
2. Under `Service Provider`, paste the values of `Client ID` and `Client Secret` from OneLogin into the boxes labeled `Client ID` and `Client secret`, respectively.



3. Fill in `User Identified By` with ${given_name}.
4. Select `Default role` option, choose the appropriate role under `User Role`, and click `Add` to complete.

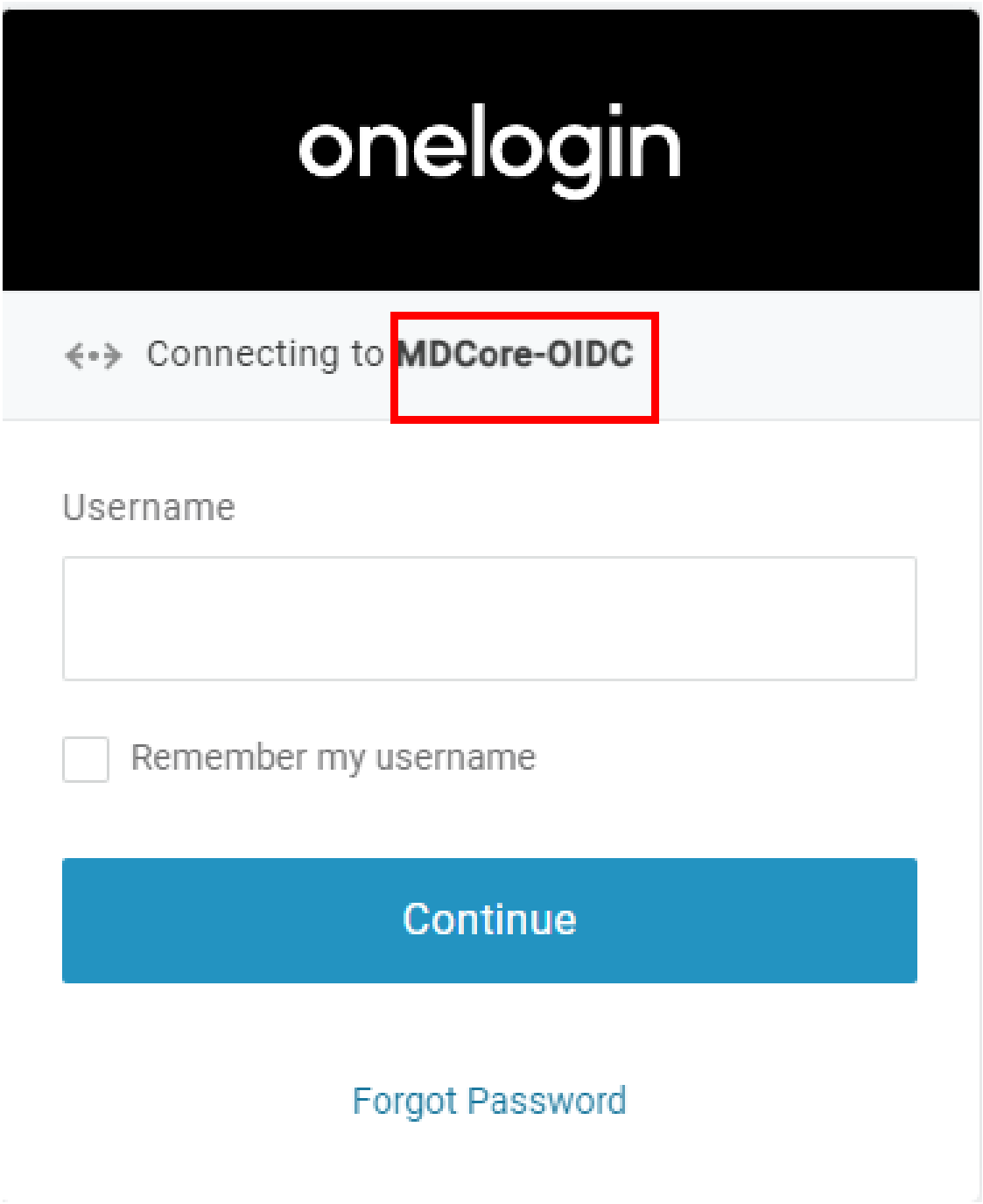


5. In `User Management` page, toggle the new directory, MDCORE-OIDC in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all sessions will expire immediately.



Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to OneLogin page.



2. Sign in using the account registered with OneLogin.

3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

PingFederate

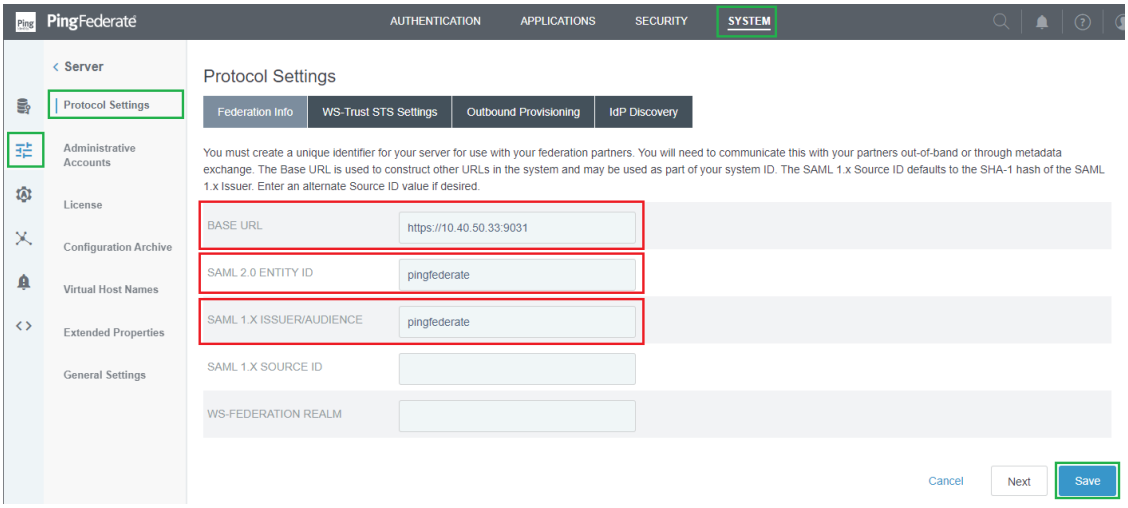
The instructions guide users on how to configure PingFederate SAML and OIDC Single Sign-on for MetaDefender Core.

Configure SAML SSO

Set up PingFederate

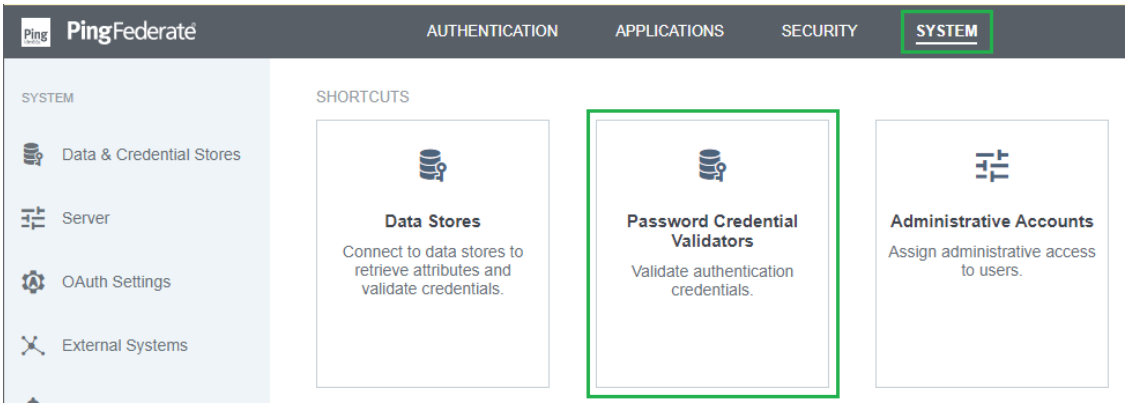
Configurate PingFederate

1. Sign in to PingFederate console at `https://service-ip:service-port`, where `service-ip` and `service-port` refer to the IP address and port of the machine hosting PingFederate server.
2. In homepage, navigate to `SYSTEM` tab and select `Server` from the left sidebar.
3. Select `Protocol Settings`, enter `Base URL`, `SAML 2.0 ENTITY ID`, and `SAML 1.X ISSUER/AUDIENCE`, then click `Save` to complete.



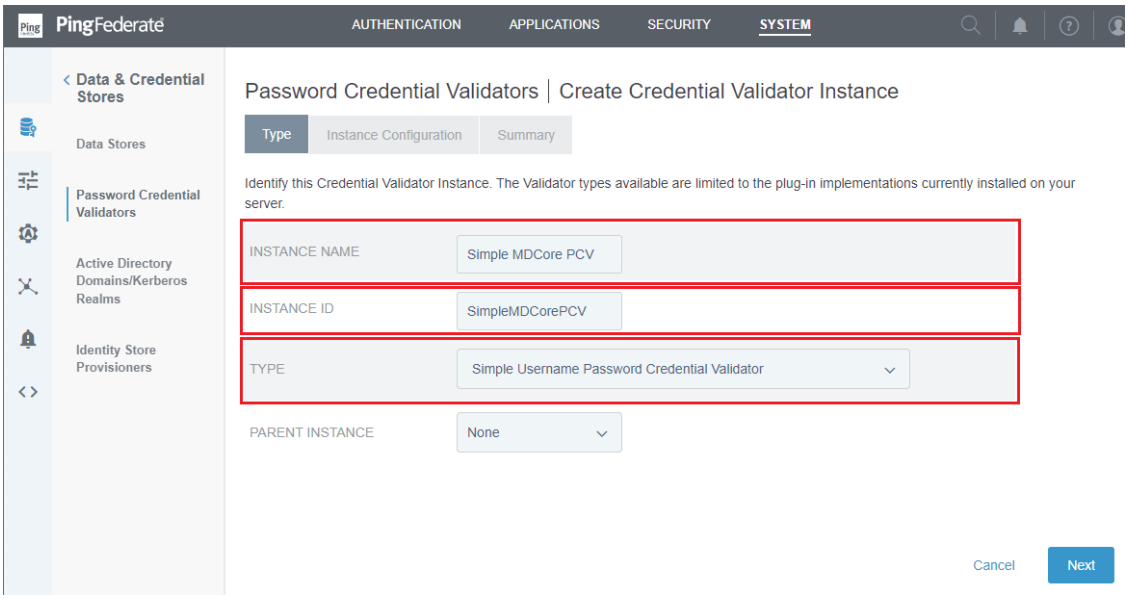
Set up simple Password Credential Validators

1. In homepage, navigate to `SYSTEM` tab and click on `Password Credential Validators`.

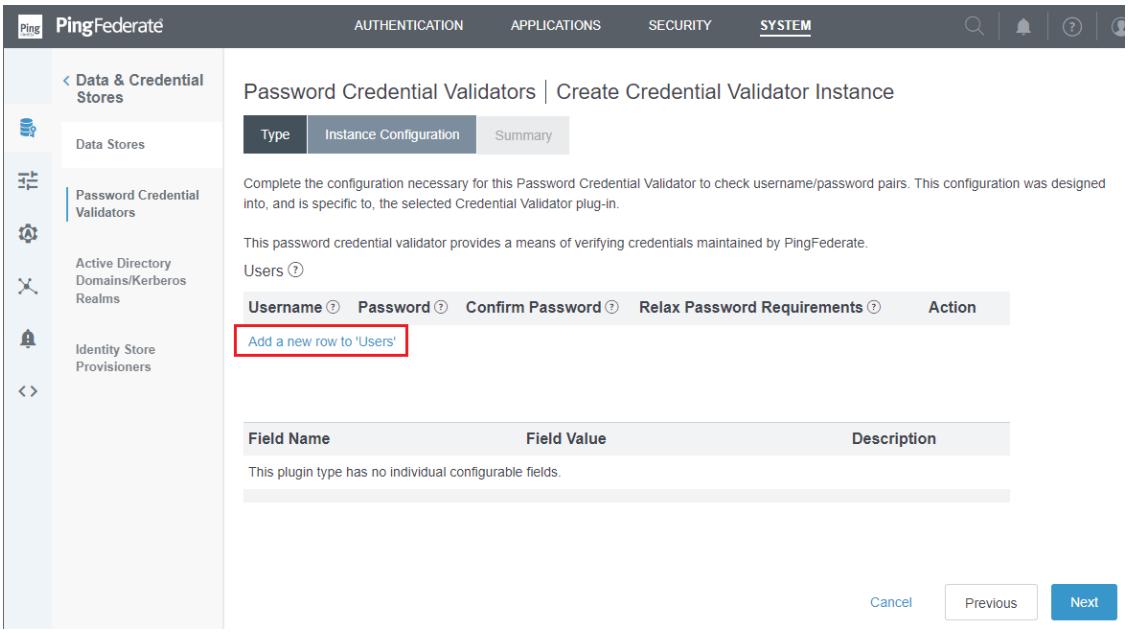


2. Click `Create New Instance`.
3. Enter `INSTANCE NAME` and `INSTANCE ID`, select `Simple Username Password Credential Validator` for `TYPE`,

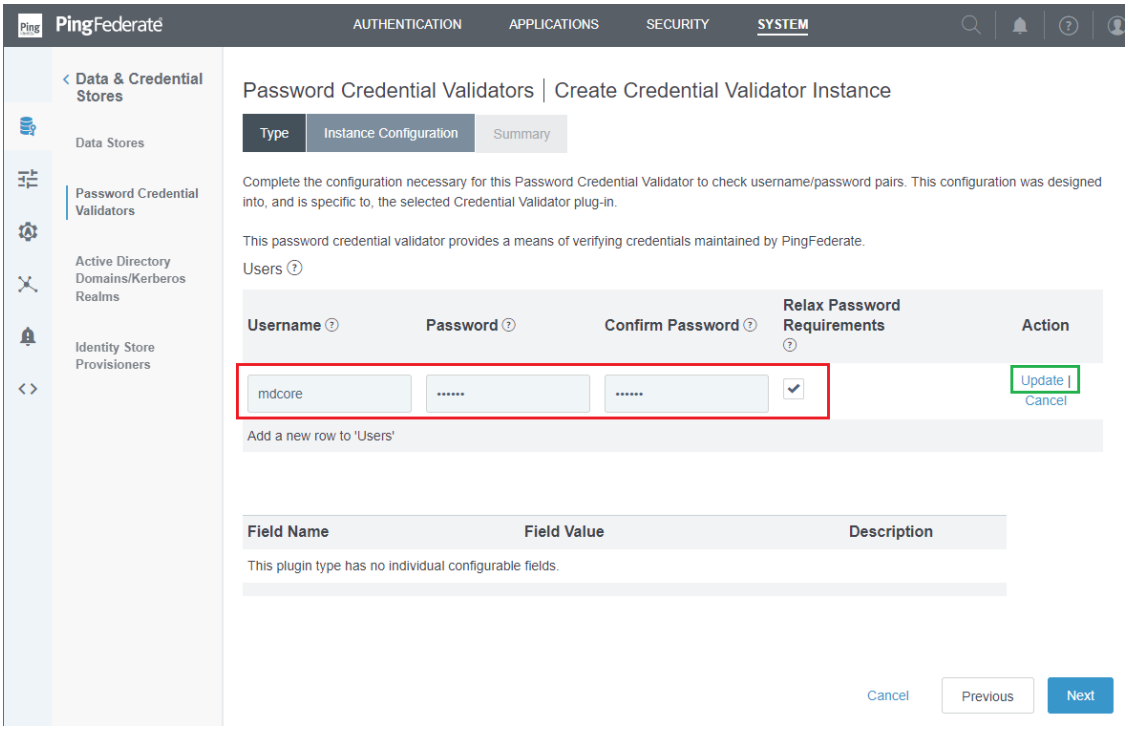
and click `Next`.



4. In `Create Credential Validator Instance` page, click `Add a new row to 'Users'.`



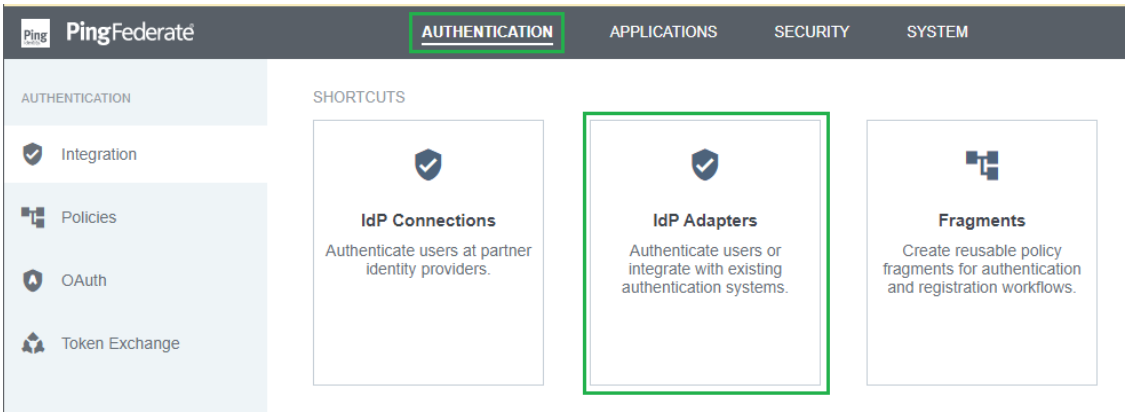
5. Enter `Username`, `Password`, and `Confirm Password`, then click `Update` and `Next`.



6. Review all selected options in `Summary` tab, and click `Save`.

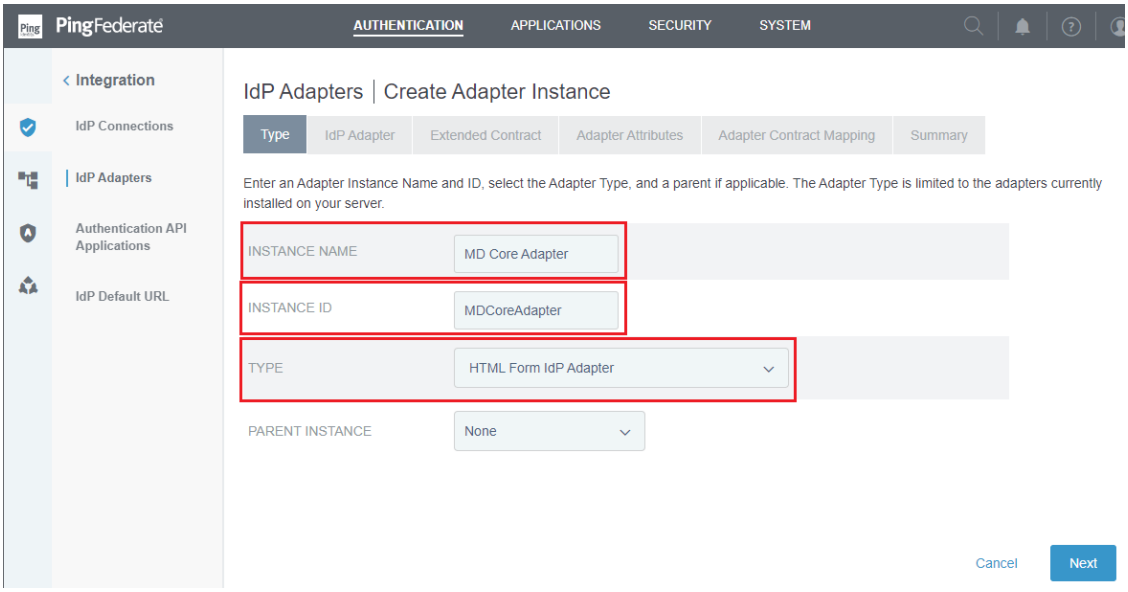
Set up a simple IdP Adapter

1. In homepage, navigate to `AUTHENTICATION` tab and click on `IdP Adapters`.

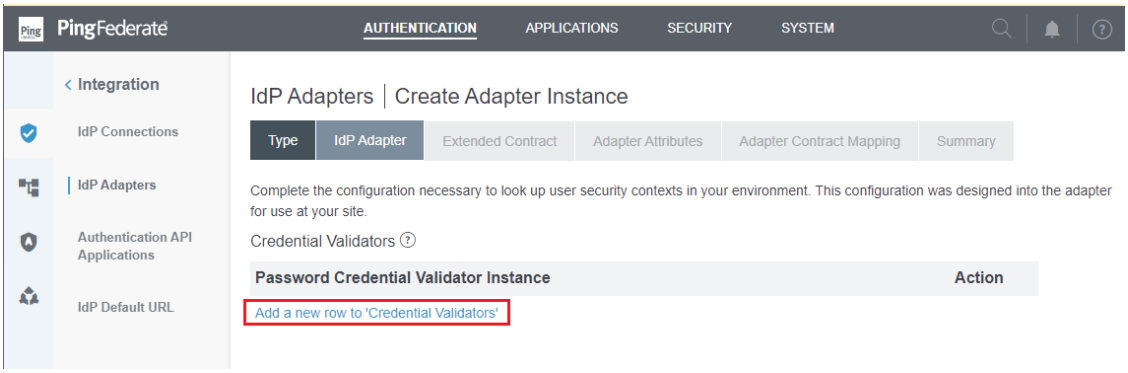


2. Click `Create New Instance`.

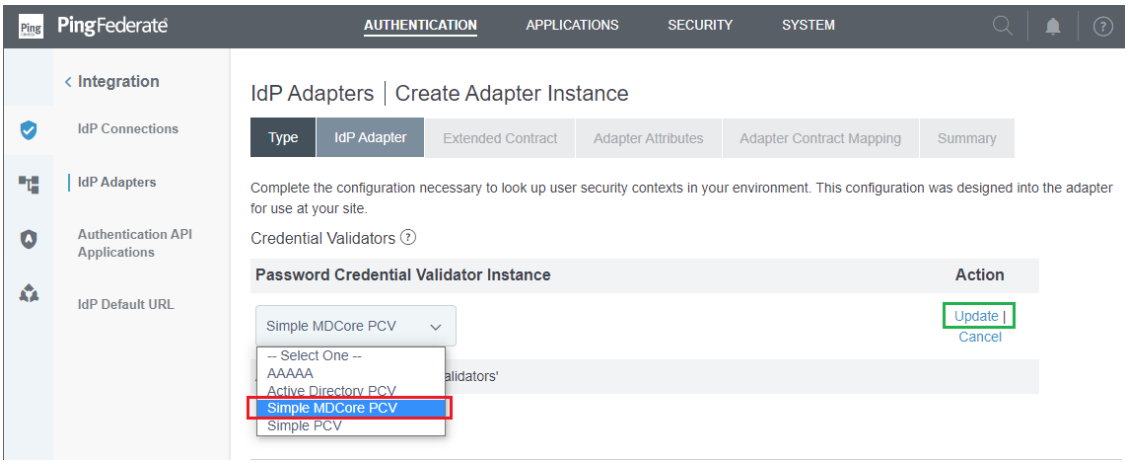
3. Enter `INSTANCE NAME`, `INSTANCE ID`, and select `HTML Form IdP Adapter` for `TYPE` . Click `Next`.



4. Click `Add a new row to 'Credential Validators'`.

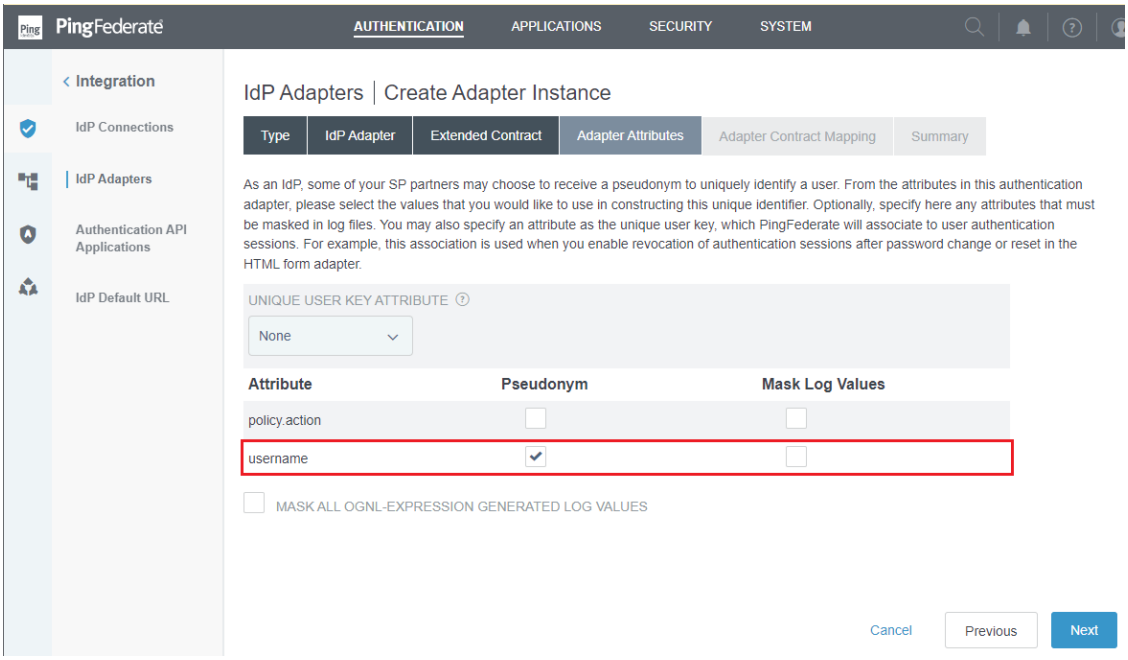


5. Select the Password Credential Validator created in the previous stage, that is "Simple MDCore PCV", then click `Update` and `Next`.



6. Click `Next` on `Extended Contract` tab.

7. In `Adapter Attributes` tab, toggle `username` attribute under `Pseudonym` column, and click `Next`.

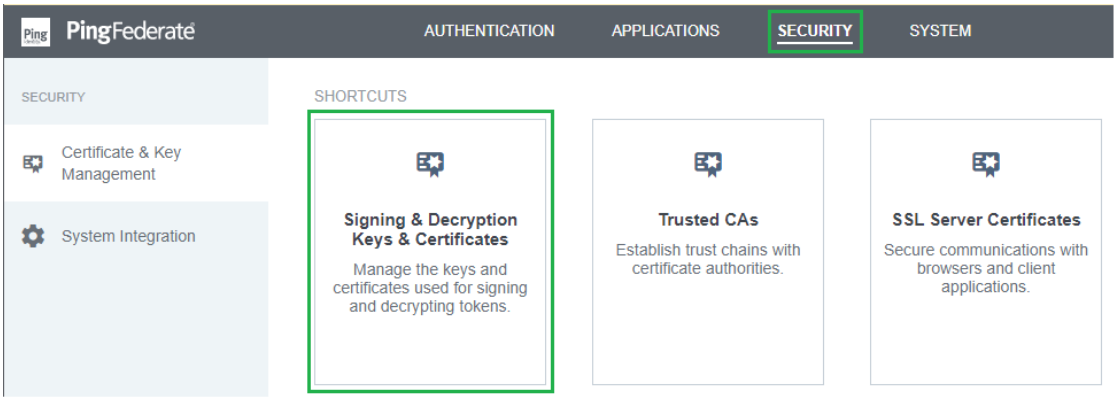


8. Click `Next` on `Adapter Contract Mapping` tab.

9. Review all selected options in `Summary` tab and click `Save` to complete.

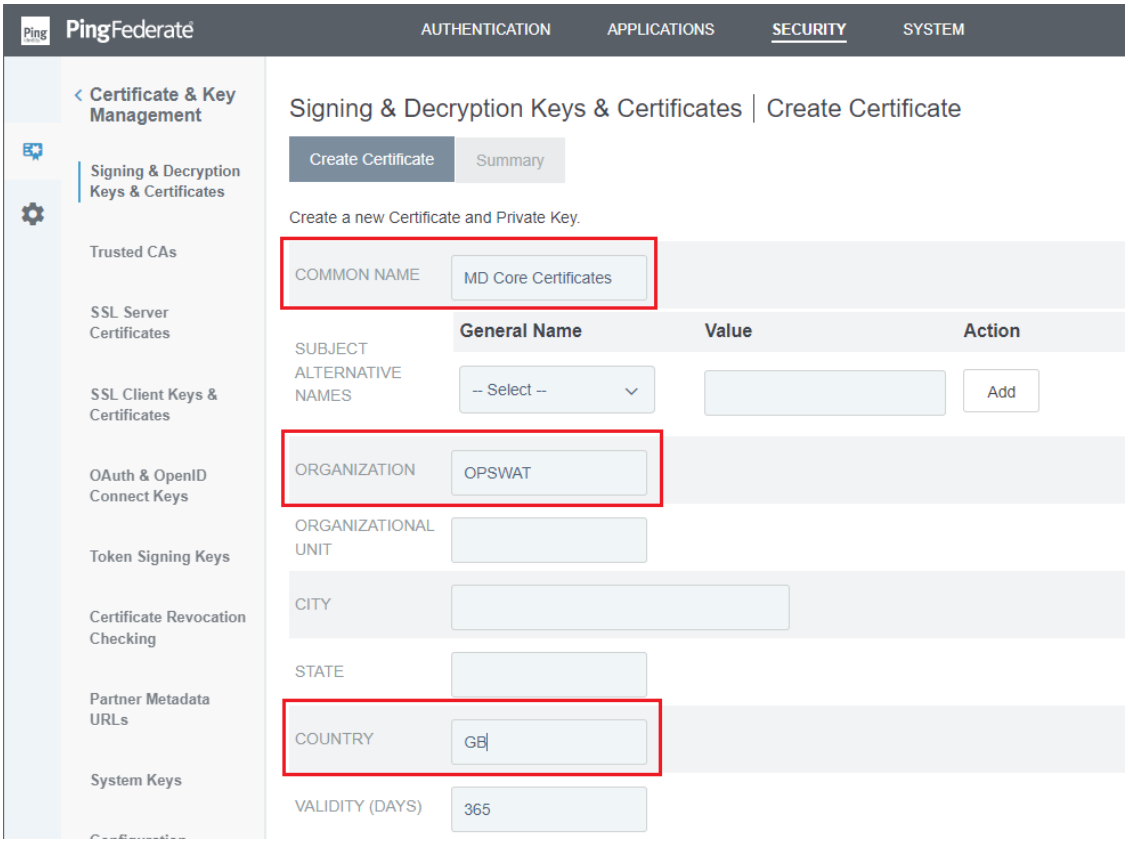
Set up Signing, Decryption Keys and Certificates

1. In homepage, navigate to `SECURITY` tab and select `Signing & Decryption Keys & Certificates`.



2. Click `Create New`.

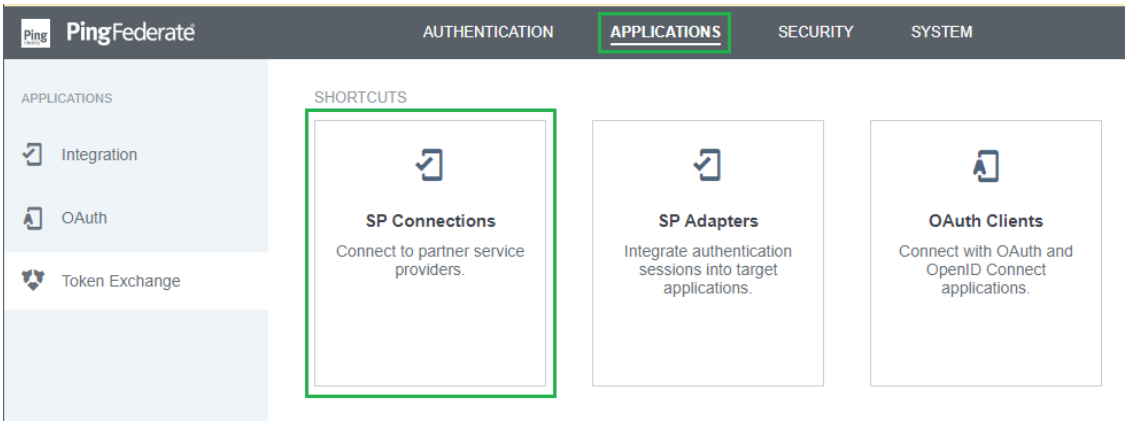
3. Enter `COMMON NAME`, `ORGANIZATION`, and `COUNTRY`, along with any other necessary information, then click `Next`.



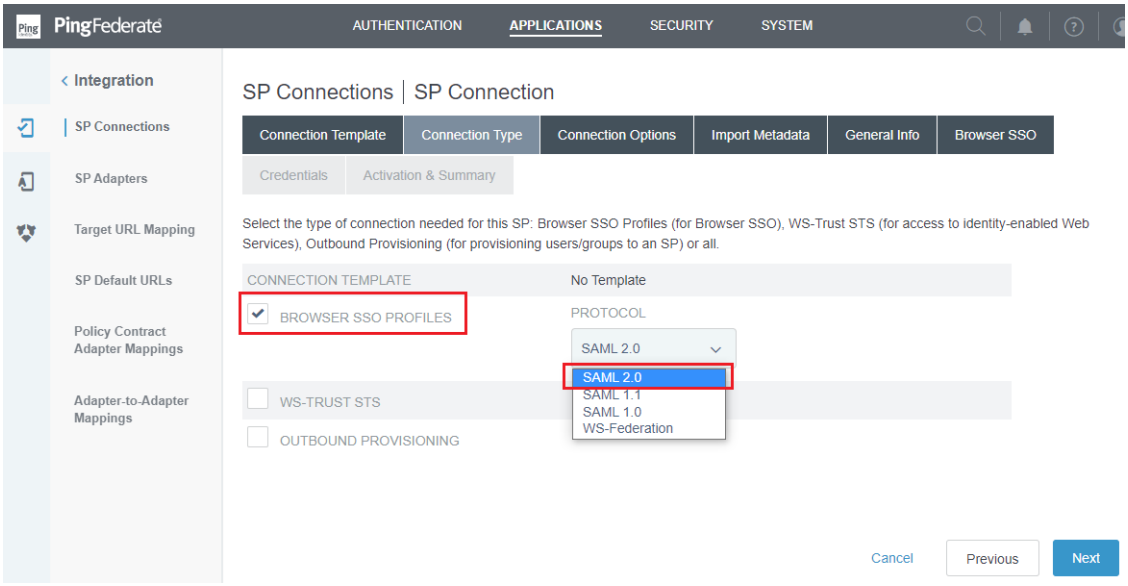
4. Review all selected options in `Summary` tab and click `Save` to complete.

Set up Service Provider connections

1. In homepage, go to `APPLICATIONS` tab and choose `SP Connections`.



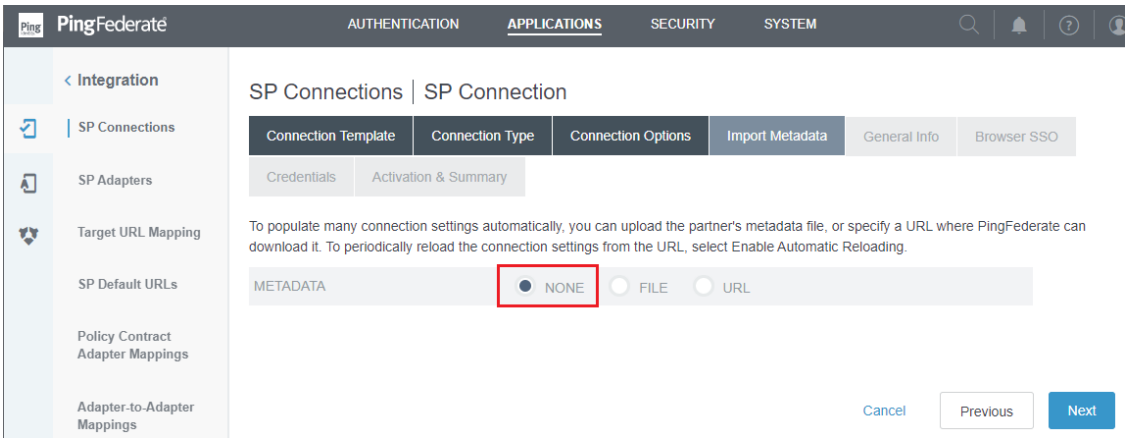
2. Select `Create Connection`.
3. Select `DO NOT USE A TEMPLATE FOR THIS CONNECTION` and then click `Next`.
4. In `Connection Type` phase, toggle `BROWSER SSO PROFILES` and choose `SAML 2.0` for `PROTOCOL`. Press `Next`.



5. In `Connection Options` phrase, select `BROWSER SSO` and then click `Next`.



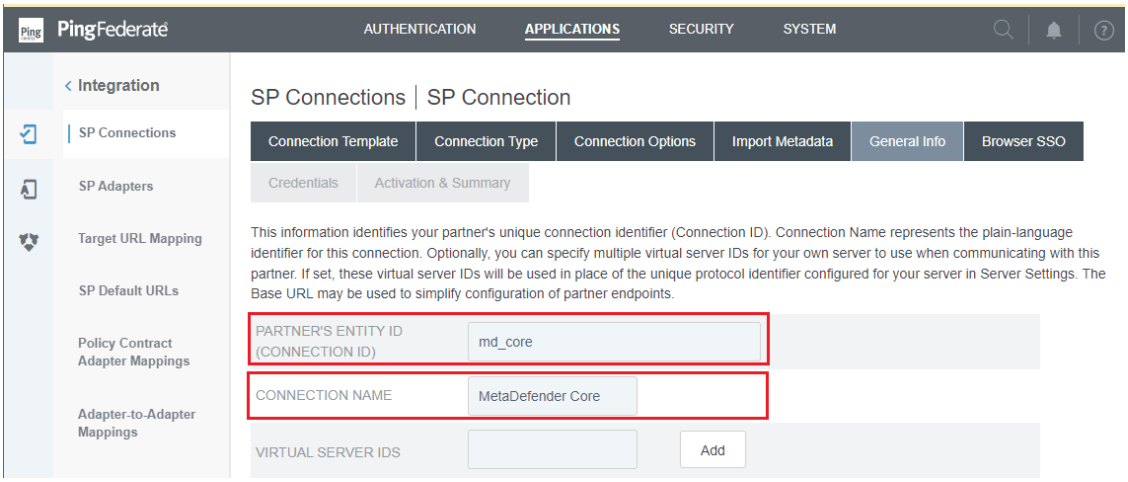
6. In `Import Metadata` phase, select `NONE` for `METADATA` subtab and press `Next`.



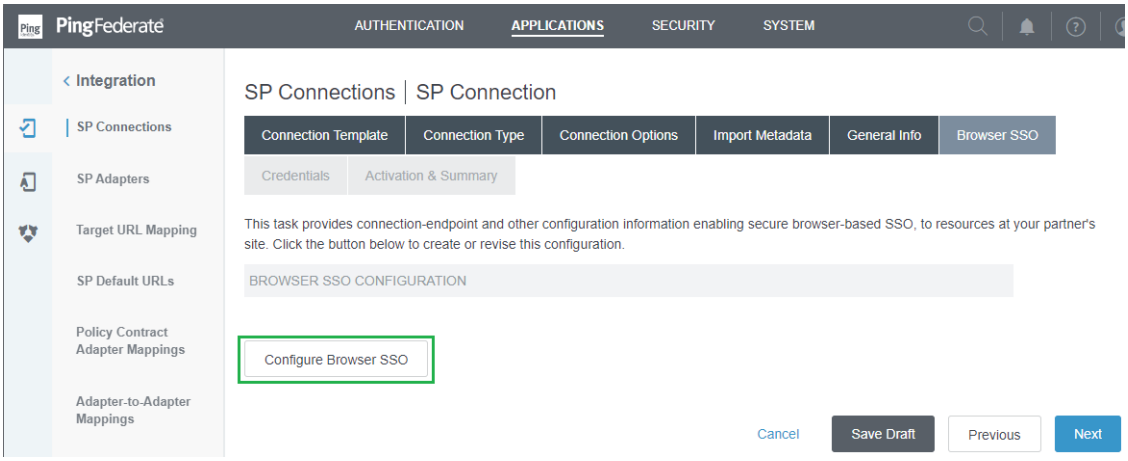
7. In `General Info` section, enter `PARTNER'S ENTITY ID (CONNECTION ID)` and `CONNECTION NAME`, then click `Next`.

Info

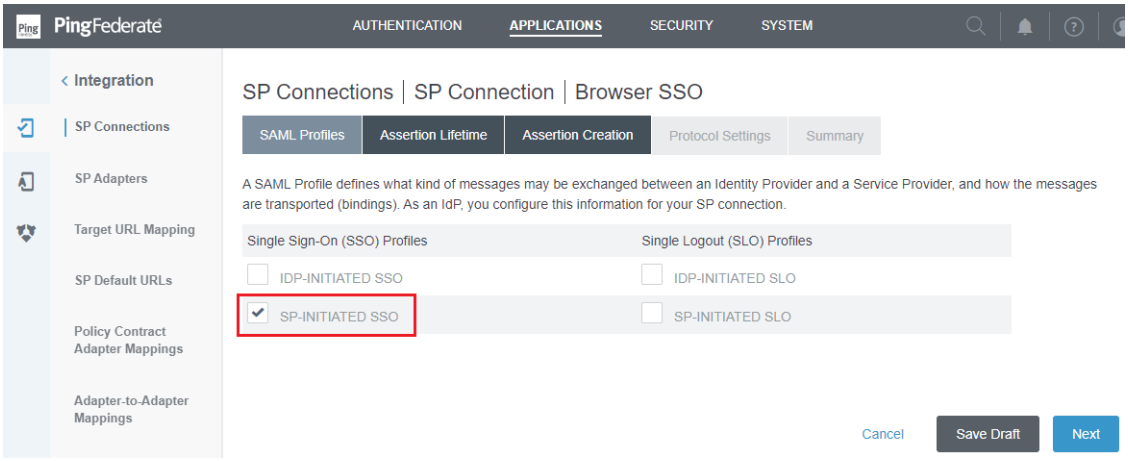
`PARTNER'S ENTITY ID`, 'md_core' in this case, will later be used to configure `Custom entity ID` in MetaDefender Core.



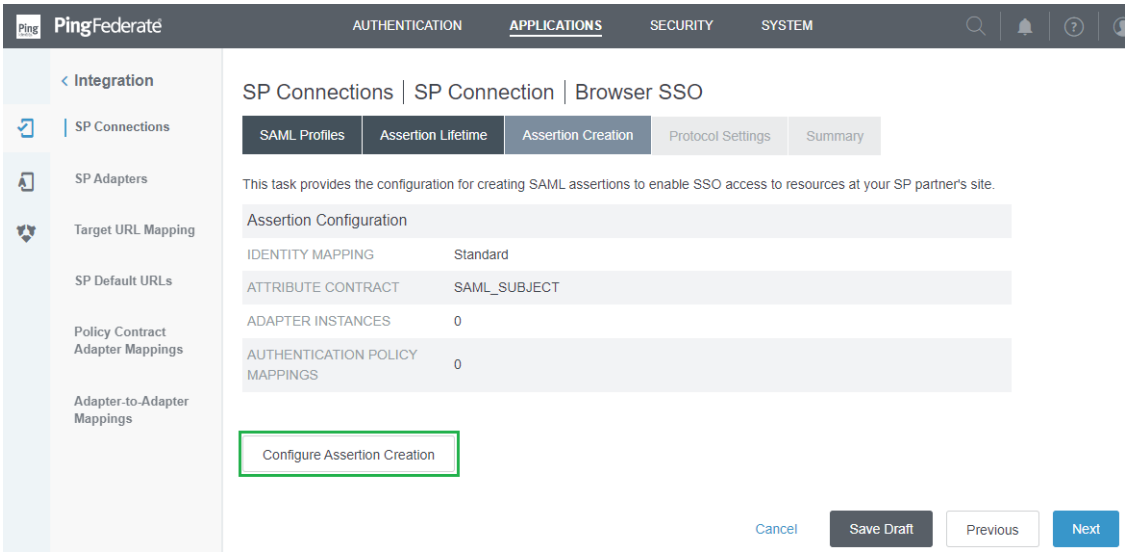
8. In `Browser SSO` phrase, select `Configure Browser SSO`.



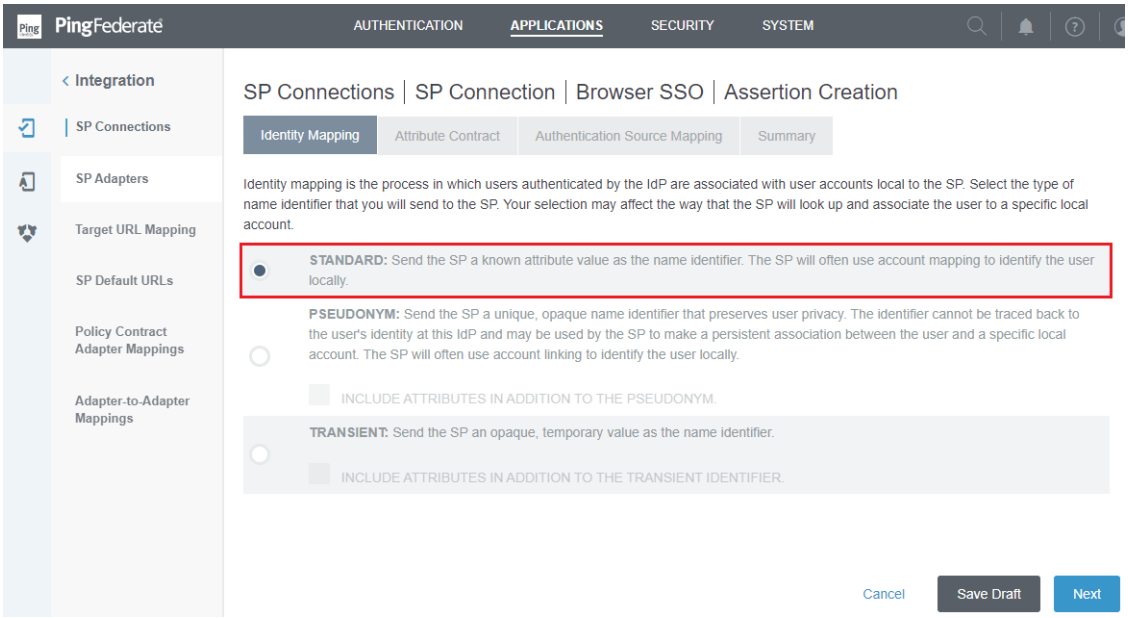
9. In `SAML Profiles` phrase, check `SP-INITIATED SSO` under `Single Sign-On (SSO) Profiles` column. Press `Next` once you are finished.



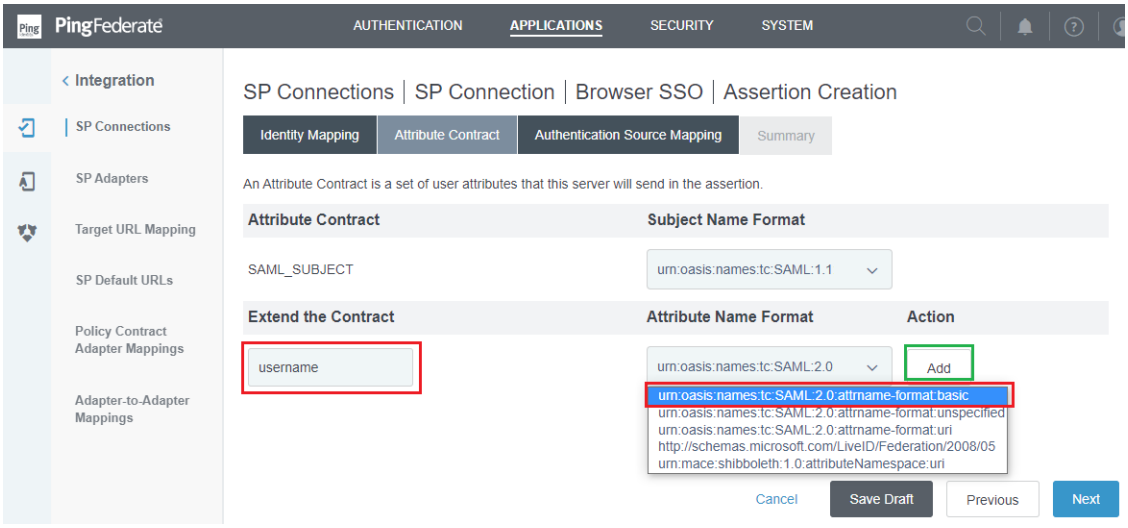
10. Select `Next` in `Assertion Lifetime` phase.
11. In `Assertion Creation` phase, select `Configure Assertion Creation`.



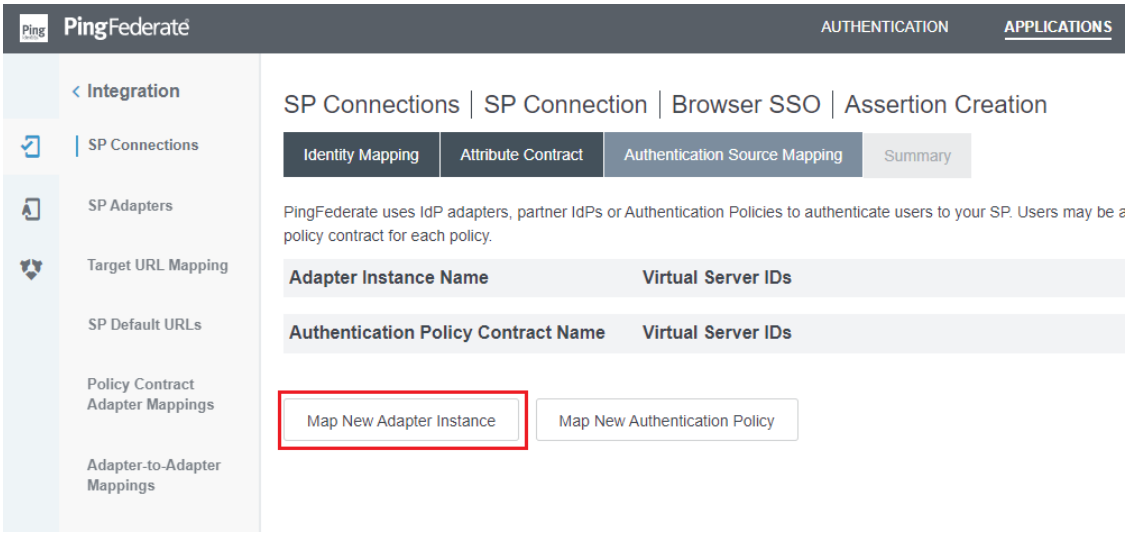
12. In `Identity Mapping` phrase, choose `STANDARD` and press `Next`.



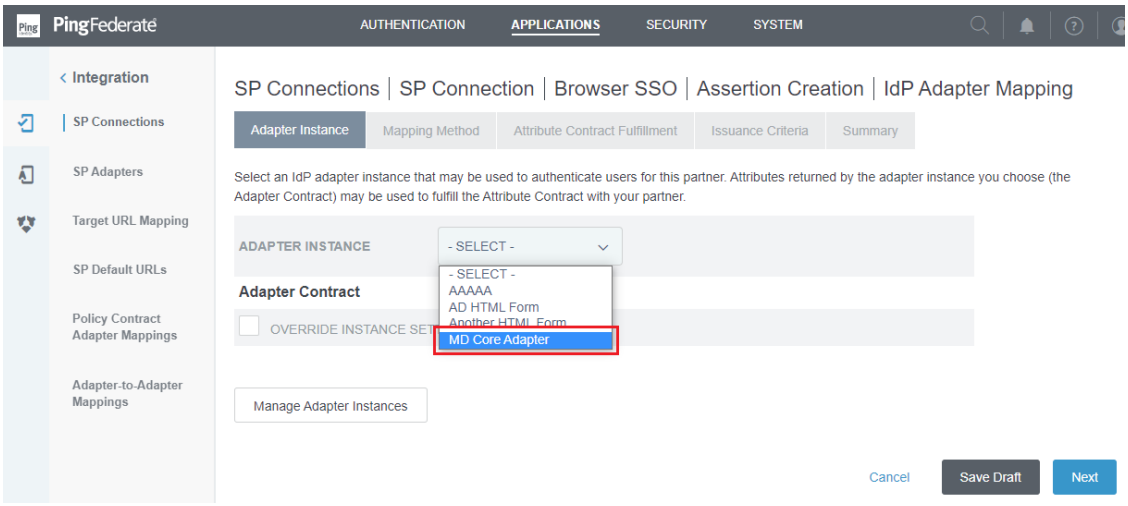
13. In `Attribute Contract` phase, navigate to `Extend the Contract`, complete the contract needed by MetaDefender Core, using 'username' as a sample. Select the basic format for `Attribute Name Format`, then click `Add` followed by `Next`.



14. In `Authentication Source Mapping` phase, select `Map New Adapter Instance`.

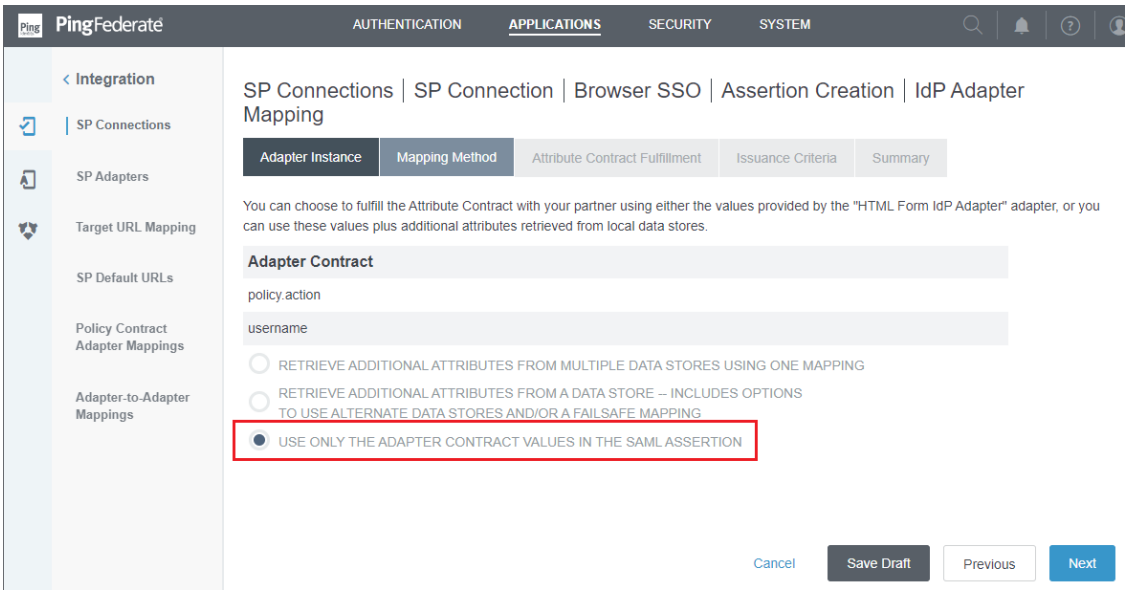


15. In `Adapter Instance` phase, choose the appropriate IdP Adapter for `ADAPTER INSTANCE` and then click `Next`.

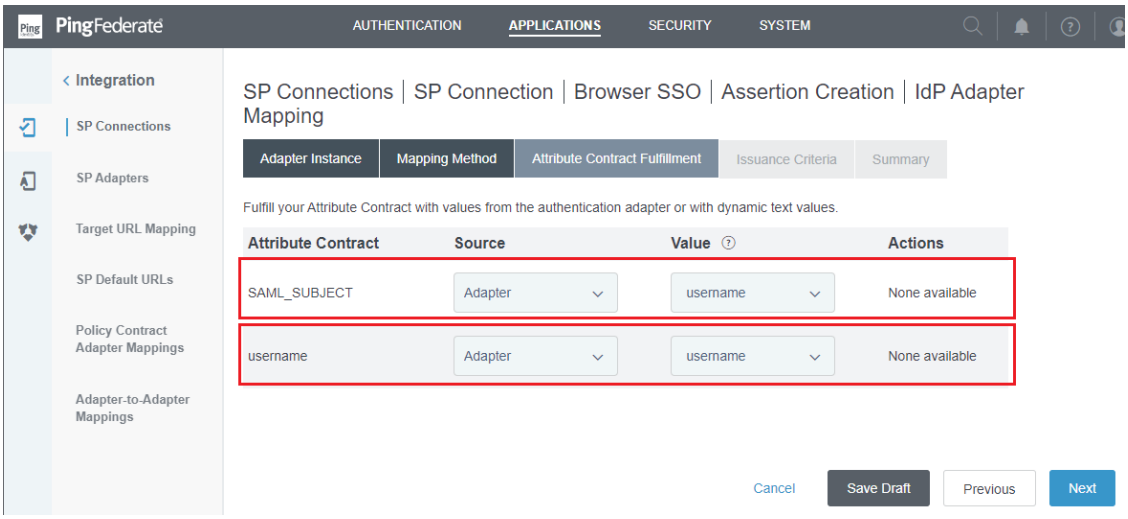


16. In `Mapping Method` phase, select `USE ONLY THE ADAPTER CONTRACT VALUES IN THE SAML ASSERTION`, then

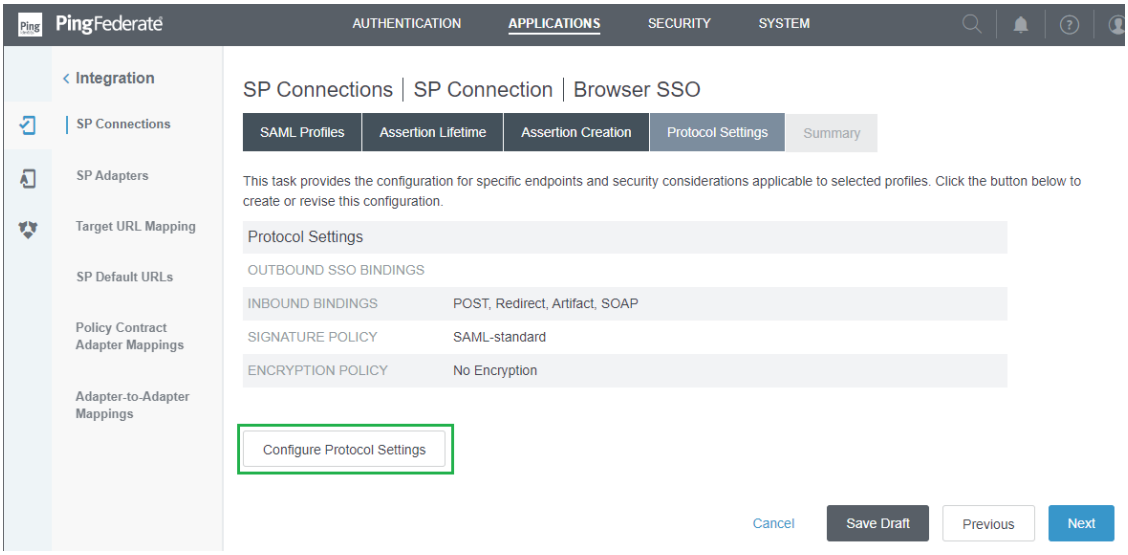
press `Next`.



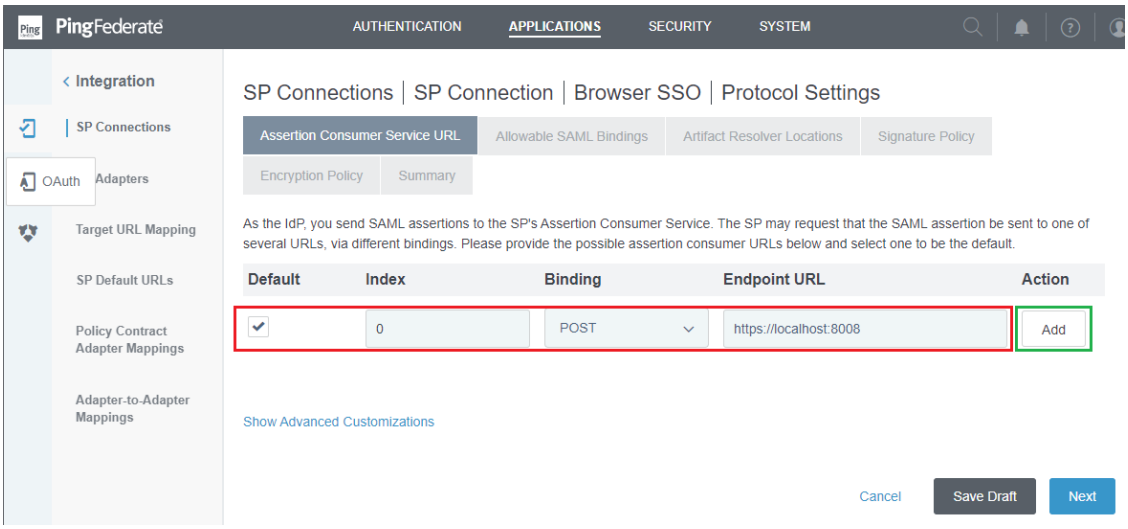
17. In `Attribute Contract Fulfillment` phase, choose 'Adapter' for `Source` column and 'username' for `Value` column, then click `Next`.



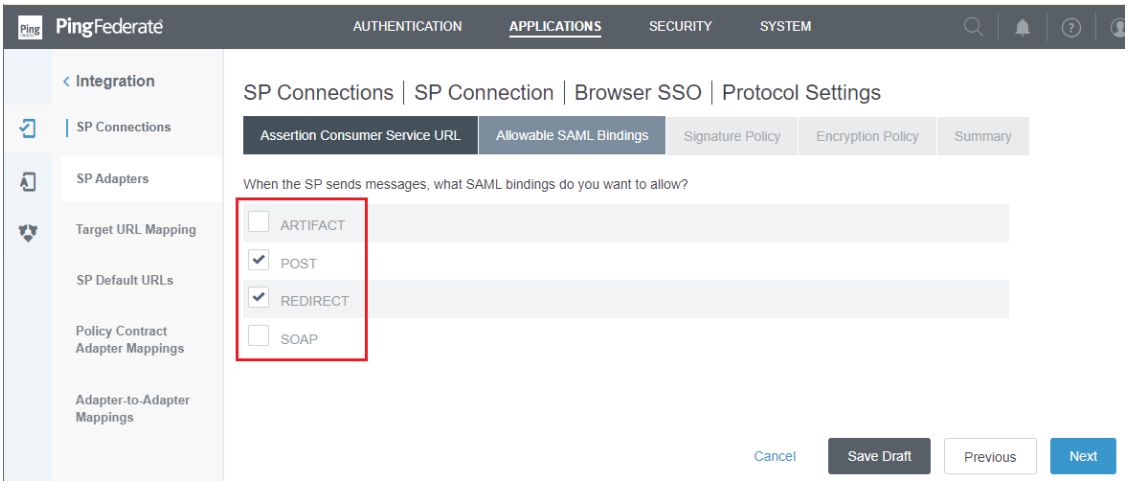
18. Press `Next` on `Issuance Criteria` phrase.
19. Examine all chosen options in `Summary`, then select `Done` and `Next`.
20. Return to `Assertion Creation` phrase, examine all chosen options in `Summary`, and then click `Done` and `Next`.
21. In `Protocol Settings` phase, select `Configure Protocol Settings`.



22. Check the box under `Default` column, enter **0** in `Index` column, select POST for `Binding`, and input the address of the machine hosting MetaDefender Core in `Endpoint URL` column, then click `Add` and `Next`.

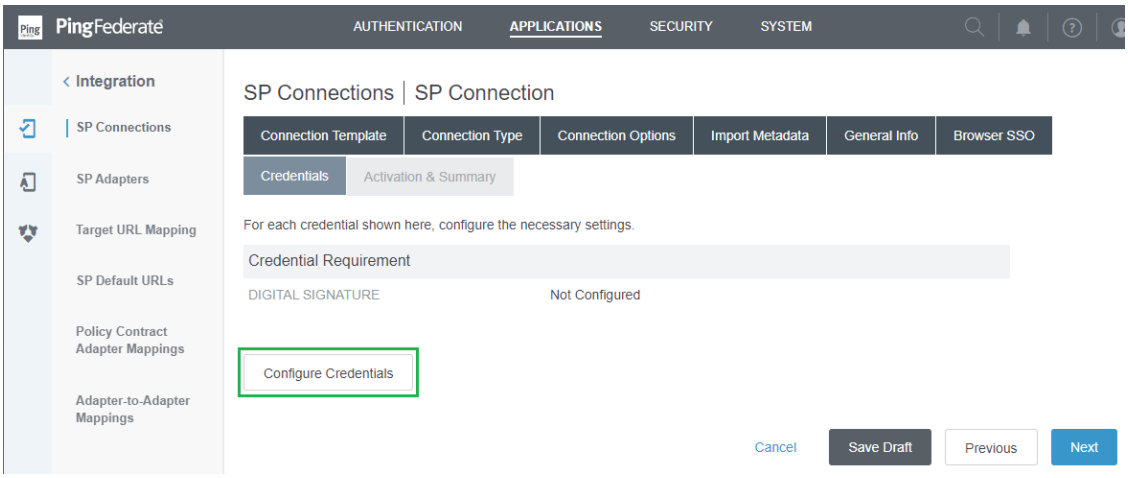


23. In `Acceptable SAML Bindings` phase, select only 'POST' and 'REDIRECT' choices, then click `Next`.

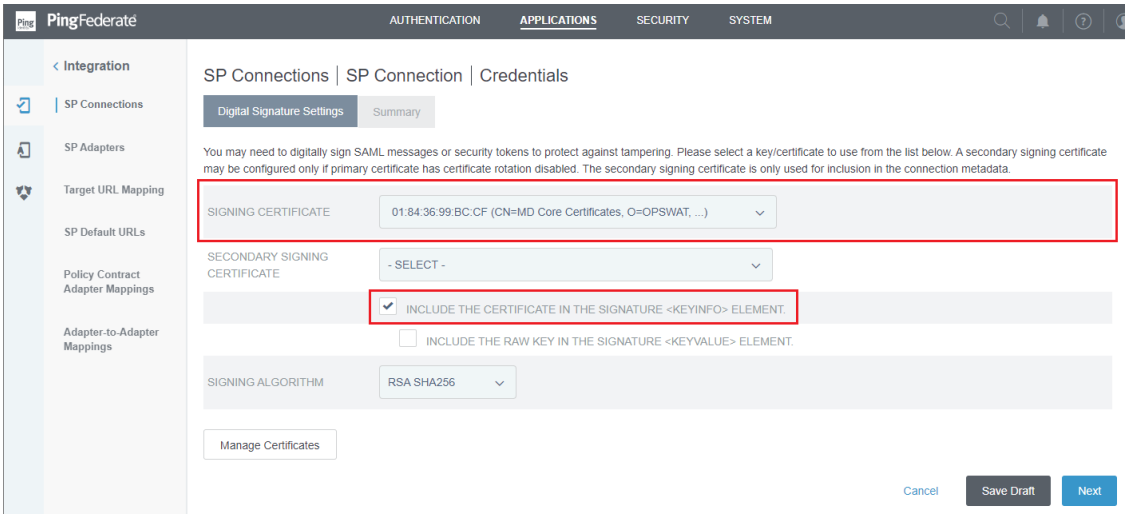


24. Press `Next` on the phrases for `Signature Policy` and `Encryption Policy`.

25. Examine all chosen options in `Summary` and click `Done`.
26. Select `Next` on `Browser SSO`.
27. Examine all chosen options in `Summary`, then select `Done` and `Next`.
28. In `Credentials` phase, select `Configure Credentials`.



29. In `Digital Signature Settings` phase, go to `SIGNING CERTIFICATE`, choose the signing signature configured in the earlier step, activate the option `INCLUDE THE CERTIFICATE IN THE SIGNATURE <KEYINFO> ELEMENT`, and then click `Next`.

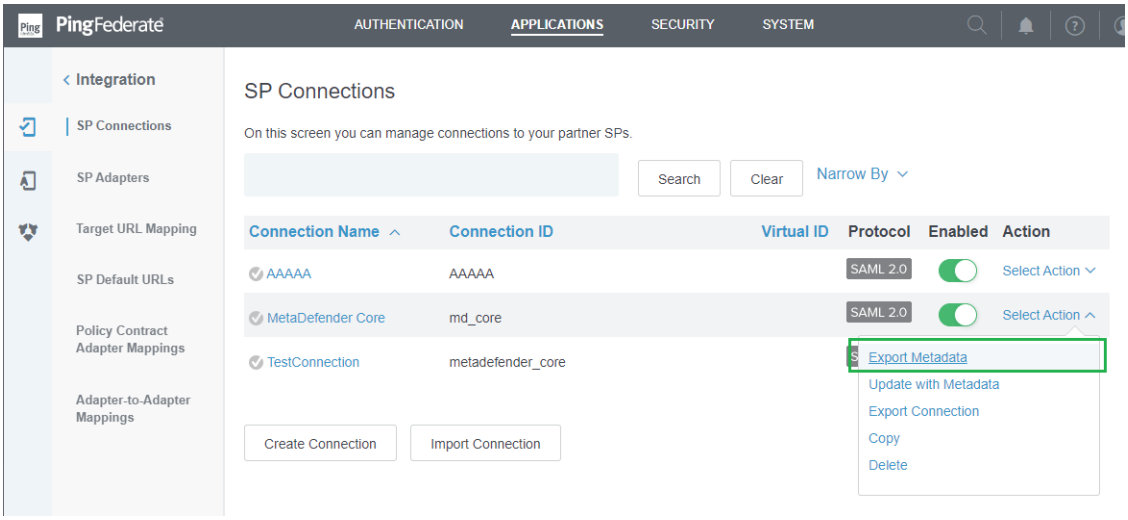


30. Examine all chosen options in `Summary` phrase, then press `Done`.
31. Return to `Credentials` tab and select `Next`.
32. Examine all chosen options in `Activation & Summary` phrase, verify that the connection is active, and then click `Save` to finish.

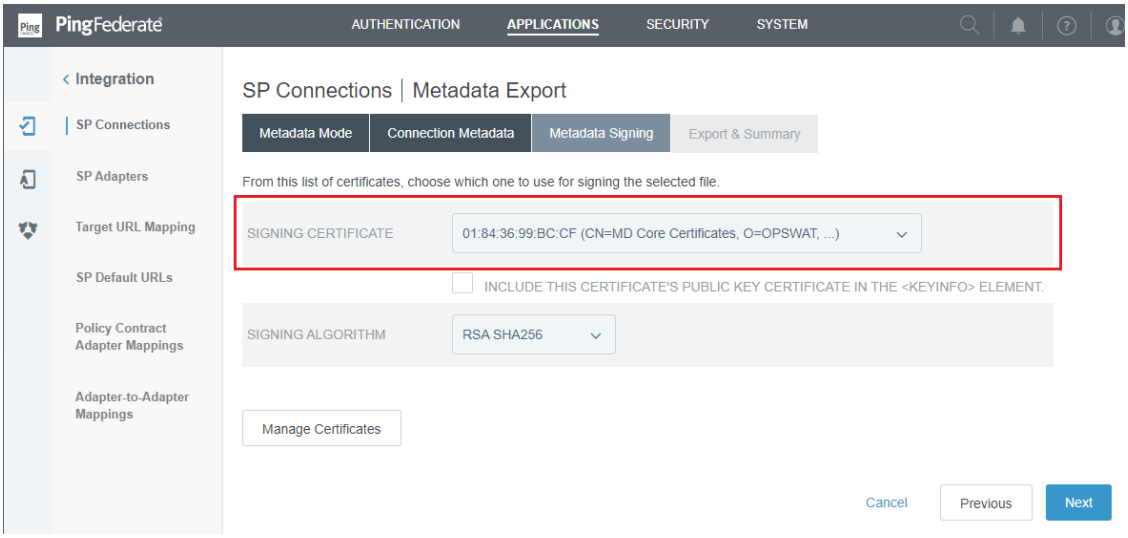


Export Identity Provider metadata

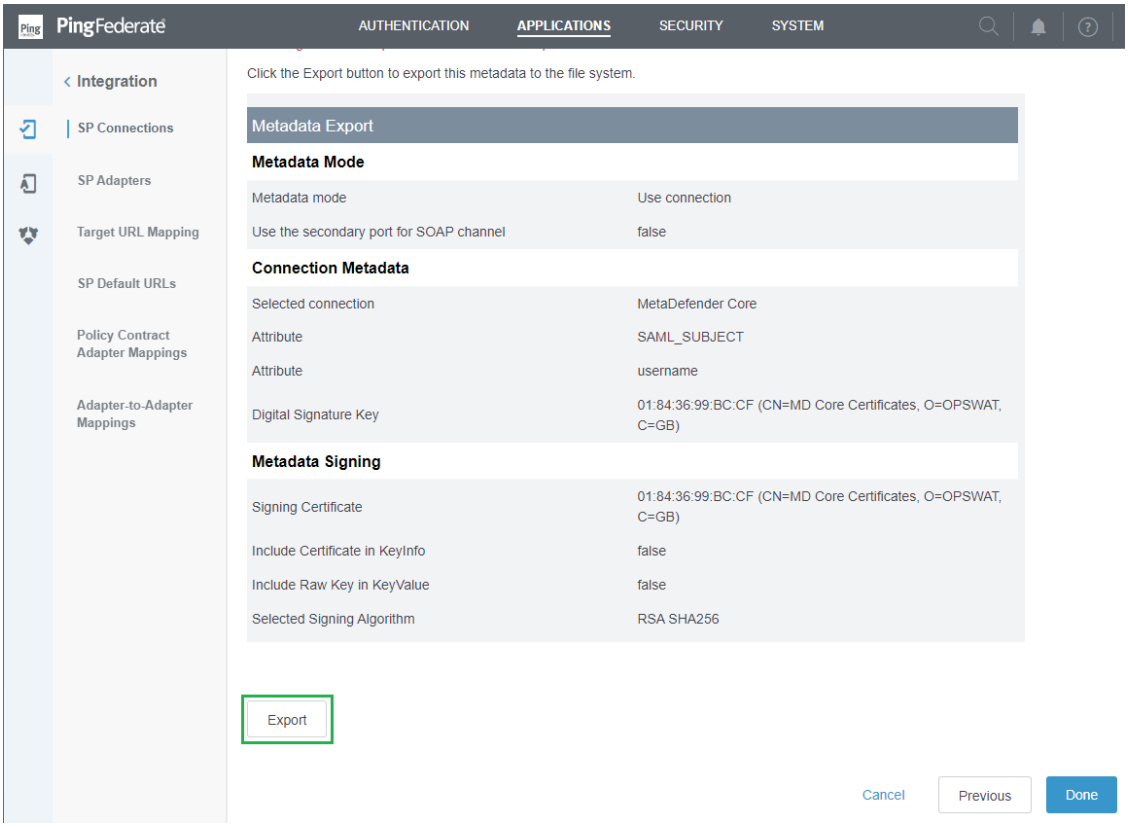
1. Go to `APPLICATIONS` and click on `SP Connections` on the left sidebar.
2. Go to the connection established in the prior step, click `Select Action` link located in `Action` column, and choose `Export Metadata`.



3. Choose the prior signing certificate, then press `Next`.



4. Examine all chosen options in `Summary` and select `Export`.



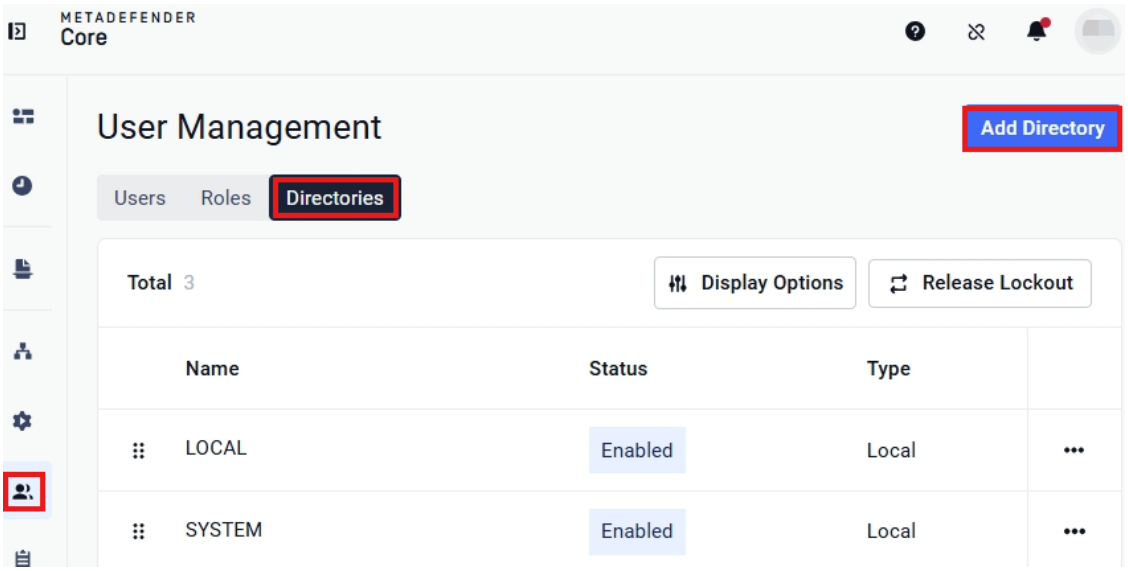
5. Save the exported file as 'metadata.xml', and then press `Done`.

Info

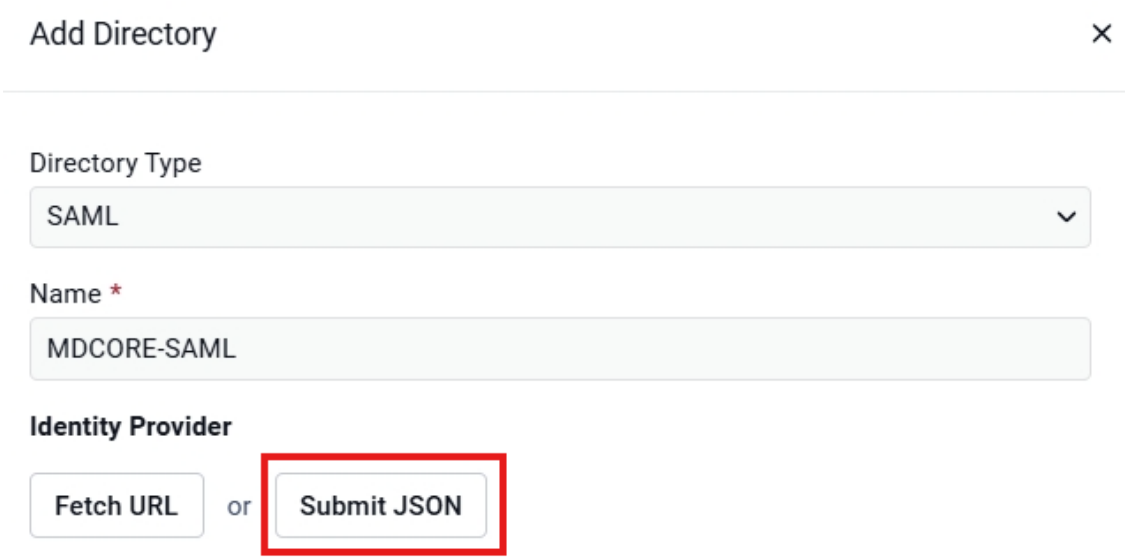
The 'metadata.xml' file will be used in the upcoming phase to set up `Identity Provider` in MetaDefender Core.

Configure SAML in MetaDefender Core

1. Sign in to MetaDefender Core.
2. In the dashboard, click on `User Management` in the left sidebar.
3. On `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.



4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCORE-SAML.
5. Click `Submit JSON` and upload the 'metadata.xml' file exported in the earlier stage.

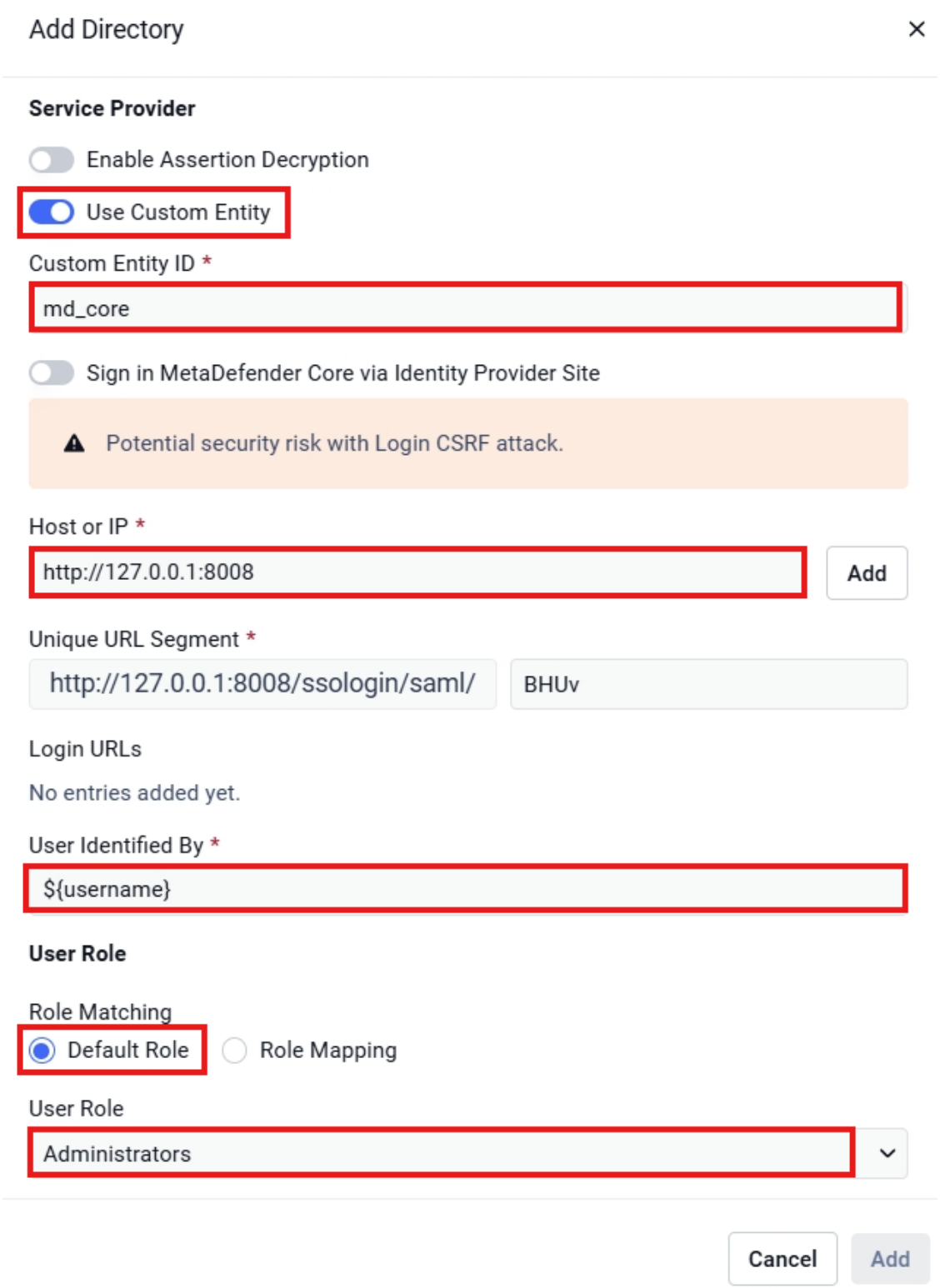


6. Navigate to `Service Provider`, toggle `Use custom entity ID` and fill in `Custom entity ID` with the partner's entity ID configured in PingFederate. That is 'md_core' for this case.
7. Enter value or `Host or IP`, click `Copy` and save it in *redirect_url*.

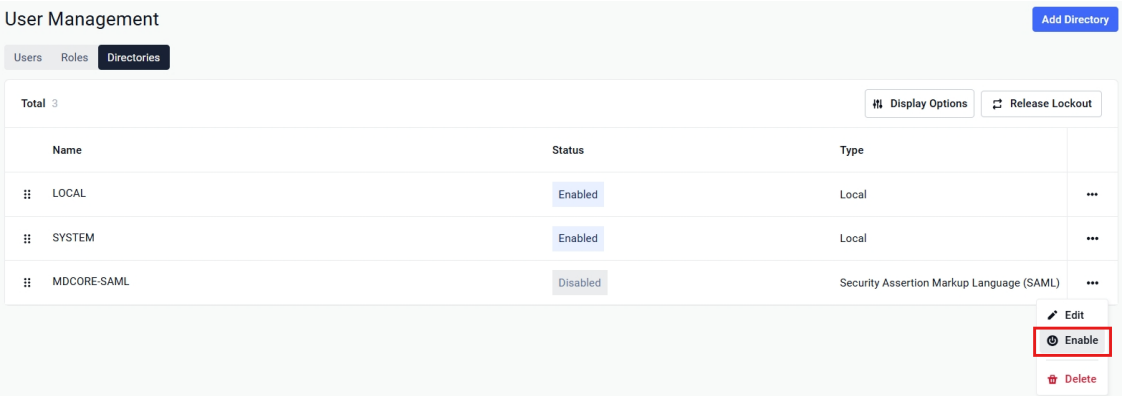
Info

redirect_url will serve to establish the value for `Endpoint URL` in the final configuration phase in PingFederate.

8. Establish `User Identified by` using ${username}.
9. Choose `Default role` and select the appropriate role for the logged-in user.
10. Press `Add` when finished.

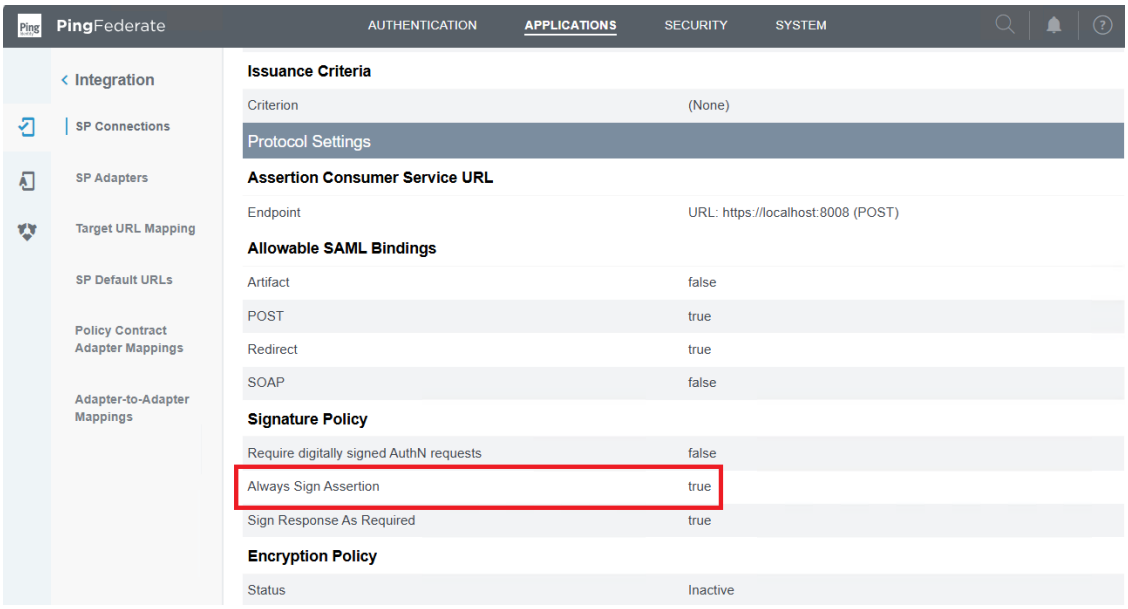


11. In `User Management` page, toggle the new directory, MDCORE-SAML, in this instance. A confirmation dialog box will pop up to verify the action. After clicking `Enable`, all sessions will end instantly.

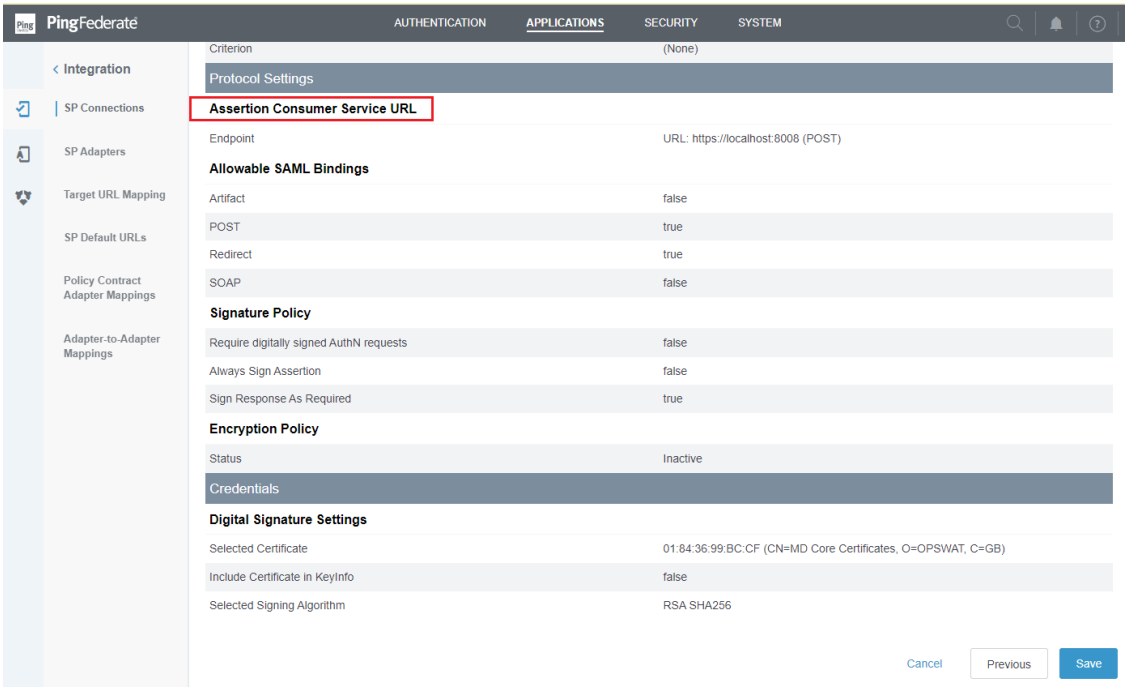


Complete configuration in PingFederate

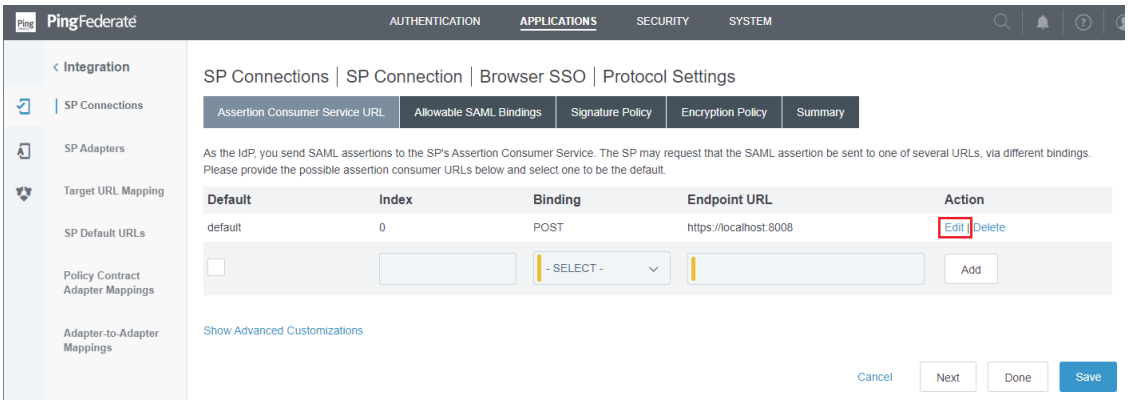
1. Access the PingFederate management console.
2. Go to `APPLICATION` and choose `SP Connections` from the left sidebar.
3. In `Signature Policy`, set `Always Sign Assertion` to 'true'.



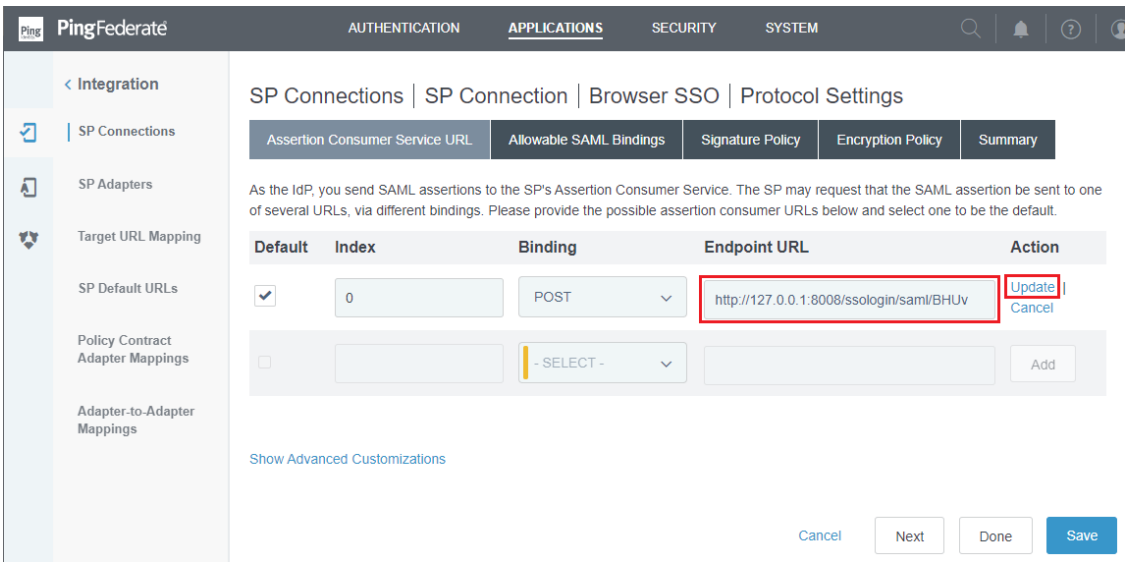
4. Go to `Assertion Consumer Service URL` and select the link next to `Endpoint`.



5. Click `Edit`.

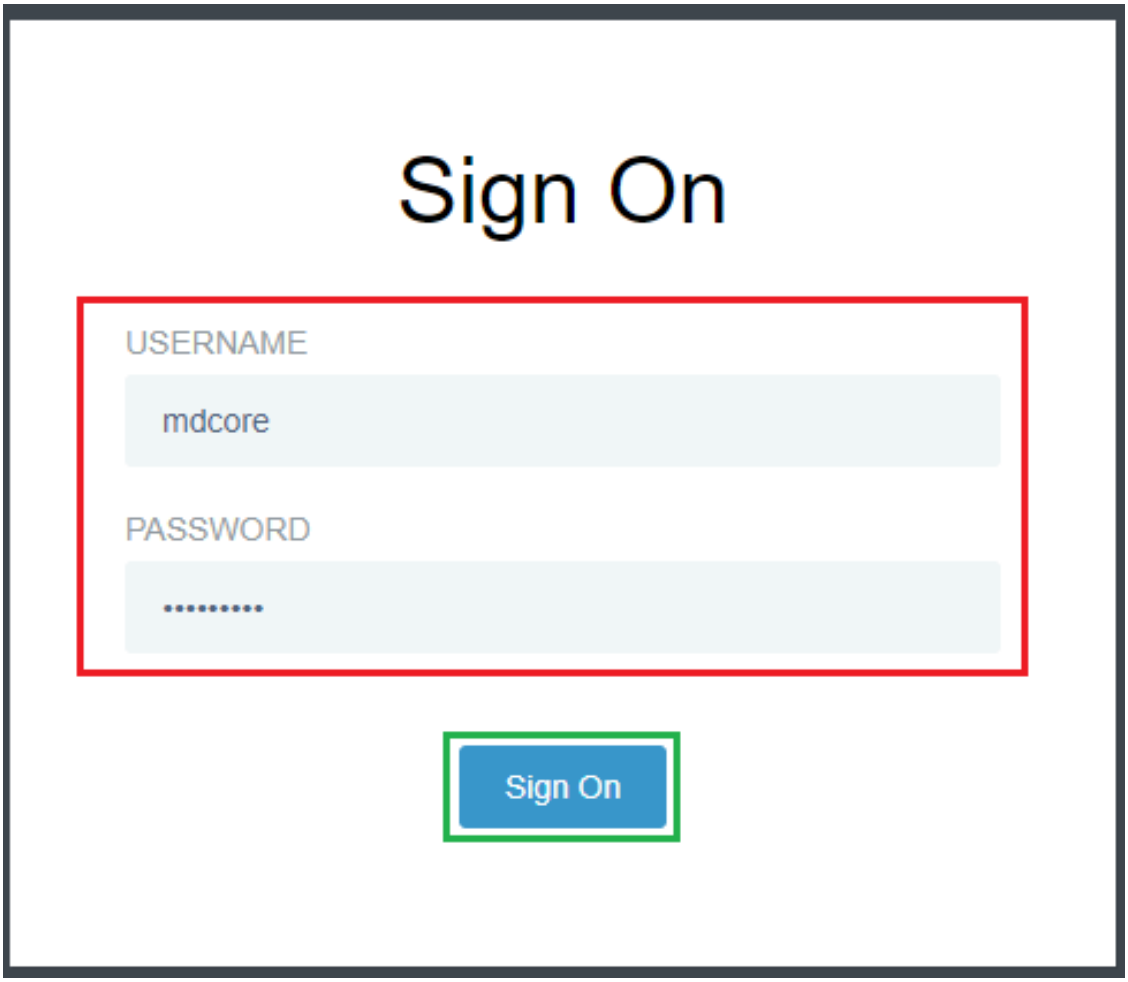


6. Modify the URL located below `Endpoint URL` to the value found in *redirect_url*, then click `Update` and `Save`.

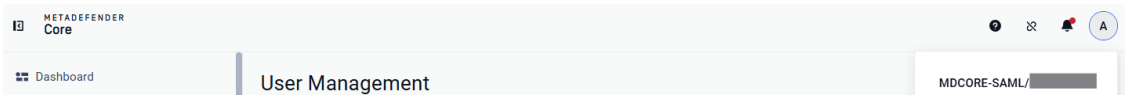


Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to PingFederate Sign-on page.



2. Sign in using the account registered with PingFederate.
3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

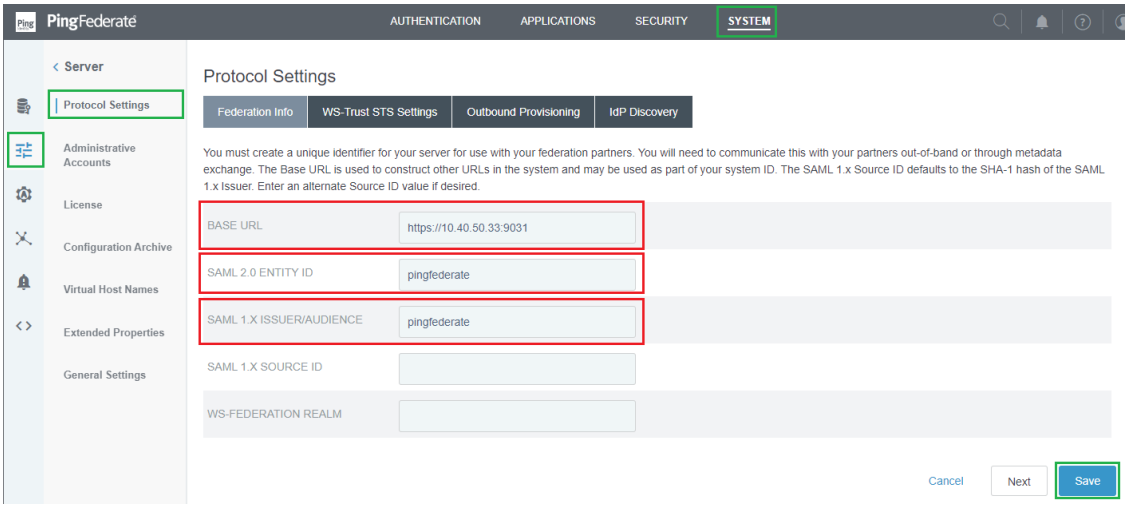
Configure OIDC SSO

Set up PingFederate

Configurate PingFederate

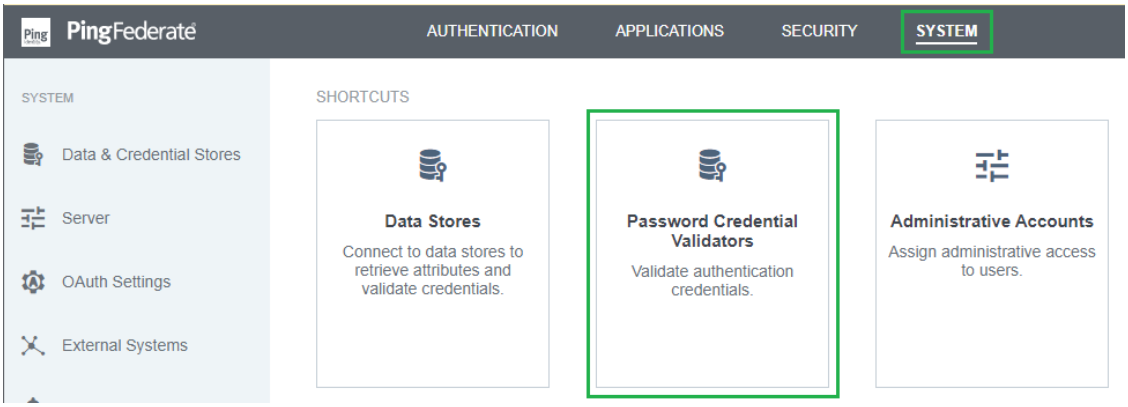
1. Sign in to PingFederate console at `https://service-ip:service-port`, where `service-ip` and `service-port` refer to the IP address and port of the machine hosting PingFederate server.

2. In homepage, navigate to `SYSTEM` tab and select `Server` from the left sidebar.
3. Select `Protocol Settings`, enter `Base URL`, `SAML 2.0 ENTITY ID,` and `SAML 1.X ISSUER/AUDIENCE`, then click `Save` to complete.

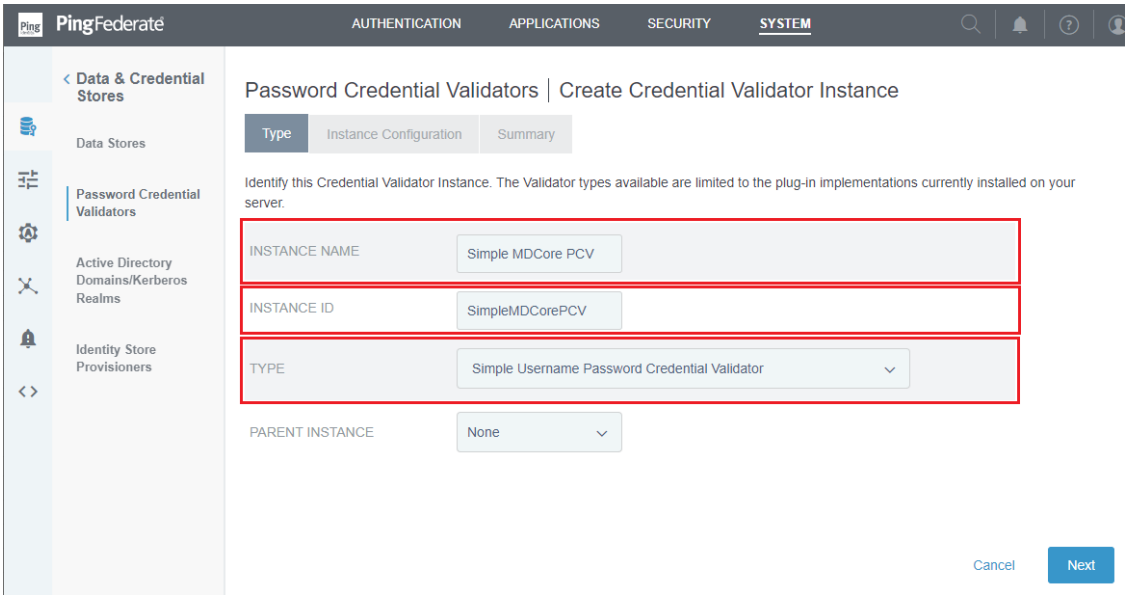


Set up simple Password Credential Validators

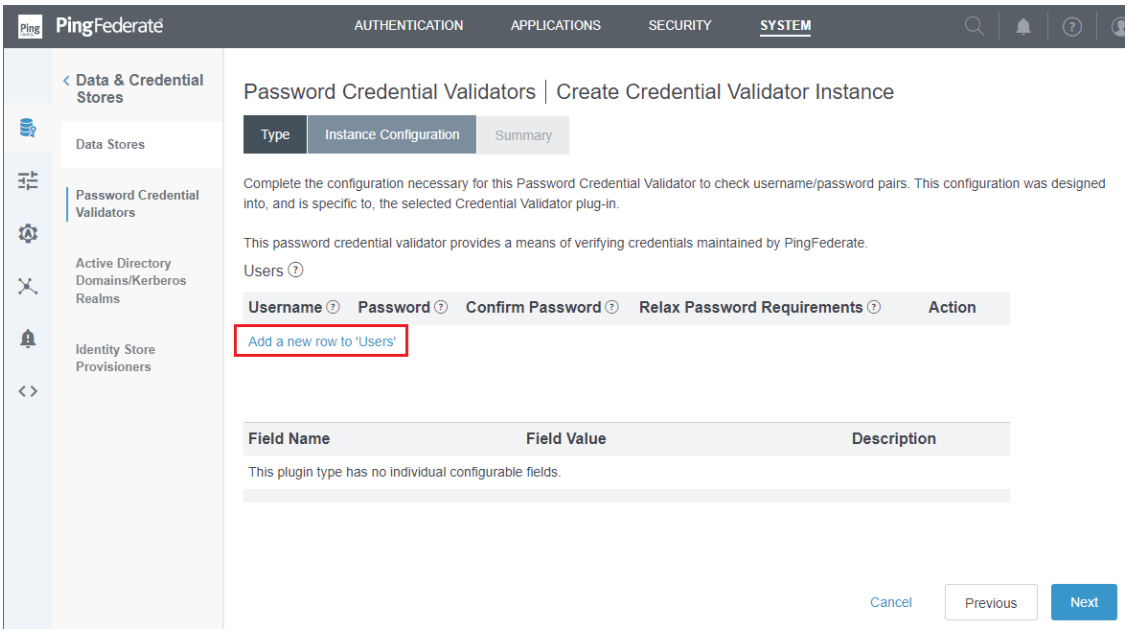
1. In homepage, navigate to `SYSTEM` tab and click on `Password Credential Validators`.



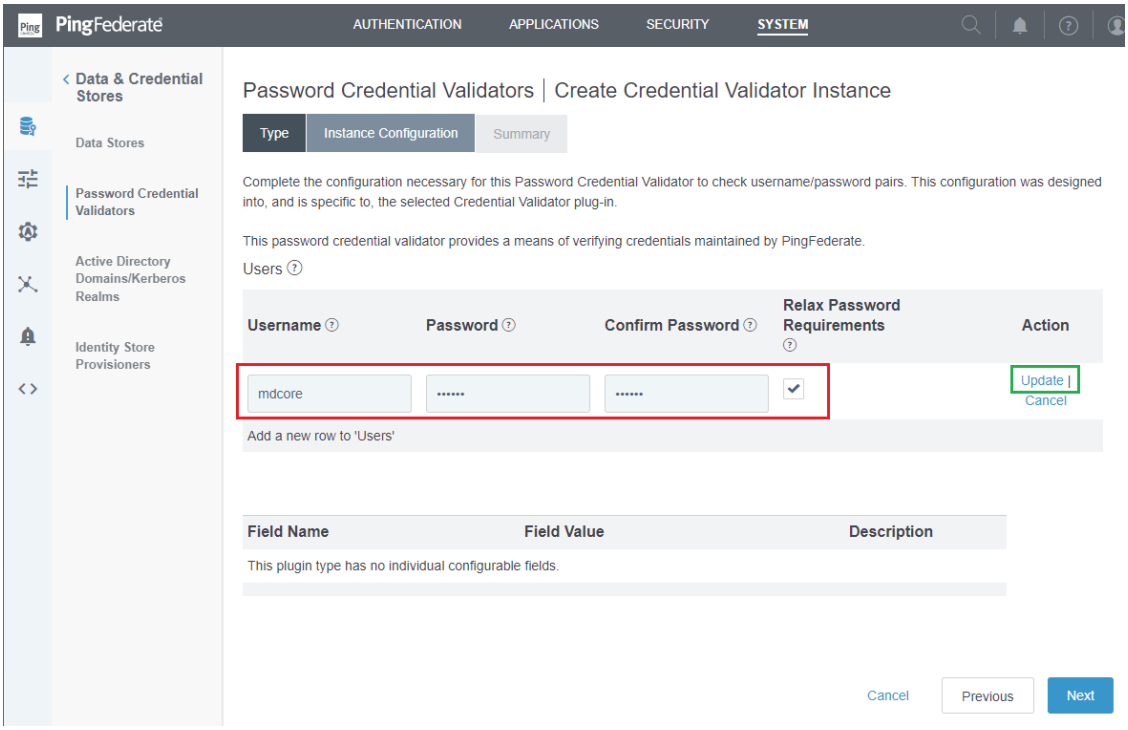
2. Click `Create New Instance`.
3. Enter `INSTANCE NAME` and `INSTANCE ID`, select `Simple Username Password Credential Validator` for `TYPE`, and click `Next`.



4. In `Create Credential Validator Instance` page, click `Add a new row to 'Users'.`



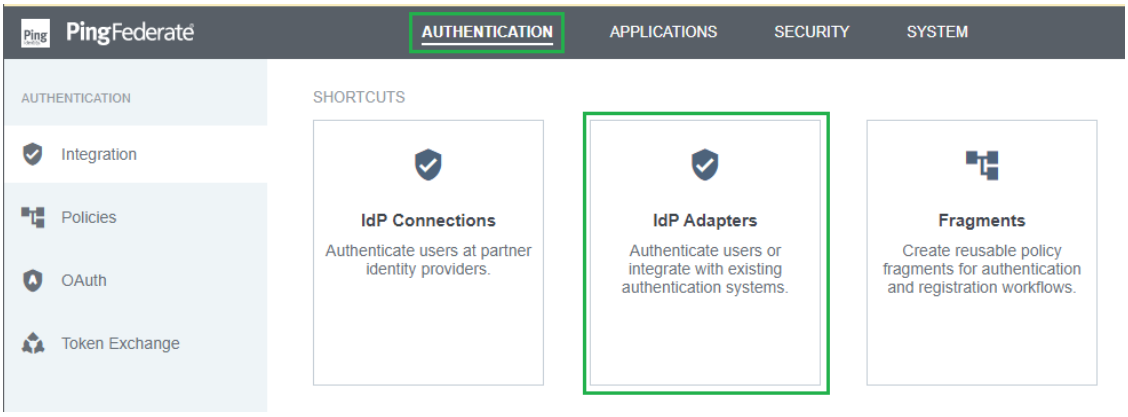
5. Enter `Username`, `Password`, and `Confirm Password`, then click `Update` and `Next`.



6. Review all selected options in `Summary` tab, and click `Save`.

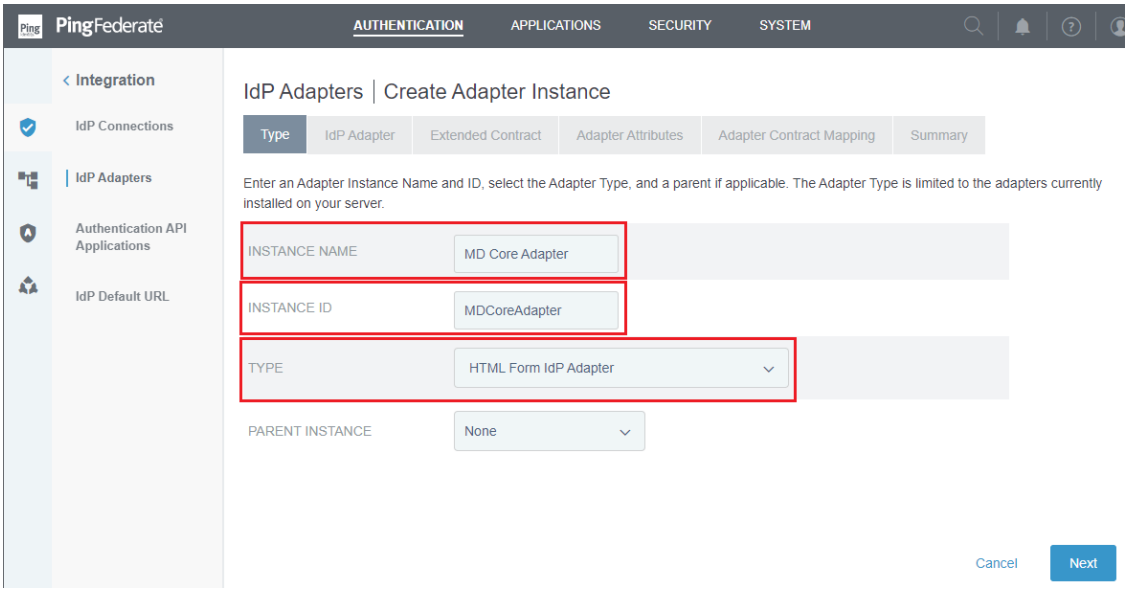
Set up a simple IdP Adapter

1. In homepage, navigate to `AUTHENTICATION` tab and click on `IdP Adapters`.

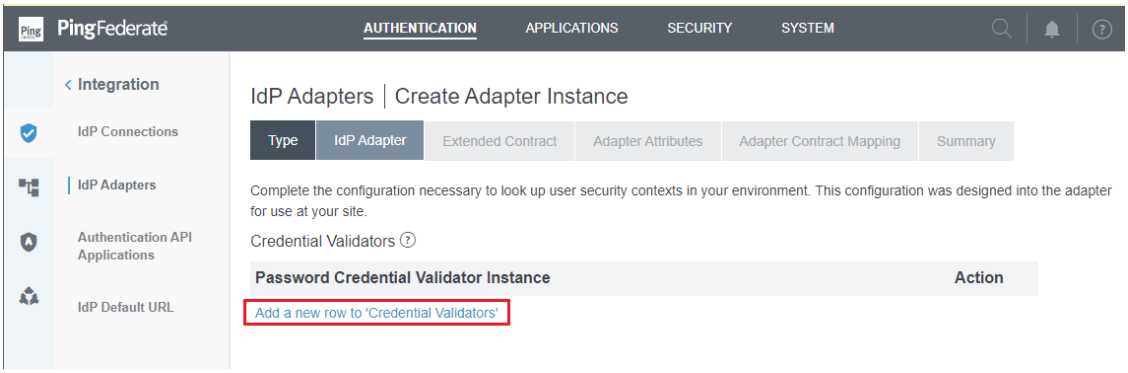


2. Click `Create New Instance`.

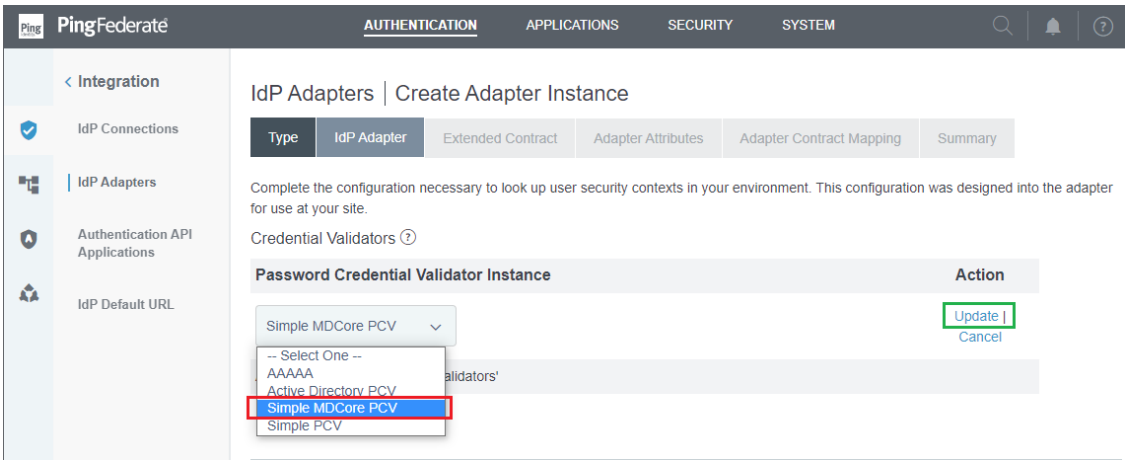
3. Enter `INSTANCE NAME`, `INSTANCE ID`, and select `HTML Form IdP Adapter` for `TYPE` . Click `Next`.



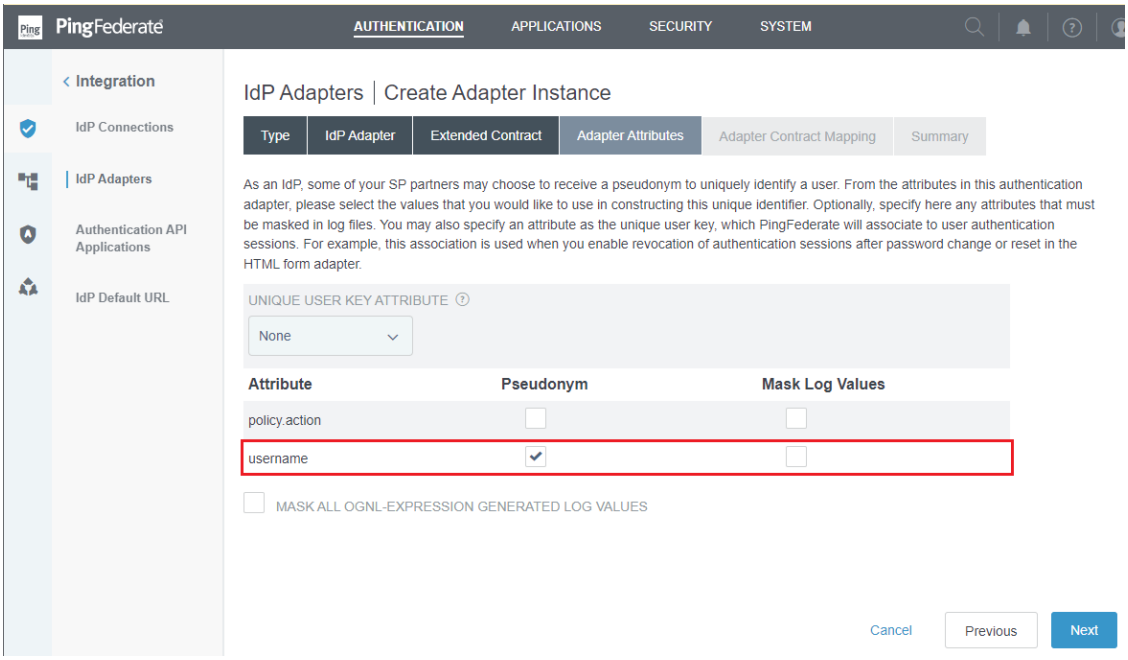
4. Click `Add a new row to 'Credential Validators'`.



5. Select the Password Credential Validator created in the previous stage, that is "Simple MDCore PCV", then click `Update` and `Next`.



6. Click `Next` on `Extended Contract` tab.
7. In `Adapter Attributes` tab, toggle `username` attribute under `Pseudonym` column, and click `Next`.

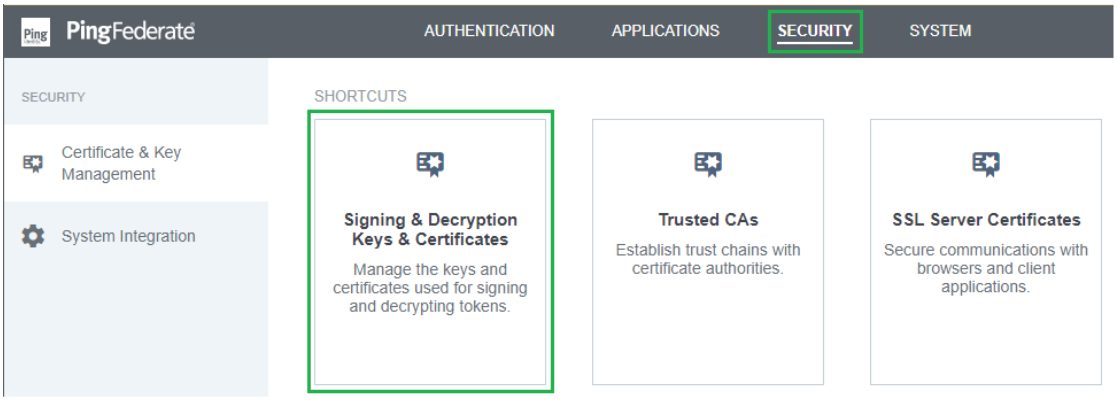


8. Click `Next` on `Adapter Contract Mapping` tab.

9. Review all selected options in `Summary` tab and click `Save` to complete.

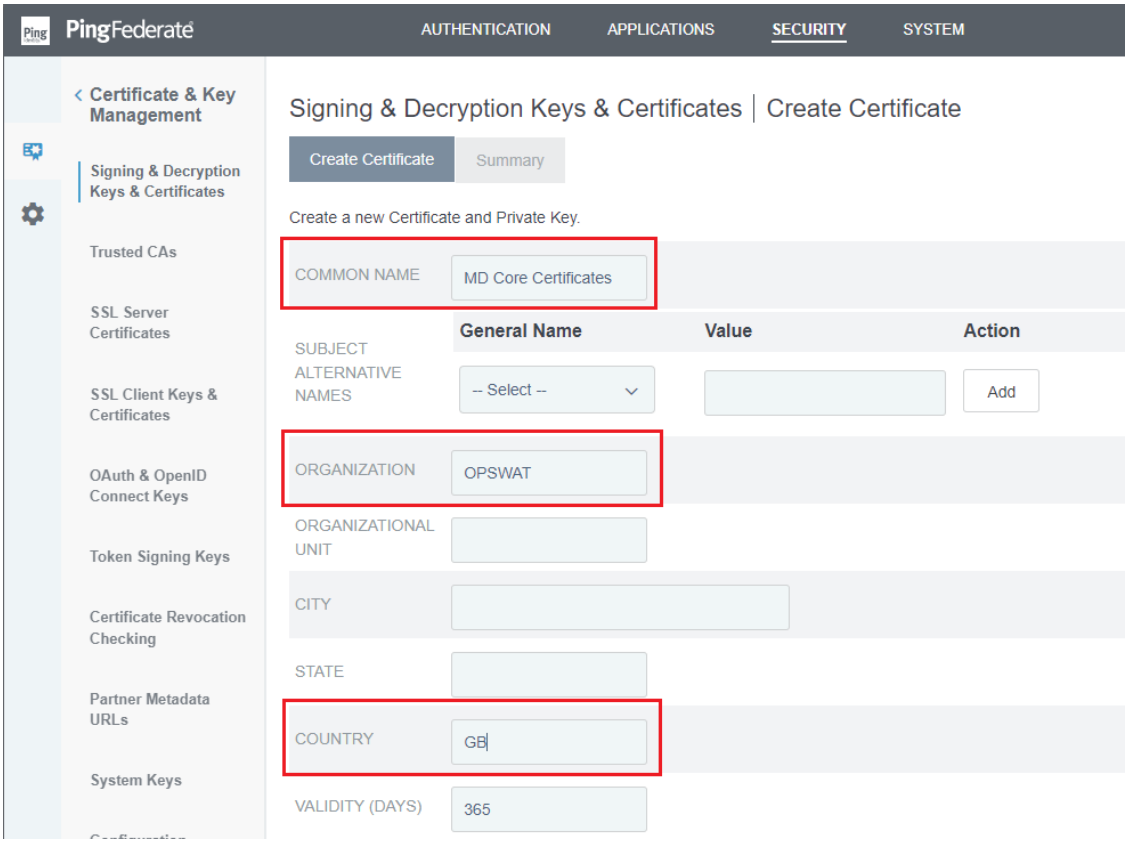
Set up Signing, Decryption Keys and Certificates

1. In homepage, navigate to `SECURITY` tab and select `Signing & Decryption Keys & Certificates`.



2. Click `Create New`.

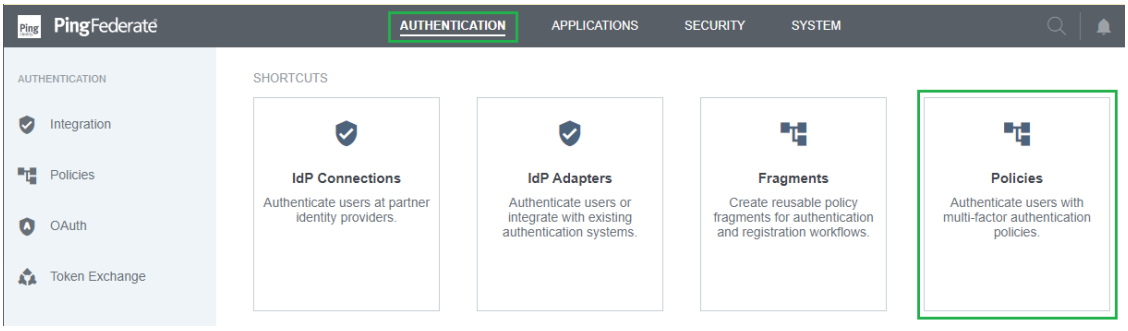
3. Enter `COMMON NAME`, `ORGANIZATION`, and `COUNTRY`, along with any other necessary information, then click `Next`.



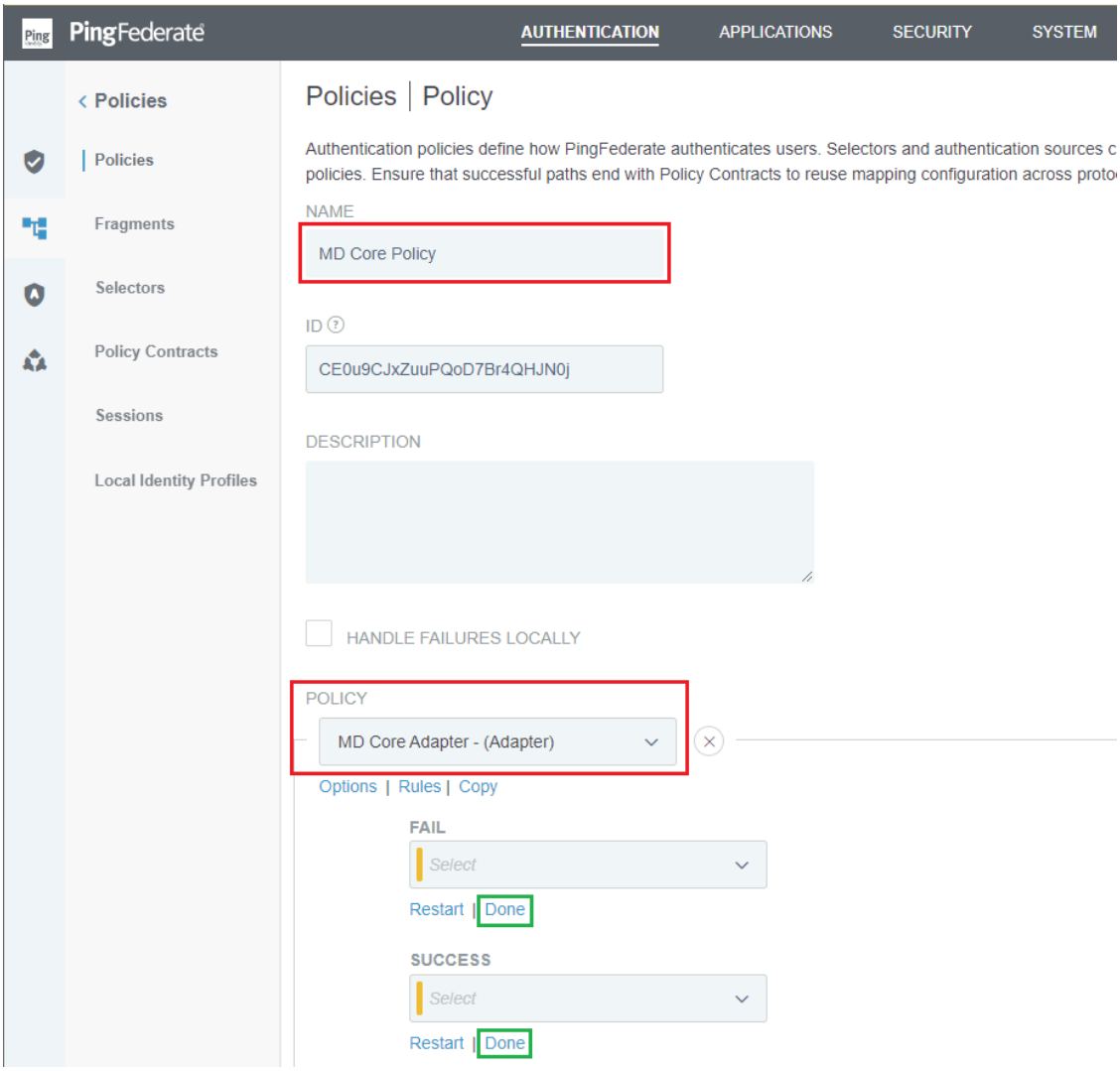
4. Review all selected options in `Summary` tab and click `Save` to complete.

Set up Authentication Policies

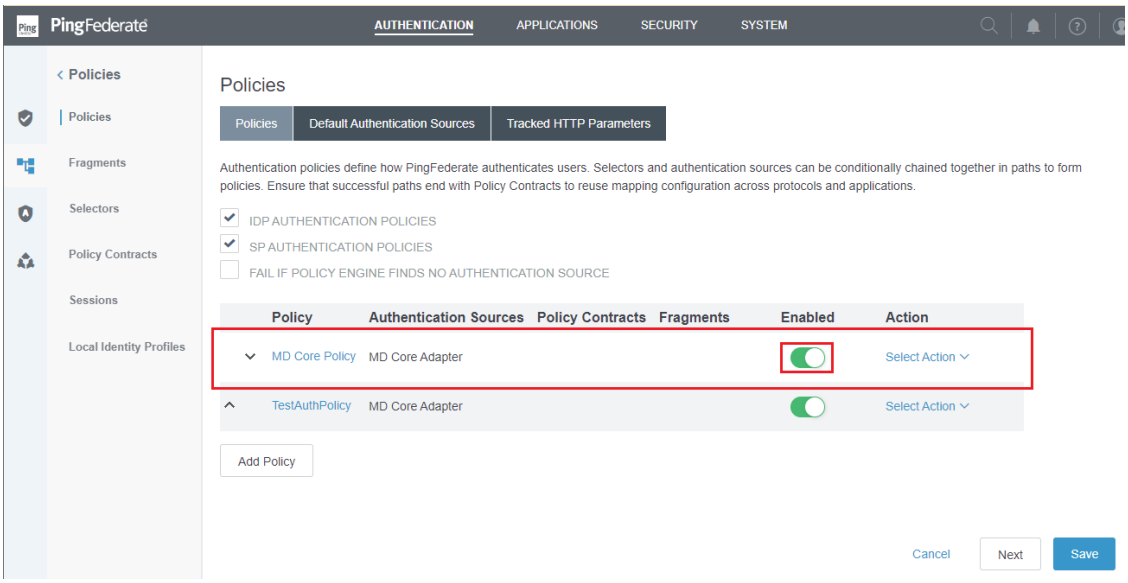
1. In homepage, navigate to `AUTHENTICATION` tab, and click `Policies`.



2. Click `Add Policy` in the next page.
3. Enter `NAME`, select the IdP Adapter from the previous stage for `POLICY`. Click `Done` for `FAIL` and `SUCCESS` options. Once finished, click `Done`.

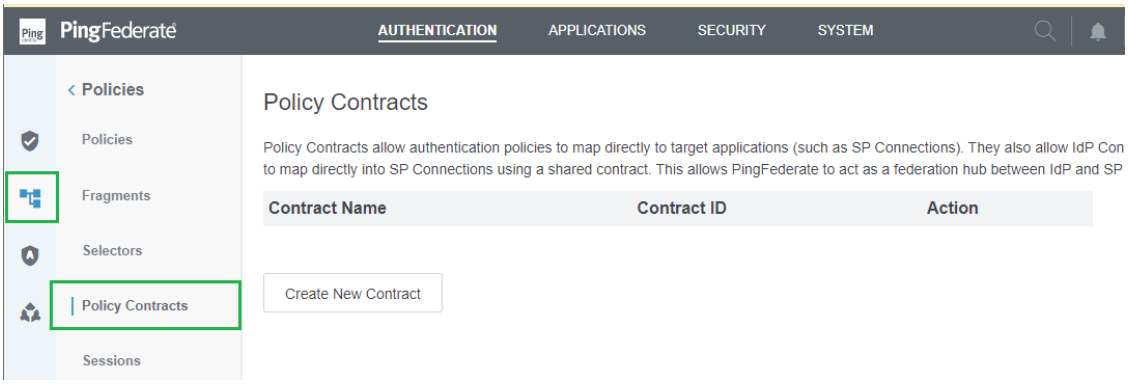


4. Ensure the new policy is enabled, then click `Save` to complete.

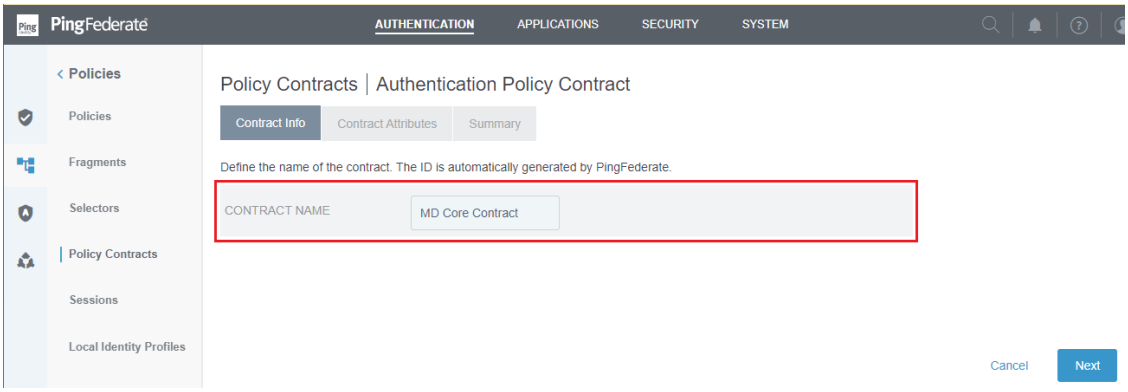


Set up Policy Contracts

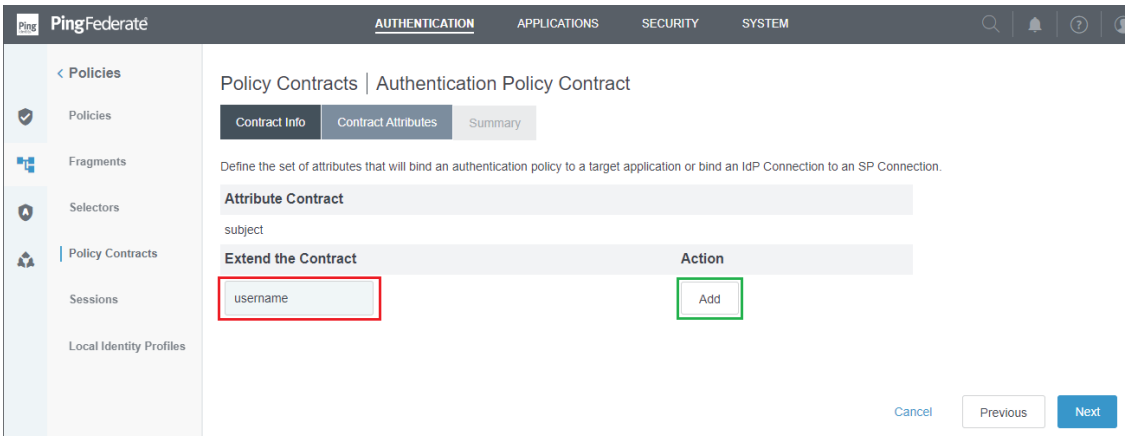
1. In homepage, navigate to `AUTHENTICATION` tab. Select `Policies` icon on the left sidebar, then click `Policy Contracts`.



2. Press `Create New Contract` on the next page.
3. In `Contract Info` step, enter `CONTRACT NAME` and click `Next`.



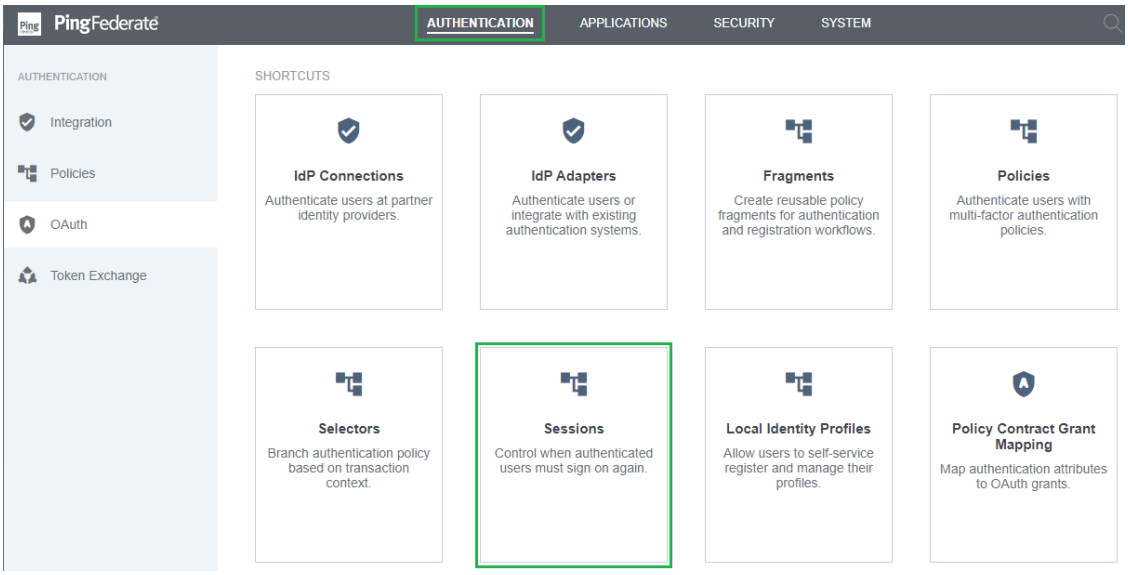
4. In `Contract Attributes` step, navigate to `Extend the Contract`, fill in claims that will be returned to MetaDefender Core, click`Add` to add the claim, and then `Next`.



5. Review all selected options in `Summary` tab, then click `Save` to complete.

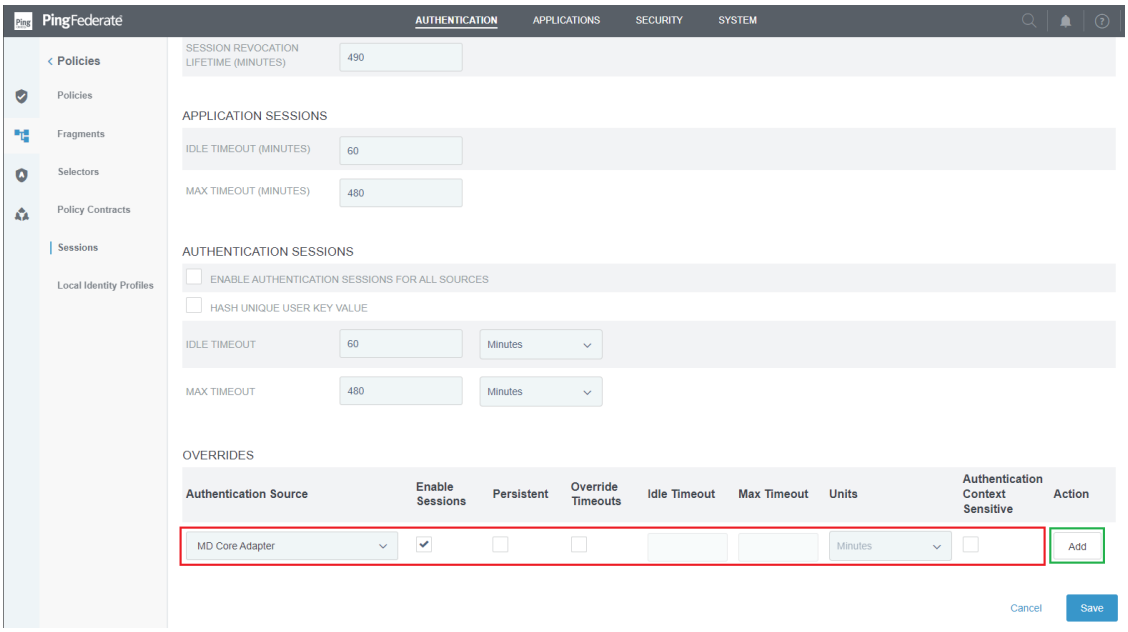
Set up Policy Sessions

1. In homepage, navigate to `AUTHENTICATION` tab and click `Sessions`.



2. In `Sessions` menu, find `OVERRIDES` subtab.

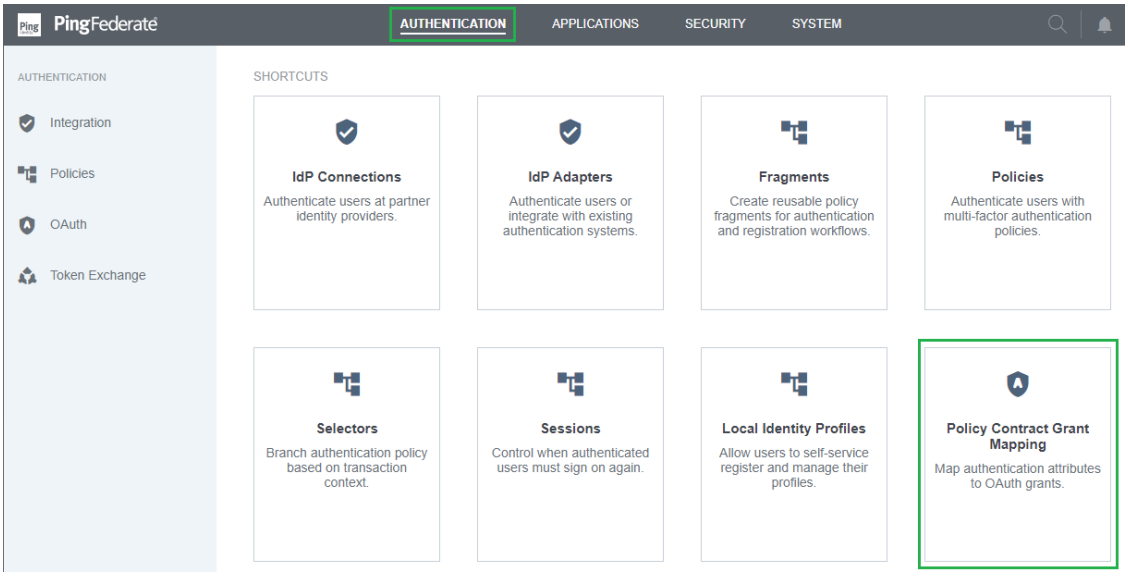
3. Select "MD Core Adapter" for Authentication Source, tick `Enable Sessions`, and click `Add` to include the authentication source.



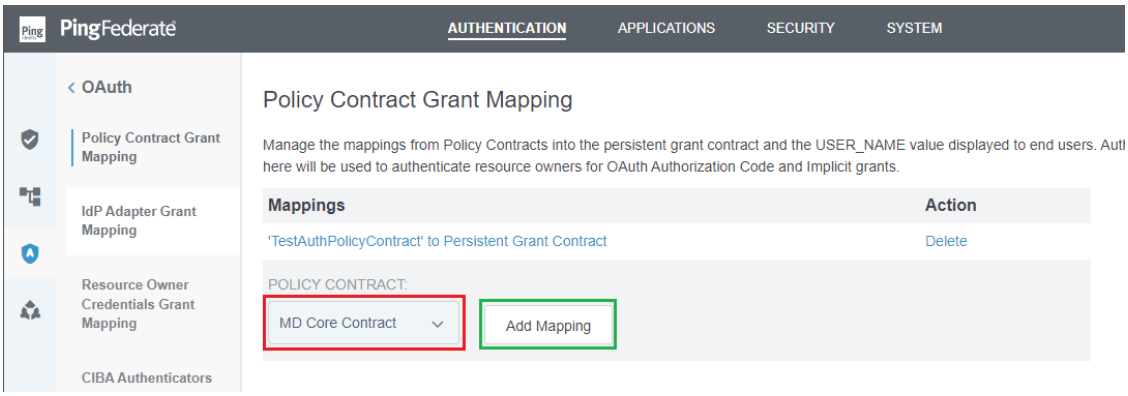
4. Click `Save` to complete.

Set up Policy Contract Grant Mapping

1. In homepage, navigate to `AUTHENTICATION` and click `Policy Contract Grant Mapping`.

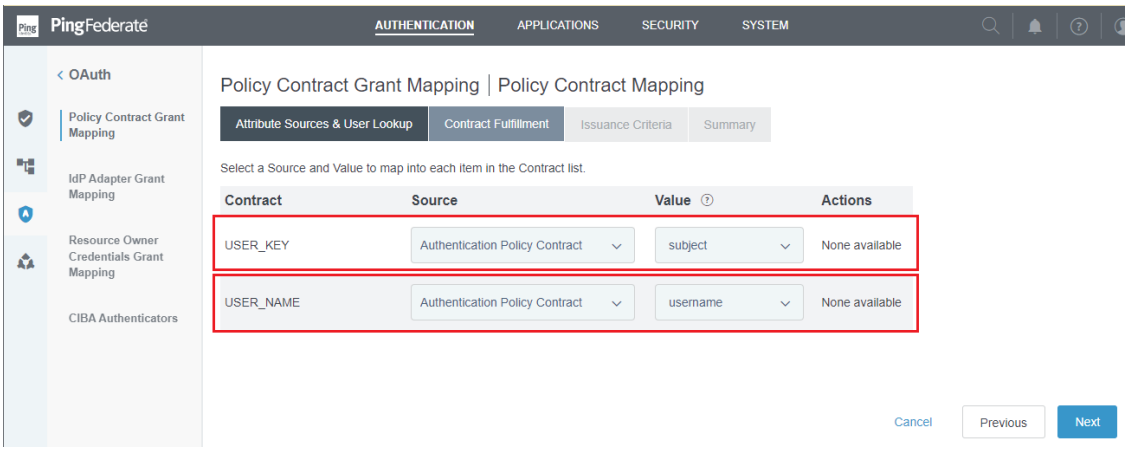


2. Select the Policy contract in previous stage, then click `Add Mapping`.



3. In `Attribute Sources & User Lookup` step, click `Next`.

4. In `Contract Fulfillment` step, select the appropriate mapping for `Source` and `Value` columns, then click `Next`.

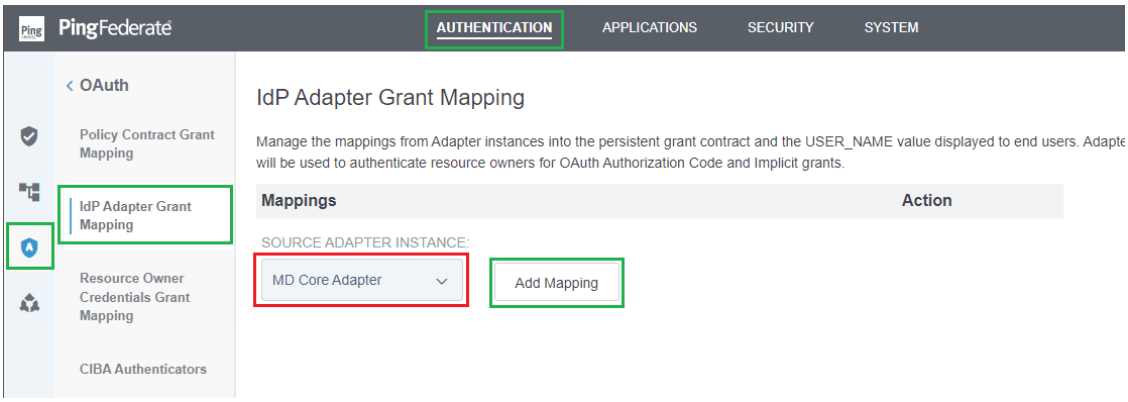


5. Click `Next` in `Issuance Criteria` step.

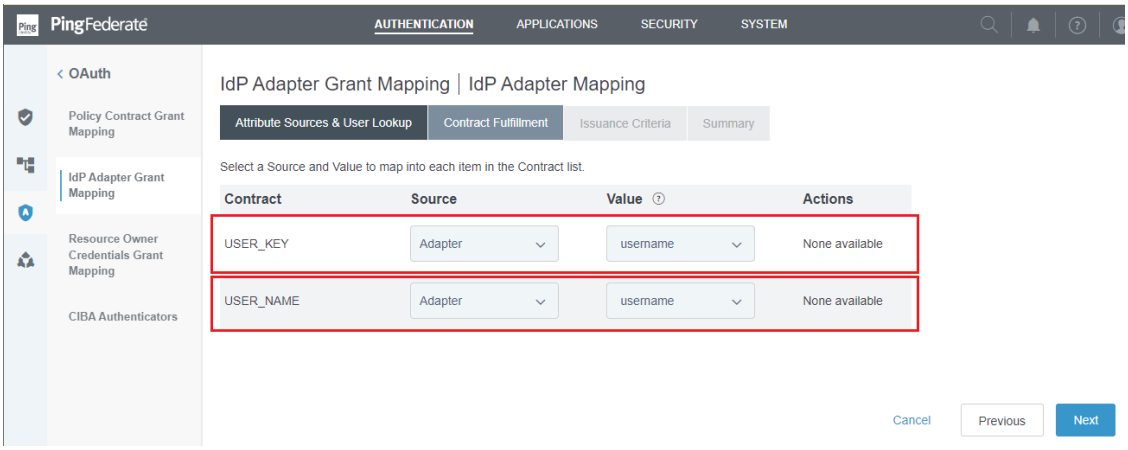
6. Review all selected options in `Summary` tab, then click `Save` to complete.

Set up IdP Adapter Grant Mapping

1. In homepage, navigate to `AUTHENTICATION`, select `OAUTH` in the left side bar, and click `IdP Adapter Grant Mapping`.
2. Under `SOURCE ADAPTER INSTANCE`, select the IdP Adapter that was set up in the previous stage and click `Add Mapping`.



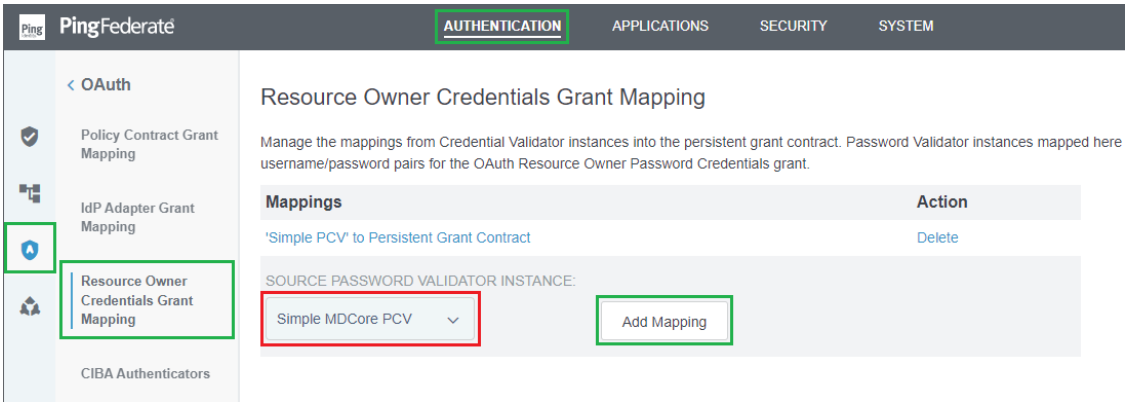
3. Click `Next` in `Attribute Sources & User Lookup` step.
4. In `IdP Adapter Mapping` step, select the appropriate mapping for `Source`, `Value`, and click `Next`.



5. Click `Next` in `Issuance Criteria` step.
6. Review all selected options in `Summary` tab, then click `Save` to complete.

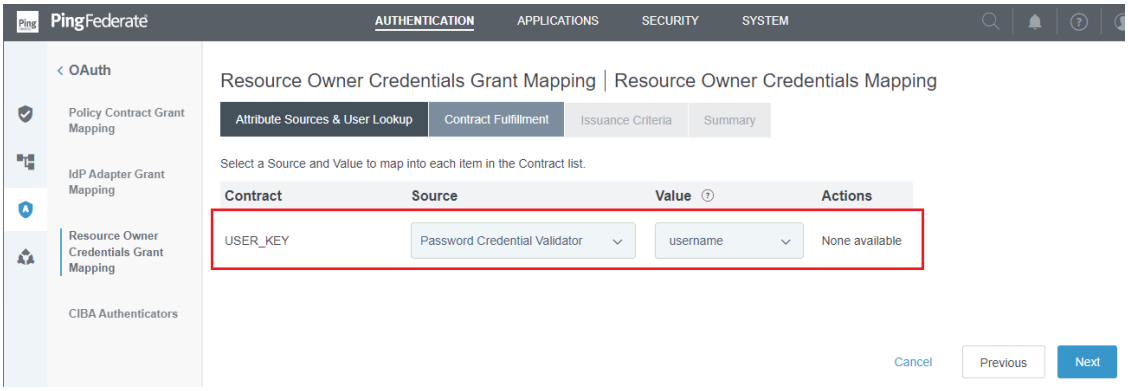
Set up Resource Owner Credentials Grant Mapping

1. In homepage, navigate to `AUTHENTICATION`, select `OAUTH` in the left side bar, and click `Resource Owner Credentials Grant Mapping`.
2. Under `SOURCE PASSWORD VALIDATOR INSTANCE`, select the Password Credential Validator that was set up in the previous stage, which is "Simple MDCore PCV", and click `Add Mapping`.



3. Click `Next` in `Attribute Sources & User Lookup` step.

4. In `Contract Fulfillment` step, choose the appropriate mapping for `Source` and `Value`, and then click `Next`.

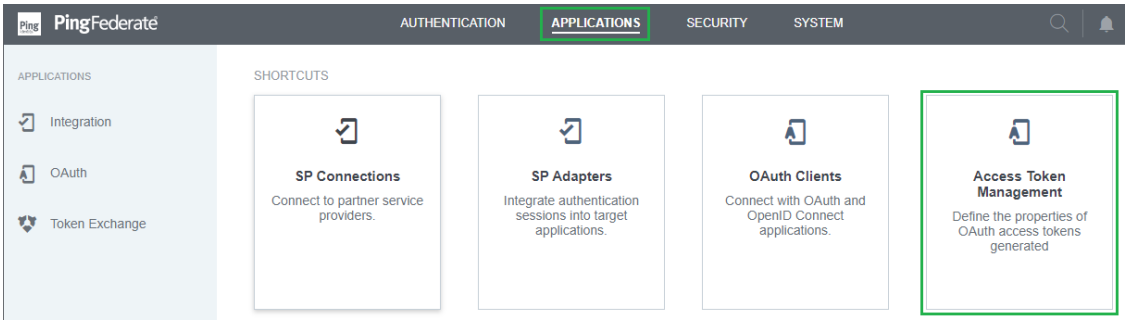


5. Click `Next` in `Issuance Criteria` step.

6. Review all selected options in `Summary` tab, then click `Save` to complete.

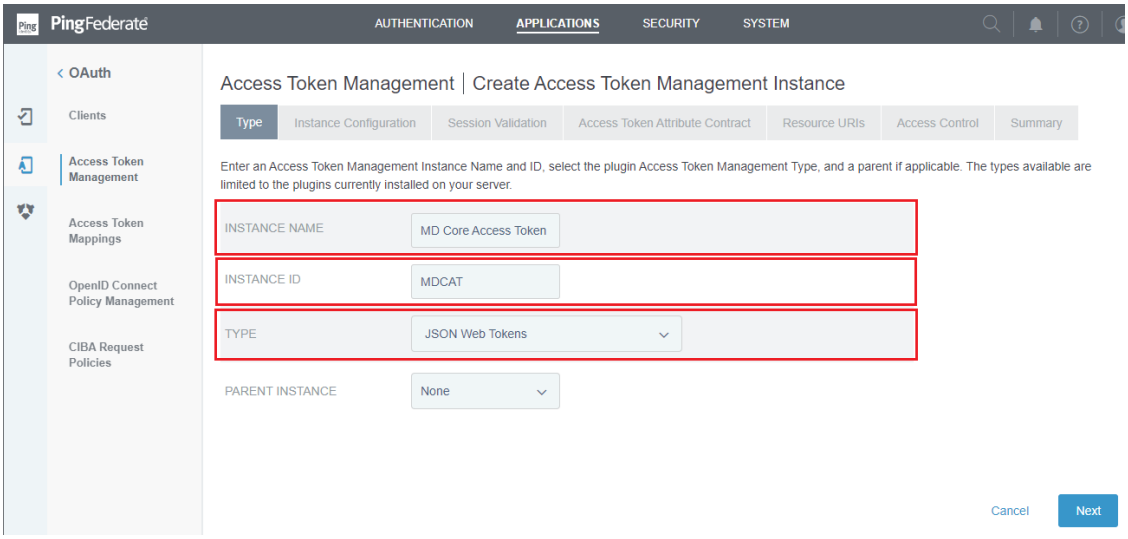
Set up Access Token Management

1. In homepage, navigate to `APPLICATIONS`, and click `Access Token Management`.

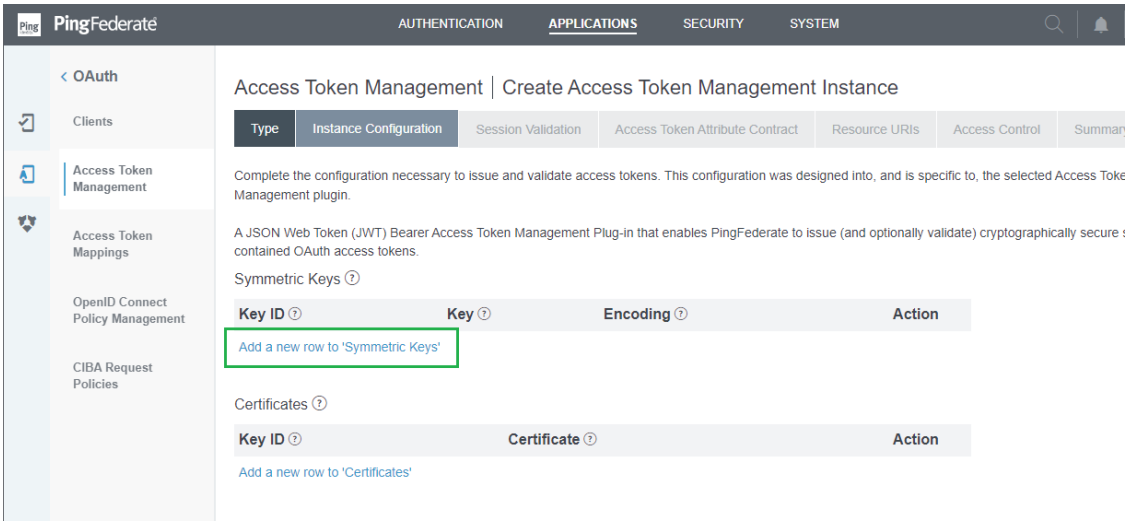


2. Click `Create New Instance`.

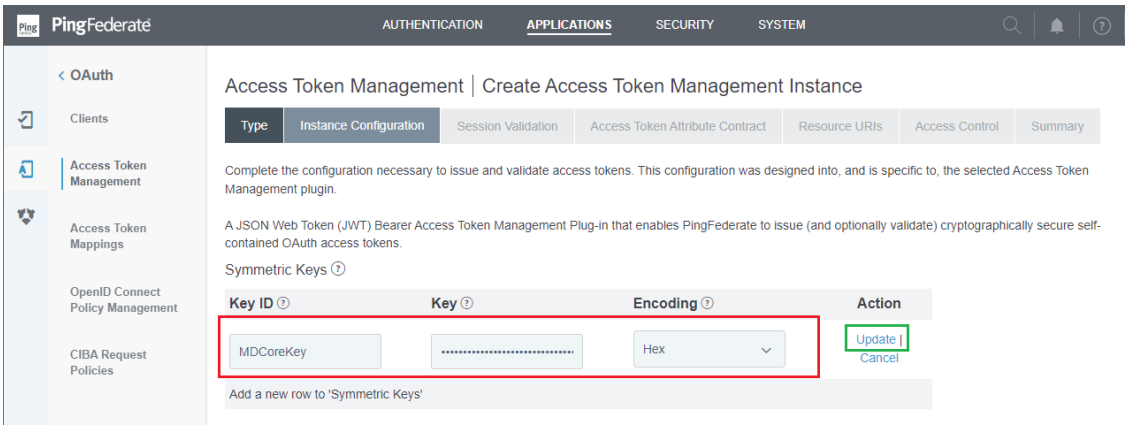
3. Enter `INSTANCE NAME`, `INSTANCE ID`, and select `JSON Web Tokens` for `TYPE`. Click `Next` to continue.



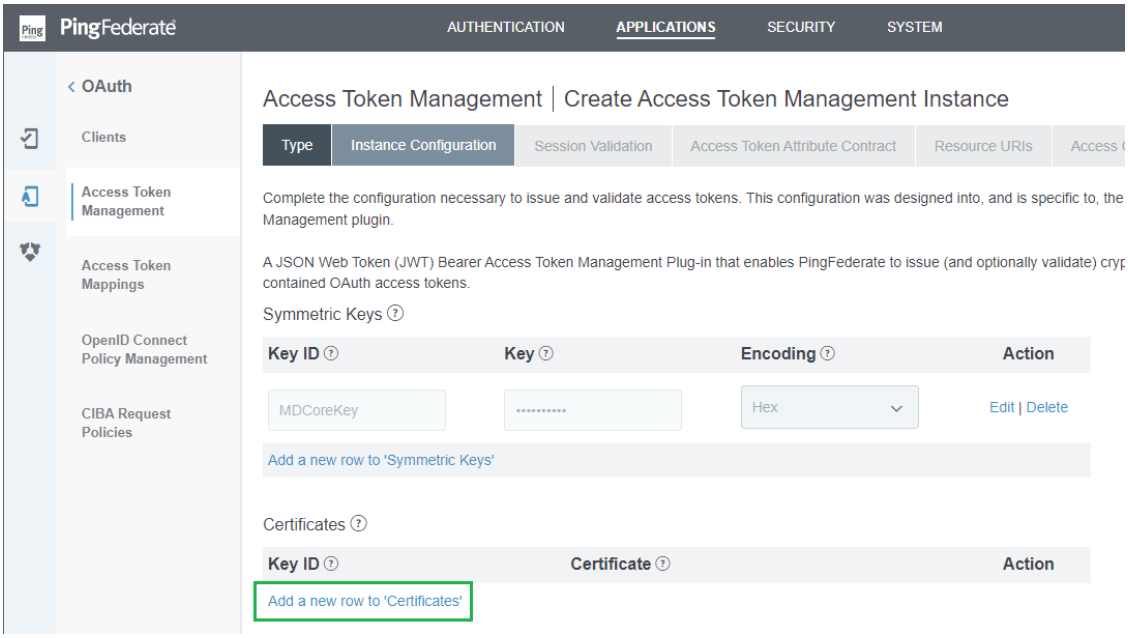
4. In `Instance Configuration` step, click `Add a new row to 'Symmetric Keys'` under `Symmetric Keys`.



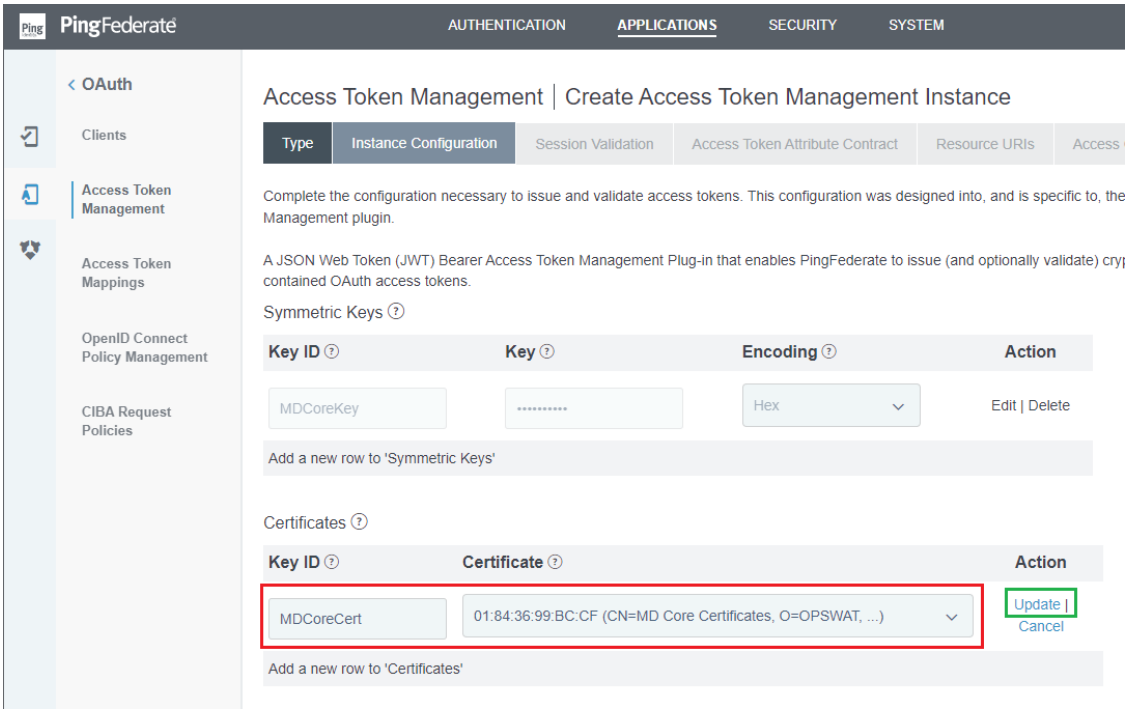
5. Enter `Key ID`, `Key`, and select the correct `Encoding`. Click `Update`.



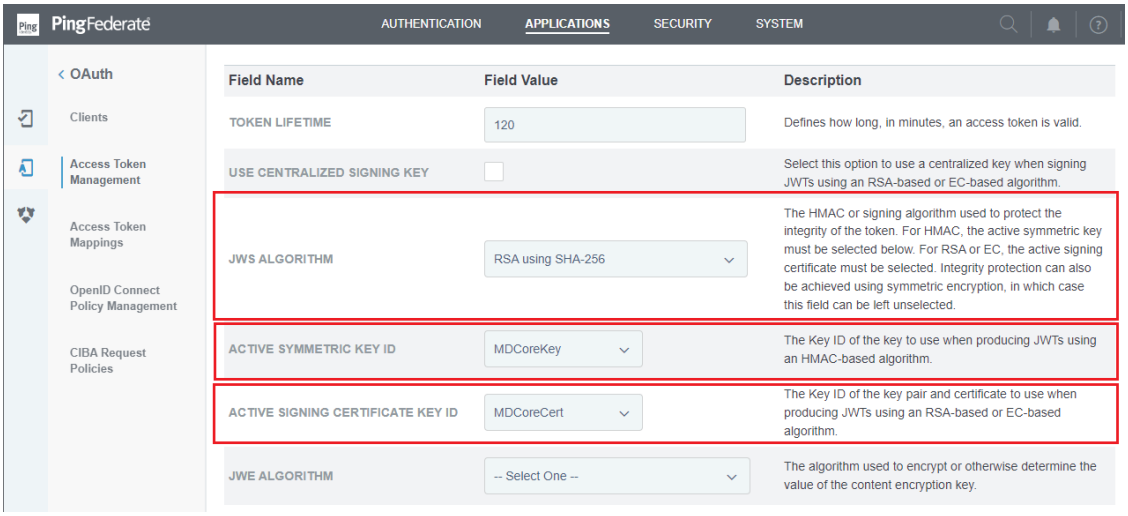
6. Back to `Instance Configuration` step, click `Add a new row to 'Certificates'` under `Certificates`.



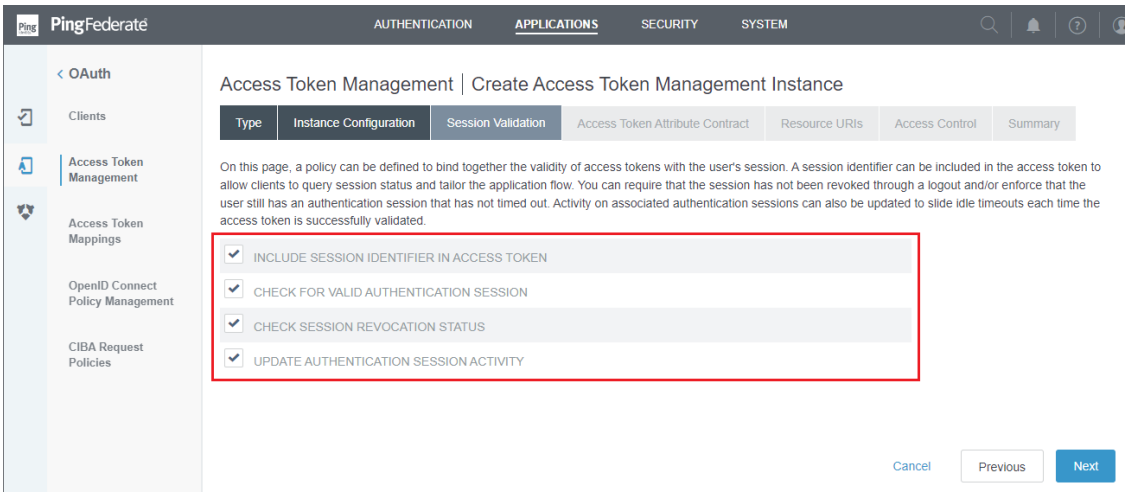
7. Enter `Key ID` and choose `Certificate` from the previous stage. Then click `Update`.



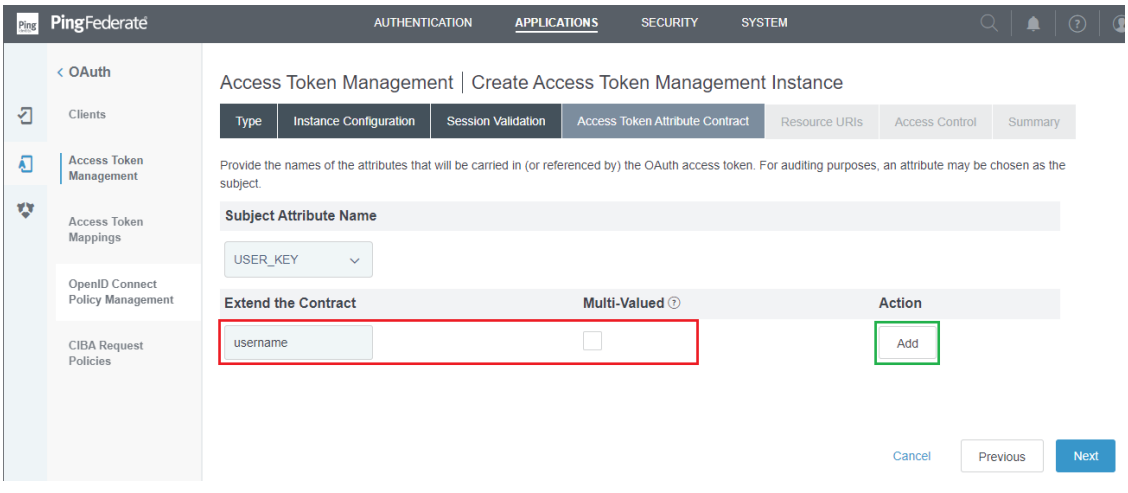
8. Back in `Instance Configuration` step, navigate to `JWS ALGORITHM` and select the appropriate algorithm. Proceed to `ACTIVE SYMMETRIC KEY ID` and `ACTIVE SIGNING CERTIFICATE KEY ID`, then select the key and certificate that were set up in steps 4 and 5 of this stage. Click `Next` when finished.



9. In `Session Validation` step, check all options and click `Next`.



10. In `Access Token Attribute Contract` step, navigate to `Extend the Contract` and add the same claims that have already been set up in the stages: `Policy Contract Grant Mapping`, `IdP Adapter Grant Mapping`, and `Resource Owner Credentials Grant Mapping`. Click `Next`.

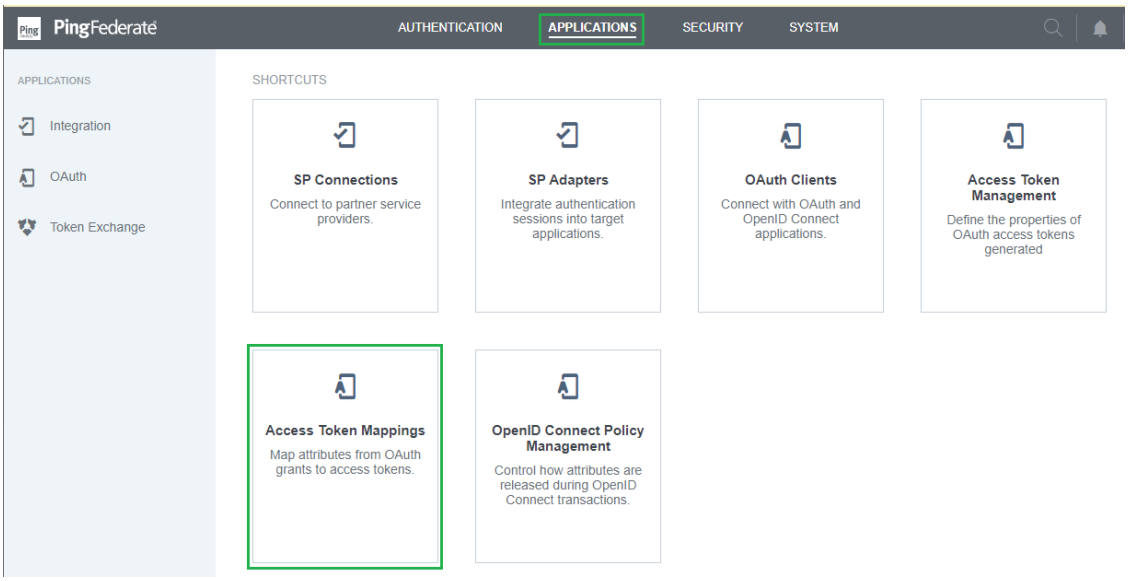


11. Click `Next` in `Resource URIs` and `Access Control` steps.

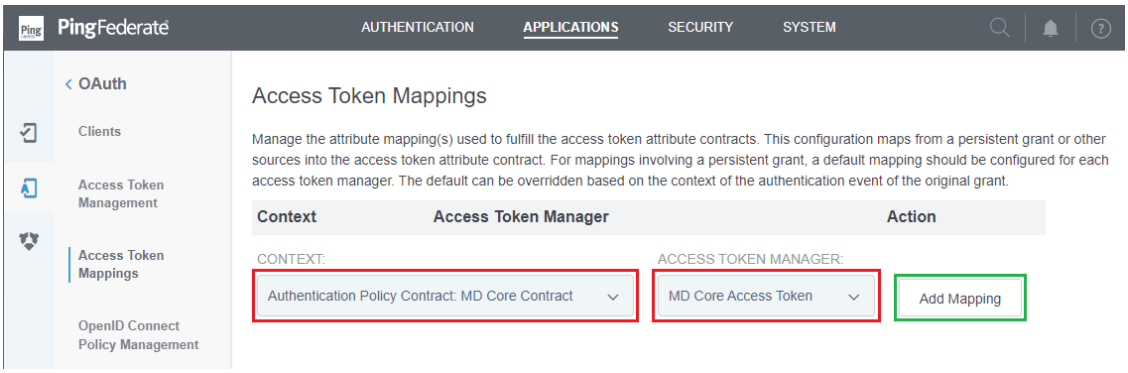
12. Review all selected options in `Summary` tab, then click `Save` to complete.

Set up Access Token Mappings

1. In homepage, navigate to `APPLICATIONS` and click on `Access Token Mappings`.

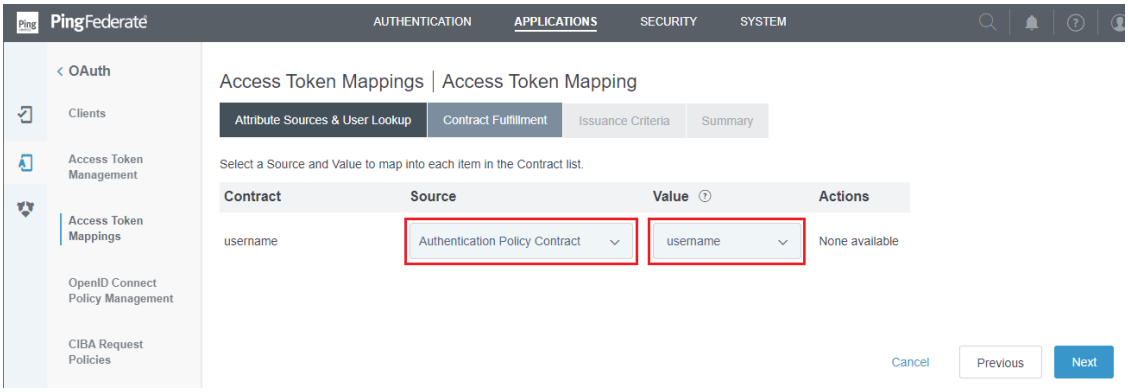


2. In `Access Token Mappings` page, select the Authentication Policy Contract from the previous step to map to `ACCESS TOKEN MANAGER`. Click `Add Mapping`.



3. Click `Next` in `Attribute Sources & User Lookup` step.

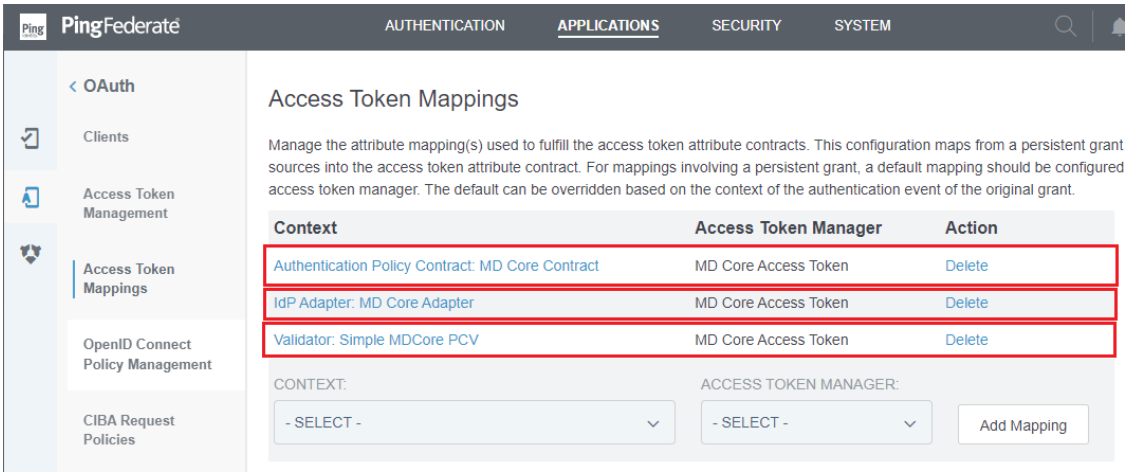
4. In `Contract Fulfillment` step, select the appropriate `Source` and `Value`, then click `Next`.



5. Click `Next` in `Issuance Criteria` step.

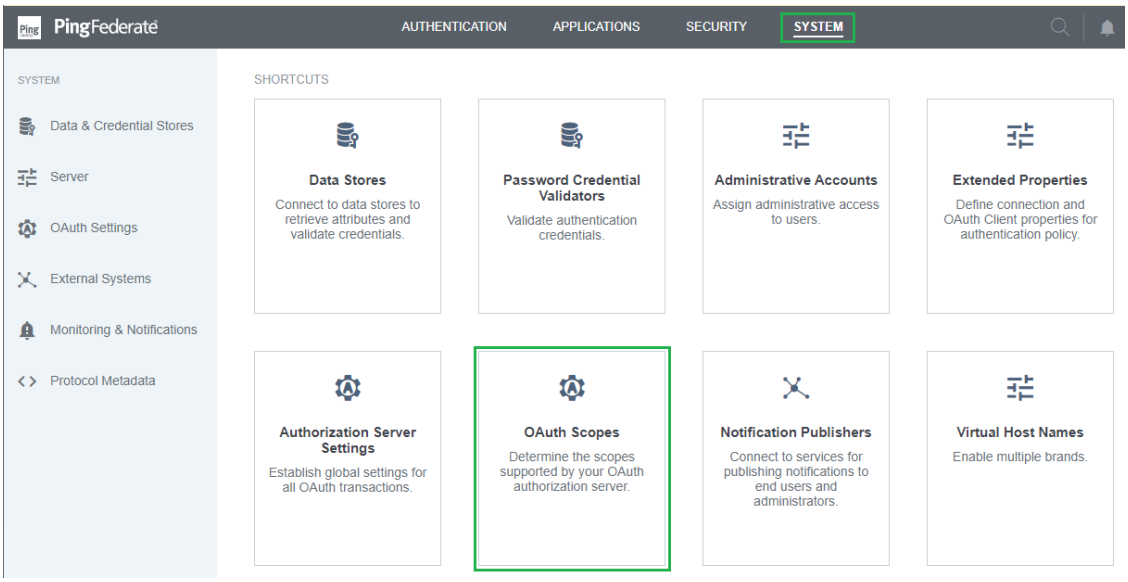
6. Review all selected options in `Summary` tab, then click `Save`.

7. Return to `Access Token Mappings` page and repeat the steps 2 to 6 for Validator and IdP Adapter.

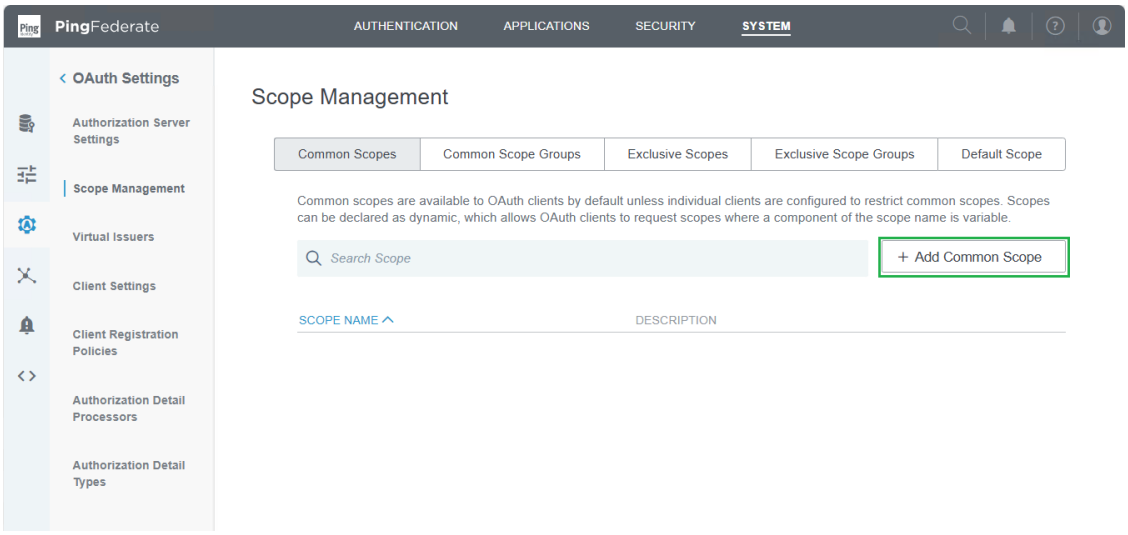


Set up Scope Management

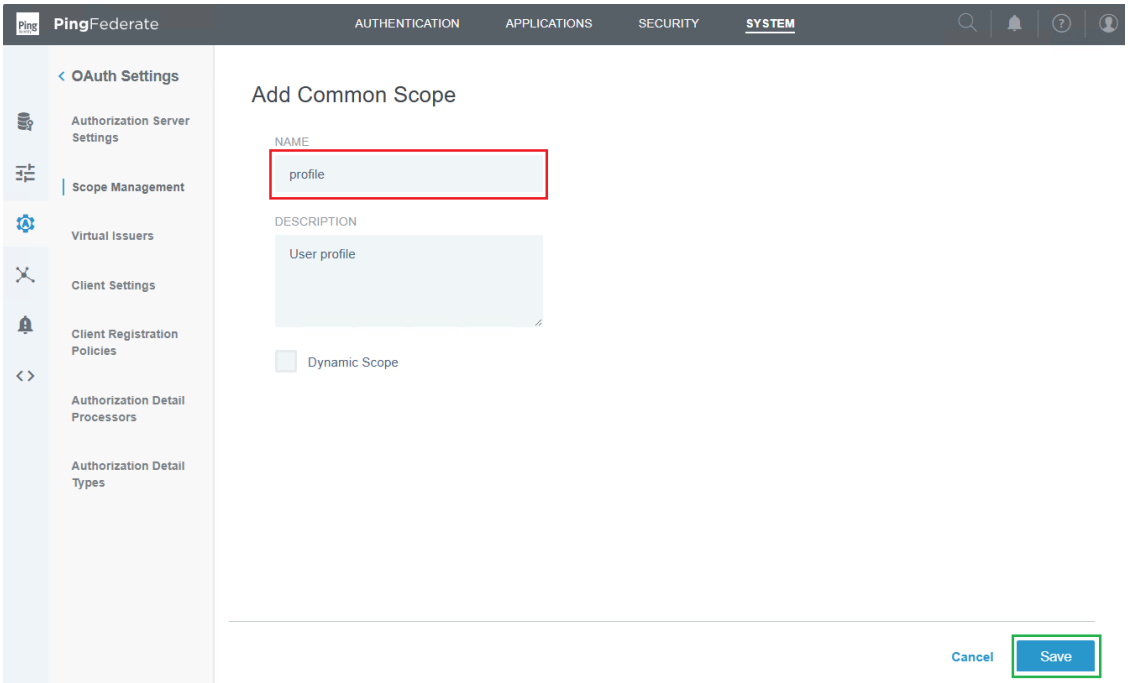
1. In homepage, navigate to `SYSTEM` and click `OAuth Scopes`.



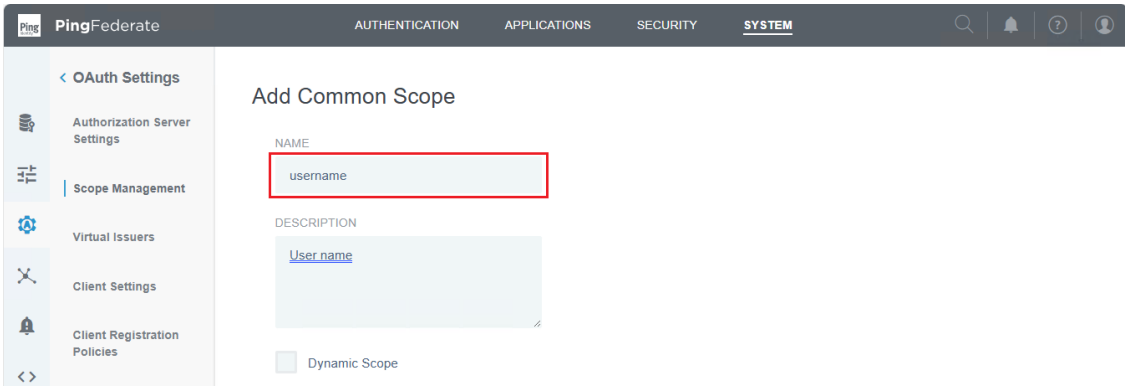
2. Under `Scope Management`, at `Common Scopes` tab, click `Add Common Scope`.



3. Due to MetaDefender Core requires **profile** scope, fill the scope name and click `Save`.



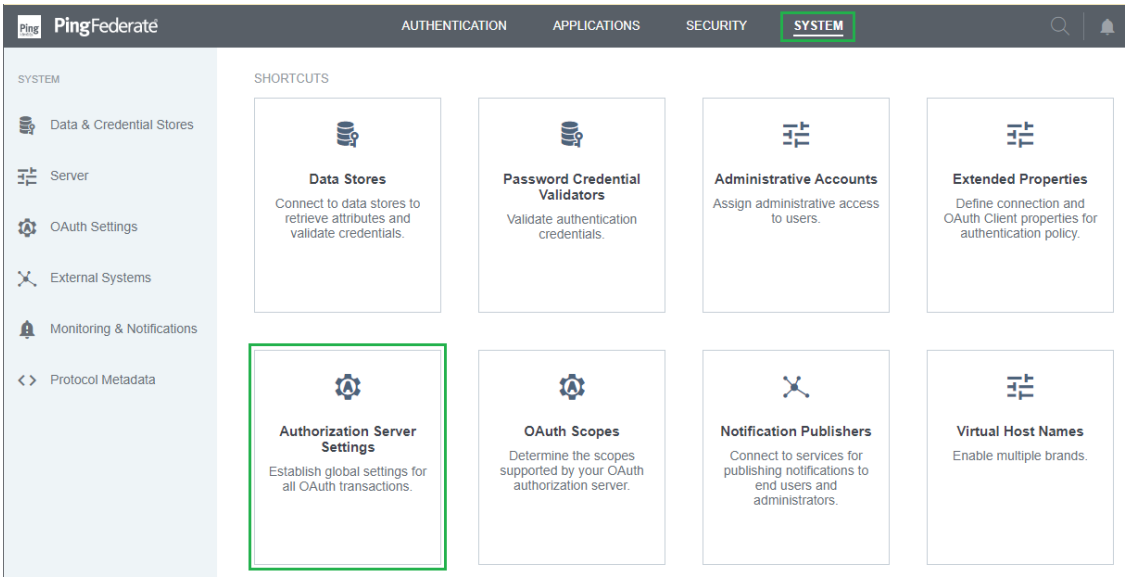
4. Repeat the above step to add `username` scope.



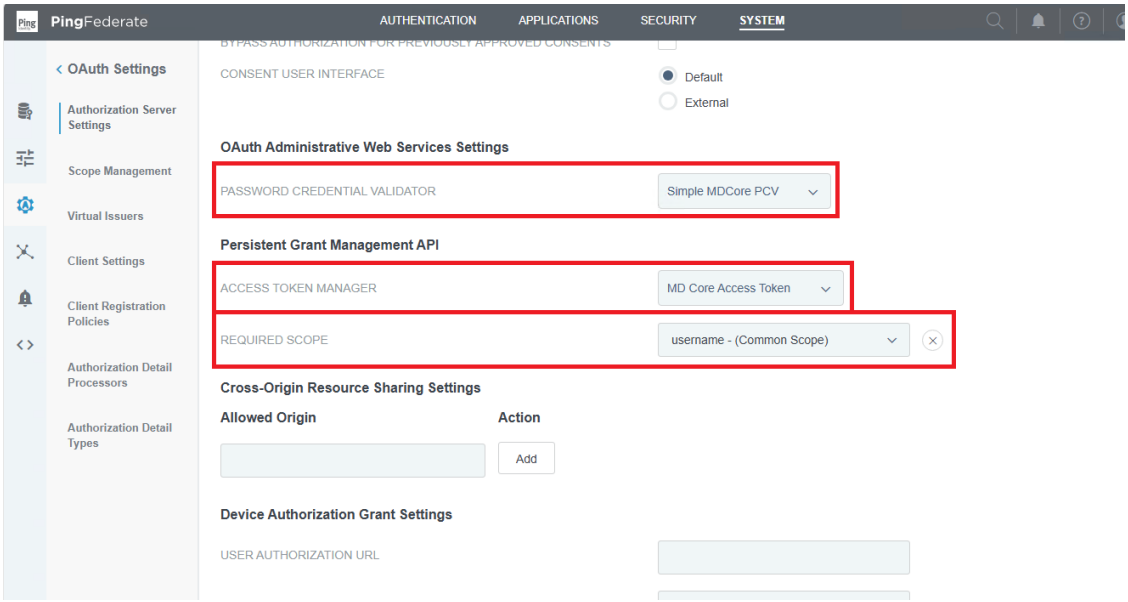
5. Click `Save` to complete.

Setup Authorization Server Settings

1. In homepage, navigate to `SYSTEM` and click `Authorization Server Settings`.



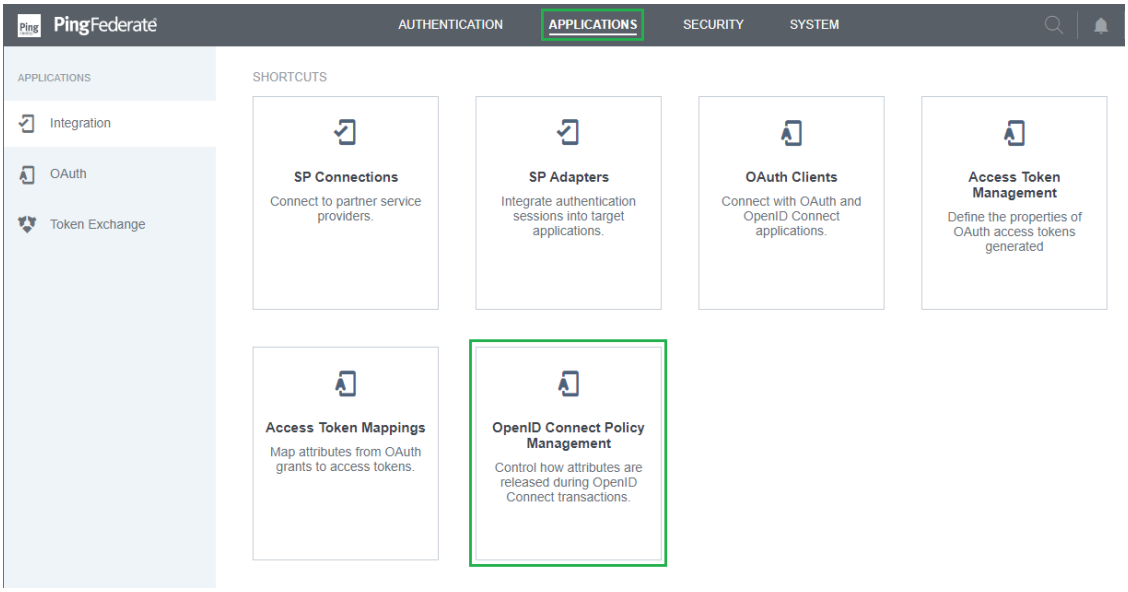
2. Navigate to `OAuth Administrative Web Services Settings` and select the password credential validator that was set up in previous stage for `PASSWORD CREDENTIAL VALIDATOR`.
3. Navigate to `Persistent Grant Management API`.
4. Select the access token manager and scope that were set up in the previous stage for `ACCESS TOKEN MANAGER` and `REQUIRED SCOPE`, respectively.



5. Click `Save` to complete.

Set up OpenID Connect Policy Management

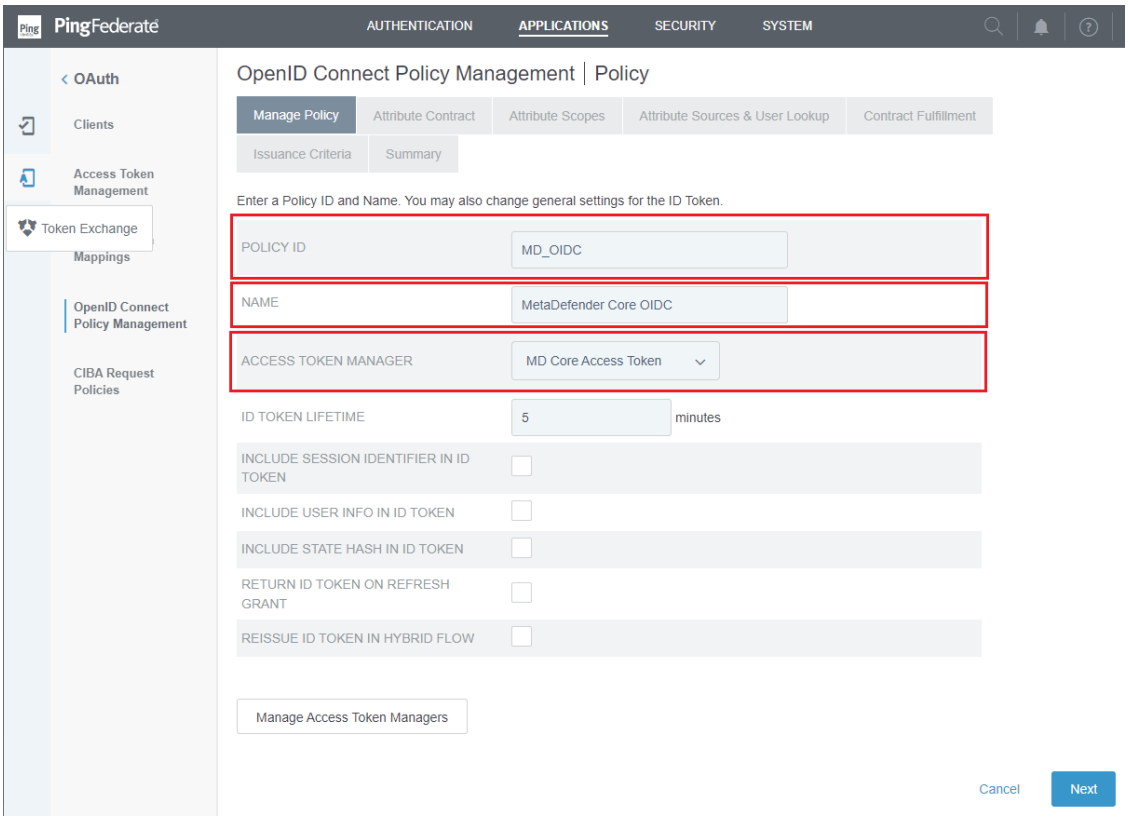
1. In homepage, navigate to `APPLICATIONS` and click `OpenID Connect Policy Management`.



2. Click `Add Policy` in the next page.

3. In `Manage Policy` step, enter `POLICY ID`, `NAME` and select the manager that was set up in previous stage for `ACCESS TOKEN MANAGER`. Click `Next` on done.

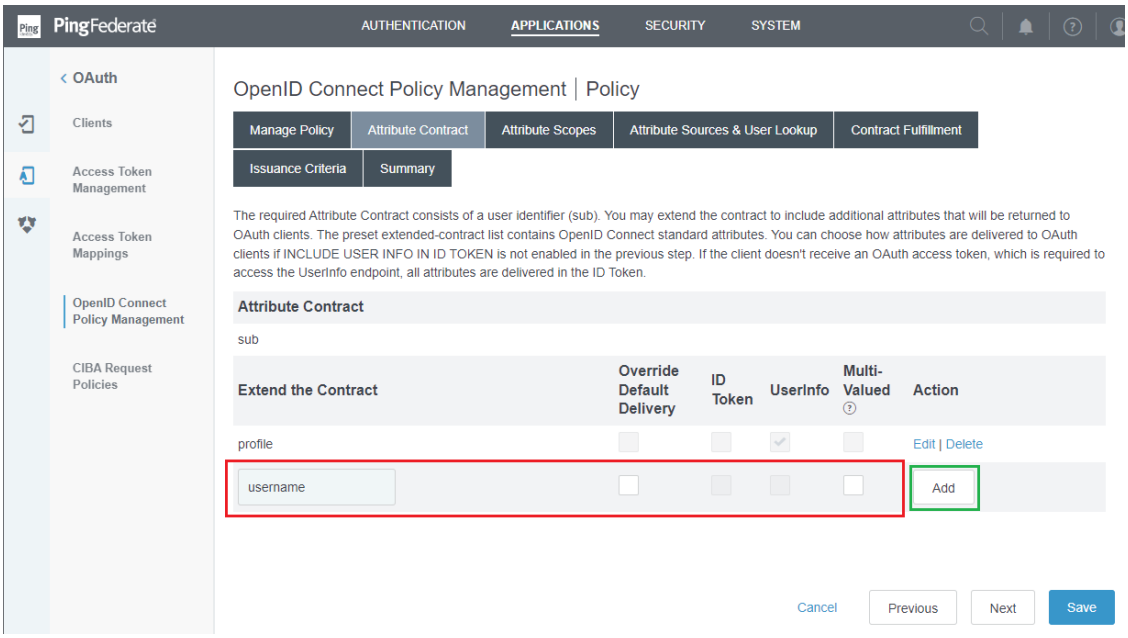
> **Info**
>
> Store the value assigned to `POLICY ID` as *policy_id*. It will be used later on fetching metadata in MetaDefender Core.



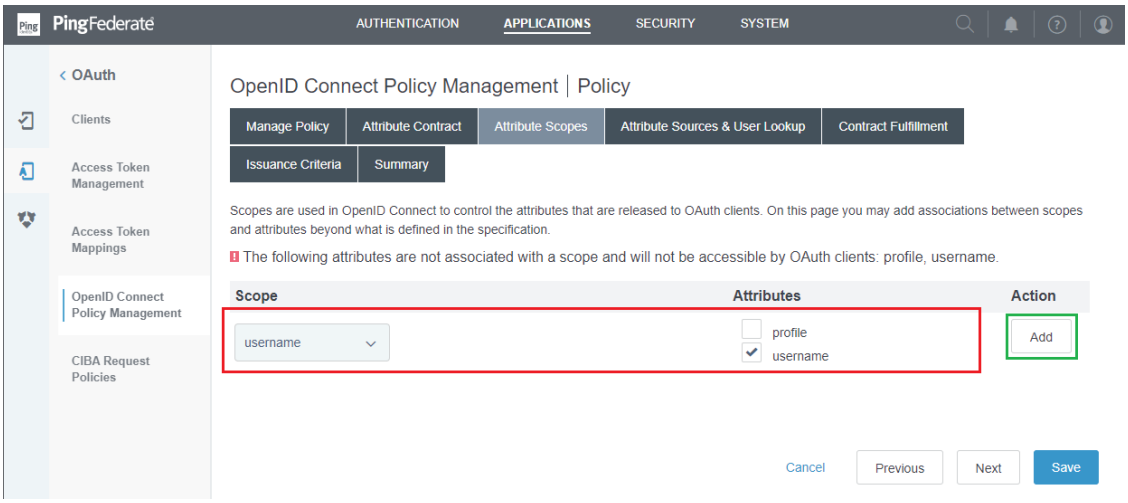
5. In `Attribute Contract` step, navigate to `Extend the Contract`, add the scopes requested by MetaDefender Core (i.e. username), and click `Next`.

Info

PingFederate, by default, lists all extended contracts. Within the scope of the instructions, only `profile` contract is required, all the remaining ones must be deleted.

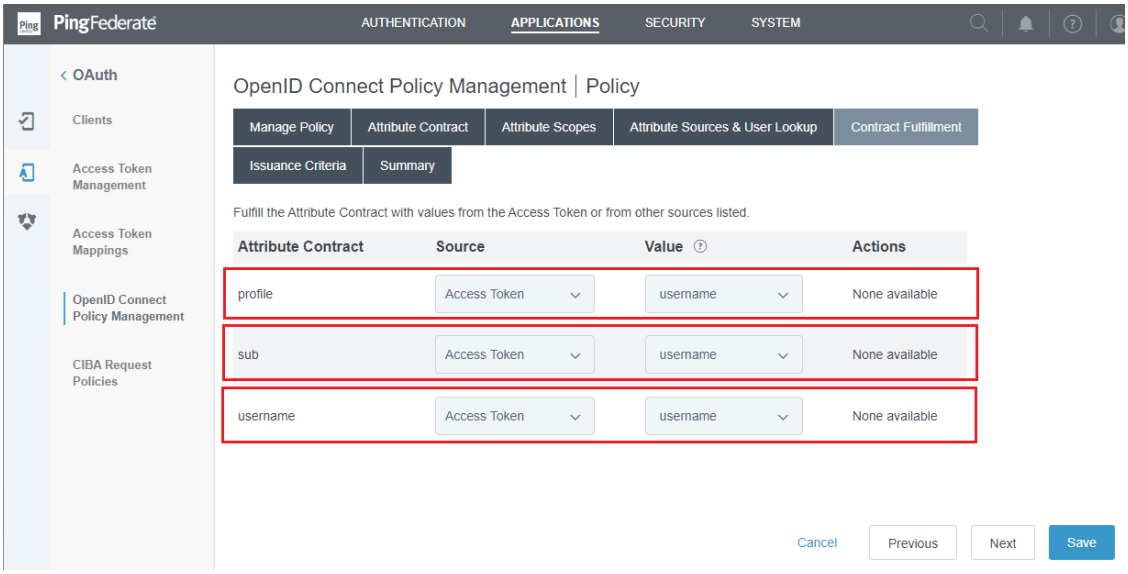


6. In `Attribute Scopes` step, match the scope with its attributes and click `Next`.



7. Click `Next` in `Attribute Sources & User Lookup` step.

8. In `Contract Fulfillment` step, select the appropriate `Source` and `Value`, and click `Next`.

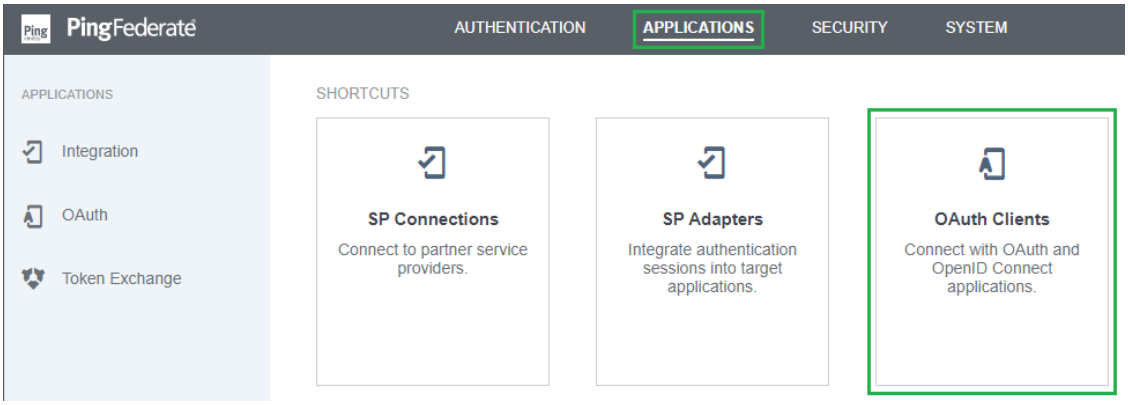


9. Click `Next` in `Issuance Criteria` step.

10. Review all selected options in `Summary` tab, then click `Save`.

Add OAuth client

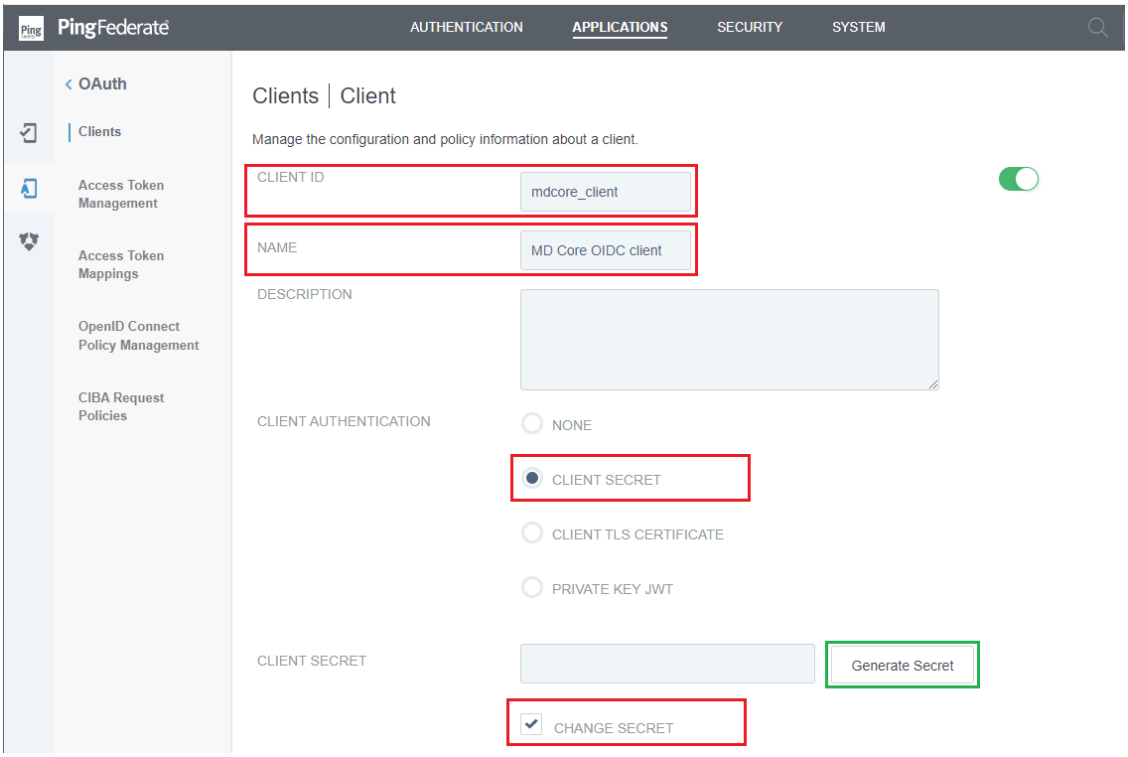
1. In homepage, navigate to `APPLICATIONS` and click `OAuth Clients`.



2. Click `Add Client` in the next page.
3. Enter `CLIENT ID`and store to *client_id*.
4. Enter `NAME`.
5. Choose `CLIENT SECRET` for `CLIENT AUTHENTICATION`.
6. Navigate to `CLIENT SECRET`, check `CHANGE SECRET` box, click `Generate Secret`, and store the generated string as *client_secret*.

Info

client_id and *client_secret* will be used later on setting up MetaDefender Core.



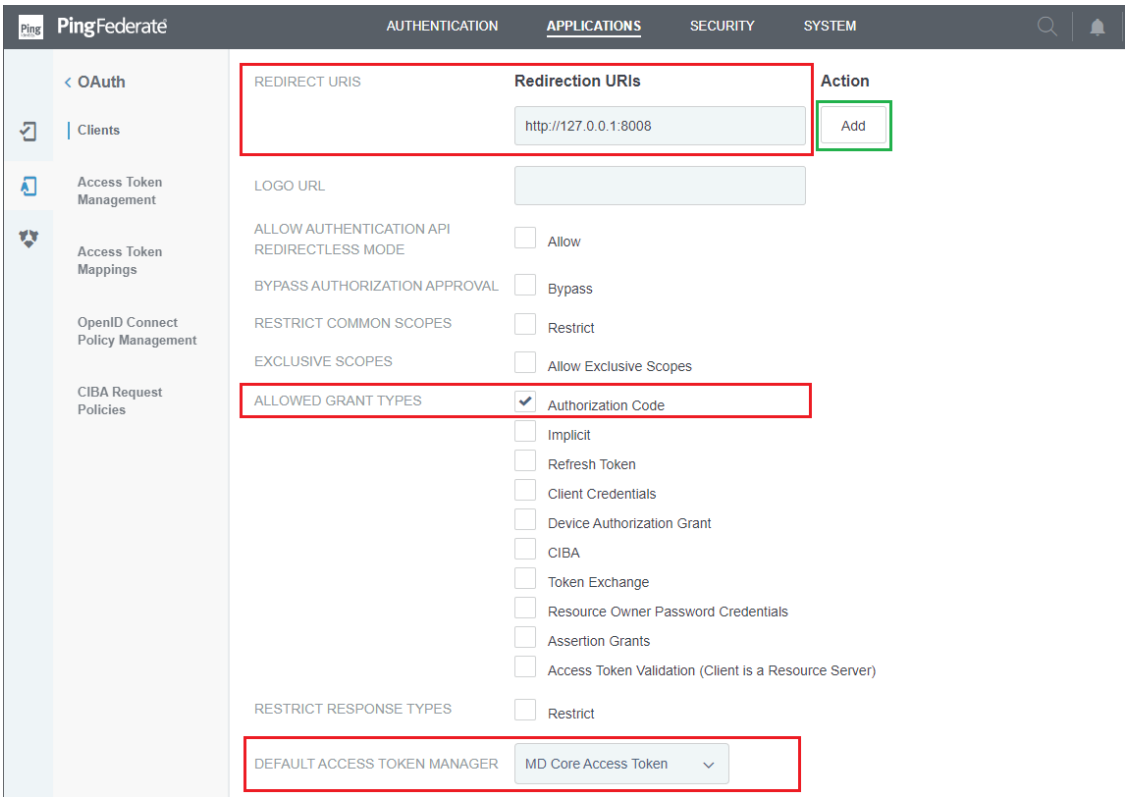
7. Navigate to `REDIRECT URIS`, fill in a temporary Redirection URI, and click `Add`.

Warning

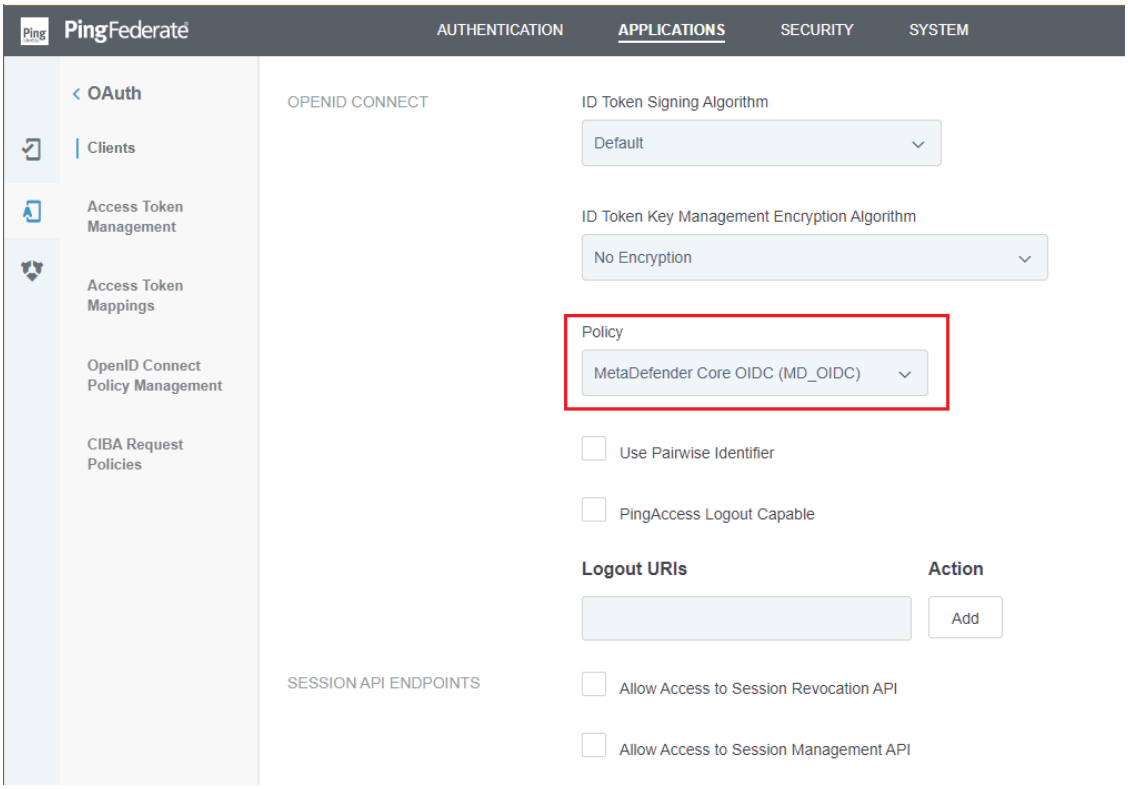
A temporary redirection URI is used at this step to complete configuration in PingFederate. We will return to update this setting with the correct URI from MetaDefender Core later.

8. Navigate to `ALLOWED GRANT TYPES`, and tick `Authorization Code`.

9. Navigate to `DEFAULT ACCESS TOKEN MANAGER`, and select Access Token Manager that was set up previously.



10. Navigate to `OPENID CONNECT`, under `Policy`, select OpenID connect policy that was set up previously.

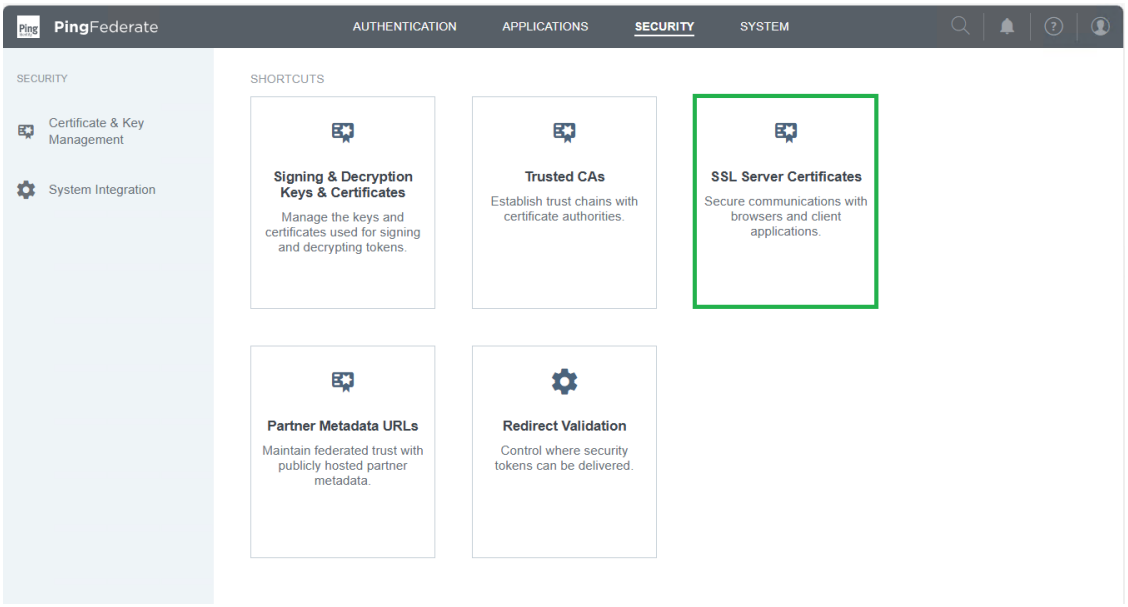


11. Click `Save` to complete.

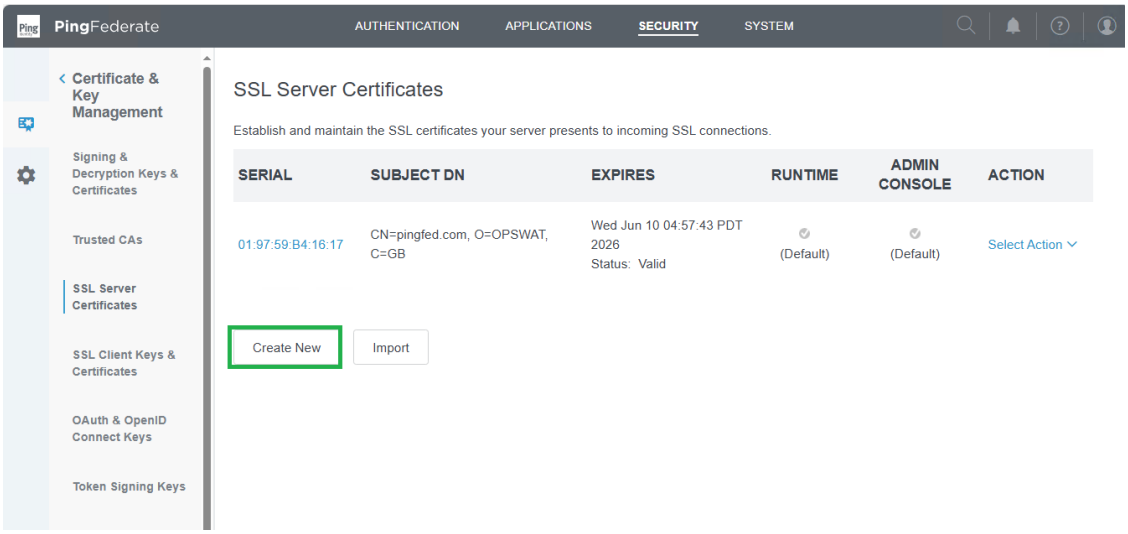
Set up MetaDefender Core

Install PingFederate certificate

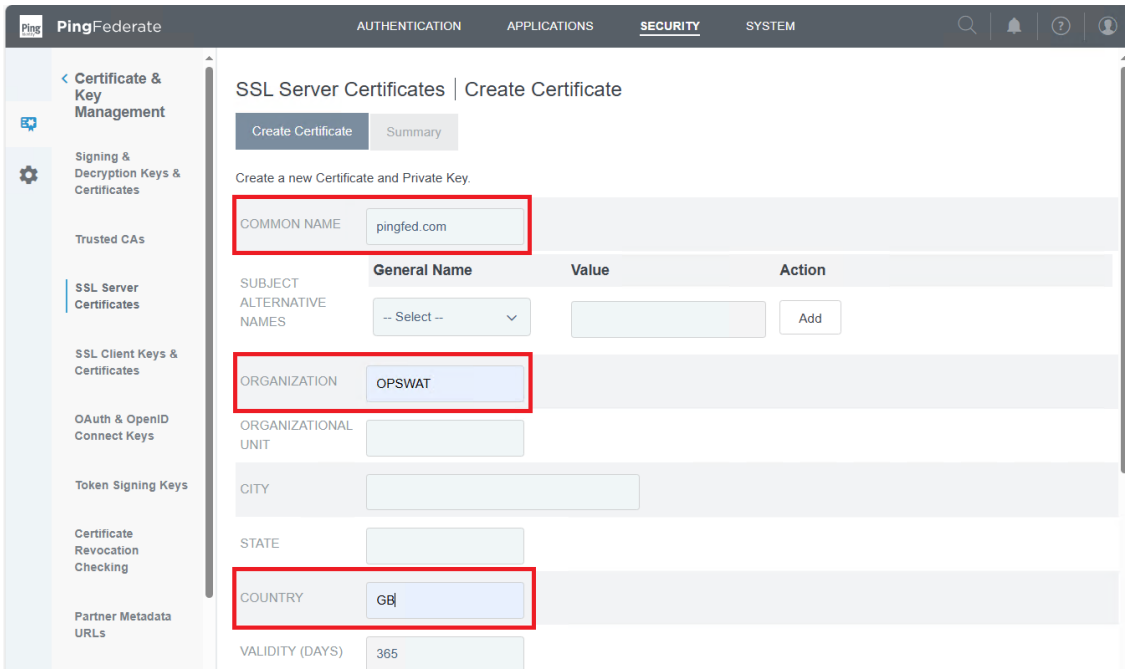
1. In homepage, go to `SECURITY` and click `SSL Server Certificates`.



2. Click `Create New` to generate a new certificate.



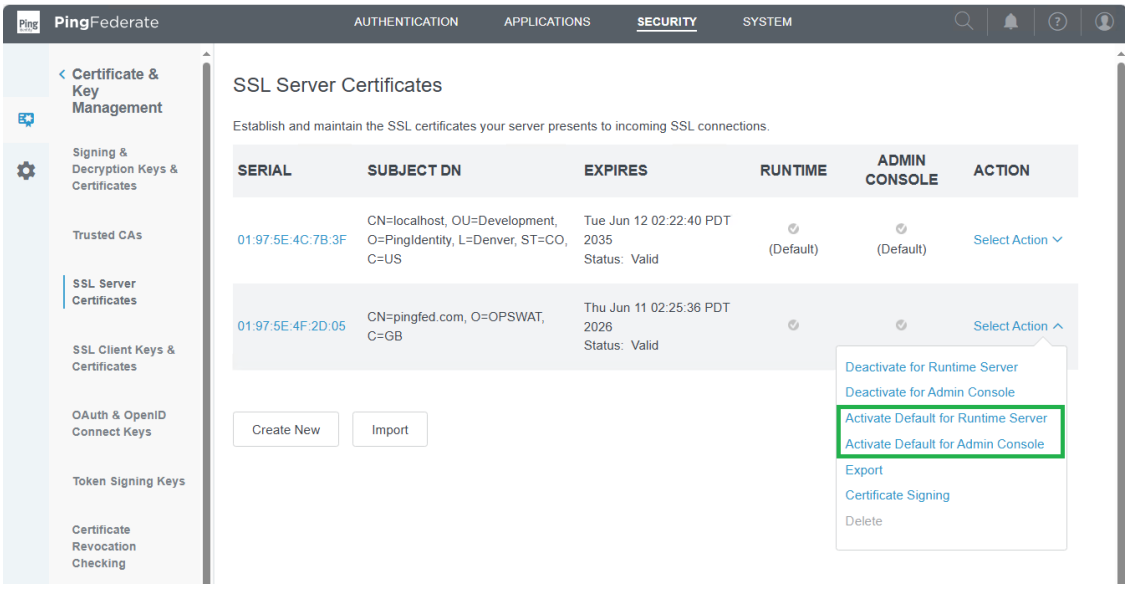
3. Complete the information for the new certificate, click `Next` and `Save`.



Info

`COMMON NAME` must refer to the domain or IP address hosting PingFederate.

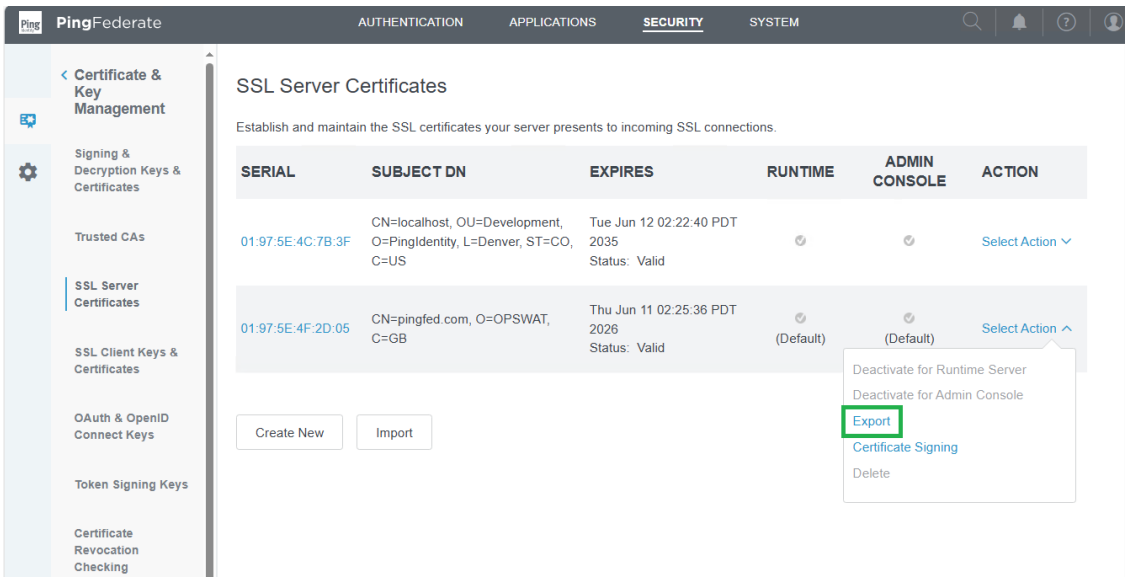
4. Activate the new certificate.



Warning

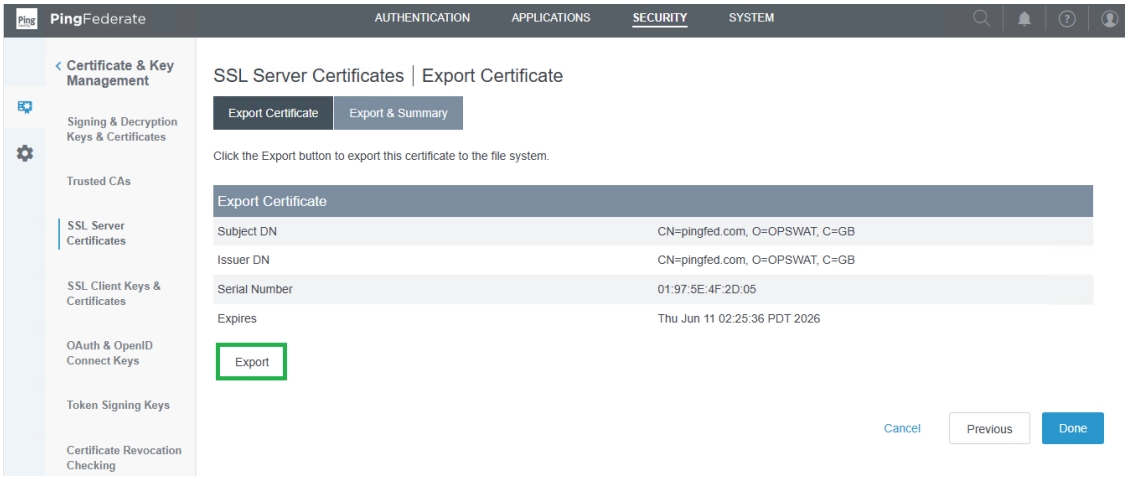
Activating the certificate will end all current sessions, and users will need to sign in again into PingFederate using the domain or IP specified in the certificate.

5. Select the certificate and select `Export`.



6. Retain the default settings and click `Next`.

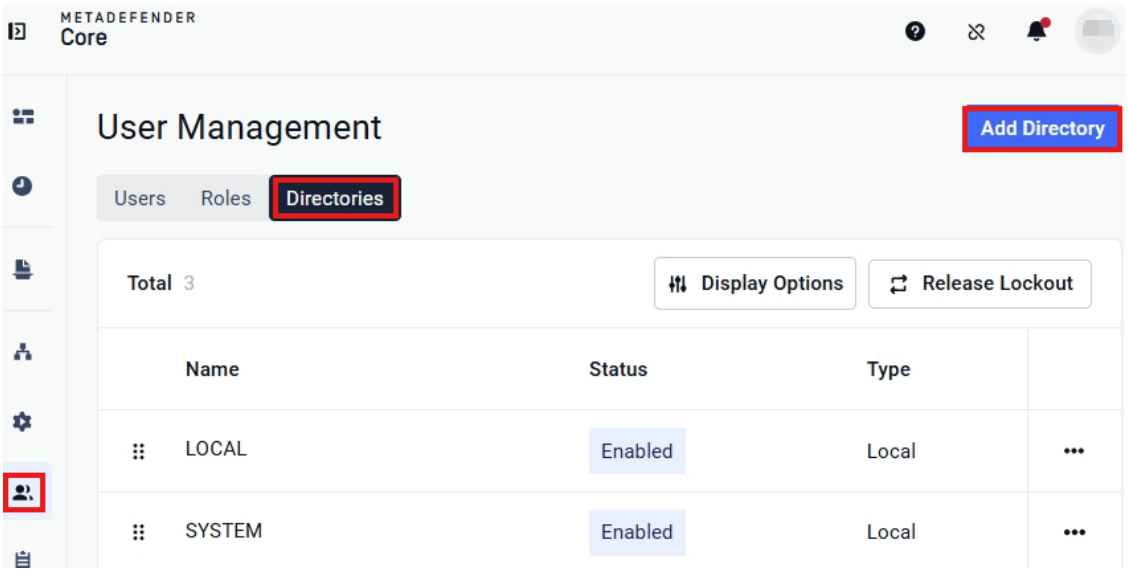
7. Click `Export` to download the certificate.



8. Install the certificate on the server running MetaDefender Core.
9. Restart MetaDefender Core for the changes to take effect.

Configure OIDC in MetaDefender Core

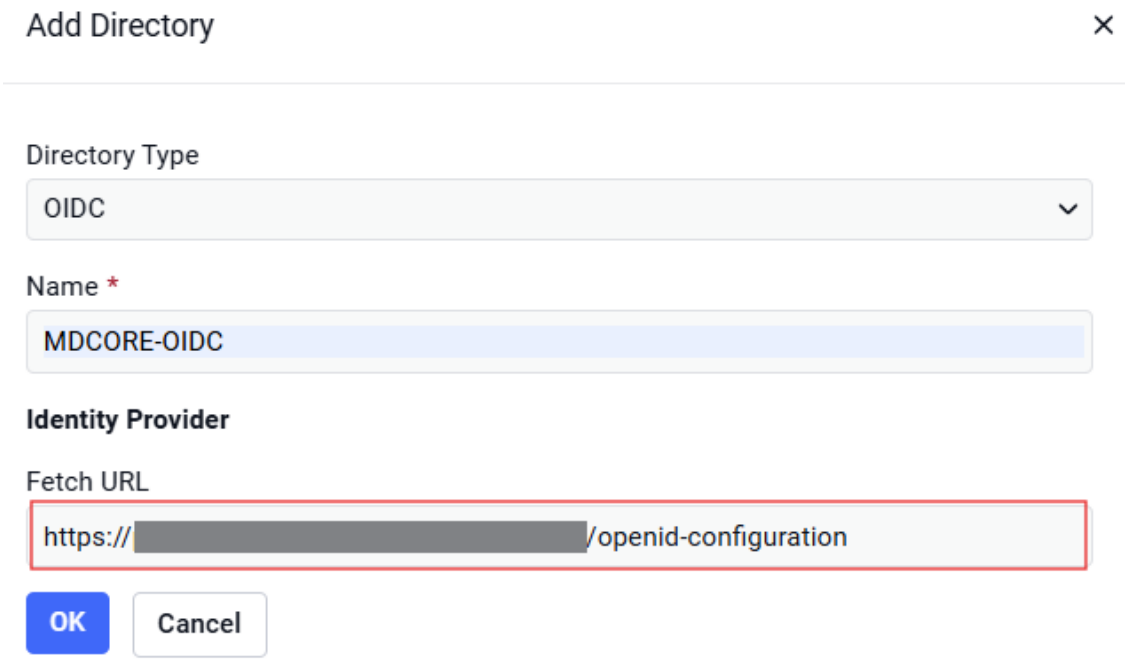
1. Sign in to MetaDefender Core.
2. In the dashboard, click on `User Management` in the left sidebar.
3. On `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.



4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCore-OIDC.
5. Click `Fetch URL`.



6. Enter PingFederate metadata URL and click `OK`.



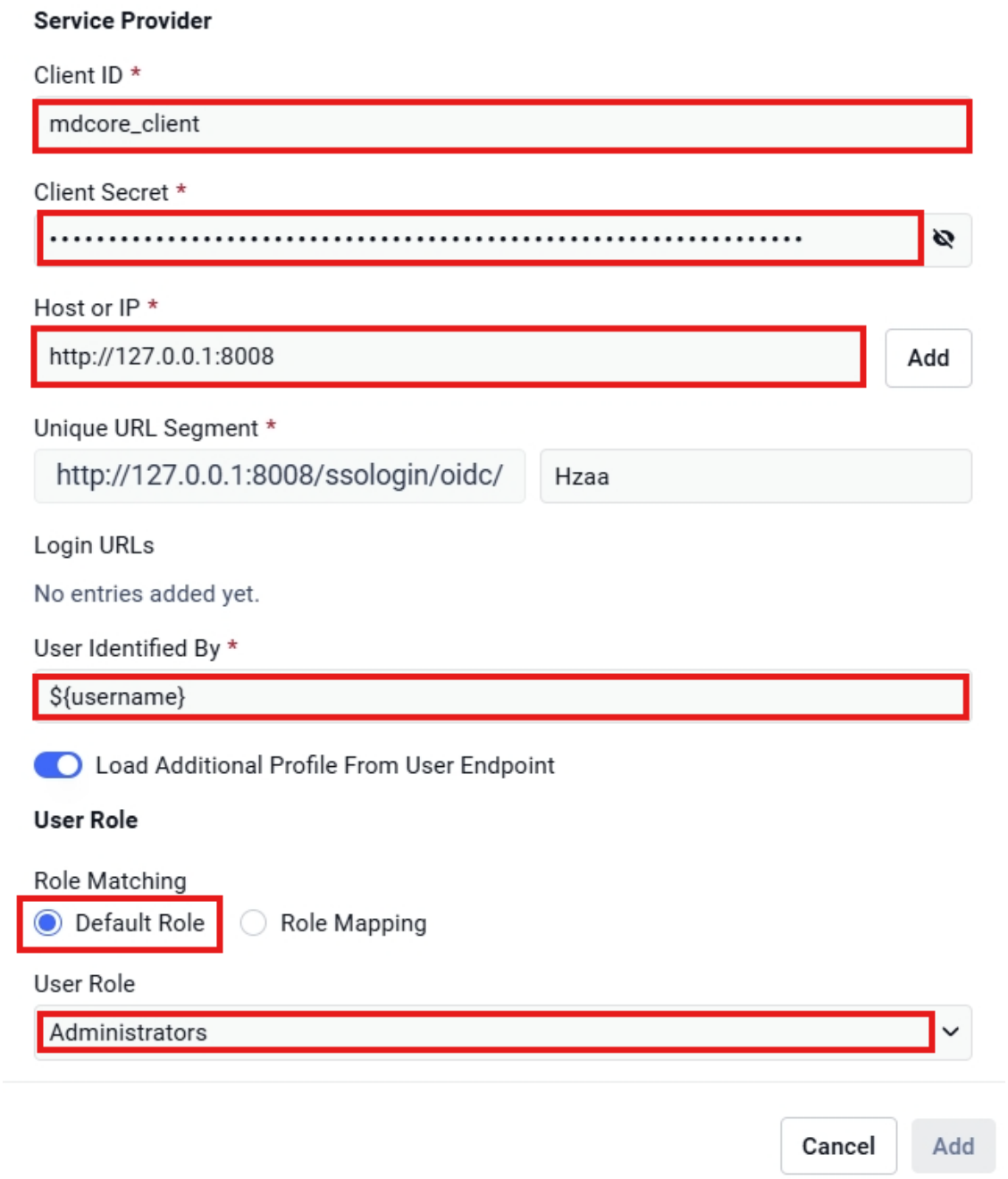
Info

PingFederate metadata URL is in format of `<host>:<port>/.well-known/openid-configuration?policy_id=<policy_id>` , in which:

- `host` and `port` are the host/IP and port of the machine hosting PingFederate. The default port is 9031.
- `policy_id` is *policy_id* that is set to `POLICY_ID` in the stage of **Setup OpenID Connect Policy Management**.

7. Under `Service Provider`, paste *client_id* and *client_secret* to boxes under `Client ID` and `Client secret` respectively.

8. Fill in `Host or IP` with the host or IP where MetaDefender Core is hosted, such as http://127.0.0.1:8008 for this example.



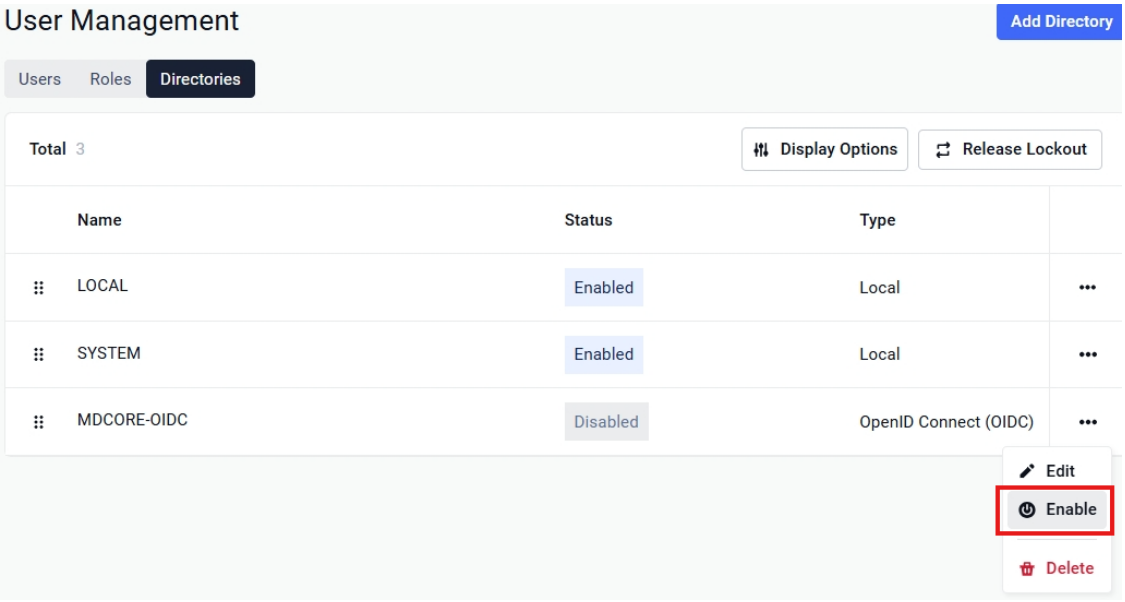
9. Copy `Login URL` and store it as *login_url*.

> **Info**
>
> *login_url* is used to update Redirection URI setting in PingFederate in the next stage.

10. Fill in `User Identified by` with **${username}**.

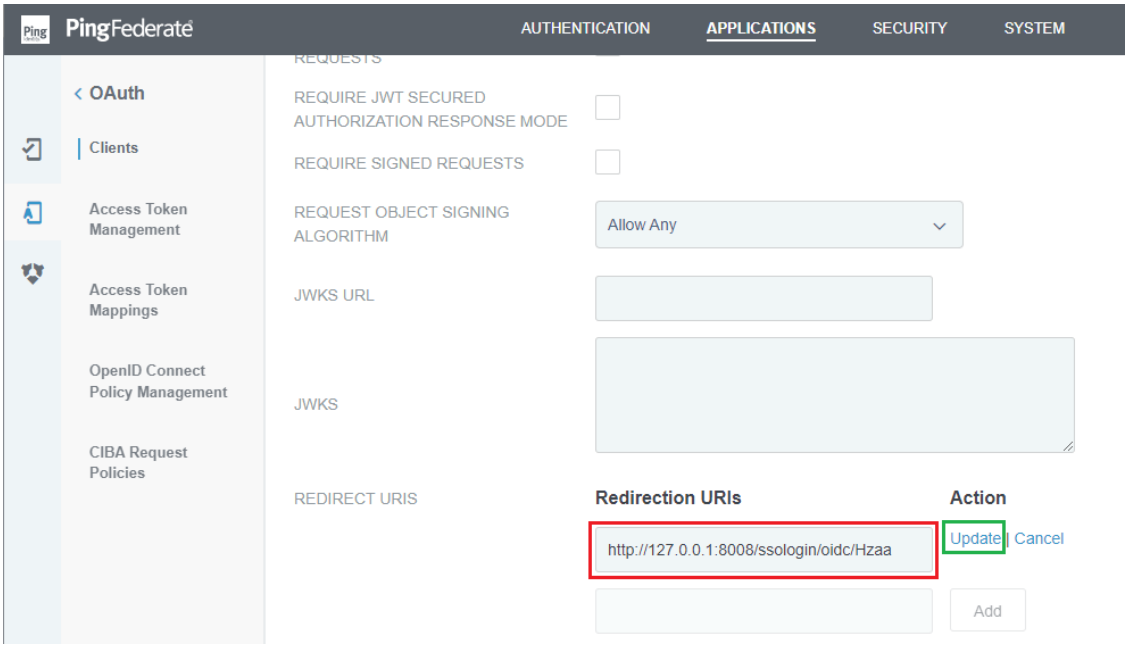
11. Select the default role and choose the appropriate role for the login user.

12. Click `Add` to complete.
13. In `User Management` page, toggle the new directory, MDCORE-OIDC in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all sessions will expire immediately.



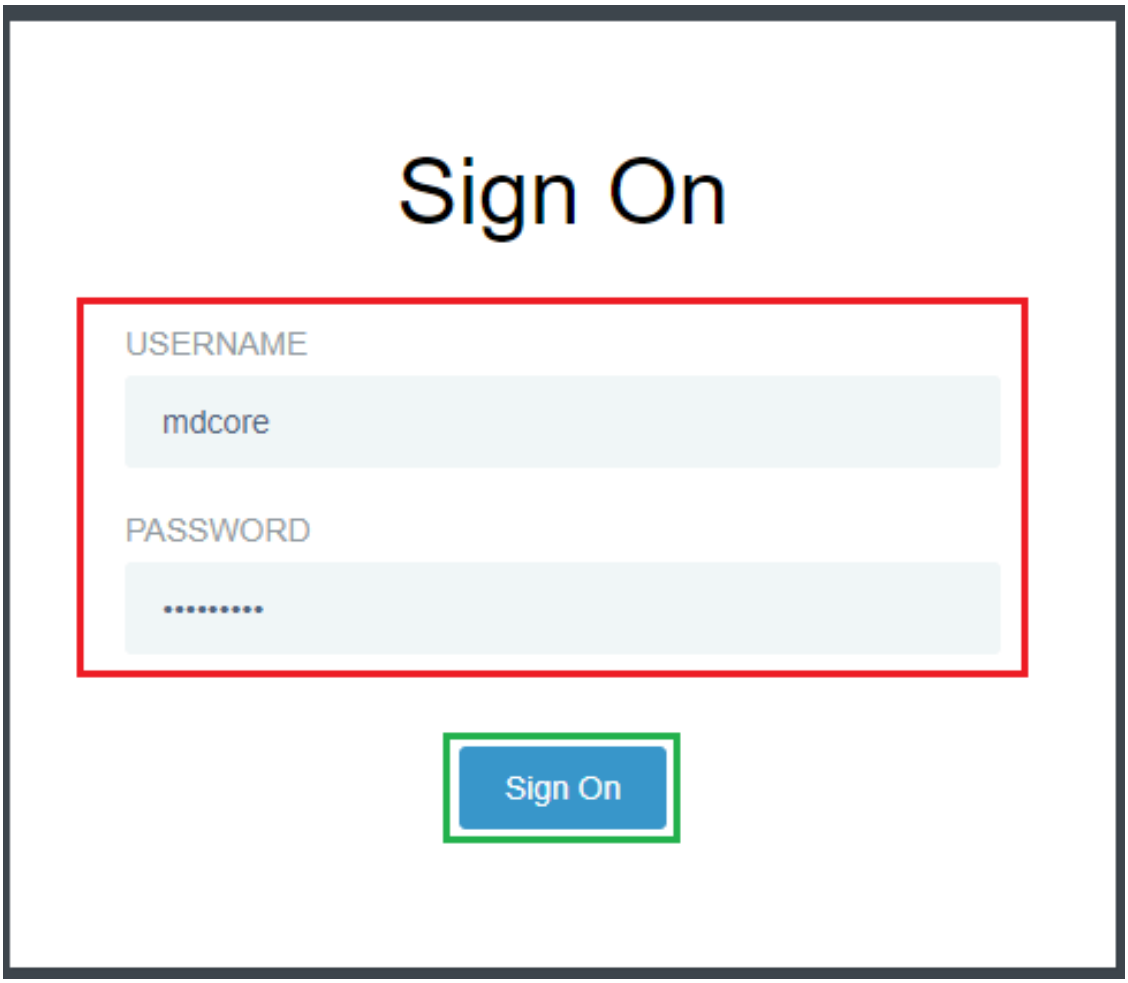
Complete configuration in PingFederate

1. Back to PingFederate, in `OAuth`, navigate to `Clients` in the sidebar.
2. In `APPLICATIONS` tab, navigate to `Redirection URIs`, click `Edit`, modify the URI with *login_url* copied from MetaDefender Core.
3. Click `Update` and `Save`.



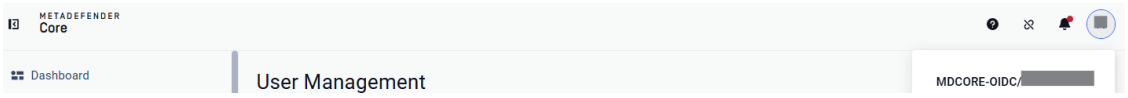
Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to PingFederate Sign-on page.



2. Sign in using the account registered with PingFederate.

3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.



4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

PingOne

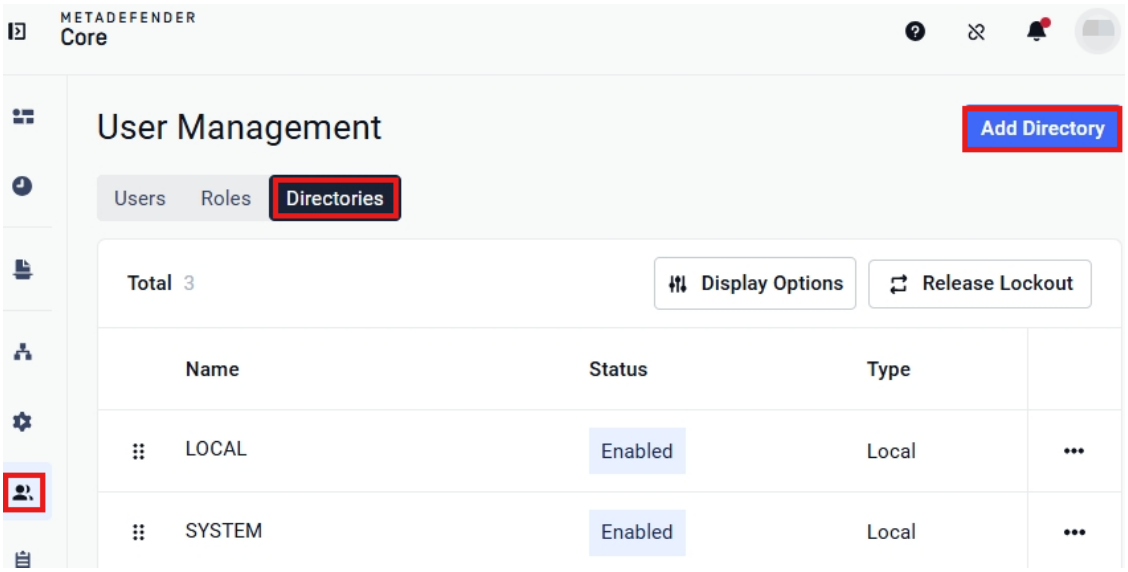
The instructions guide users on how to configure PingOne SAML and OIDC Single Sign-on for MetaDefender Core.

Configure SAML SSO

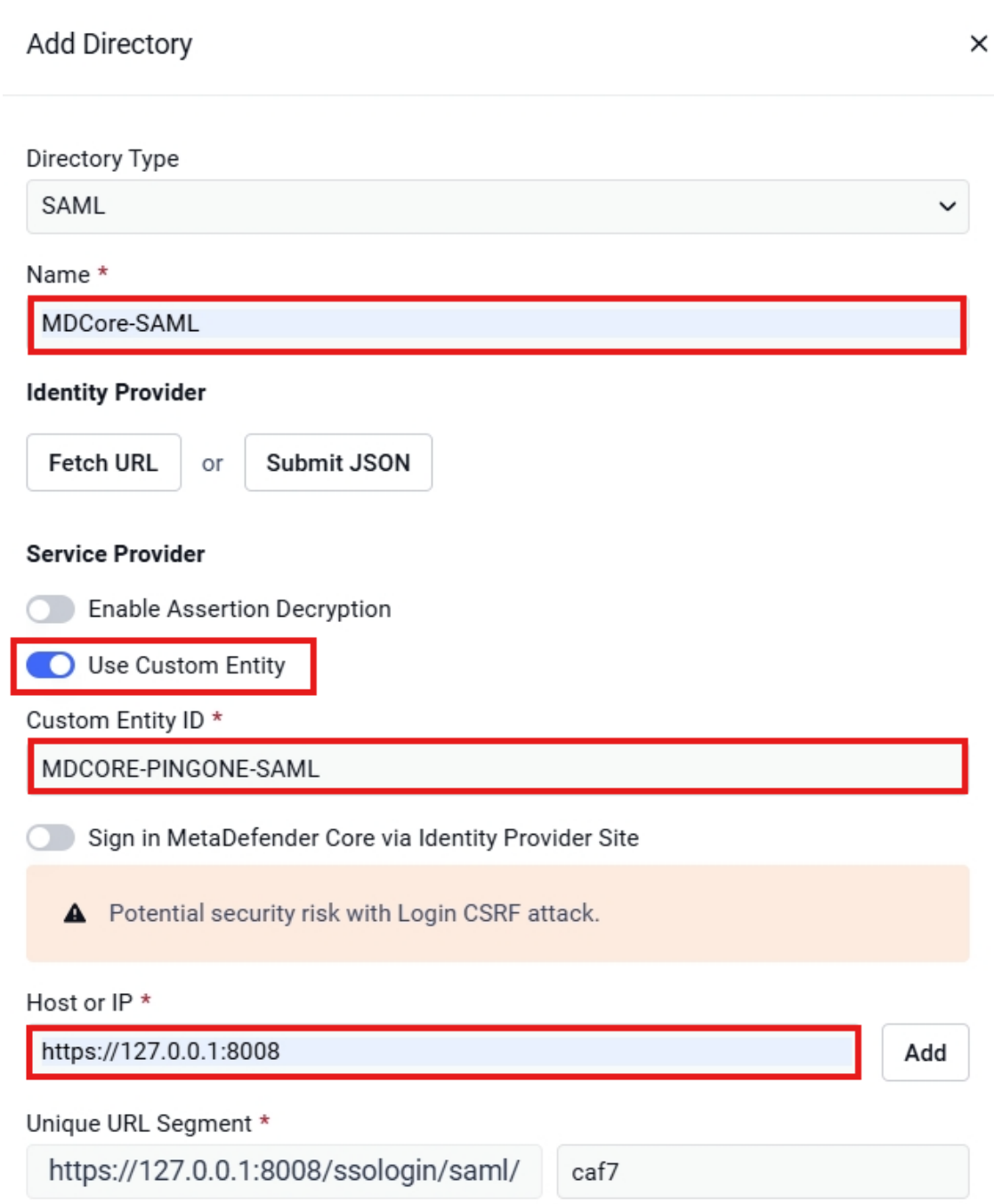
Create SAML directory on MDCore

1. Login to MDCore

2. At dashboard, hit `User Management` in sidebar
3. Under `User Management` page, choose `Directories` tab and hit `Add directory` on the top right

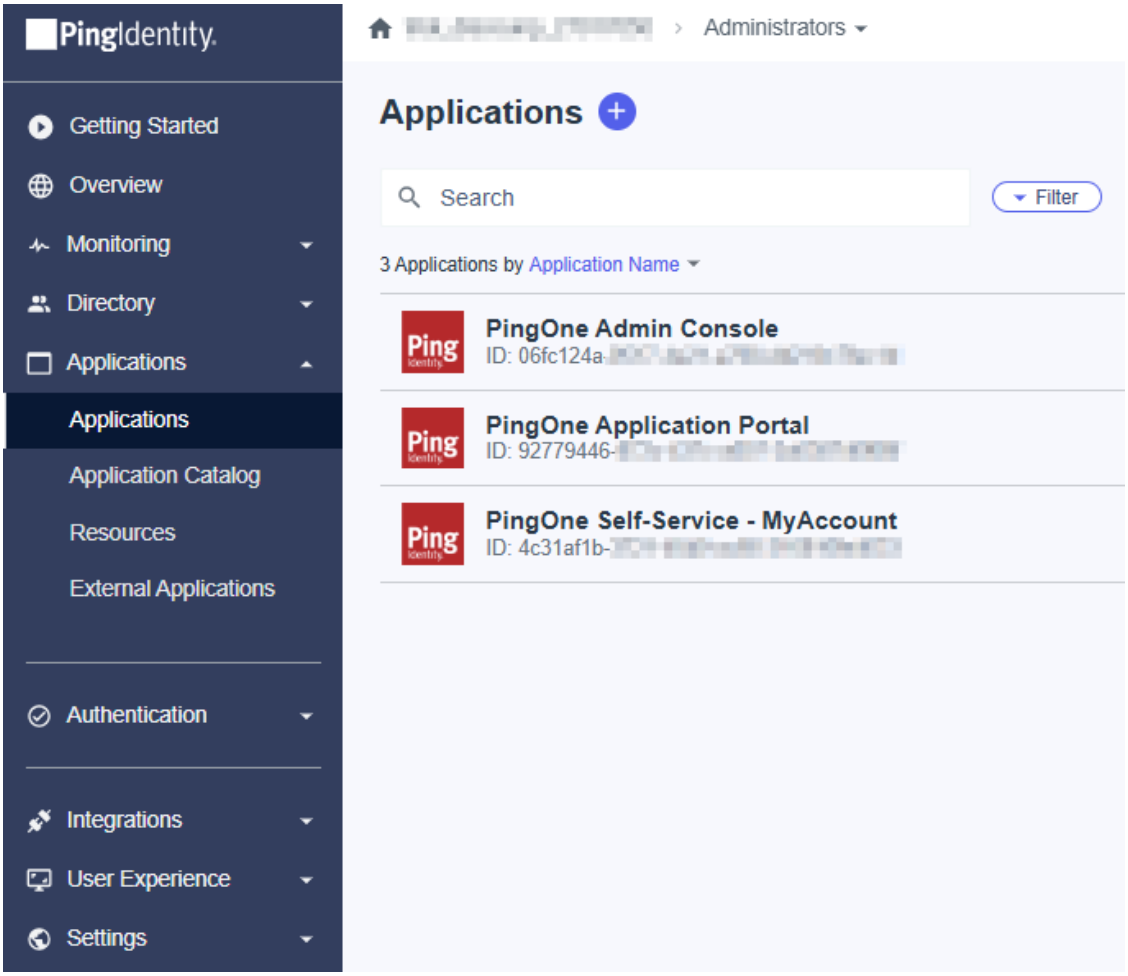


4. In `Add Directory` page, choose `SAML` in `Directory Type`, fill `Name` for the new directory, MDCore-SAML for example
5. Under `Service Provider` section, enable option `Use custom entity ID` and fill out the ID, MDCORE-PINGONE-SAML for example, then store it in *entity_id*
6. Fill `Host or IP` where MDCore is hosting, *https://127.0.0.1:8008* for this example
7. Copy string generated under `Login URL` and store to *reply_uri*

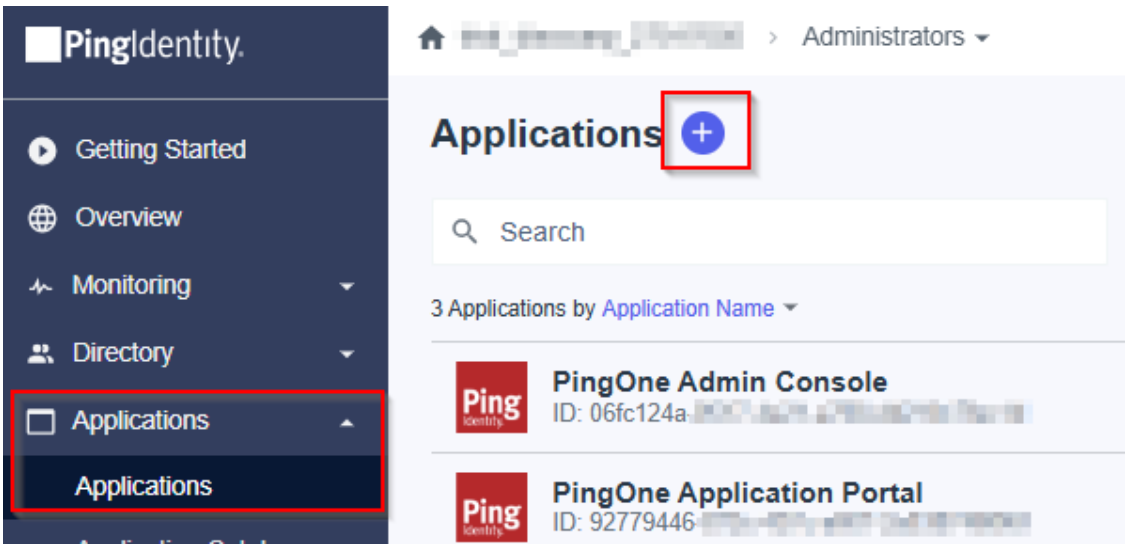


Create and configure SAML application in PingOne

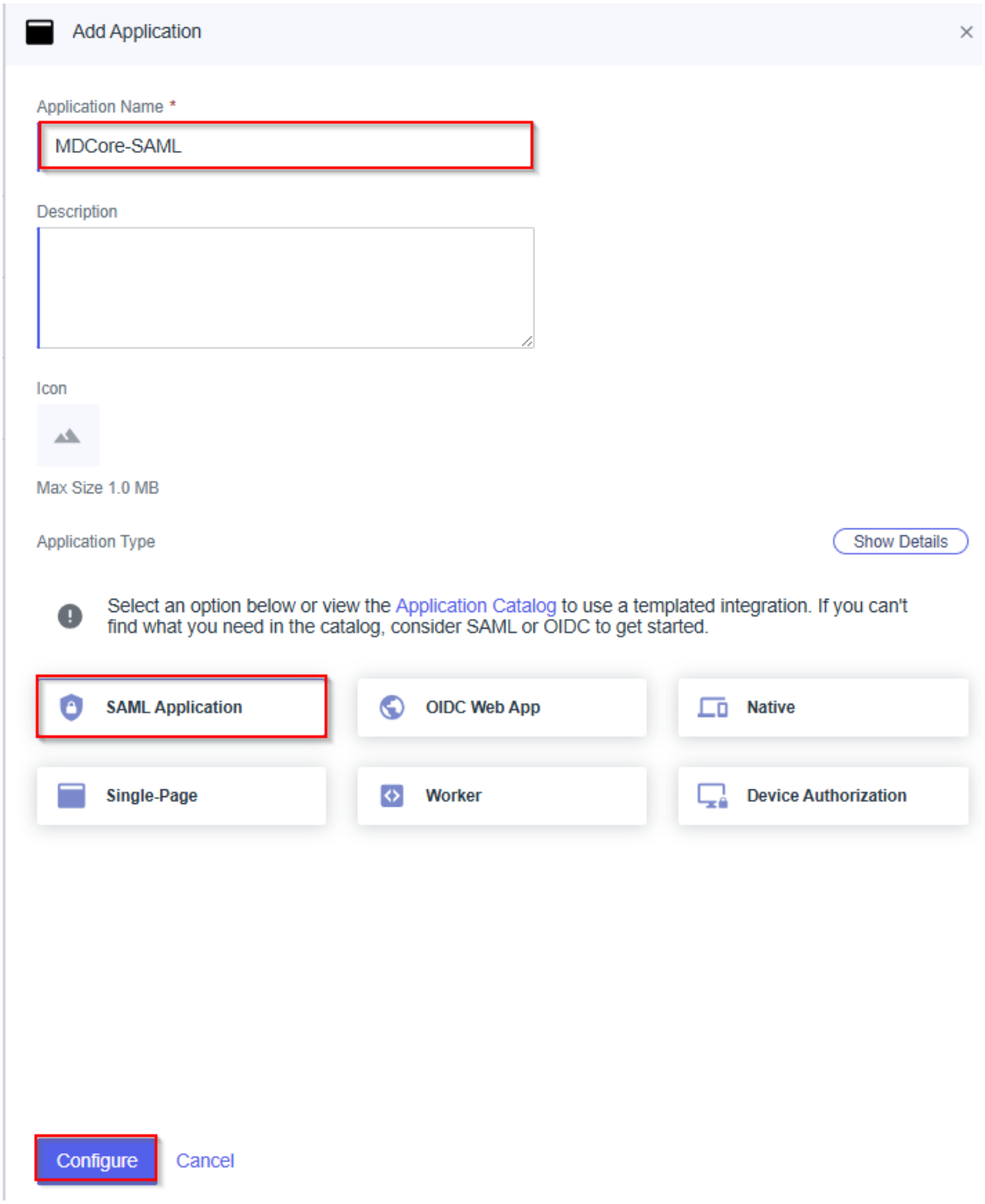
1. In the PingOne Overview page, navigate `Applications` on sidebar



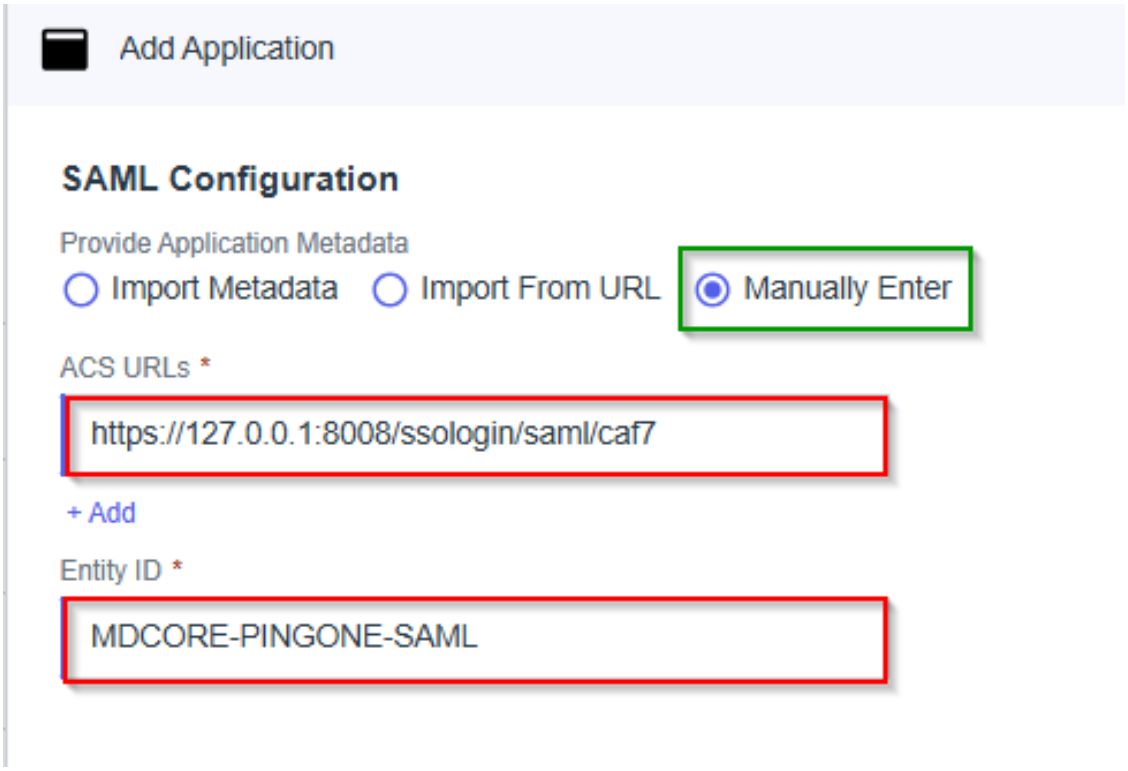
2. Hit on plus button to add new application



3. Fill application name, MDCore-SAML for an example, choose `SAML Application` for Application Type and hit `Configure`

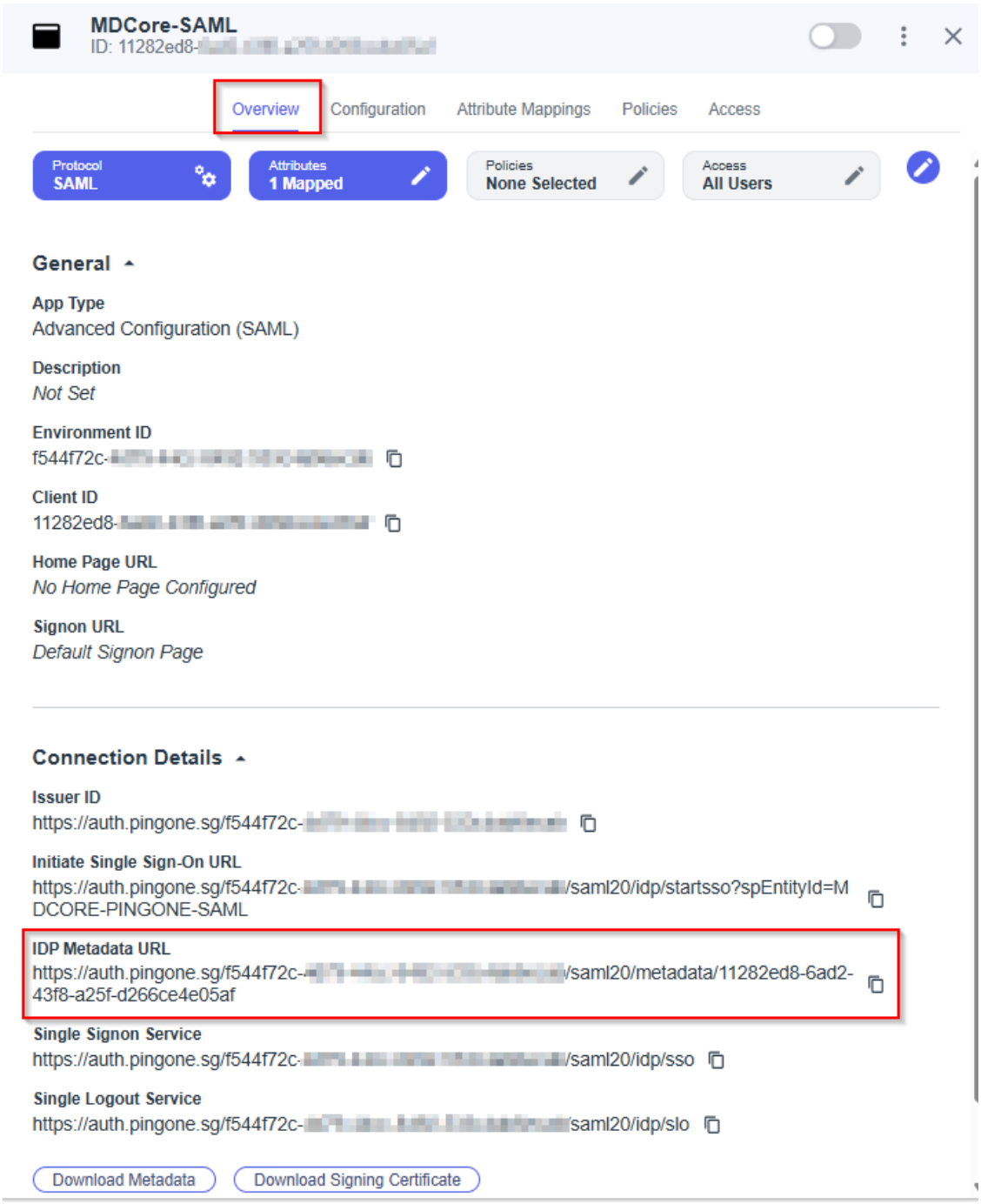


4. Select `Manually Enter` for SAML Configuration, fill `ACS URLs` with *reply_uri*, and `Entity ID` with *entity_id*, and hit `Save`

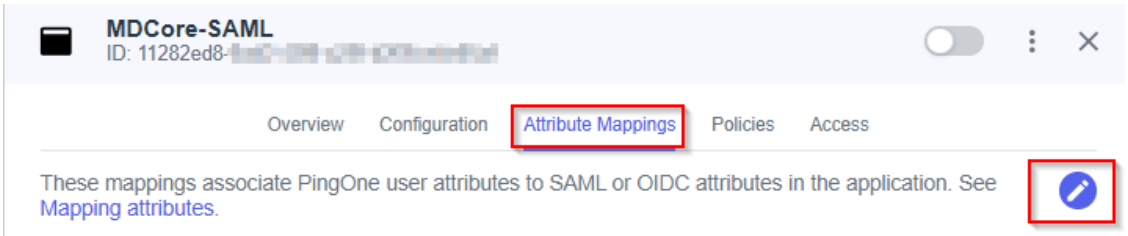


5. Select `Overview` tab, navigate to `IDP Metadata URL`, copy the link below and store to *metadata_url*

—



6. Select `Attribute Mappings` tab and click Edit button.

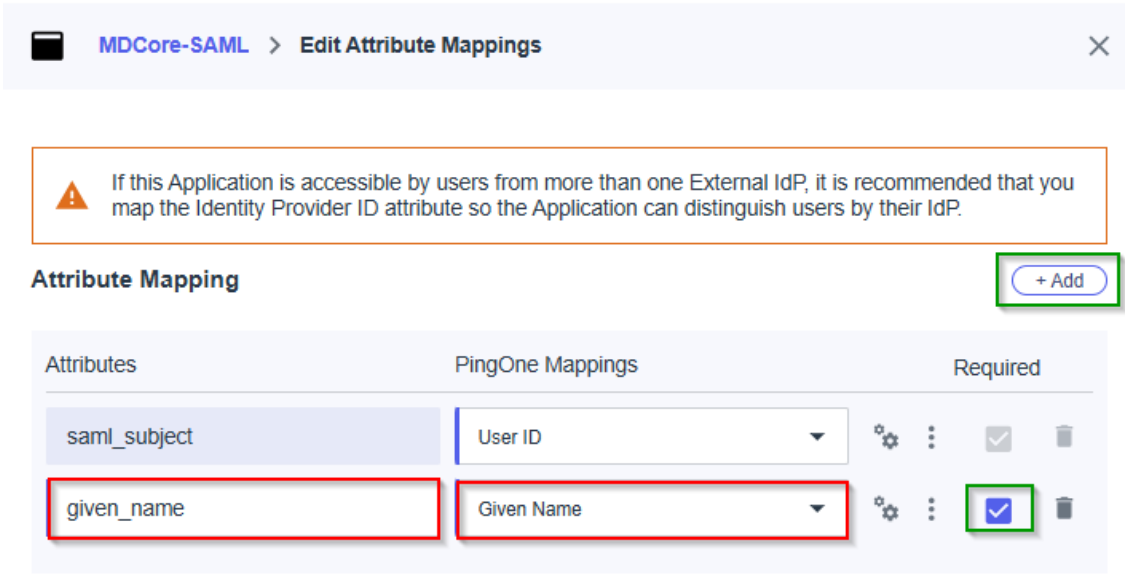


7. In `Edit Attribute Mappings` page, hit `Add` to add new mapping

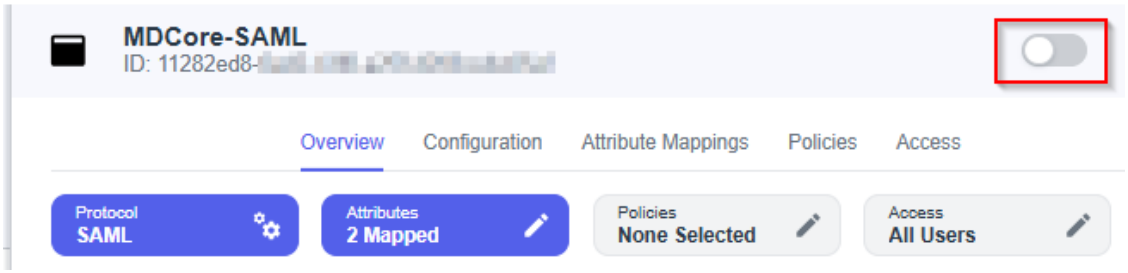
Info

In this step, we make a mapping from the key name used by MDCore and the name exported by PingOne. For example, we set `given_name` mapped to `Given Name` exported by PingOne. Later on, `given_name` is used by MDCore to identify the login user

8. Fill attribute name, `given_name` in this example, select item `Given Name` in PingOne Mappings, enable `Required` option and hit `Save` to complete



9. Enable the new application on PingOne

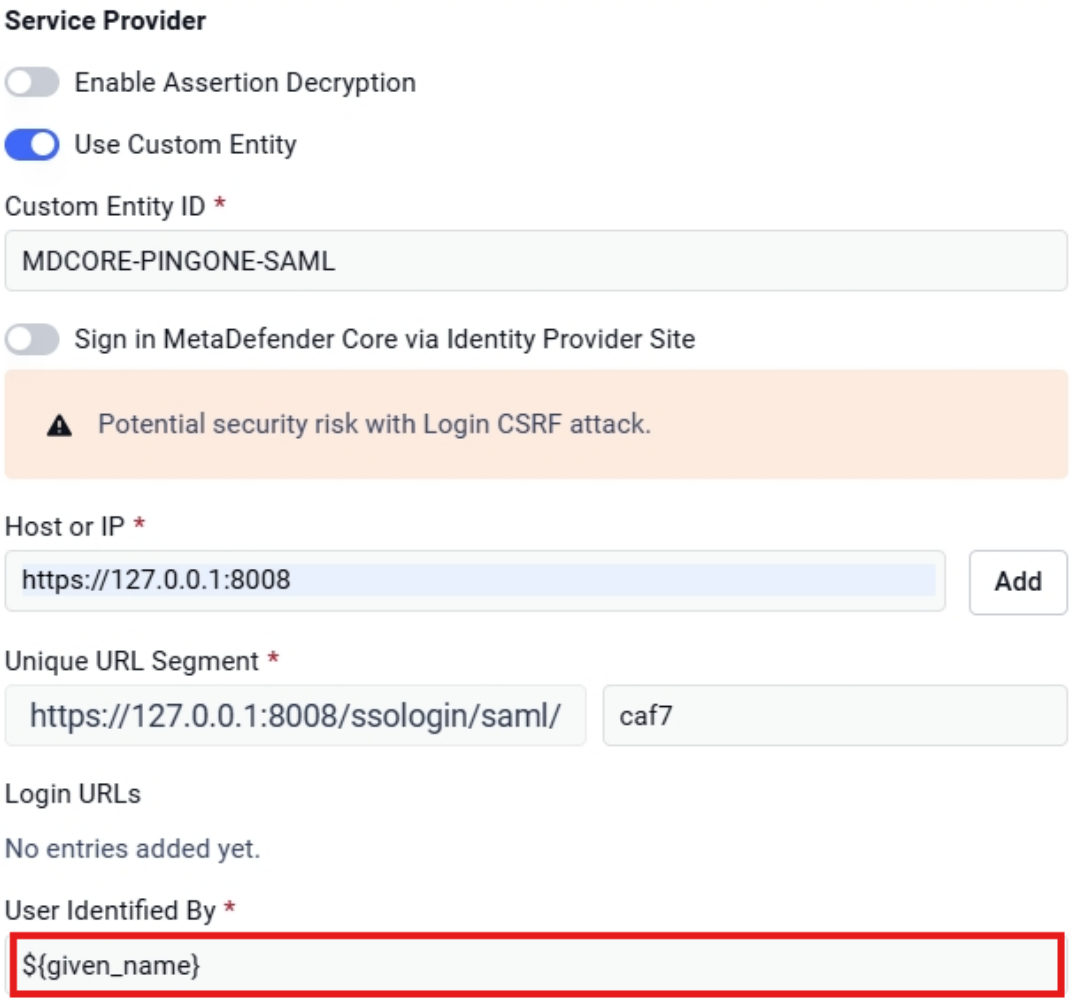


Complete configuration on MDCore

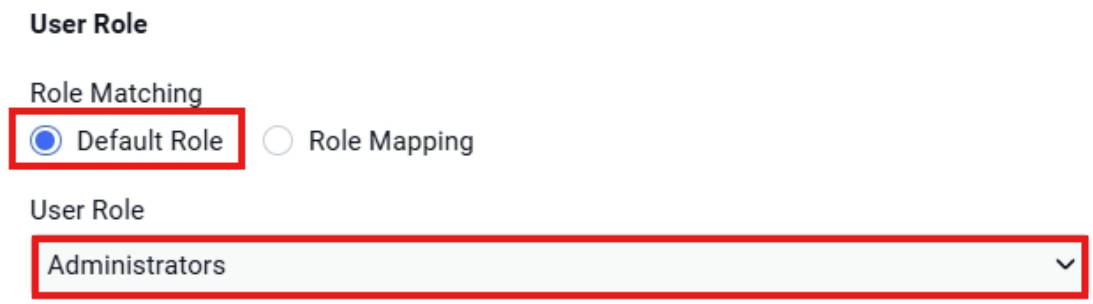
1. Switch back to MDCore, under `Identity Provider`, hit `Fetch URL`, paste *metadata_url* to the box under, and then hit `OK`, which requests MDCore to check and set PingOne as its IDP if succeed



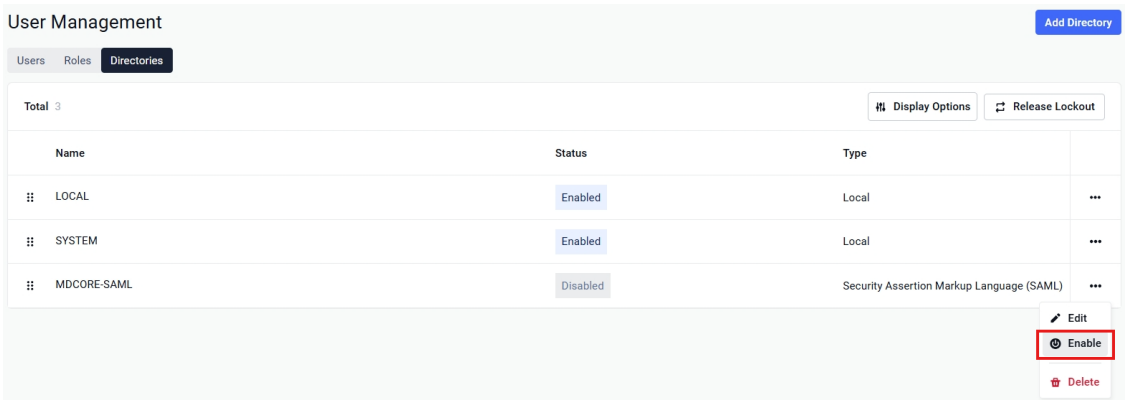
2. Under `Service Provider`, fill ${given_name} in the box under `User identified by`



3. Select `Default role` option, choose the role to assign to login user under `User Role` and hit `Add` to complete setting on MDCore

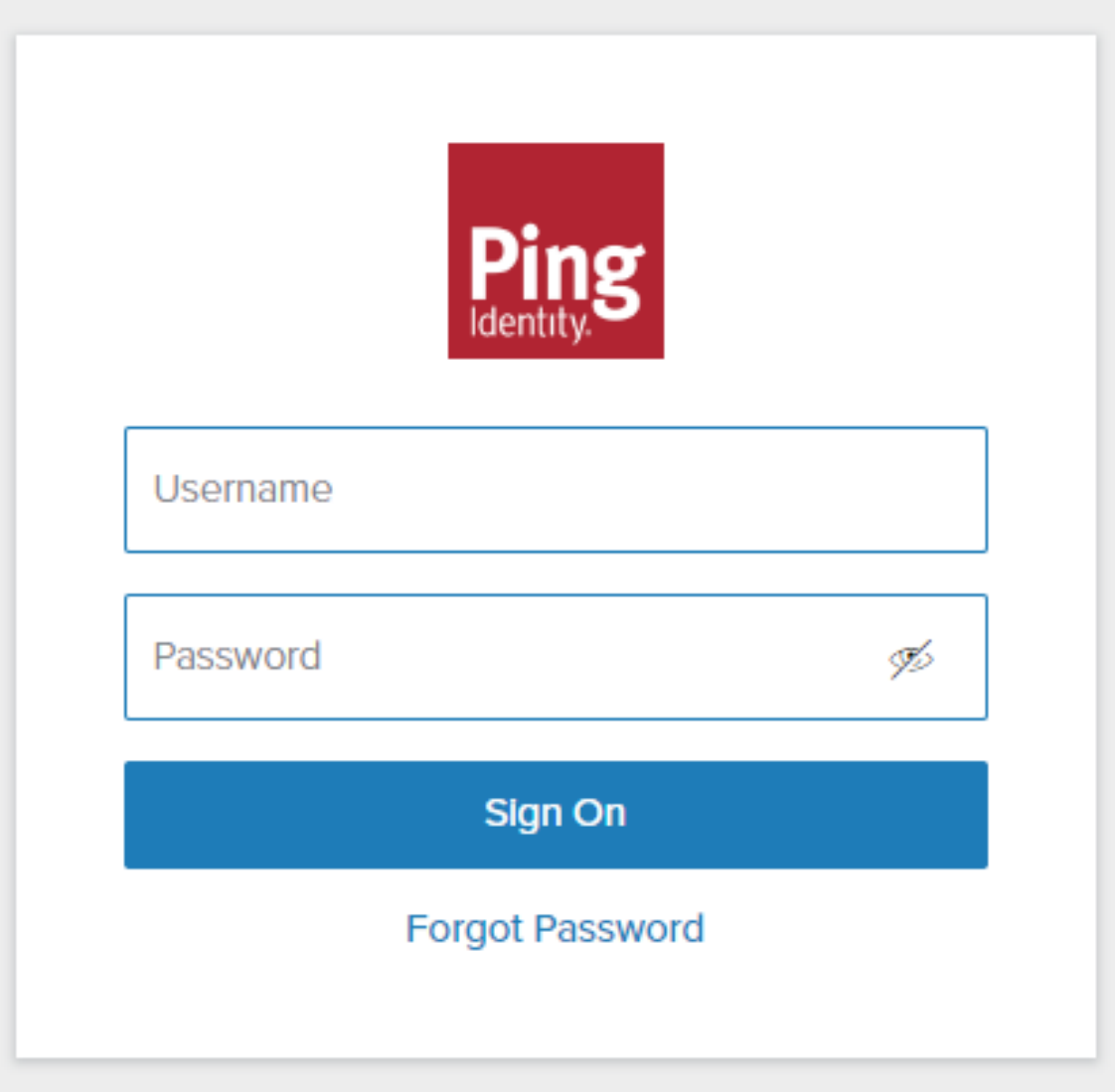


4. In `User Management` page, toggle the new directory, MDCORE-SAML in this example. A dialog box is shown to confirm the action. Once `Enable` is hit, all sessions are expired immediately



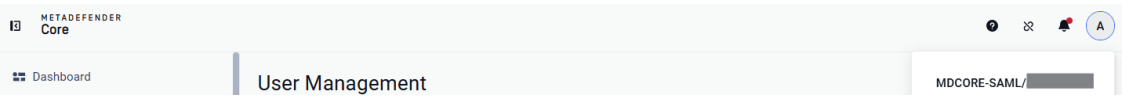
Test the integration

1. In home screen on MDCore, hit `Login`, the user is redirected to login page from PingIdentity



2. Login by the account registered to PingIdentity

3. If everything goes right, MDCore dashboard is shown with user identity set at the top right corner



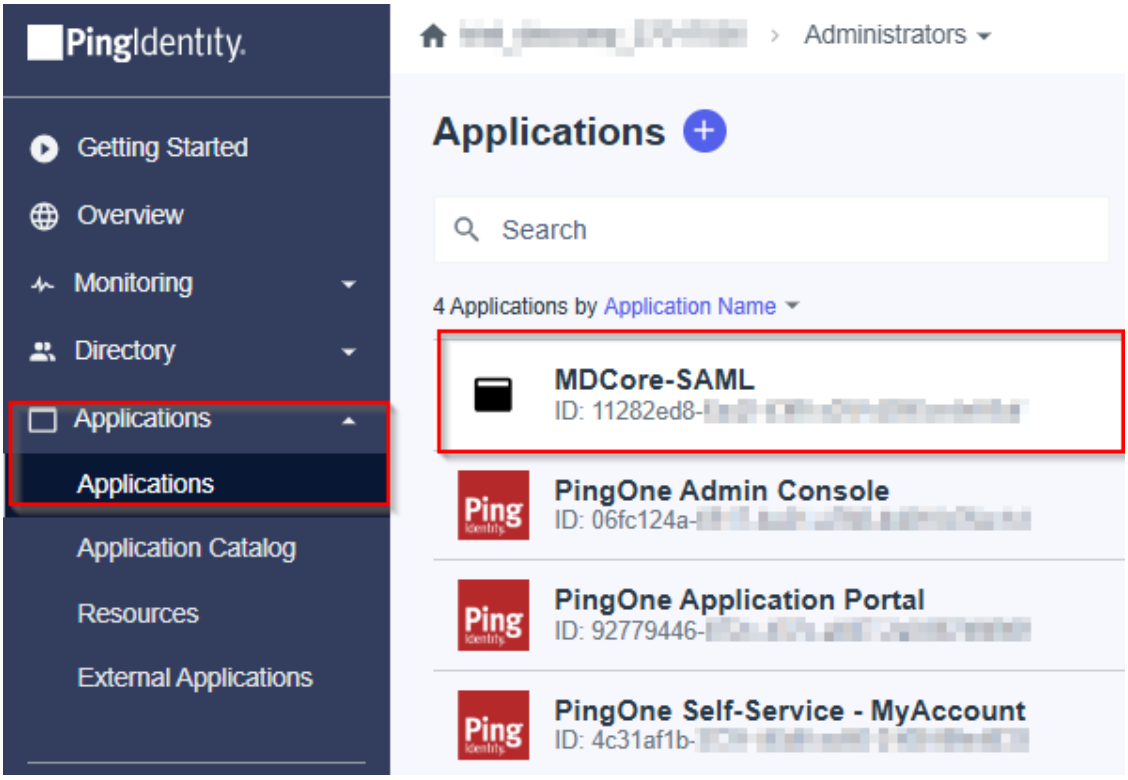
4. Otherwise, access back login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Test IdP-initiated login

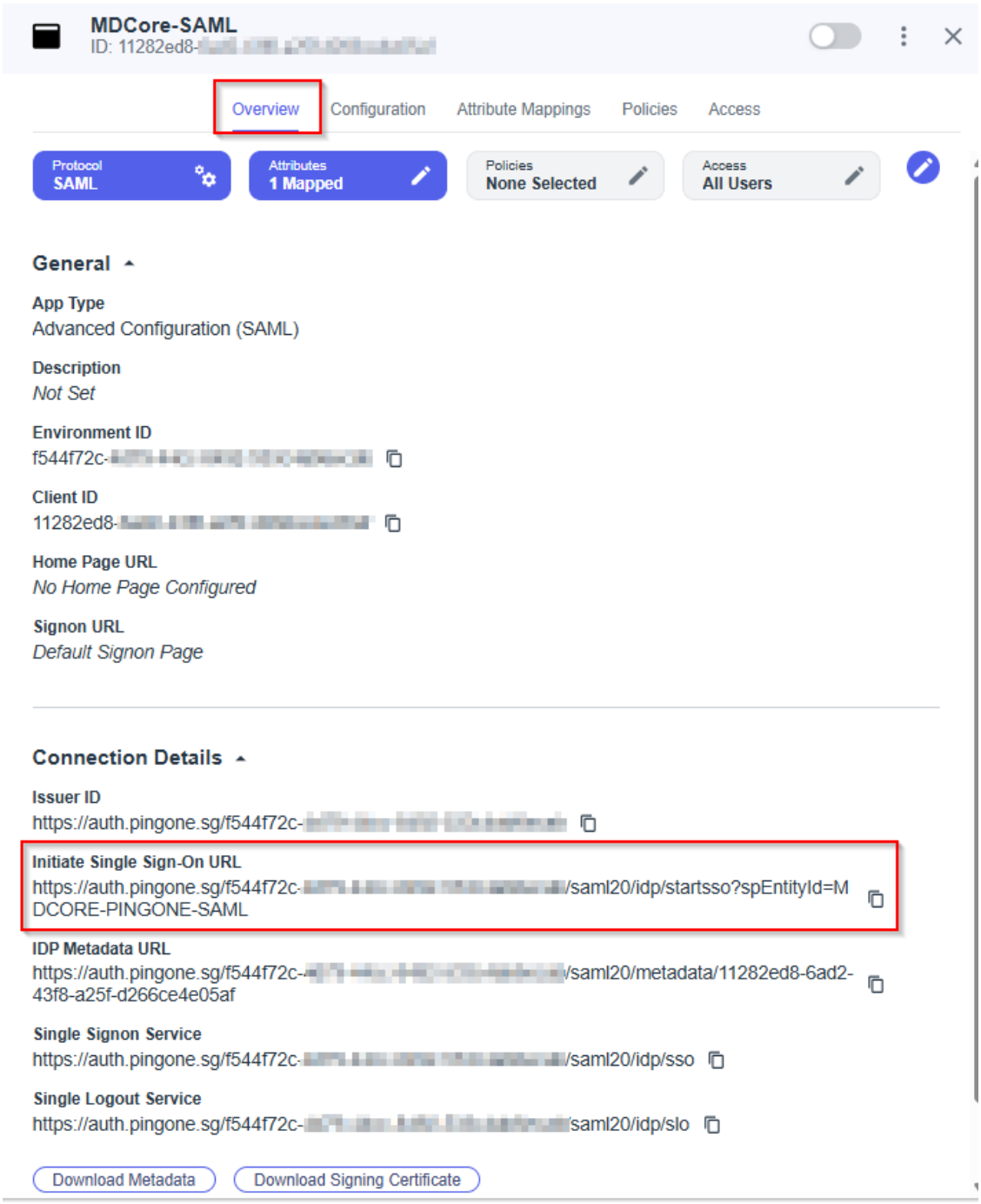
1. Login to PingOne management page.

2. On the left sidebar, hit `Applications`, then `Applications`.

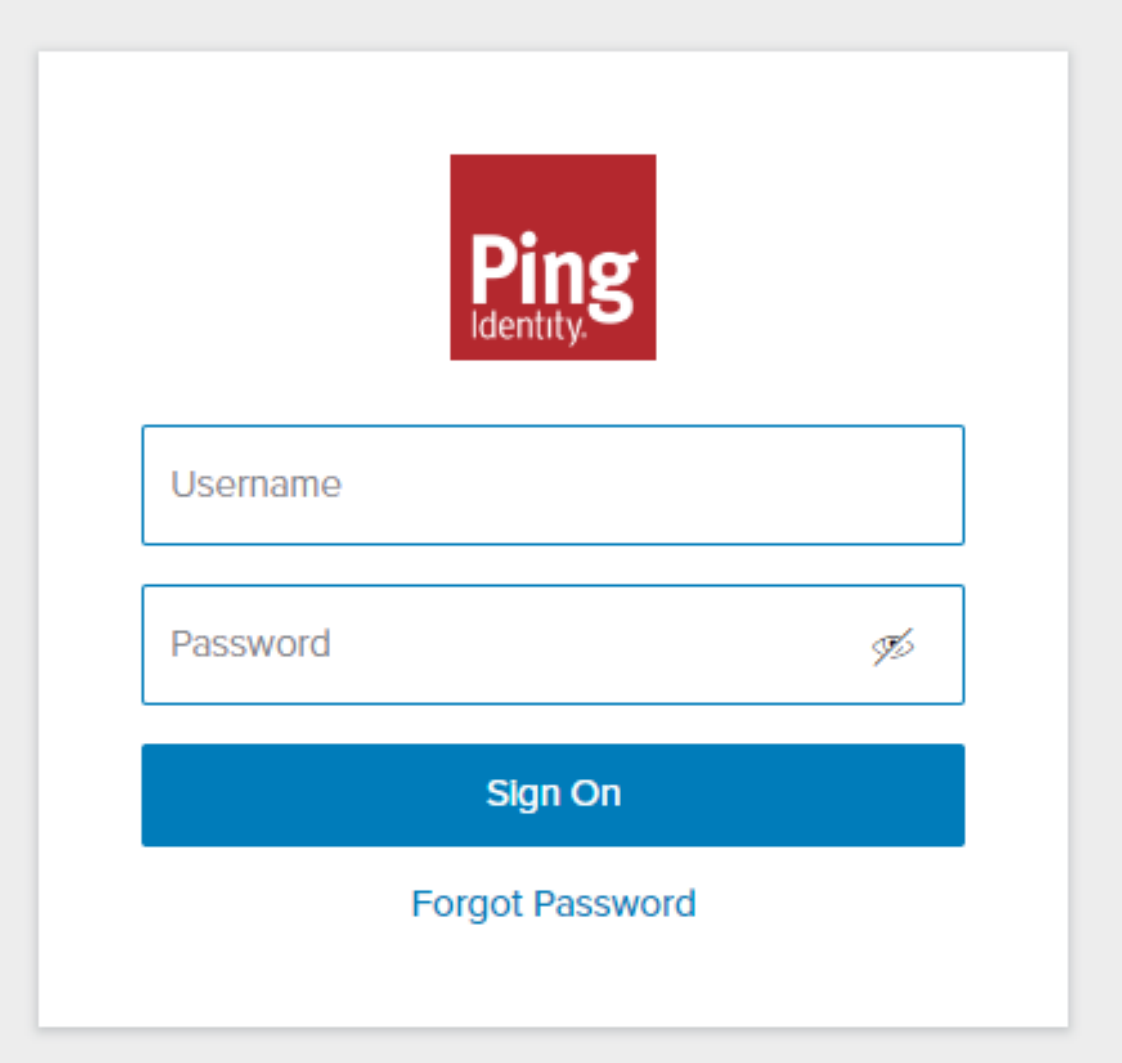
3. Select the SSO directory from which IdP-initiated login is enabled. MDCore-SAML in this example.



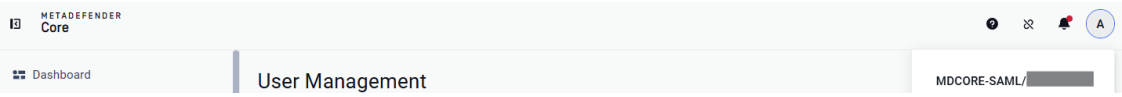
4. On the right sidebar, hit tab `Overview`. Copy the URL under `Initiate Single Sign-On URL`.



5. Paste the URL into your browser and try to sign in.



6. If everything goes right, MDCore dashboard is shown with user identity set at the top right corner

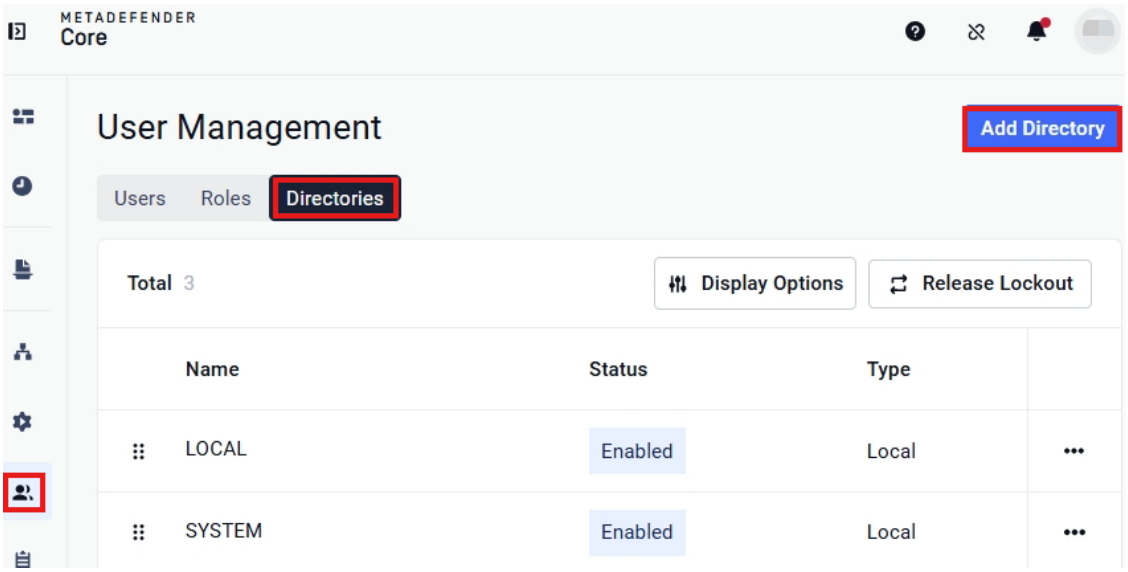


7. Otherwise, access back login page at `<mdcore-host>#/public/backuplogin` for troubleshooting.

Configure OIDC SSO

Create OIDC directory in MDCore

1. Login to MDCore
2. At dashboard, hit `User Management` in sidebar
3. Under `User Management` page, choose `Directories` tab and hit `Add directory` on the top right

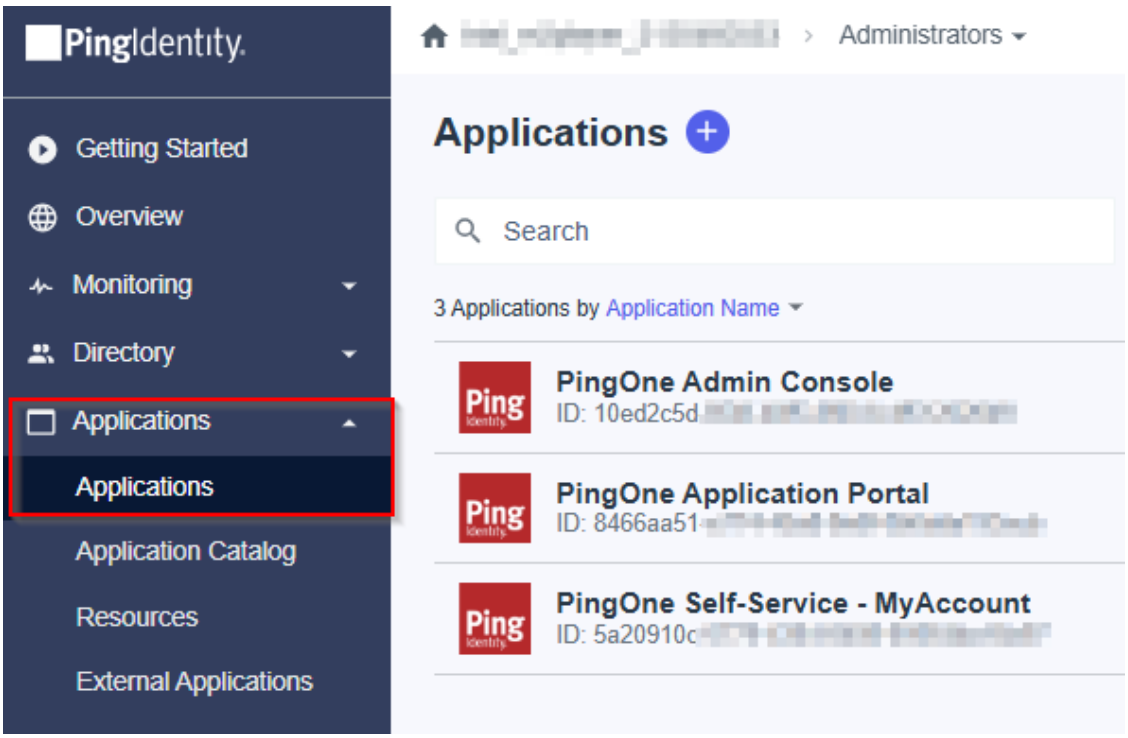


4. In `Add Directory` page, choose `OIDC` in `Directory Type`, fill `Name` for the new directory, MDCore-OIDC for example
5. Fill `Host or IP` where MDCore is hosting, ***https://127.0.0.1:8008*** for this example
6. Copy string generated under `Login URL` and store to *redirect_uri*

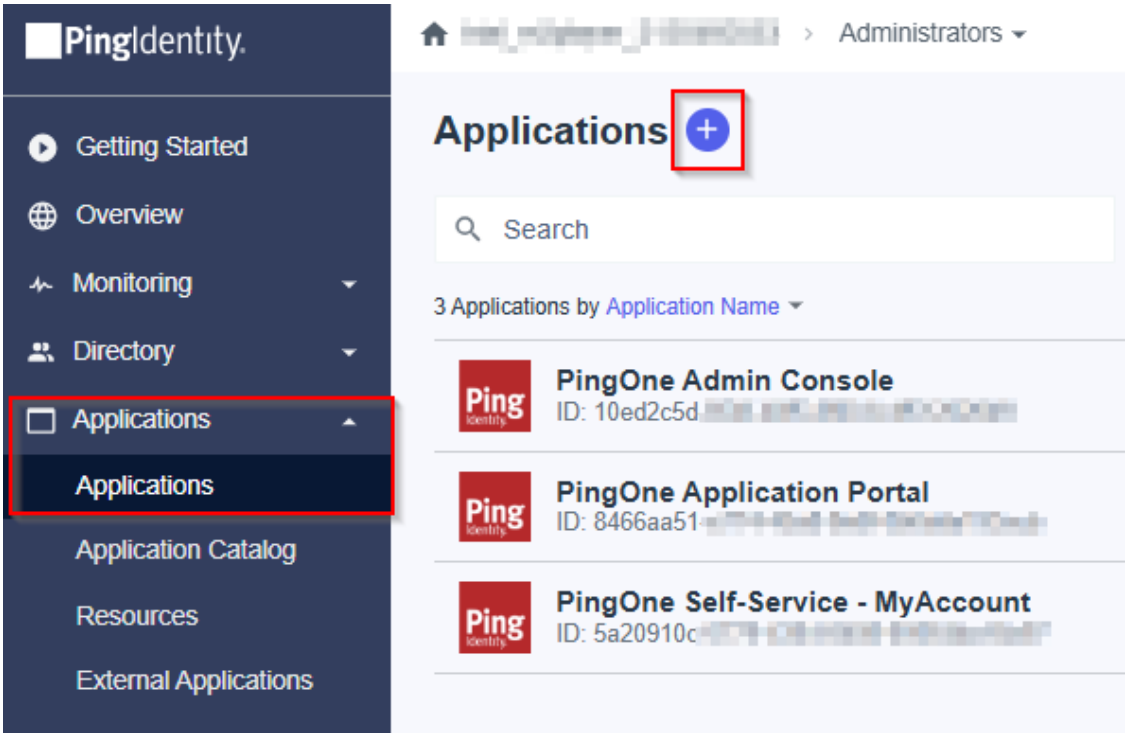
—

Create and configure OIDC application on PingOne

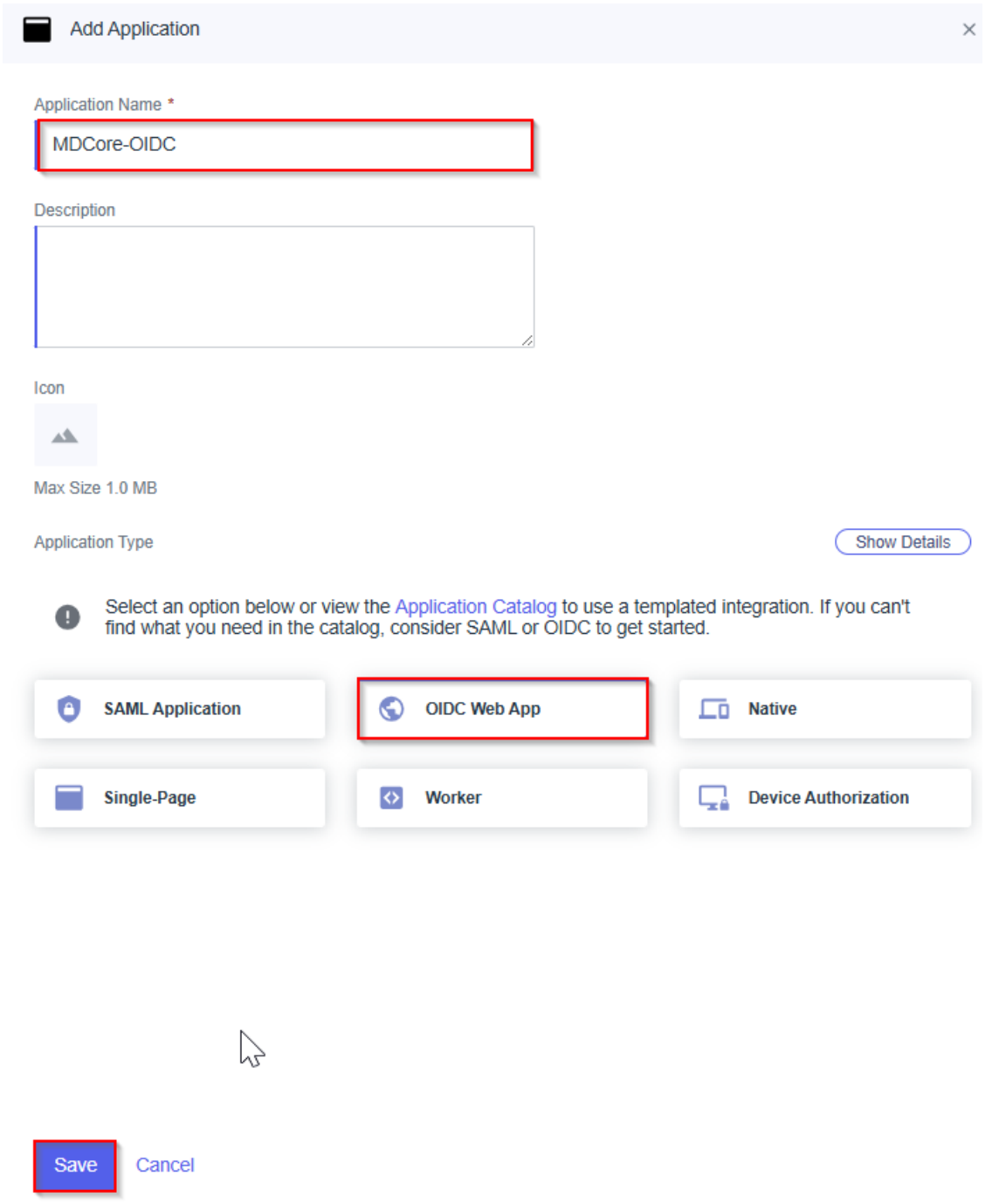
1. In the PingOne Overview page, navigate `Applications`on sidebar



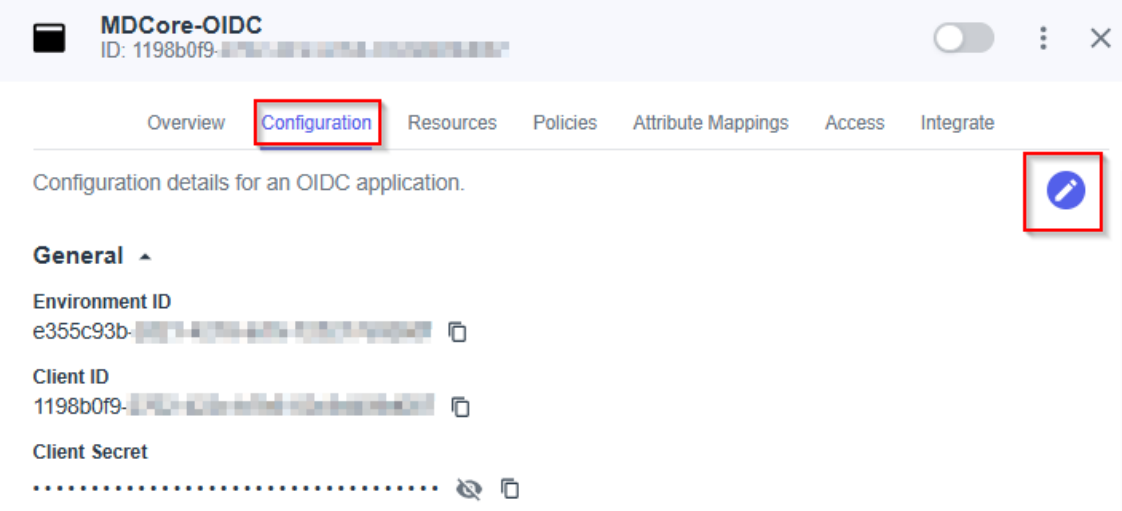
2. Hit on plus button to add new application



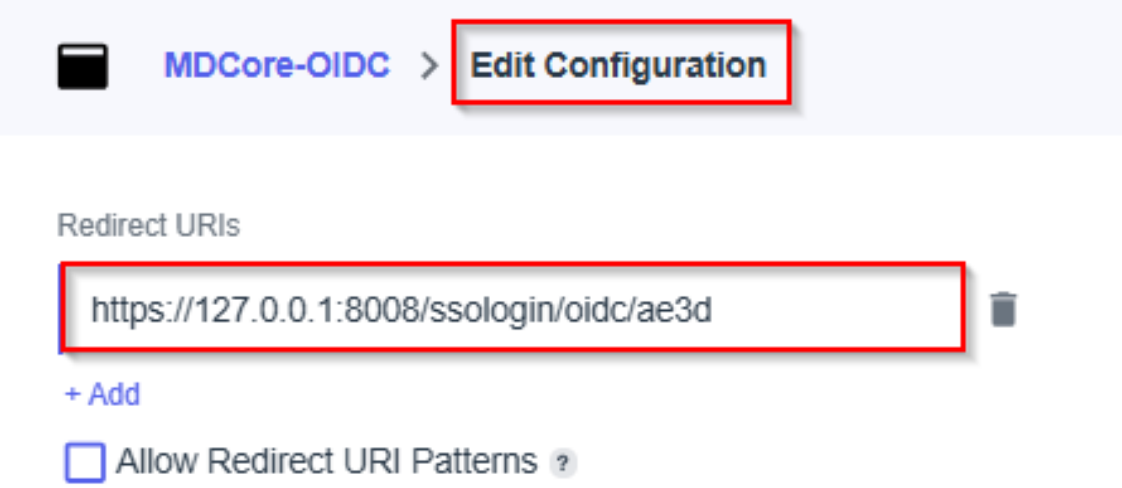
3. Fill application name, MDCore-OIDC for an example, choose `OIDC Web App` for Application Type and hit `Save`



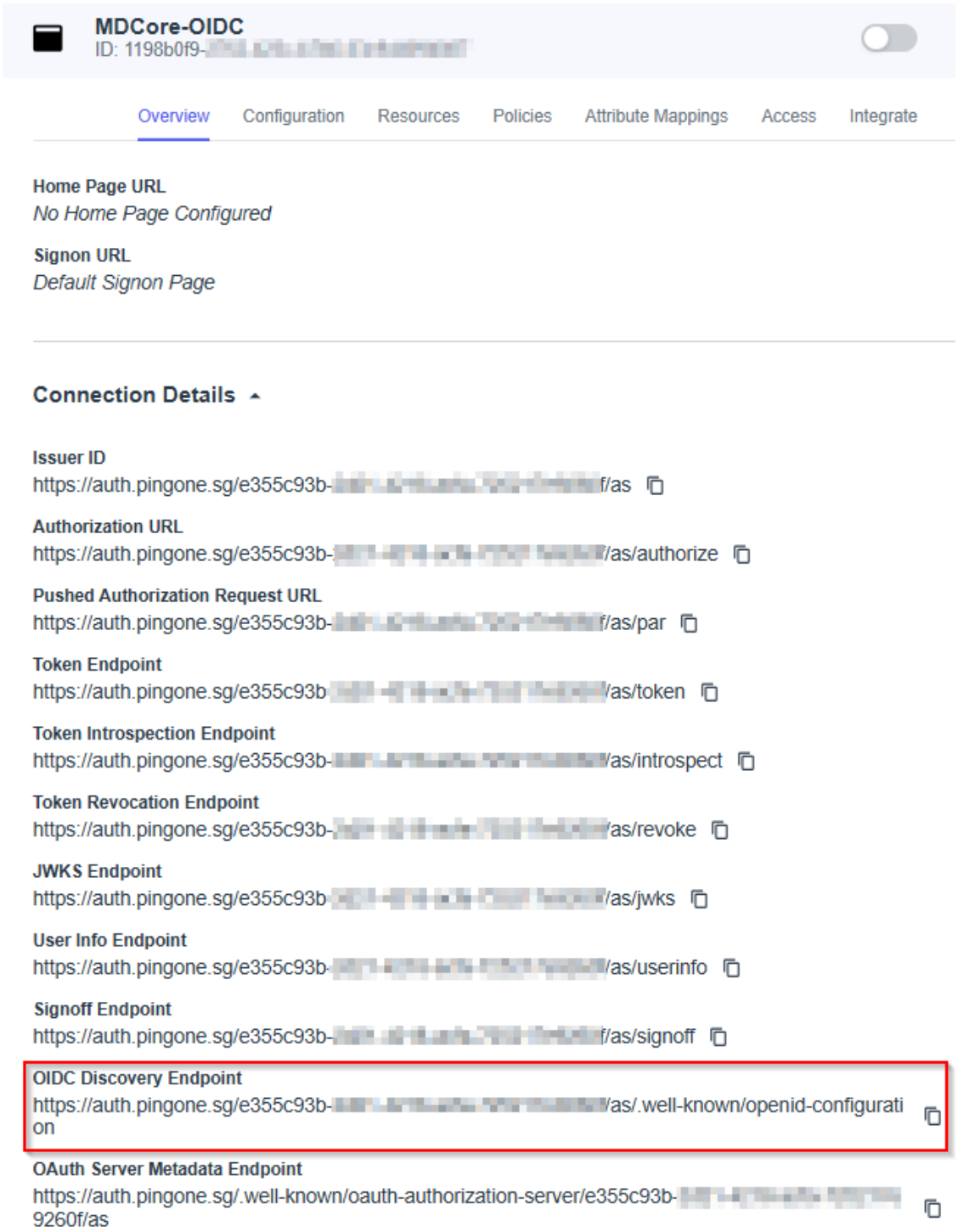
4. Select `Configuration` tab, hit Edit button



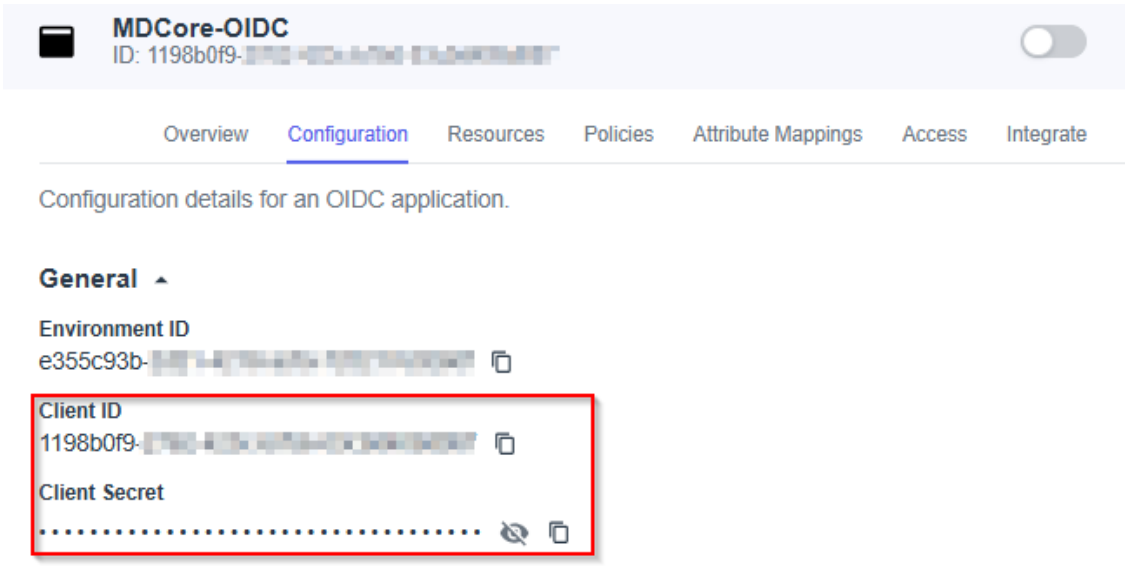
5. In `Edit Configuration` page, navigate to `Redirect URIs` and paste ***redirect_uri*** to the box below, then hit `Save`



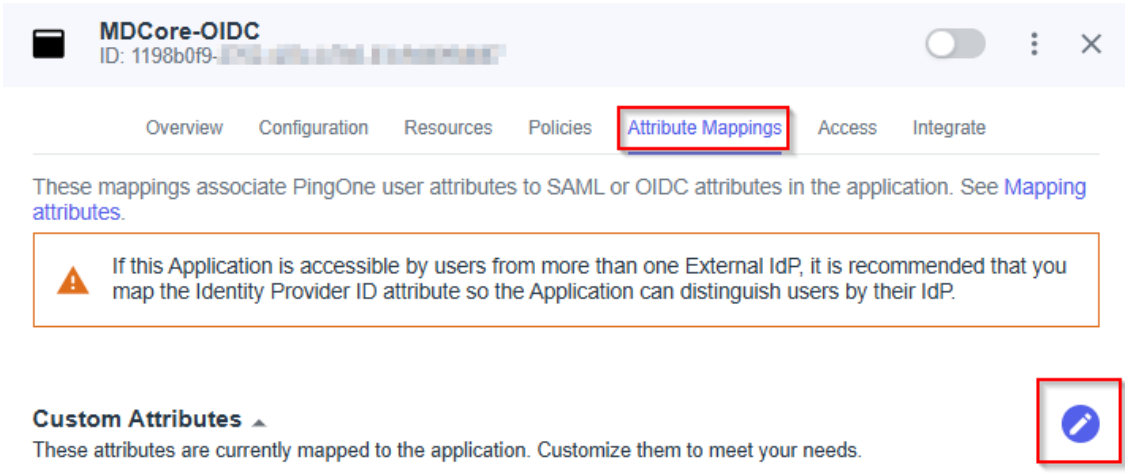
6. In `Overview` tab, navigate to `OIDC Discovery Endpoint`, copy the link below and store to *metadata_url*



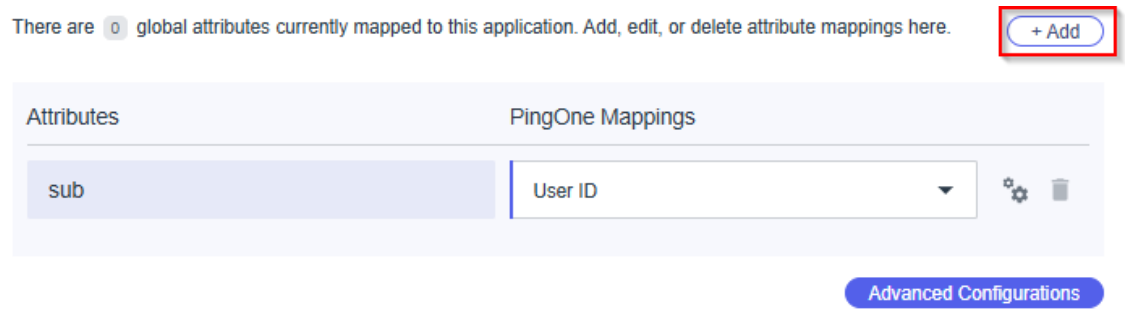
7. Select Configuration tab, expand General section, navigate to Client ID and Client Secret, copy and store them in client_id and client_secret.



8. Select Attribute Mappings tab and hit Edit button



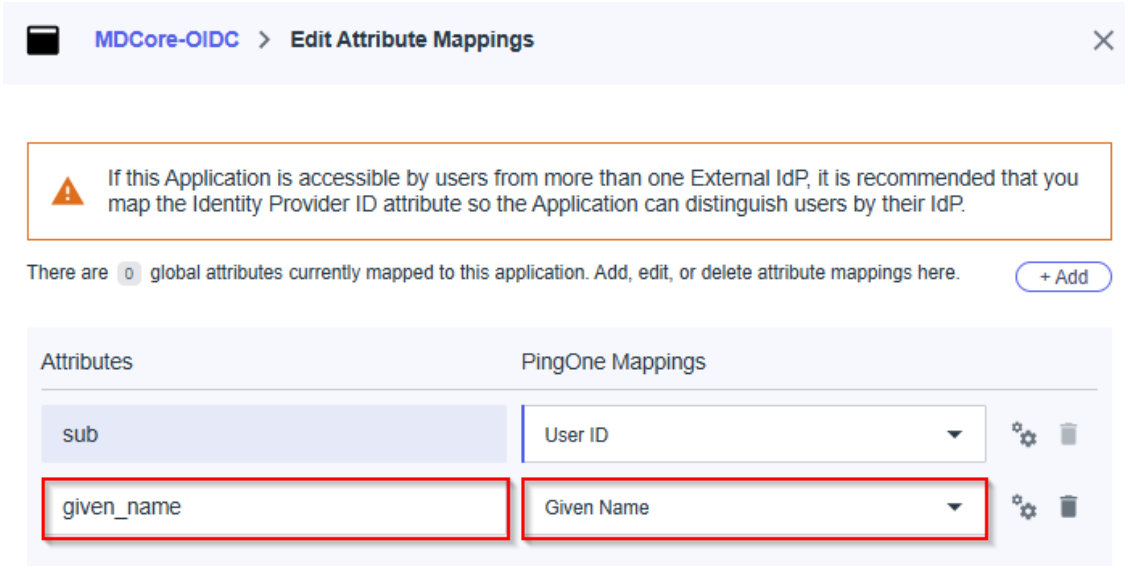
9. In Edit Attribute Mappings page, hit Add to add new mapping



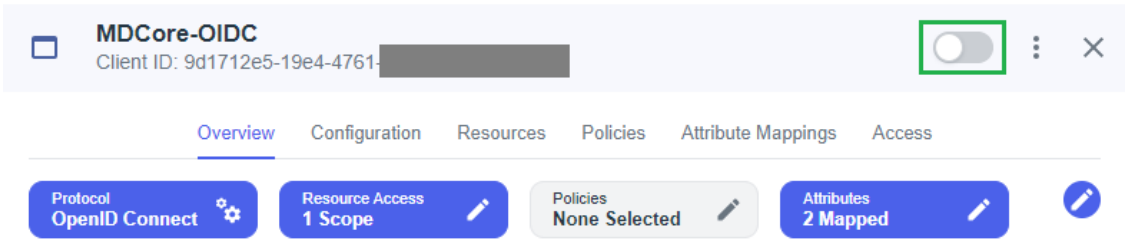
Info

In this step, we make a mapping from the key name used by MDCore and the name exported by PingOne. For example, we set `given_name` mapped to `Given Name` exported by PingOne. Later on, `given_name` is used by MDCore to identify the login user

10. Fill attribute name, `given_name` in this example, select item `Given Name` in PingOne Mappings and hit `Save` to complete

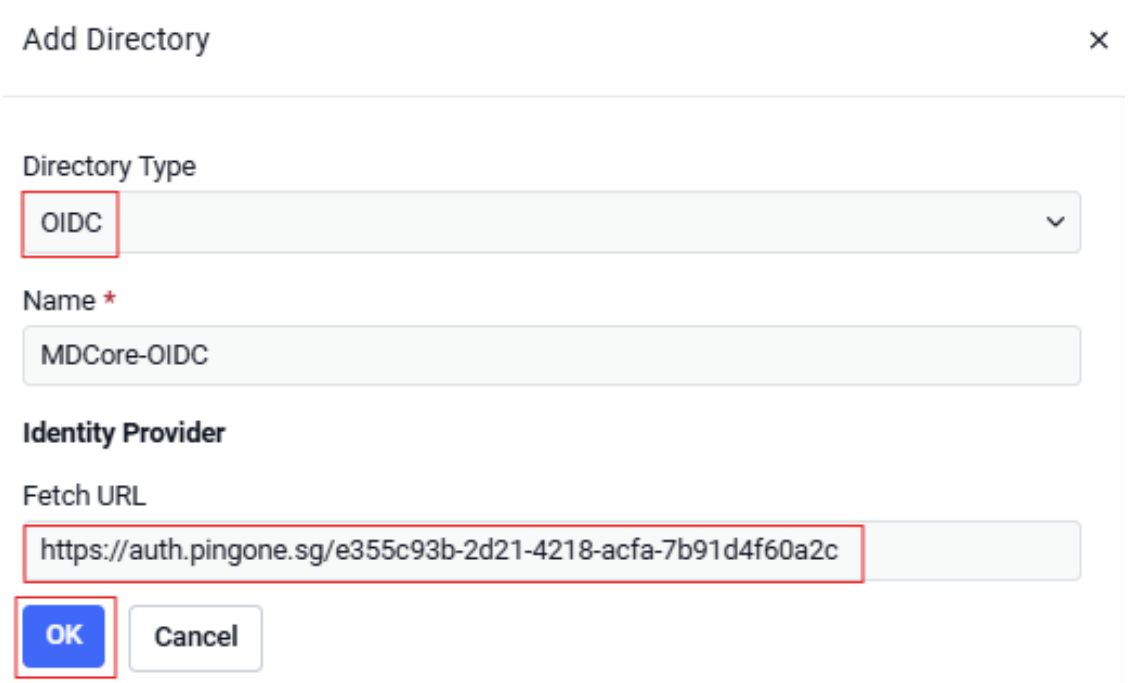


11. Enable the new application on PingOne

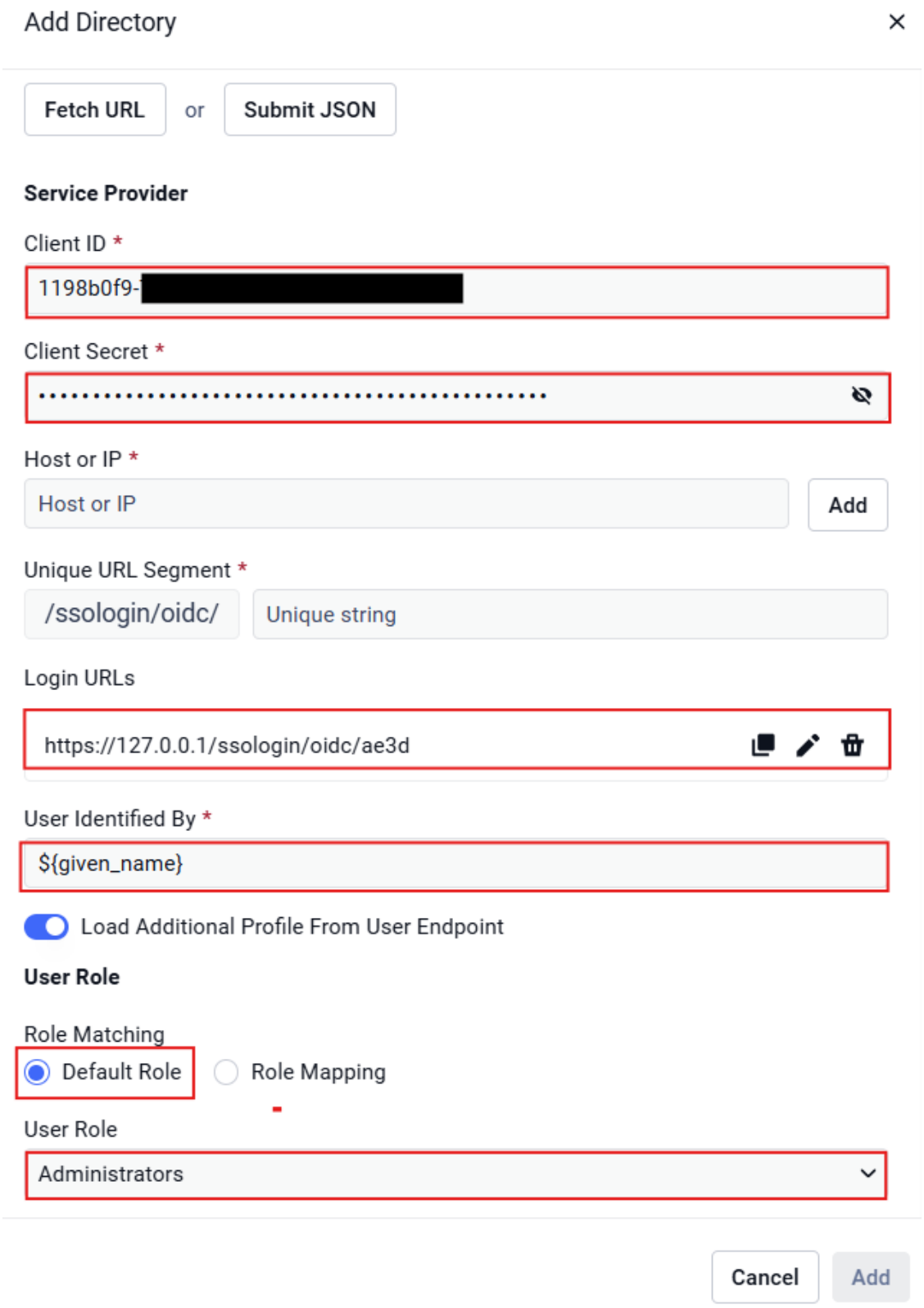


Complete configuration on MDCore

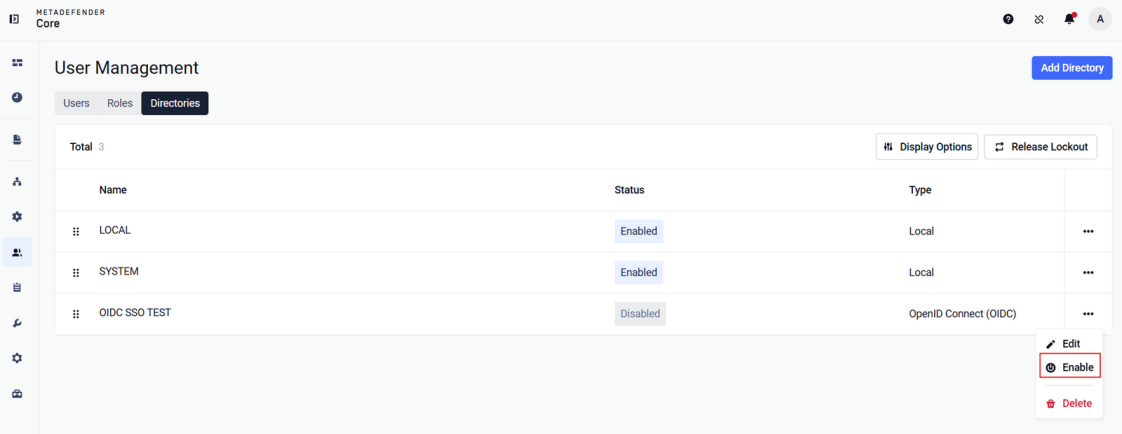
1. Switch back to MDCore, under Identity Provider, hit Fetch URL, paste metadata_url to the box under, and then hit OK, which requests MDCore to check and set PingOne as its IDP if succeed



2. Under `Service Provider`, paste ***client_id*** and ***client_secret*** to boxes under `Client ID` and `Client secret` respectively
3. Fill ${given_name} in the box under `User identified by`
4. Select `Default role` option, choose the role to assign to login user under `User Role` and hit `Add` to complete setting on MDCore

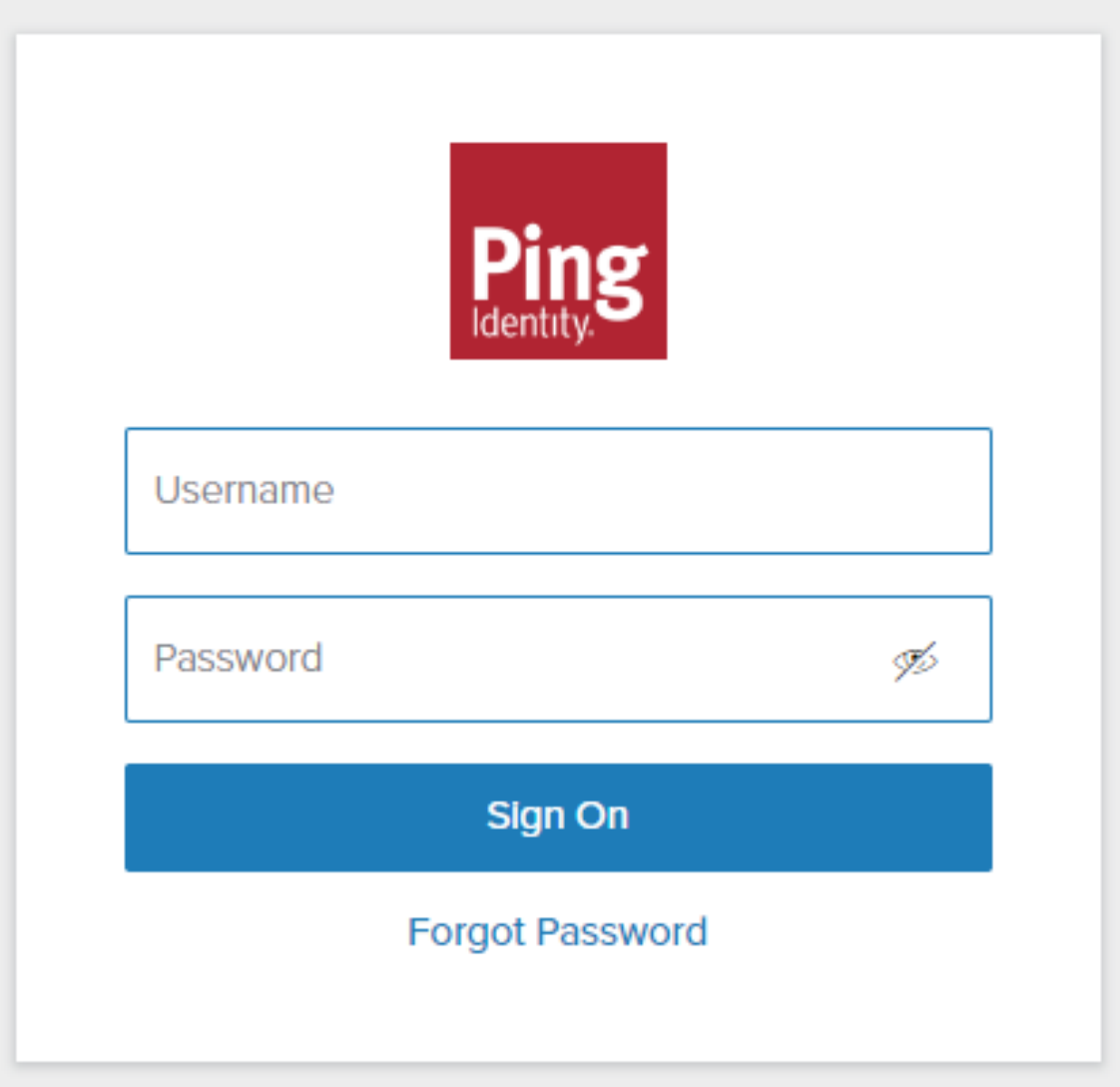


5. In `User Management` page, toggle the new directory, MDCORE-OIDC in this example. A dialog box is shown to confirm the action. Once `Enable` is hit, all sessions are expired immediately



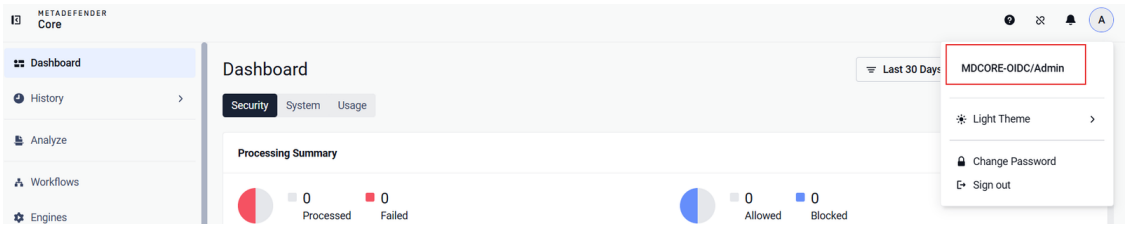
Test the integration

1. In Home screen on MDCore, hit `Login`, the user is redirected to login page from PingIdentity



2. Login by the account registered to PingIdentity

3. If everything goes right, MDCore dashboard is shown with user identity set at the top right corner



4. Otherwise, access back login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Best Practices for Configuration & Security Hardening Guide

Introduction

This page provides recommended baseline configurations for a standard MetaDefender Core deployment. The goal is to offer secure, stable, and maintainable defaults that can be applied to new installations or used as a reference to validate existing environments. It focuses on three main areas:

- **PostgreSQL** configuration (performance, stability, SSL/TLS)
- **NGINX** configuration (HTTPS enablement, access restrictions, surface-area reduction)
- **MetaDefender Core** application settings (account lockout, password policy, data retention, email notifications, user roles, and proxy usage, …)

PostgreSQL recommendations

Remote PostgreSQL

Recommended baseline remote PostgreSQL settings for MetaDefender Core to balance stability, performance, and maintainability. Adjust values to your environment size (connections, RAM, workload) and monitor over time. You can apply these metrics for `postgresql.conf` .

postgresql.conf · bash

```
max_connections                 100
shared_buffers                  128MB
max_wal_size                    2GB
min_wal_size                    80MB
logging_collector               on
log_file_mode                   0640
work_mem                        100MB
maintenance_work_mem            200MB
autovacuum_vacuum_threshold     100000
autovacuum_vacuum_scale_factor  0
log_min_duration_statement      200
```

Info

The recommendation above applies primarily to **remote** PostgreSQL deployments. For bundled PostgreSQL that ships with MetaDefender Core, the corresponding configuration is already handled by the installer and default setup.

For some settings related to the hardware resources like `work_mem`, `maintenance_work_mem` (RAM), `max_wal_size` (disk), etc. You can adjust those setting to match with your current system.

PostgreSQL SSL

Enable SSL/TLS for connections between MetaDefender Core and PostgreSQL to encrypt credentials, queries, and results in transit, protect against eavesdropping on the network, and meet compliance requirements for data-in-transit encryption.

Info

Enabling SSL introduces additional CPU overhead for connection setup and encryption, which can have an impact on database **performance**.

For more details, please check SSL Connection for bundled PostgreSQL.

NGINX recommendations

Enable HTTPS

For more details, please check: Enabling HTTPS.

Enabling HTTPS encrypts all traffic between clients and MetaDefender Core, ensuring data confidentiality and integrity in transit.

Use TLS 1.3 for stronger security, faster handshakes, and modern cipher suites. Disable older protocols (TLS 1.2 and below) unless strict compatibility is required.

Restrictions

Apply additional NGINX security restrictions to limit which hosts, clients, and referrers can access MetaDefender Core. These rules help reduce attack surface by only allowing trusted domains, blocking abusive user-agents, filtering referrer spam, and denying access to sensitive APIs.

For more details, please check: NGINX Configuration.

1. **Only allow access to our domain only** - Restrict access so that only the allowed MetaDefender Core domains can reach this service. Any request with a different Host header is immediately dropped with status code 444 (no response), which helps reduce attack surface and prevent abuse through unauthorized virtual hosts.

Code · bash

```
if ($host !~ ^(metadefendercore.in|www.metadefendercore.in|images.metadefendercore.in)$ ) {
    return 444;
}
```

** 2) Deny certain user-agents - Block abusive or unwanted user-agents such as automated scanners, generic download tools, and known spam bots that may overload or probe your server for vulnerabilities. The following rules return HTTP 403 for matching user-agents to reduce noise, protect resources, and keep logs cleaner. **

Code · bash

```
## Block download agents ##
if ($http_user_agent ~* LWP::Simple|BBBike|wget) {
    return 403;
}
##

## Block robots ##
if ($http_user_agent ~* msnbot|scrapbot) {
    return 403;
}
##
```

** 3) Block referral spam - Block requests that come from unwanted external websites (referrer spam). Only direct access or requests from legitimate pages are allowed; any request whose `Referer` header matches the patterns below is rejected with HTTP 403 to reduce spam traffic and potential malicious redirects. **

Code · bash

```
## Deny certain Referers ###
if ( $http_referer ~* (babes|forsale|girl|jewelry|love|nudit|organic|poker|porn|sex|teen) ) {
    return 403;
}
##
```

4) Block particular APIs - Deny direct access to sensitive administration endpoints that should never be exposed over the public interface. The following rule blocks requests to `/admin/config/session`

Code · bash

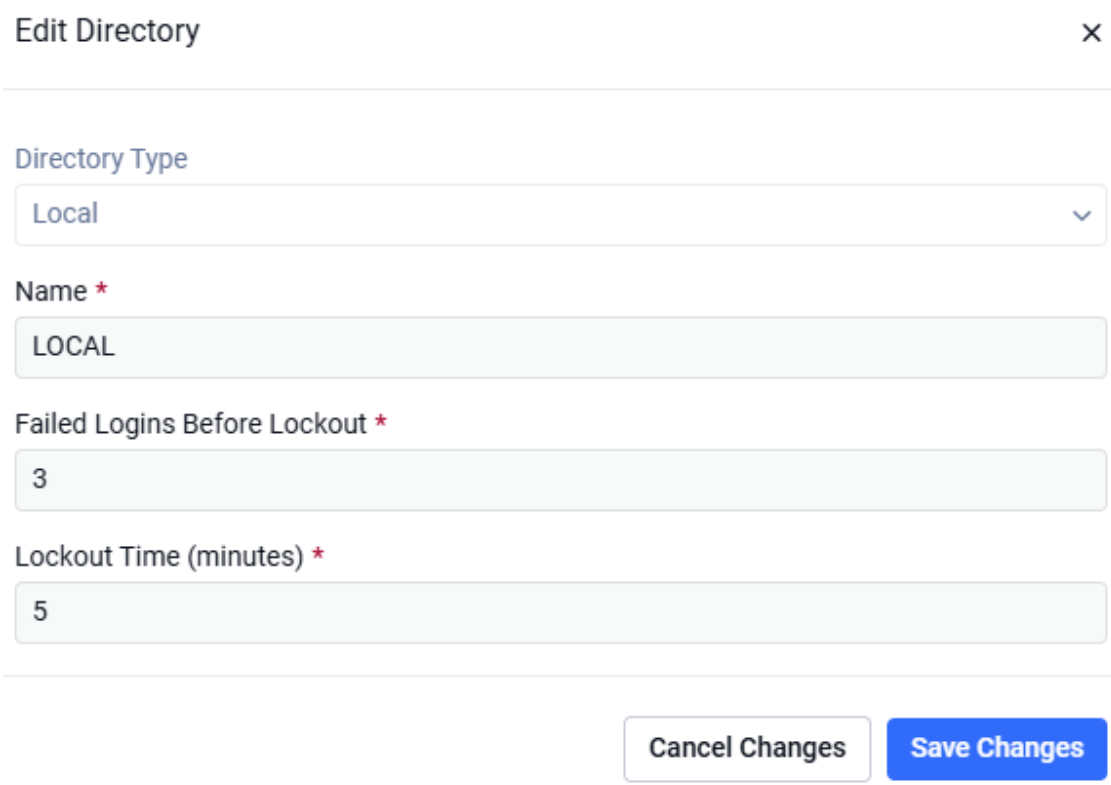
```
## Block particular api ###
location ~ "^/admin/config/session$" {
  return 403;
}
##
```

MetaDefender Core recommended settings

Account lockout

Configure lockout thresholds to slow down brute-force attempts while minimizing impact on legitimate users. Use a low retry limit and a short, temporary lockout to balance security and usability.

Account lockout settings are accessible under **Users Management > Directories** tab.



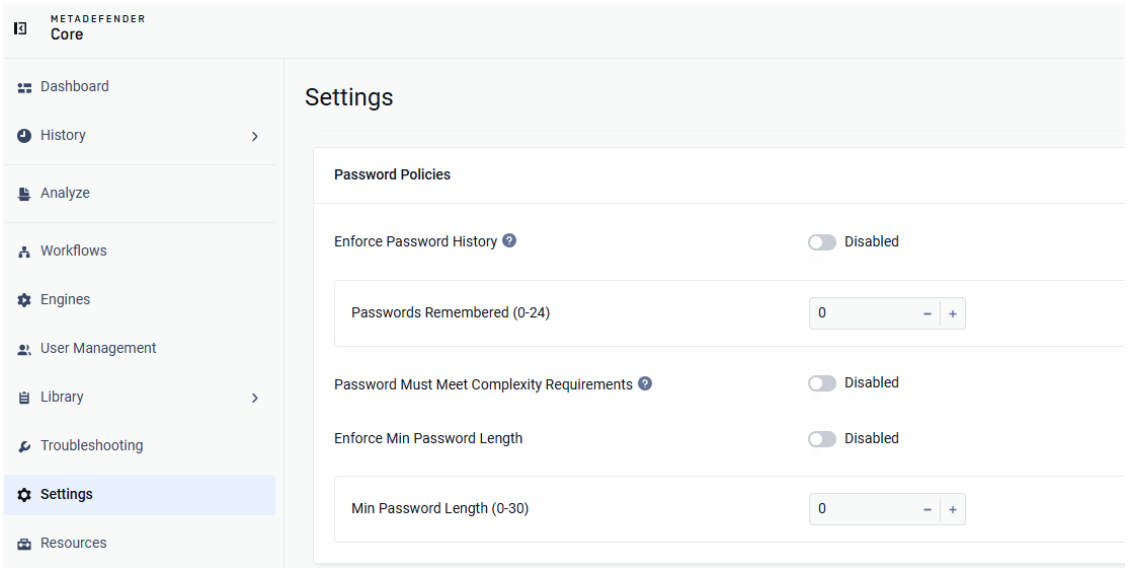
Info

Only LOCAL/SYSTEM users have account lockout.

Password policy

Define a minimum password complexity to protect local MetaDefender Core accounts from guessing and brute-force attacks. Use the following baseline policy as a secure default and tighten it further if your organization has stricter compliance requirements.

Password policies settings are accessible under **Settings > Security** tab.

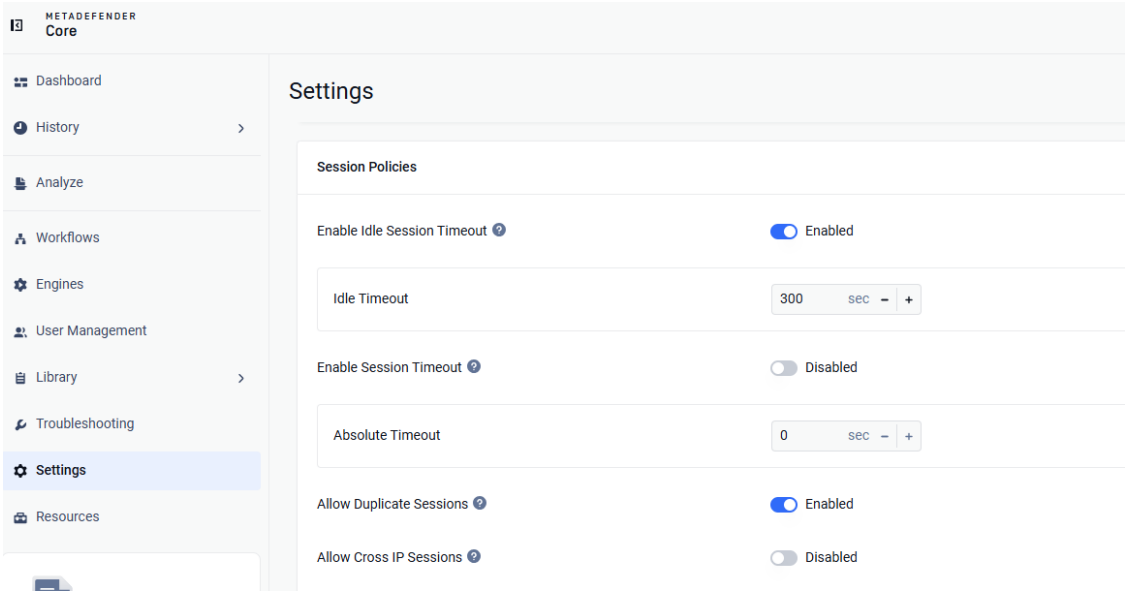


Session policy

Configure session timeout and idle limits so that unused authenticated sessions are closed automatically, reducing the risk of account misuse on shared or unattended workstations. Set shorter timeouts for environments with stricter security requirements.

For more details, please check: Session Policy.

Session Policy settings are accessible under **Settings > Security** tab.

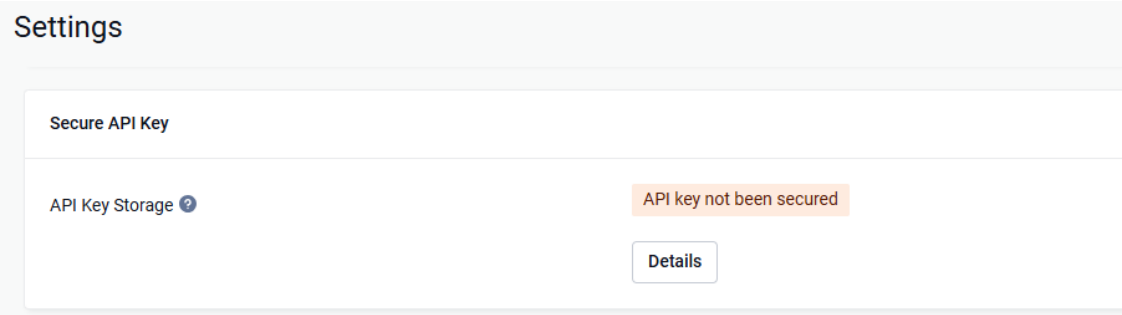


Secure API key

Store API keys in an encrypted form in the database to limit exposure in case of data leakage, unauthorized access, or backup compromise. Use strong encryption algorithm, protect encryption keys with strict access controls.

> **Info**
>
> Only LOCAL/SYSTEM users have API key

Secure API key settings are accessible under **Settings > Security** tab

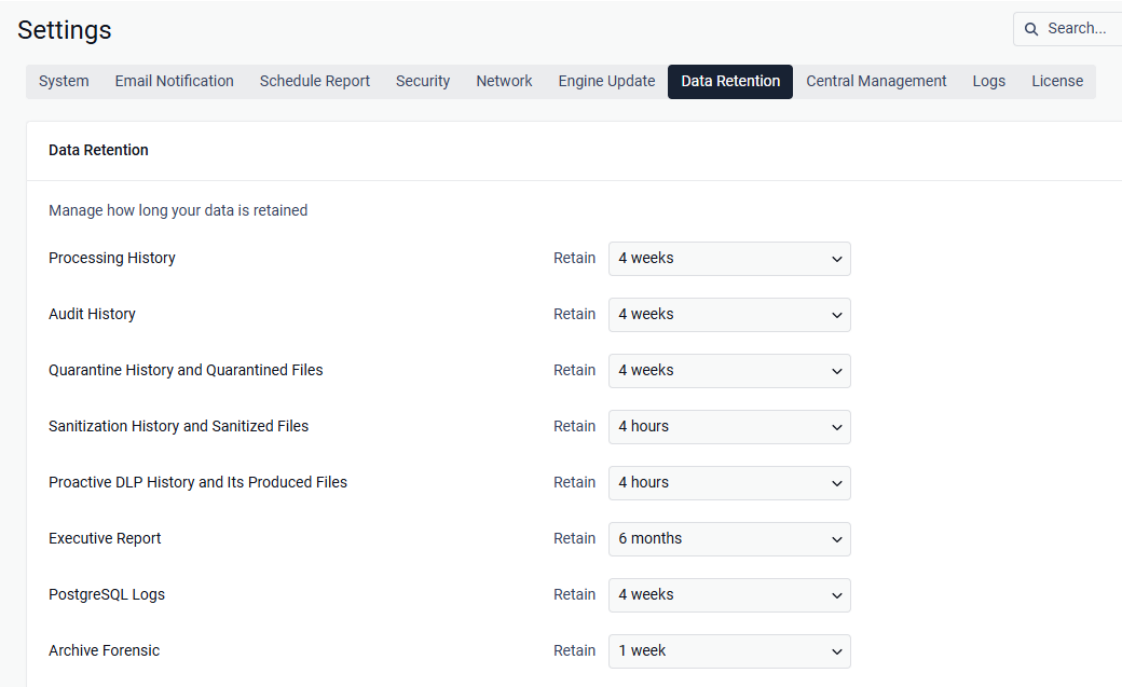


Data retention

Define how long different data types are kept in MetaDefender Core before automatic cleanup.

These retention rules help control storage growth, reduce liability, and keep the system performant while preserving the information needed for auditing and troubleshooting.

Data Retention settings are accessible under **Settings > Data Retention** tab



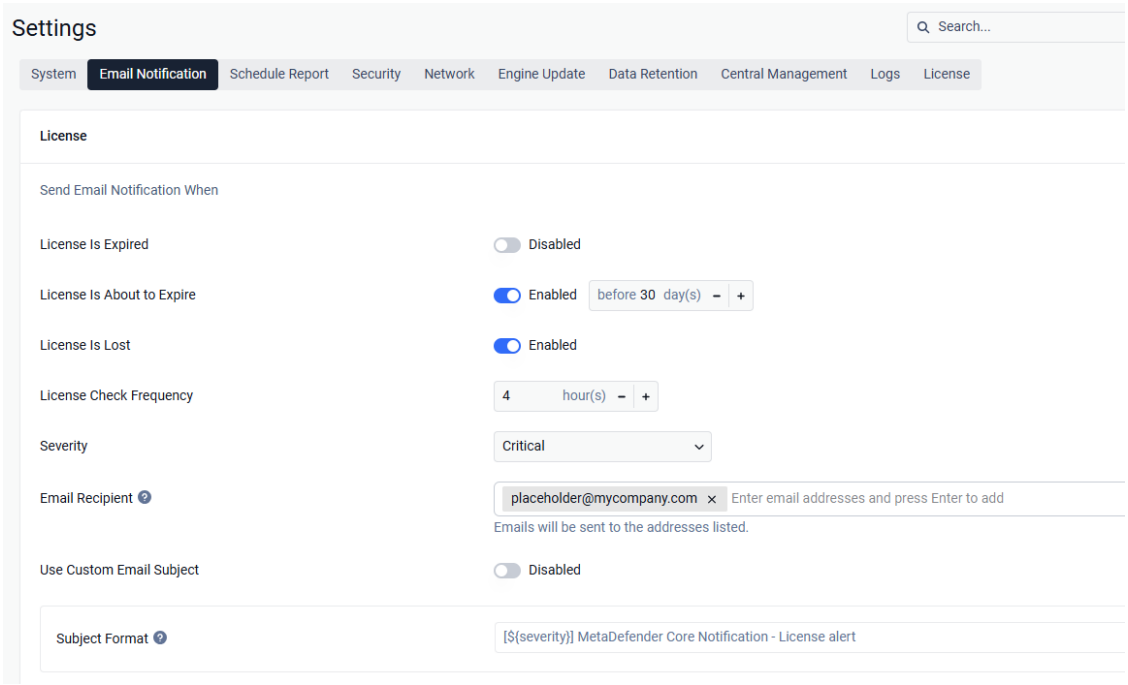
Email notification

Configure alert emails for key system events so administrators are notified before issues impact service. Use the recommended thresholds and frequencies below (license, disk space, engine updates, certificates) to balance timely warnings with minimal noise.

Notification type	Condition / Option	Recommended value
License	License is about to expire	30 days
	License check frequency	4 hours

Notification type	Condition / Option	Recommended value
Disk space	Disk space threshold	80%
Engine Update	Engine remains outdated for	7 days
Certificate	Certificate is about to expire	30 days

Email notification settings are accessible under **Settings > Email Notification** tab.



Hardening External Scanner and Post Action

Apply additional hardening for external scanners and post-action integrations to prevent abuse, credential leakage, or data exfiltration through these components. Follow the security recommendations in the official guide and enable only the scanners, post actions that are strictly required for your workflows.

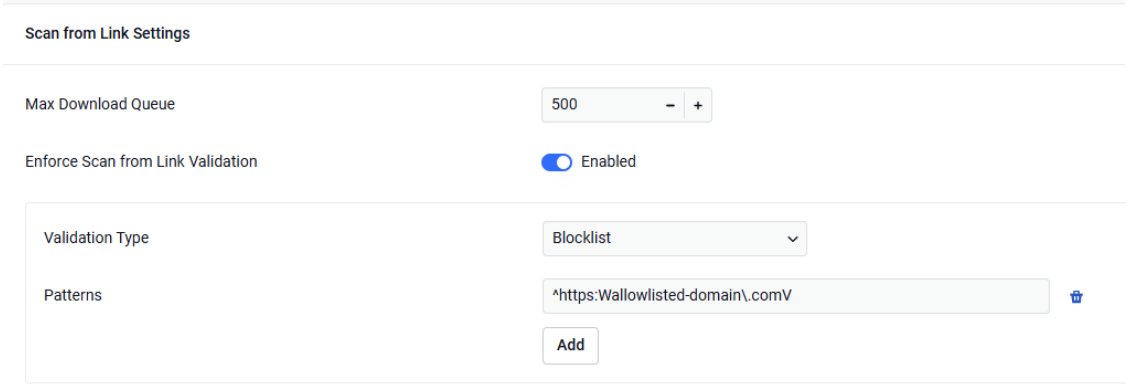
For more details, please check: Hardening External Scanner and Post Action feature.

Enforce scan from link validation: use allowlist over blocklist

When enforcing scan-from-link validation, prefer an allowlist-based approach. Only permit downloads from explicitly trusted domains or URLs and block everything else by default. This reduces the risk of users downloading files from malicious or typo squatted domains that are not yet known or listed as blocked.

For more details, please check: Scan-from-link Settings.

Enforce scan from link validation settings are accessible under **Settings > Security** tab.

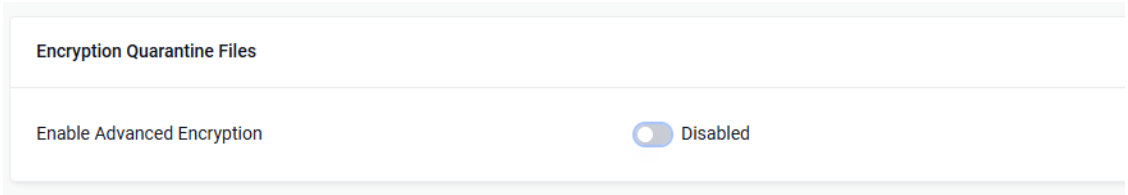


Encryption quarantined files

Enable encryption for files stored in Quarantine so that, even if the underlying storage or backups are accessed by an unauthorized party, the quarantined payloads remain unreadable. Use strong, modern encryption algorithm, protect keys carefully and ensure only strictly necessary roles can download or decrypt quarantined items.

For more details, please check: Encryption for Quarantined Files.

Encryption quarantined files settings are accessible under **Settings > Security** tab.



Request rate limiting

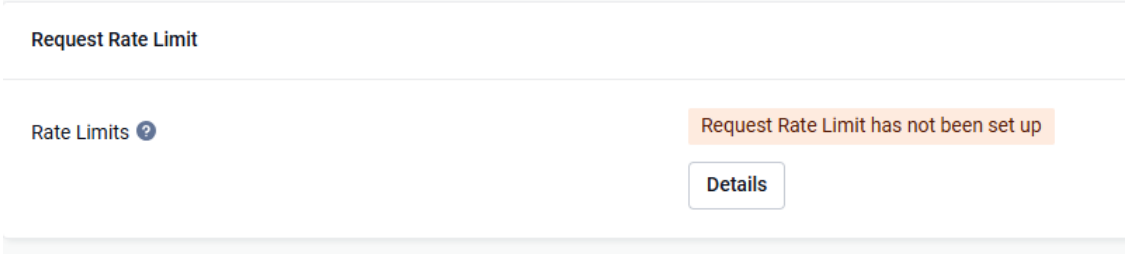
Configure request rate limits for result-fetching APIs (for example, `GET /file/{data_id}` and `GET /file/{batch_id}`) to prevent excessive polling from draining MetaDefender Core resources and degrading scan performance. Without rate limiting, high-frequency GET requests can starve scanning and processing. From MetaDefender Core 5.13.0 onward, you can set per-API limits and use webhooks (`callbackurl`) to avoid constant polling.

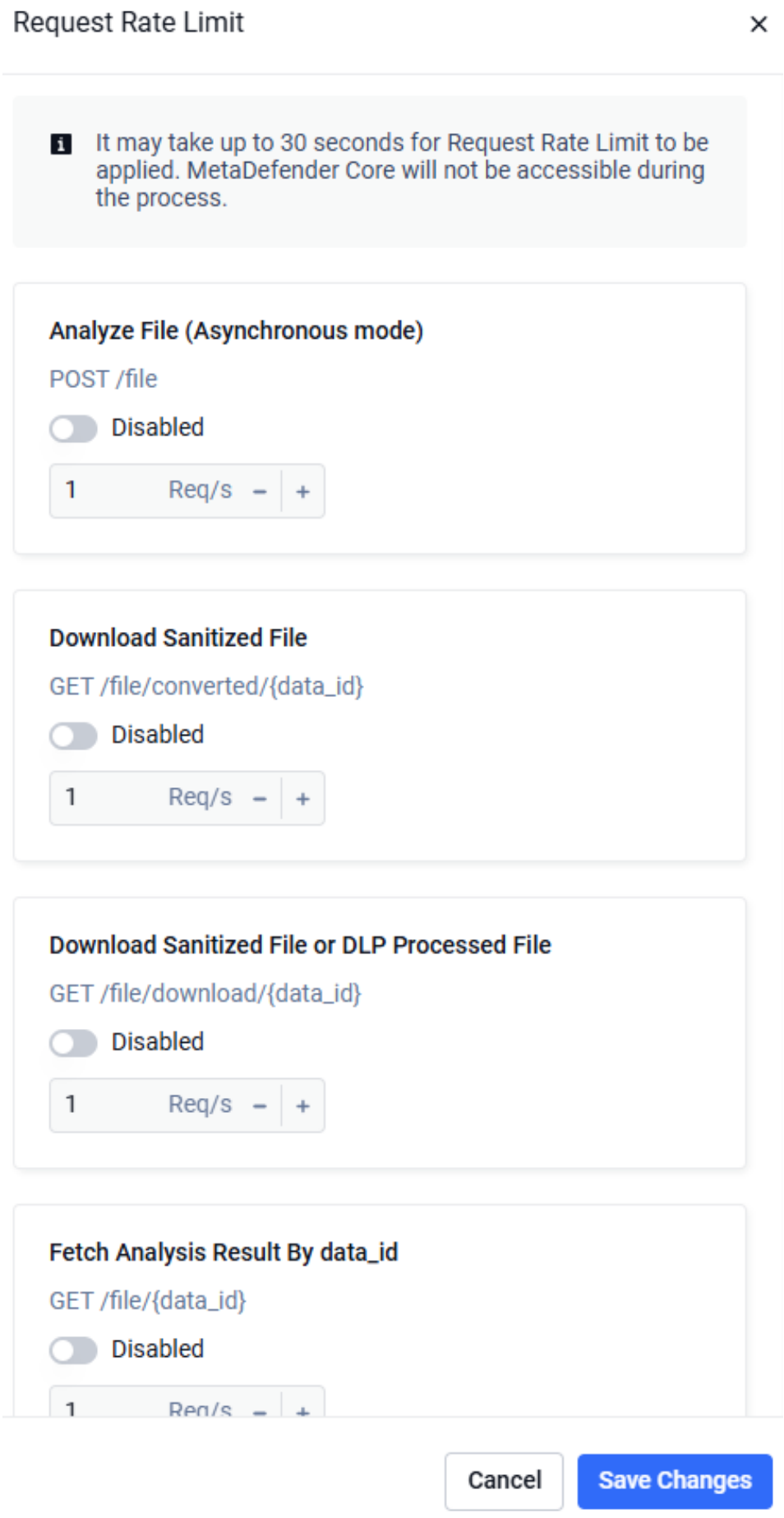
Set reasonable per-second limits for result-fetching APIs to balance system performance and how quickly clients receive results.

Prefer webhook callbacks when throughput and performance are top priority, so MetaDefender Core pushes results when ready instead of clients polling frequently.

For more details, please check: Request Rate Limit.

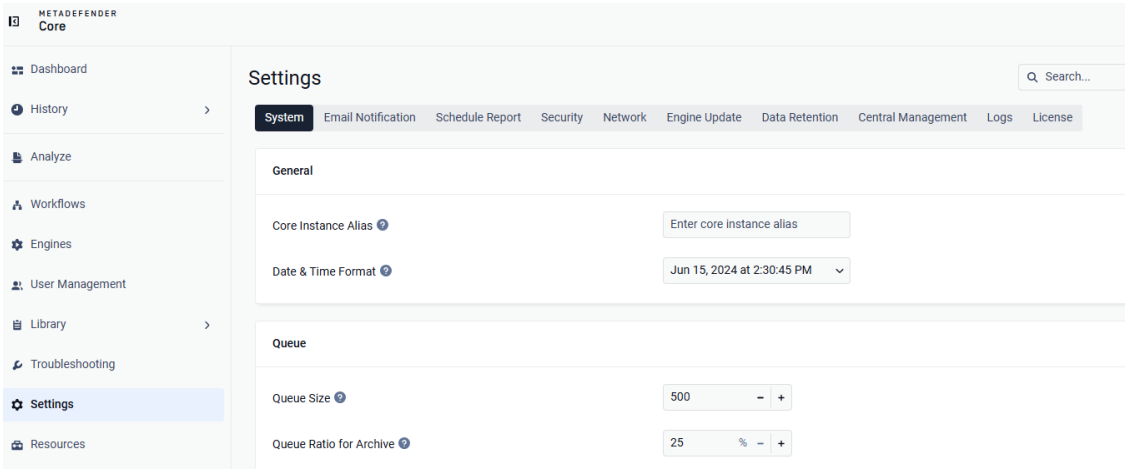
Request Rate Limit settings are accessible under **Settings > Security** tab.




Request Rate Limit
It may take up to 30 seconds for Request Rate Limit to be applied. MetaDefender Core will not be accessible during the process.
Analyze File (Asynchronous mode)
POST /file
Disabled
1 Req/s
Download Sanitized File
GET /file/converted/{data_id}
Disabled
1 Req/s
Download Sanitized File or DLP Processed File
GET /file/download/{data_id}
Disabled
1 Req/s
Fetch Analysis Result By data_id
GET /file/{data_id}
Disabled
Cancel
Save Changes

System Configuration

Date & time format, Queue size, Queue ratio for archive and Login banner are accessible under **Settings > System** after successful login.



Core instance alias

The **instance alias** functions as a nickname to help distinguish between different MetaDefender Core instances, particularly in environments with multiple deployments. **Please note that the instance alias is separate from the instance name and serves a different purpose**.

By default, the alias is an empty string. Users may choose to leave it empty to opt out of including it in email notifications.

The alias must comply with the following requirements:

- Maximum length of 64 characters
- Must not contain trailing spaces
- Must not include Unicode control characters
- Must be unique (it cannot duplicate an existing alias in the database)

> **Info**
>
> The instance alias is not included in export packages and is therefore unaffected by export/import operations or upgrade processes.

Date & time format

There are 2 supported formats

- AM/PM format
- 24 hours format

Queue size

This setting will help to adjust the overall queue size (scan queue) of Metadefender Core service.

> **Warning**
> Setting this configuration too high could affected Metadefender Core itself and other services on the server.

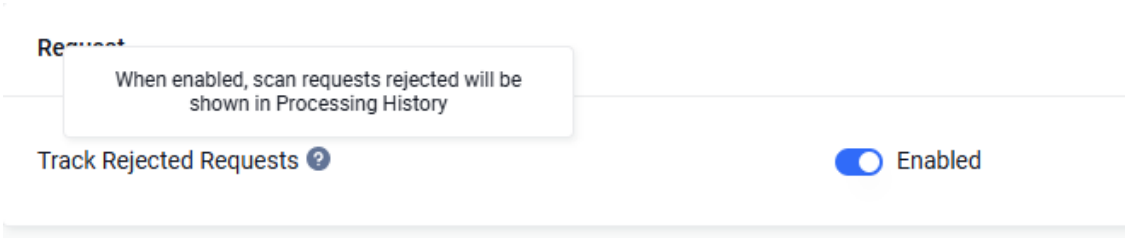
Queue ratio for archive

This setting will help to adjust the percentage of queue size spent on processing archive files. The default value is 25% and can be set to 75% for maximum.

> **Warning**
> The higher the value of queue ratio set for archive, the fewer slots are available for processing new client requests, which may cause Metadefender Core to easily respond with HTTP code 503 and message `Server is too busy. Try again later.`

> **Info**
> During the processing of nested archives, the number of queue slots allocated to extracted files **may temporarily exceed the configured ratio**. This behavior is by design. Enforcing the ratio as a hard limit could prevent child files from acquiring queue slots while the parent archive is holding one and waiting for them to finish, which would result in a deadlock.

Track Rejected Requests



By default, rejected requests (such as those exceeding file size limits or when local scan is disabled) are not stored in the processing history. Administrators can enable the option to store these rejected requests due to the following reasons for future tracing and analysis.

All requests rejected will receive a verdict of "**Rejected**" in the Processing History.

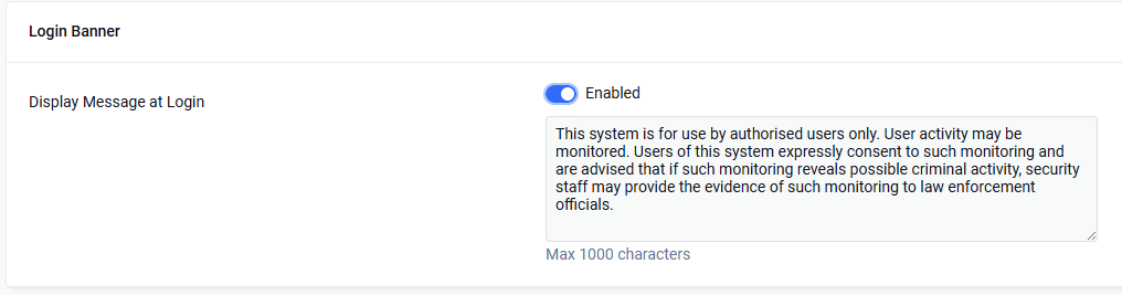
Note: This feature has not supported for multipart uploading yet.

Rejection reasons

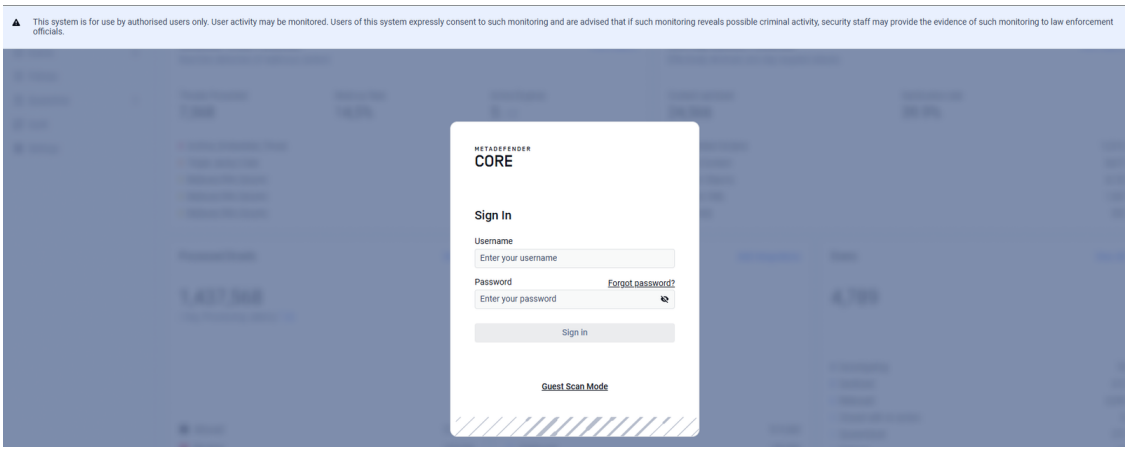
- File Size Limitations: Exceeding maximum file size and maximum download size limits
- Feature Availability: Local scan functionality is disabled
- Workflow Constraints: Workflow rule is not within its active time period
- Engine Readiness:
  - Insufficient mandatory engines available
  - Insufficient active antivirus engines
  - Insufficient up-to-date engines

Login banner

Enable this option to show login banner when user in the login page.



The message will be shown in the end of login page.



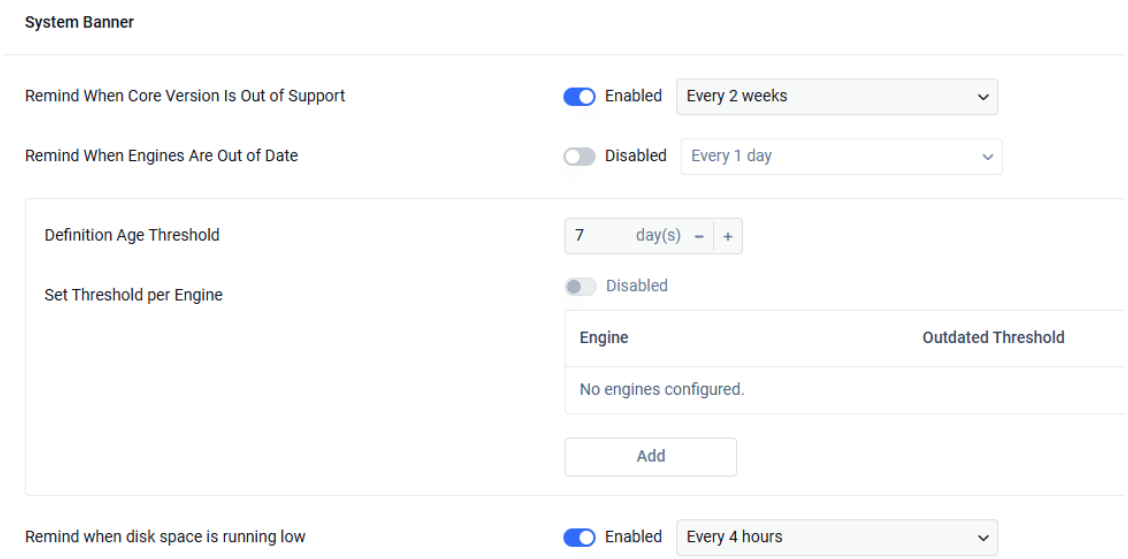
System Banner

Out-of-date notification banner

The system banner notifies users when the MetaDefender Core deployment requires attention. Two notification types are available:

- **Core version end of support** — Notifies users when the installed MetaDefender Core version is approaching or has reached end of support.
- **Out-of-date engines** — Notifies users when scan engines have not been updated within a configured timeframe. A global threshold can be set in days, with the option to override per engine.

Reminder timing is configurable for both notification types.

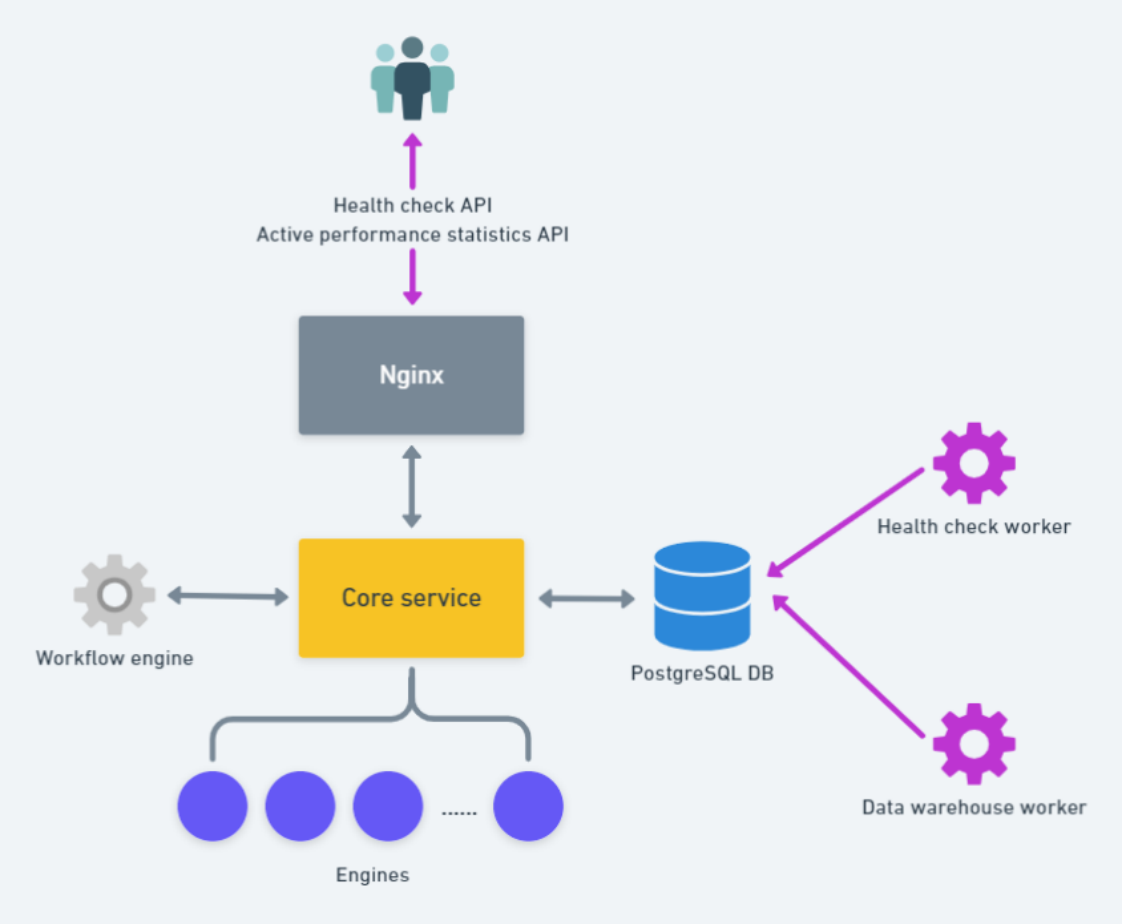


Health Check API Configuration

Product version supported

MetaDefender Core version 5.0.0 or newer

Health check API feature supports you to verify MetaDefender Core application readiness in term of file processing and licensing, before making a decision to submit files and avoid time and resources being wasted predictably.



Using the health check API

Health check API expects to return HTTP(S) 200 Successful only when all pre-configured conditions met.

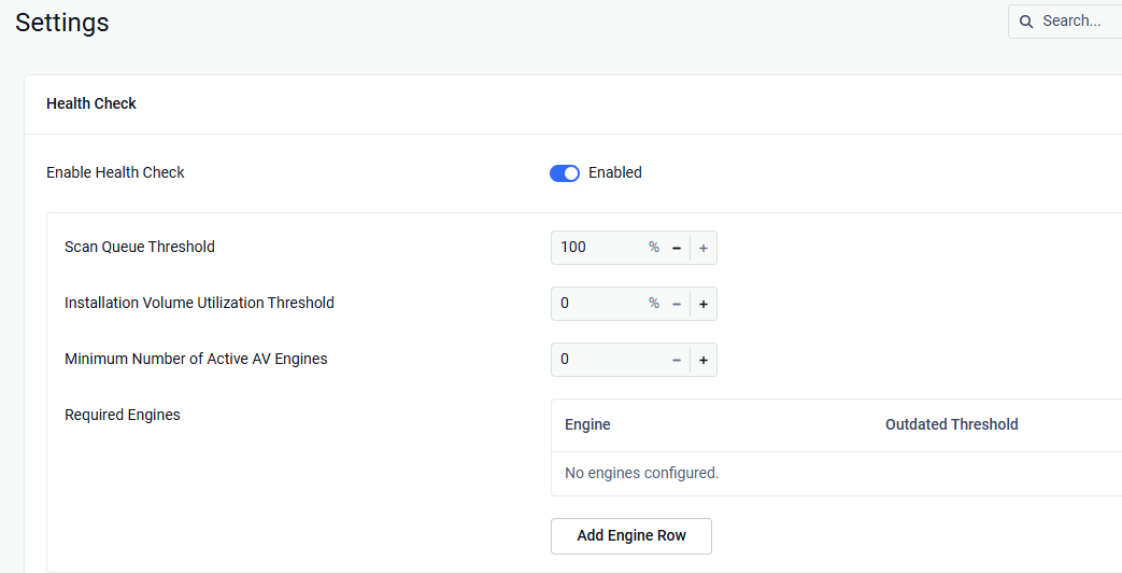
Learn more on how to use the heath check API at: GET - Get health check status

Configure health check policy

You are supported to adjust thresholds on vital criterias as your own policy tailored for heath check verification, via both REST API and MetaDefender Core management console.

Using UI

The relevant settings could be found under "Settings" page



When enabled, expecting all conditions in associated configurations met to determine MetaDefender Core's readiness for file processing.

	Value	Description
Enable health check	On / Off	Enabling this option will allow MetaDefender Core automatically run a background task to collect application operational data for health check logic.
Scan queue threshold	0% - 100%	MetaDefender Core's occupied percentage slots in queue must be equal or smaller than what configured.
Installation volume utilization threshold	0% - 100%	Monitor whether the free disk space on the MetaDefender Core installation drive falls below the configured threshold.
Minimum number of active engines	>= 0	Applicable to AV engines only. Making sure the pre-configured value is not greater than your current actual licensed AV engines.
Required engines		Select which engines are required for the health check to pass and configure their outdated threshold

	Value	Description
		in days.

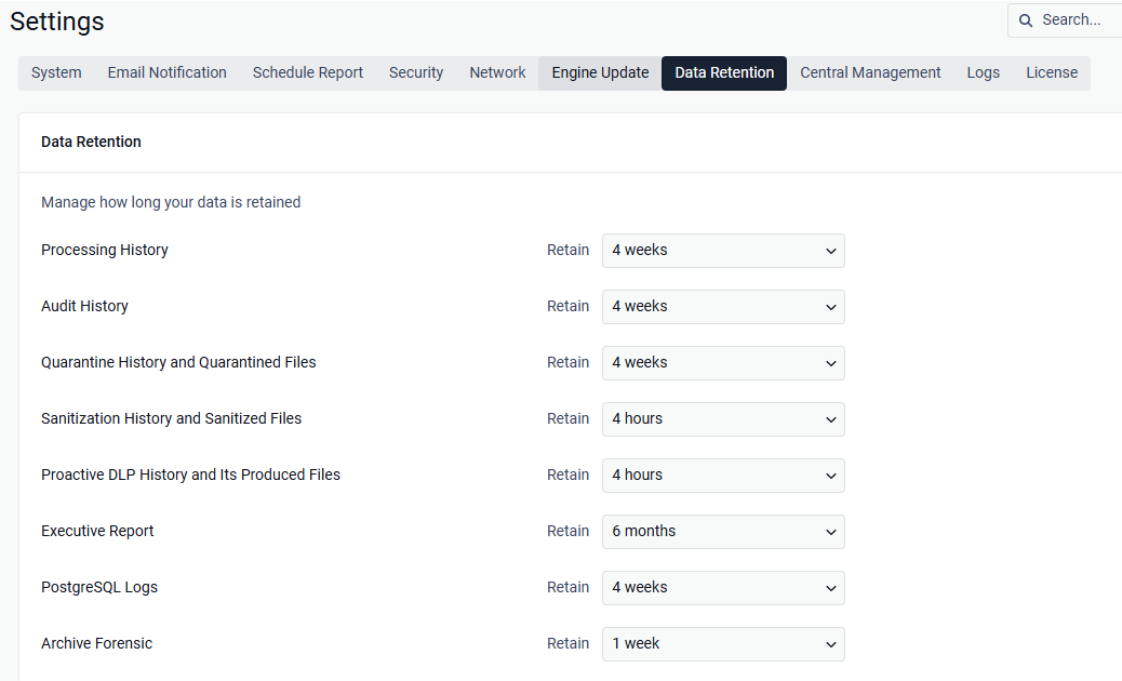
Using administrative REST API

Learn more at: GET - Get health check

Data Retention

Clean up settings are accessible under **Settings > Data retention** after successful login.

> **Info**
>
> Data marked as deleted in the MetaDefender Core database still occupies the data storage, and shall be targeted for the data retention.



Stored scan results, quarantined files, captured archive forensic files, audit log, statistics records and sanitized files that are older than the value set on this page, are permanently deleted from the server. In case you do not want to enable automatic clean up, set the value to off. This will prevent automatic removal of the scan history.

> **Warning**
>
> Setting the clean up value to off can have performance penalty.

Database Management

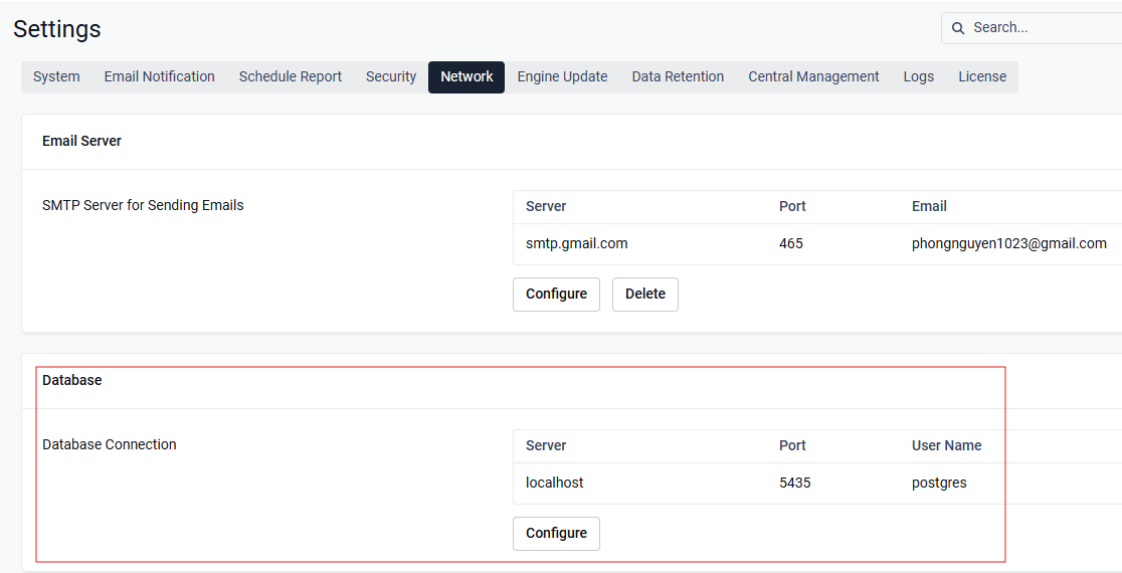
Starting MetaDefender Core 4.20.1, we have supported users to update PostgreSQL server information.

It could be done via either MetaDefender Core management console or separate CLI tool.

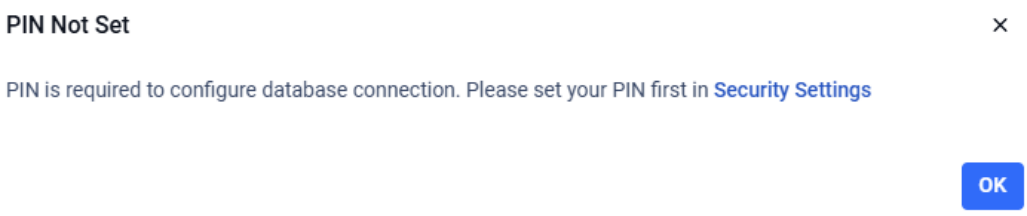
- MetaDefender Core management console
- CLI tool (ometascan-change-db-credentials)

MetaDefender Core management console

Navigate to Settings Network



You are required to setup PIN code first before changing the database information.



Info

- Local (Bult-in) PostgreSQL database setup: Only admin user credentials option displayed on the UI and allowed for modification.
- Remote PostgreSQL database setup: Both admin user credentials and database information (IP address and port) options displayed on the UI and allowed for modification.

CLI tool (ometascan-change-db-credentials)

The tool could be used when you somehow lost the PostgreSQL database connection and could not access to the MetaDefender Core management console UI.

The tool could be located at:

- Windows: <MetaDefender Core installation folder>\ometascan-change-db-credentials.exe
- Linux: /usr/sbin/ometascan-change-db-credentials

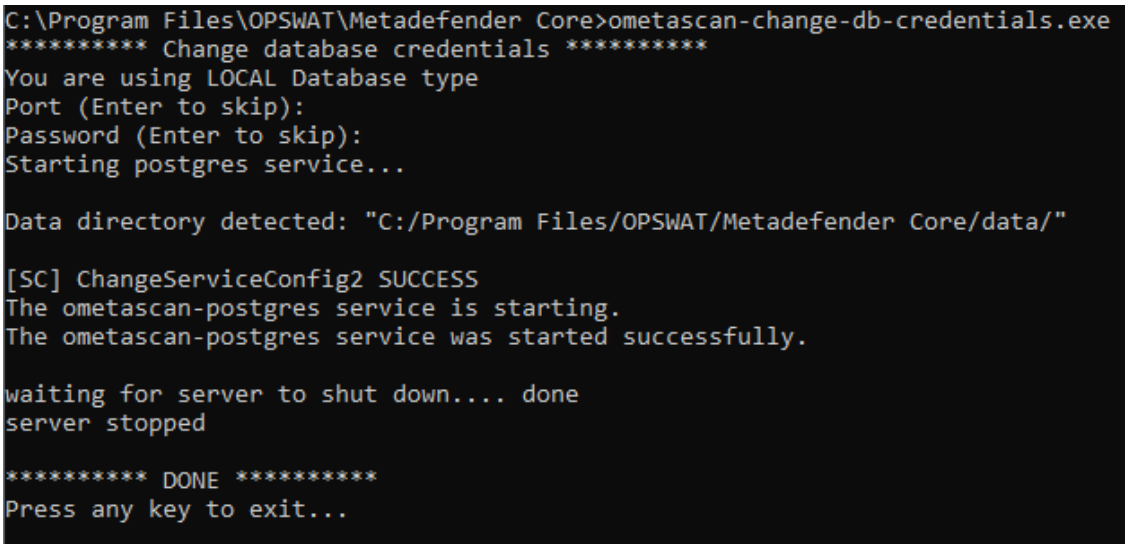
Info

The CLI tool requires admin/root privilege to run.

Make sure to stop following services yourself before running the tool (by doing so, you also confirm all active scanning activities are stopped so this modification will not impact to your production scanning service)

- MetaDefender Core (ometascan)
- MetaDefender Core PostgreSQL (Windows: ometascan-postgresql, Linux: ometascan-pg)

Upon the tool finsihed, you may need to double check to start those services back if not running.



CLI tool parameters:

Code

```
ometascan-change-db-credentials.exe -h

Usage: ometascan-change-db-credentials.exe [options]
MetaDefender Core Change DB Credentials.

Options:
  -?, -h, --help     Displays this help.
  -s, --skppassword  Only change database credentials at Core side, do not
```

```
change user password at PostgreSQL server
```

By default, this CLI tool will help change the PostgreSQL database admin credentials on both MetaDefender Core and PostgreSQL server sides. If you do not want to change the password on PostgreSQL server side, make sure you will use “-s” parameter.

Info

- Local (Bult-in) PostgreSQL database setup: Only admin user credentials option displayed and allowed for modification.
- Remote PostgreSQL database setup: Both admin user credentials and database information (IP address and port) options displayed and allowed for modification.

Encryption for Quarantined Files

Overview

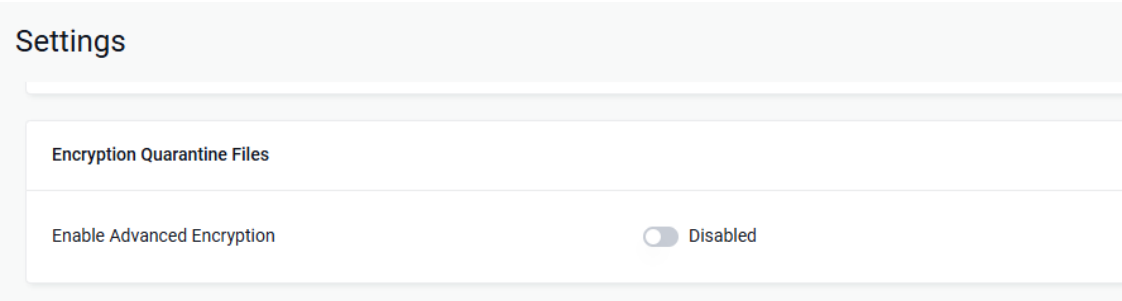
Since 5.14.0 - MetaDefender Core offers an enhanced encryption option for quarantined files, providing a significant security improvement over the standard encryption method.

Benefits of Advanced Encryption for Quarantine

- **Stronger Security**: Advanced encryption provides superior protection for quarantined files
- **Data Integrity**: Ensures quarantined files remain unaltered and protected
- **Industry Standard**: Uses encryption standards recognized across the security industry
- **Performance**: Optimized for efficient operation

Configuration

Please go to Settings -> Security > Encryption quarantine files



Info

- Standard encryption is enabled by default

- Existing quarantined files will remain encrypted with the method used when they were quarantined
- Enabling advanced encryption will only affect newly quarantined files
- There is no performance impact when using advanced encryption

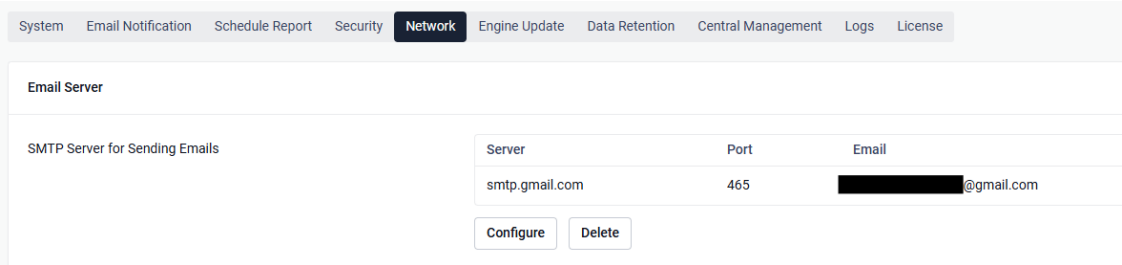
Recommendations

For environments with sensitive data or compliance requirements, we strongly recommend enabling advanced encryption for quarantined files to provide maximum security for potentially malicious content.

Email Configuration

Users with administrator privilege on MetaDefender Core are allowed to setup email configurations for SMTP in order to enable password recovery feature (please check Password Recovery).

SMTP configuration



Server configuration

Properties	Description	Option
SERVER HOST	SMTP server host	IP address / Domain all accepted
SERVER PORT	SMTP server port	Some default common ports: • 25 (TCP) • 465 (SSL) • 587 (TLS)
CONNECTION TYPE	Specify type of connection between MetaDefender Core and SMTP server	• TCP (Unsecured email transmission protocol) • SSL (Secured email transmission protocol) • TLS (Secured email

Properties	Description	Option
		transmission protocol)
AUTH METHOD	Specify the authorization protocol connecting to SMTP server	• AUTH PLAIN • AUTH LOGIN (preferred) • NONE (no authentication)

User authentication

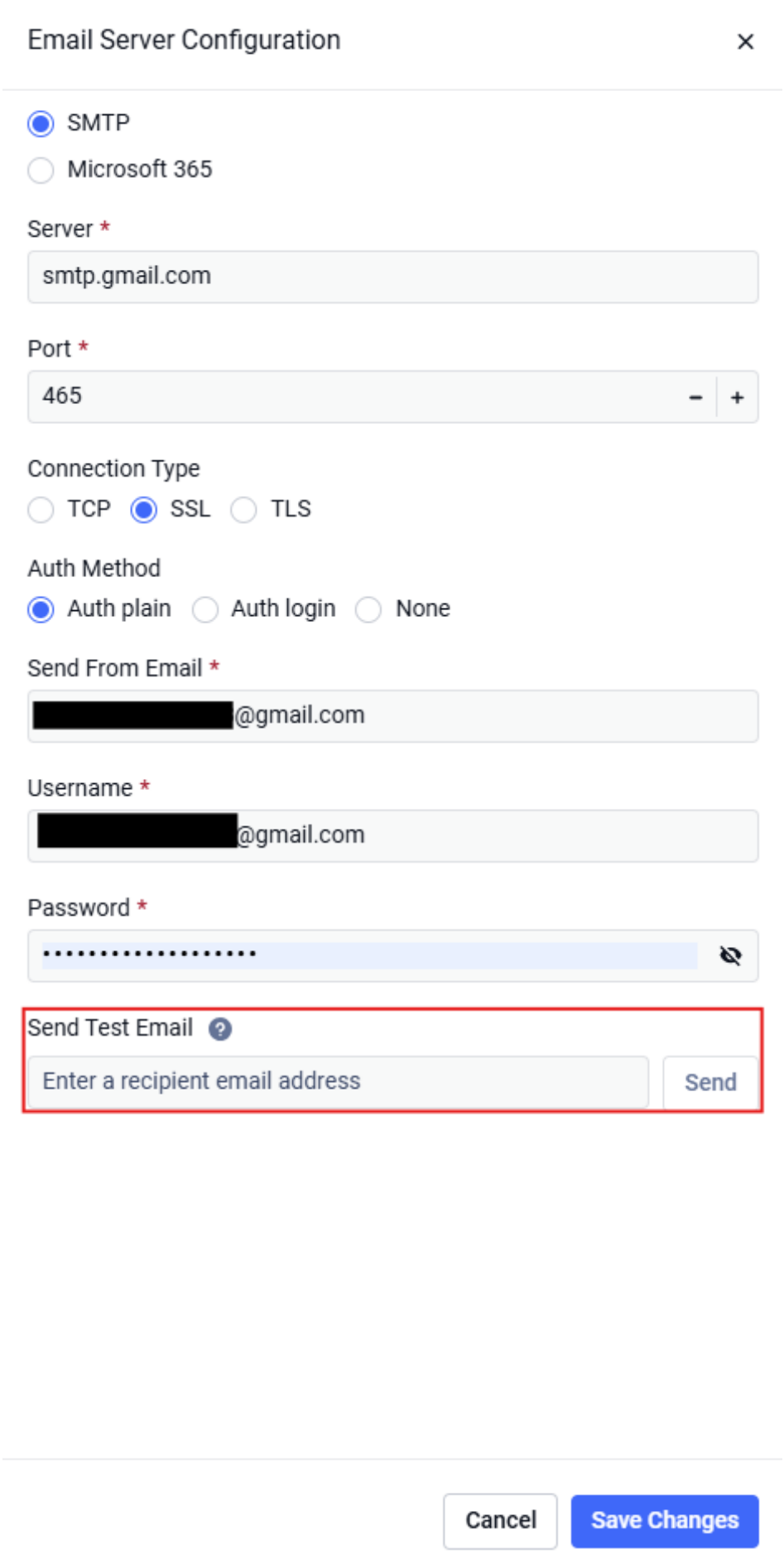
Properties	Description	Constraint
EMAIL	Email address	
USERNAME	Username for SMTP authentication (No required if AUTH METHOD = NONE)	Disallowed characters: [] : | ; = * ? < > / \ "
PASSWORD	Password for SMTP authentication (No required if AUTH METHOD = NONE)	

Please make sure to hit "SAVE SETTINGS" button once configuration is done.

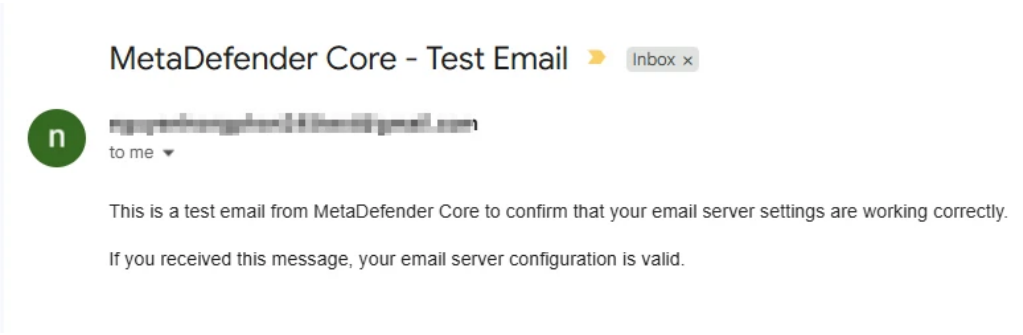
Otherwise "CLEAR SETTINGS" button once hit will clear up all mail settings on the page.

Test Email Server configuration

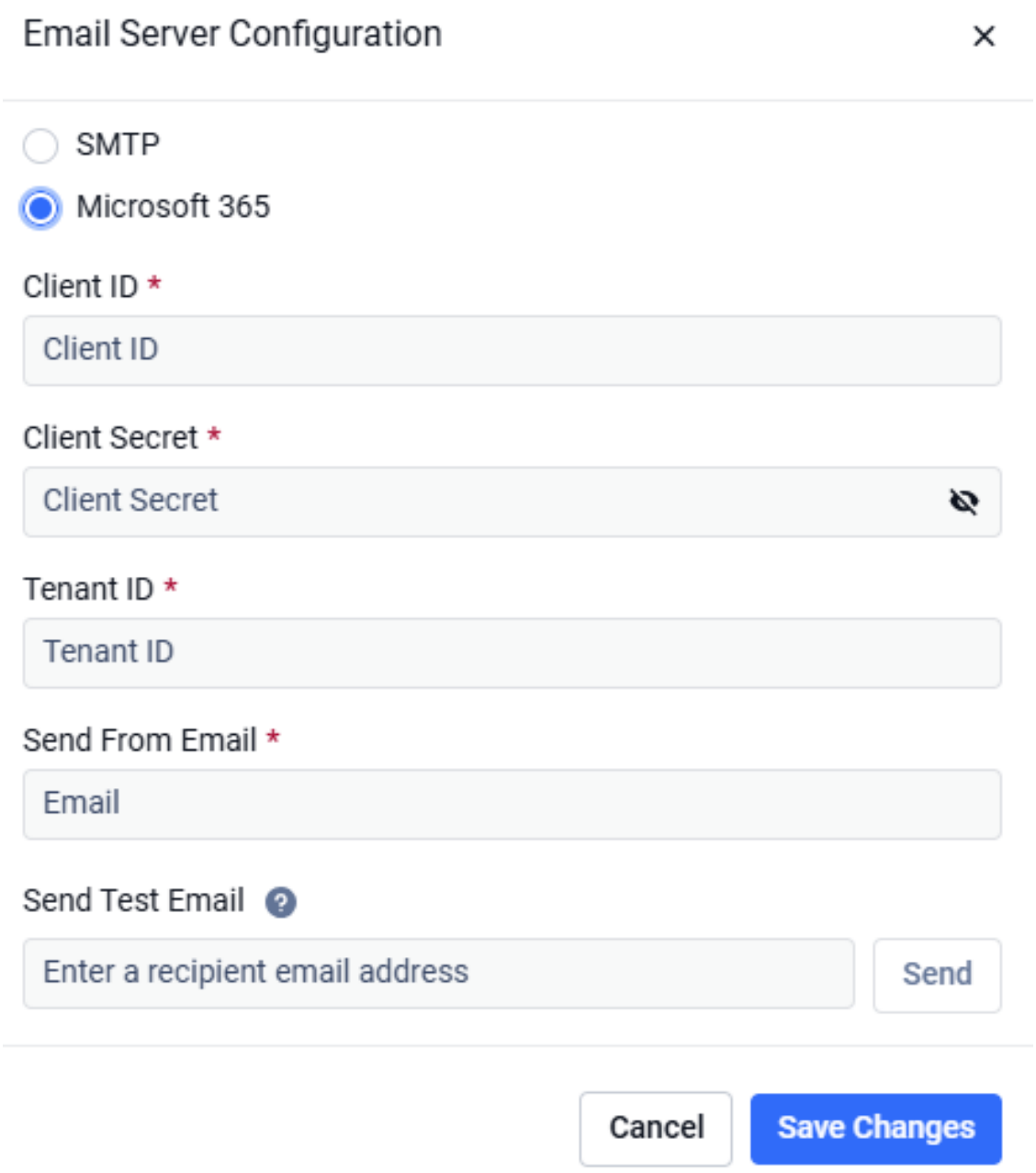
Before saving the email server configuration, the users can test the setting via the feature "Send test email"


Email Server Configuration
SMTP
Microsoft 365
Server *
smtp.gmail.com
Port *
465
Connection Type
TCP
SSL
TLS
Auth Method
Auth plain
Auth login
None
Send From Email *
[redacted]@gmail.com
Username *
[redacted]@gmail.com
Password *
Send Test Email
Enter a recipient email address
Send
Cancel
Save Changes

The test email will look like this



Microsoft 365 OAuth

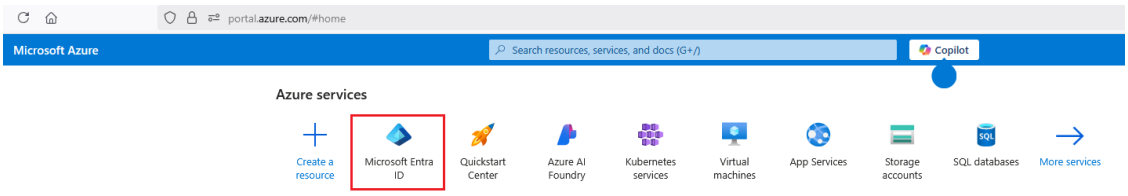


How to get Client Secret and Client ID

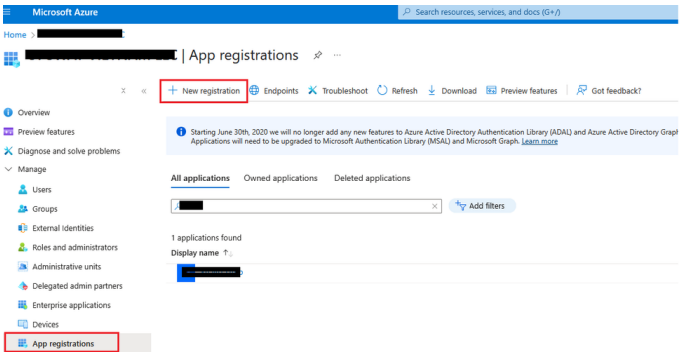
1. **If you already have this setup, please ignore those steps and goto step 7 and step 8**

2. **Log in to: Microsoft Azure**

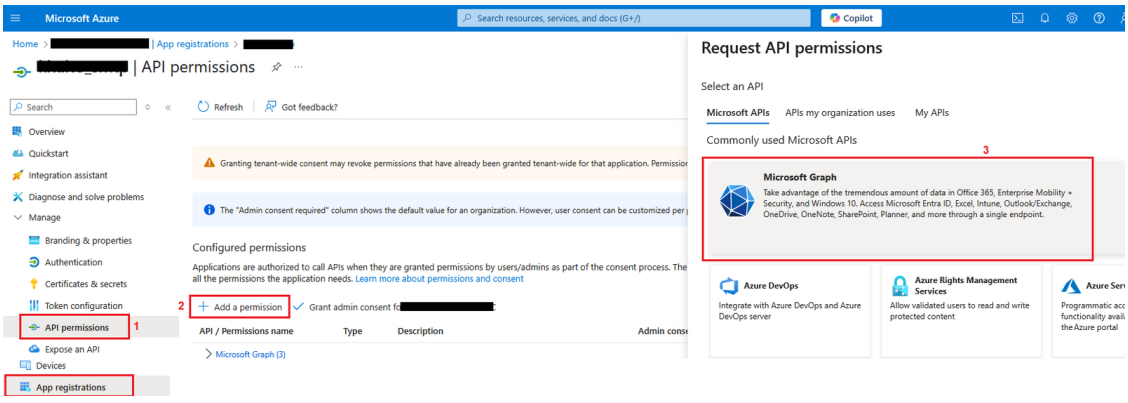
3. **Select Microsoft Azure Entra ID**



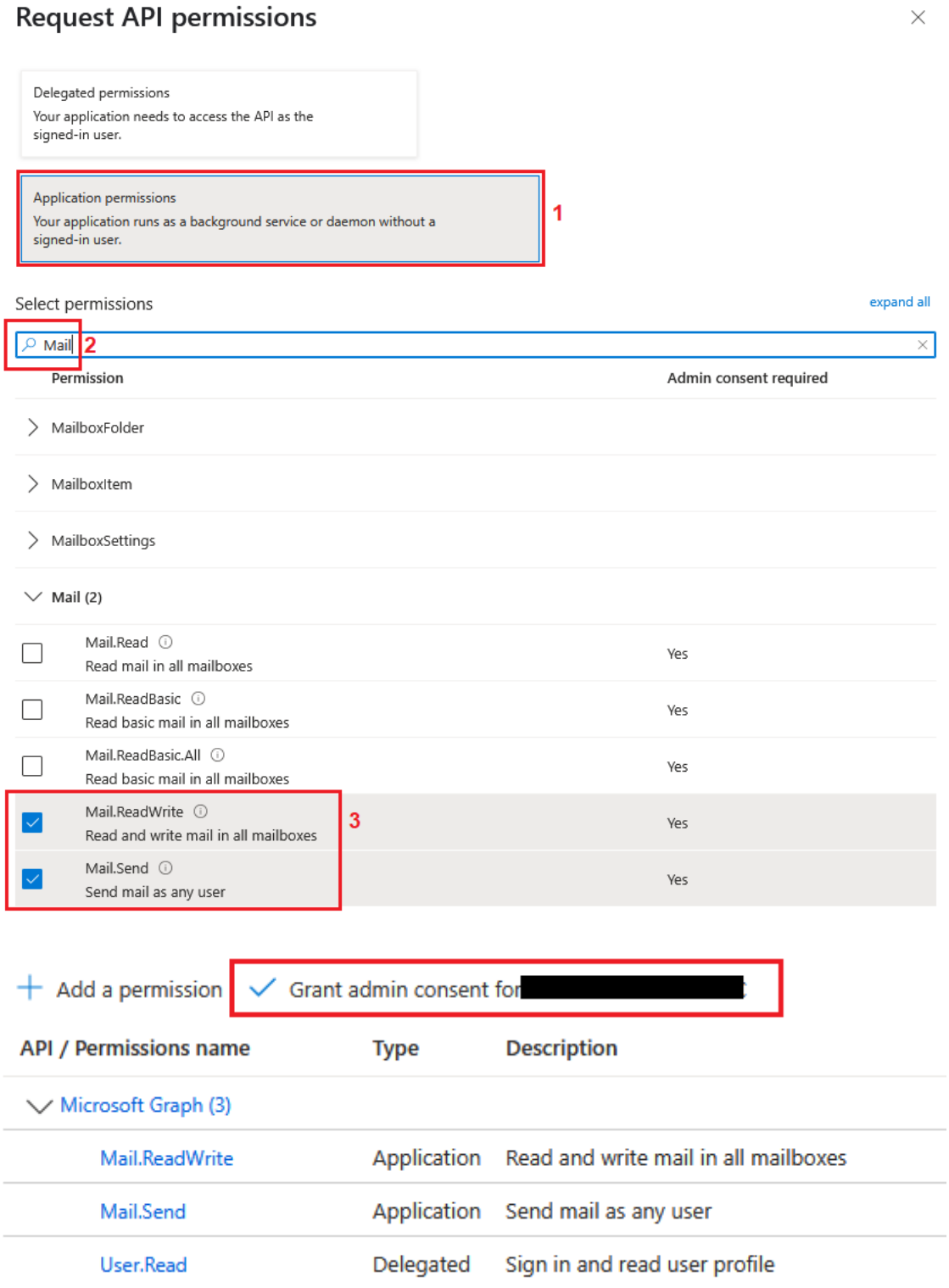
4. **Register a new app**



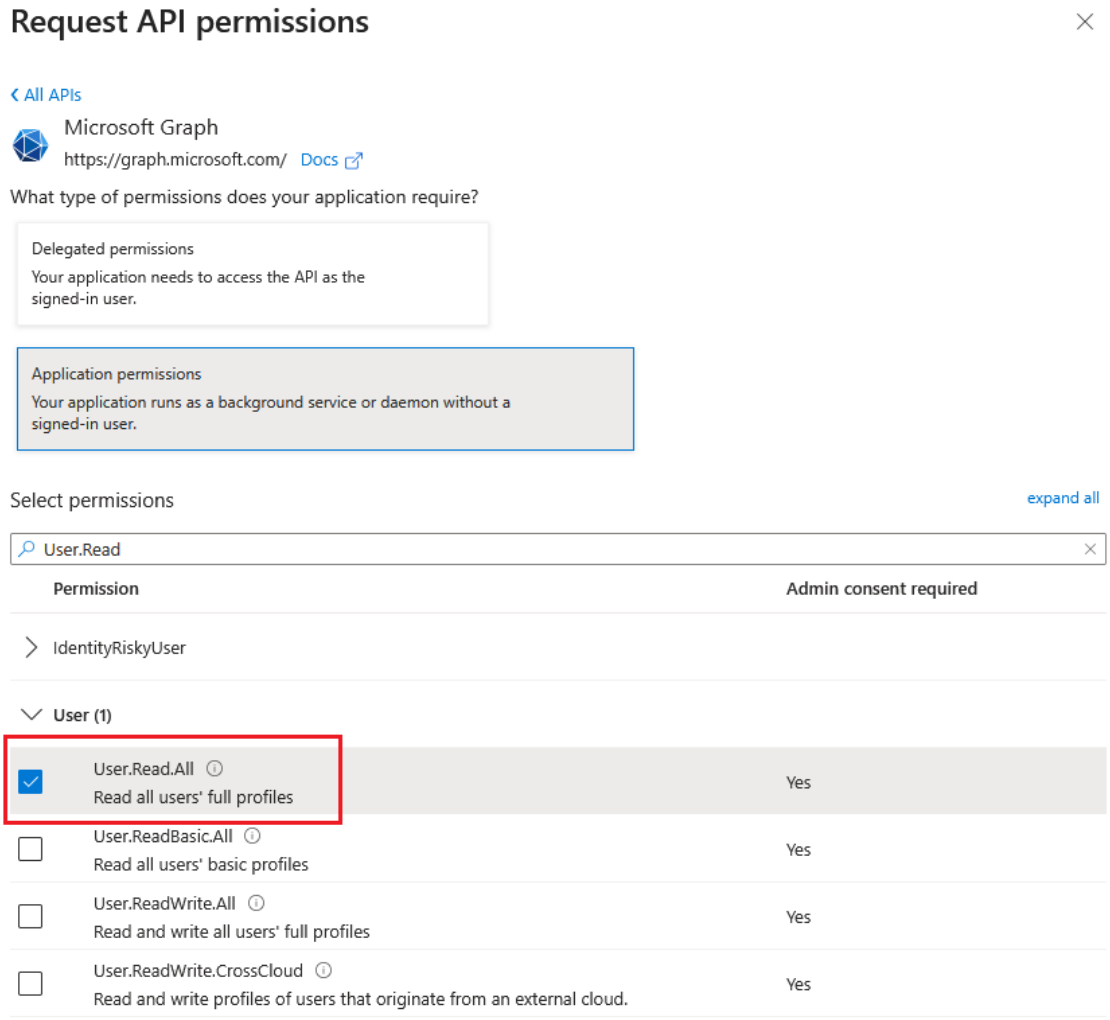
5. **Add Graph APIs**



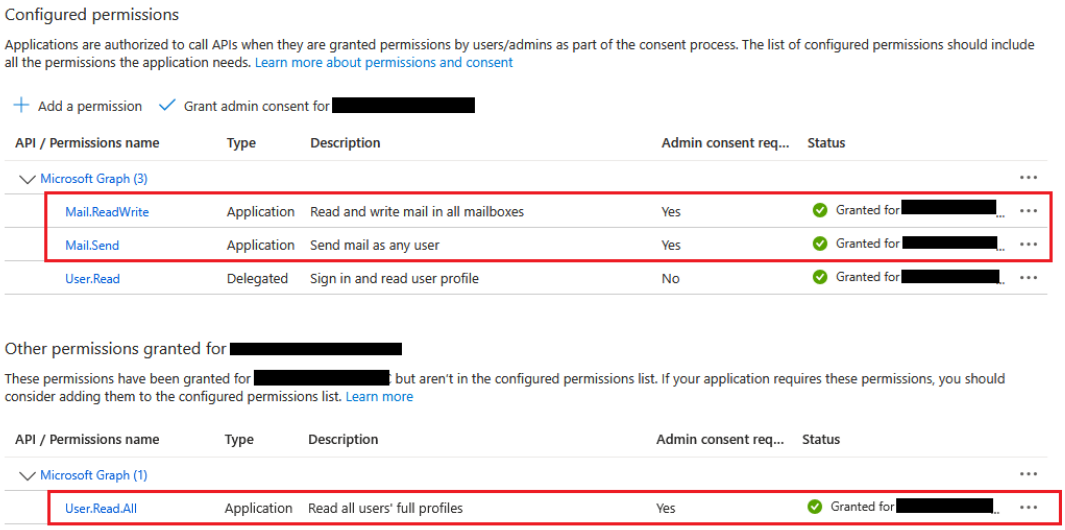
6. **Give correct permissions for Mail: Mail.ReadWrite and Mail.Send**



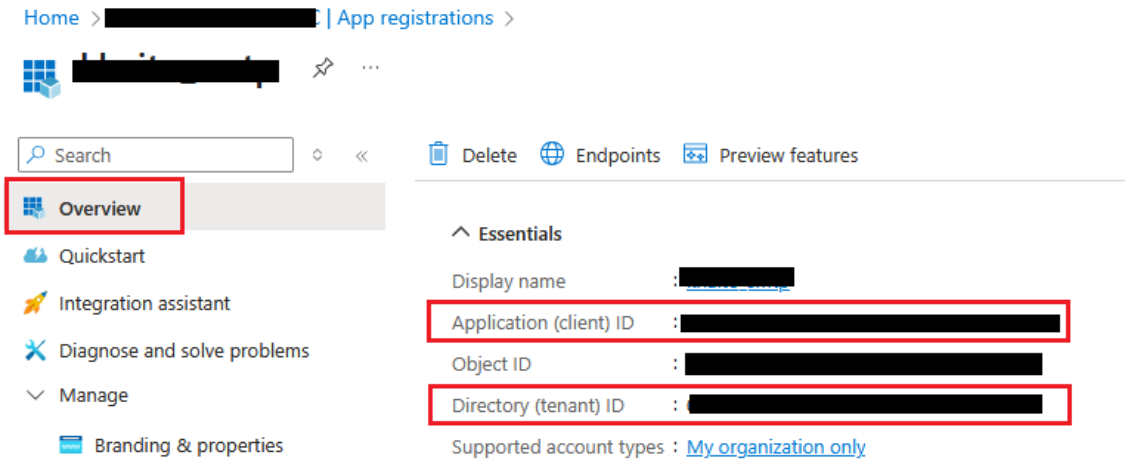
7. **Give correct permissions for User: User.Read.All**



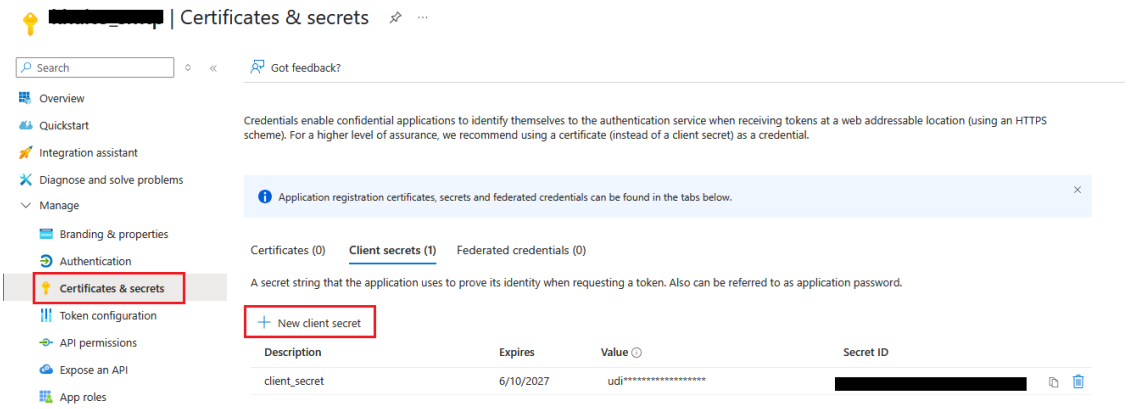
8. Expected result



9. Copy client id and tenant id



10. **Generate client secret (if it does not exist)**



Email Notification

Administrator can configure Email Notification settings to receive notifications for detected threats and incidents, providing real-time updates on potential security risks.

- **Soon-to-Expire Licensing:** Helps avoid service interruptions by facilitating timely license renewals.
- **Disk Space:** Helps avoid service interruptions by the system running out of disk space
- **Failed-to-Update Engine:** Notifies on “Download Failed”, “Permanently Failed”, Temporary Failed”, and “Content Invalid".
- **Outdated engines**: Notifies when there is any engine that not updated (update new database version) in a period of time.
- **Pre-defined Scan Outcomes:** Controls which scan results trigger an email notification. Choose one of two filtering modes:
  - **By Scan Results** — Notify based on *what was detected* (infections, suspicious activity, sensitive data, etc.).
  - **By Status** — Notify based on *what happened to the file* (blocked or allowed). Useful when you care about enforcement outcomes rather than specific findings.
- **Quarantined Files:** Alerts users about files that need review, assisting in managing false positives and potentially

malicious files.

- **Failed Scans:** Monitors the scanning process, prompting rechecks of failed scans to maintain continuity.
- **System monitor**: Alerts users about CPU and RAM usage.
- **Database monitor:** Alerts users about current status of Database, notify for users for running vacuum process if needed
  - Please read more on this article to understand about database dead tuples: Database dead-tuple alert: what it means and how to tune PostgreSQL autovacuum

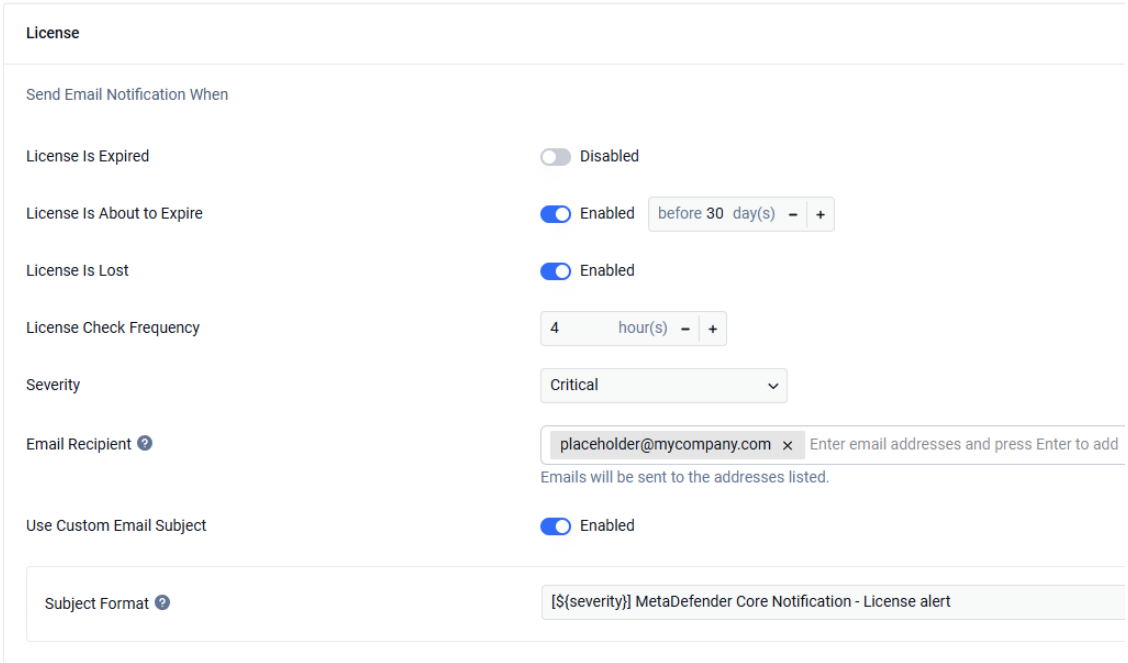
Including Instance Alias in Email Notifications

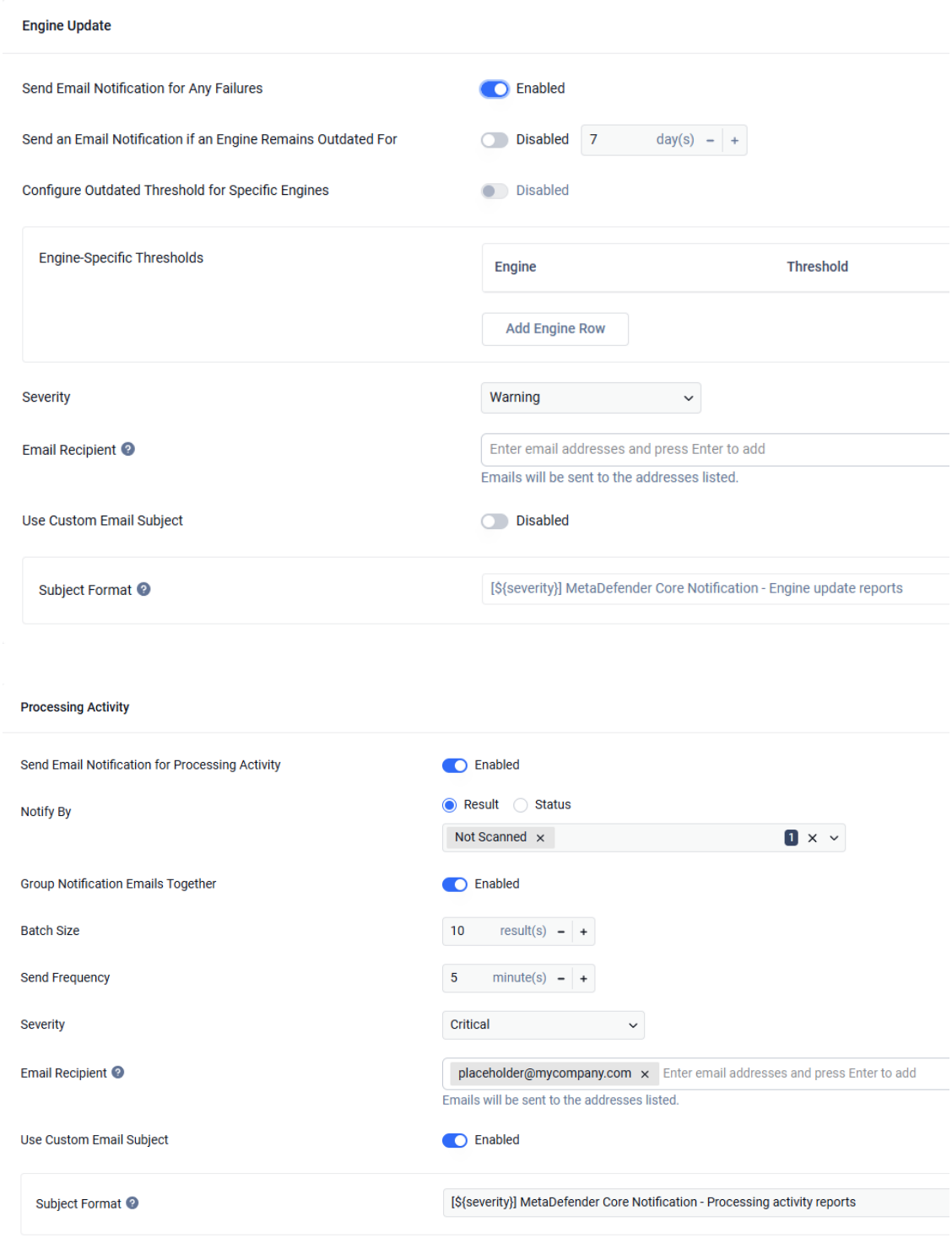
Starting from version 5.18.0, MetaDefender Core allows users to configure an instance alias to differentiate between multiple instances. When configured, the instance alias will be included in the body of email notifications. This setting can be managed in the General Configuration section: General Configuration

Custom Email Subject Format

For each section, user can define custom subject format. MetaDefender Core provide 3 pre-define macro

- ${severity} : Severity
- ${deployment_id} : Deployment ID
- ${instance_name}: identifier of the instance, configuration when fresh install, if not, it will be automatically generated. This macro **only available for shared database mode**





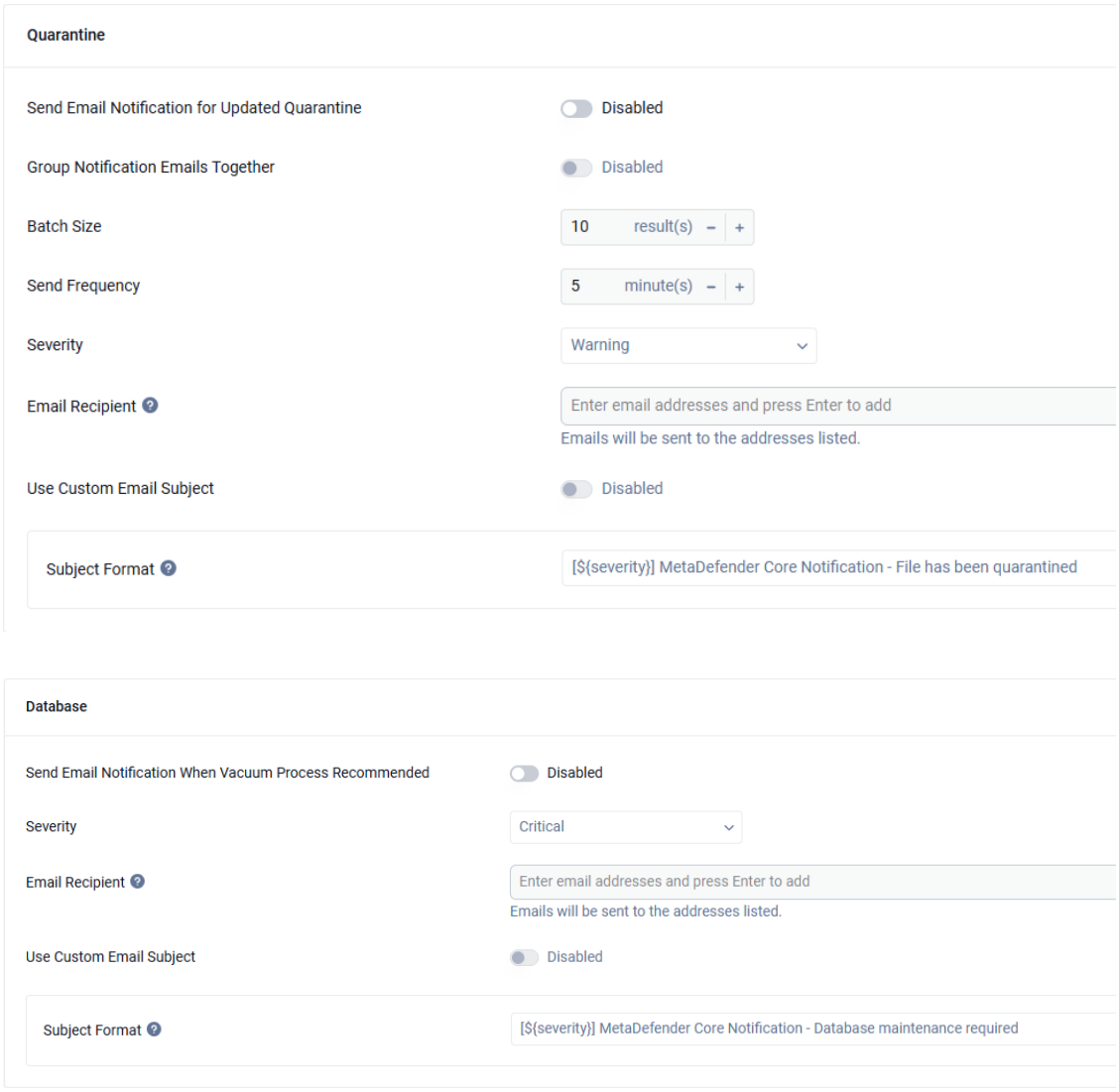
Group Notification Emails

This feature consolidates notification emails to reduce noise while ensuring important alerts are delivered promptly. It is controlled by two settings:

- **Batch Size** Defines the maximum number of requests included in a single notification email. For example, if the batch size is set to **3**, MetaDefender Core will send one email containing the results of up to three requests with the status **"Not Scanned."**
- **Send Frequency** Specifies the time interval (in minutes) at which notification emails are sent, if any

qualifying events occur. For example, if the send frequency is set to **1 minute** and only one **"Not Scanned"** request occurs within that minute, MetaDefender Core will still send a notification email, even if the batch size has not been reached.

Together, these thresholds ensure that notifications are sent in batches to minimize email spamming, while also guaranteeing timely delivery so that no important notifications are missed.



Custom email body template

The users can also custom their own email body template, please check details: Email Template

Email Template

Users can define a custom email template for each section and all outgoing emails from that section will use this layout.

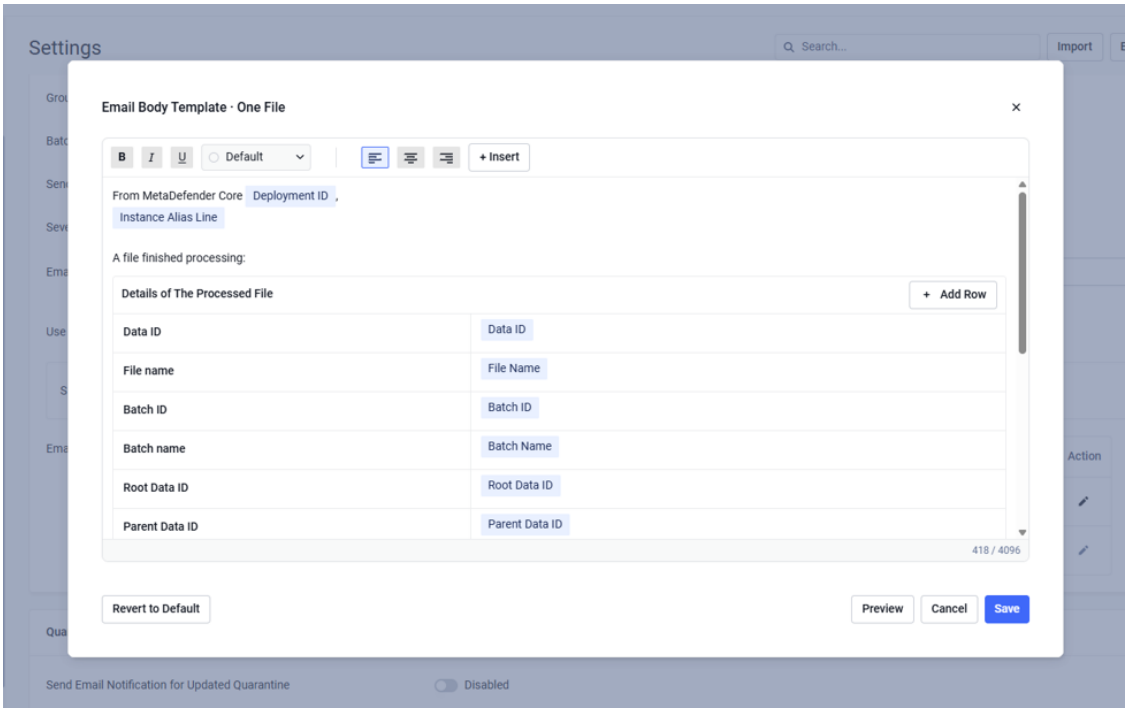
Supported formatting options include:

- Bold, italic and underlined text

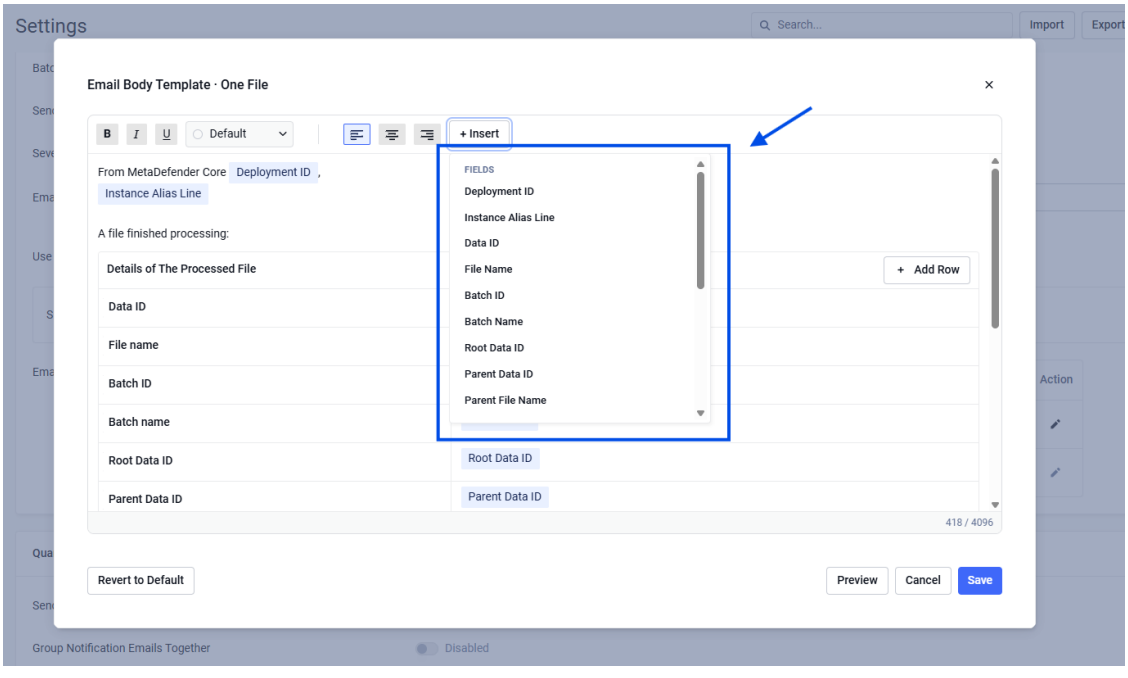
- Text color
- Text alignment
- Tables and metrics

Example

Email template for Processing Activity



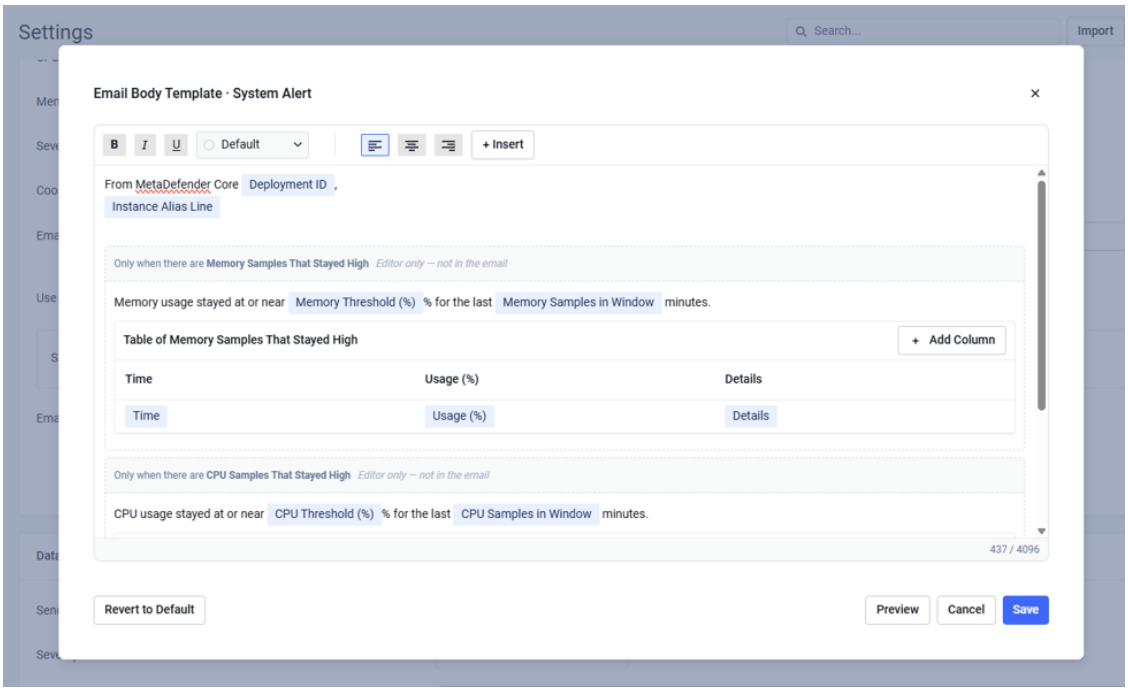
- User can insert more tables or metrics by clicking **Insert -> Metrics/Table**



- After define template, user can do quick preview their define by clicking **Preview** button



Email template for System alert



- Also click **Preview** button to view the defined template



Enabling HTTPS

MetaDefender Core supports accessing Web UI and REST interface via HTTPS. This feature is not enabled by default.

There are two ways to enable the feature:

- via Management Console or
- modifying MetaDefender Core server configuration via configuration files.

Info

TLS v1.2 and v1.3 are supported HTTPS connection.

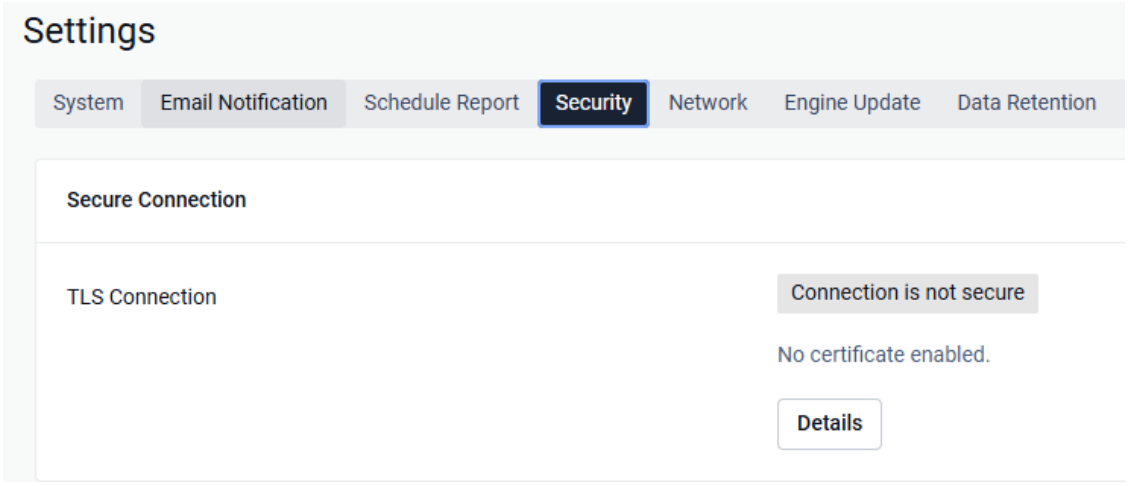
Warning

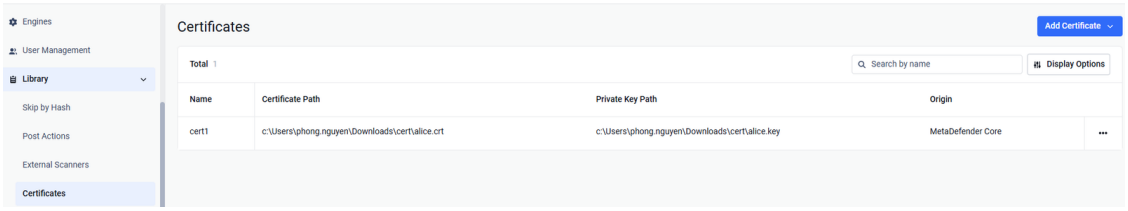
If HTTPS is configured via both: Web Management console and configuration file, only the settings made on Management Console will take effect. It is highly recommended not to use both configuration files and Management Console for HTTPS settings at the same time.

Enabling HTTPS via Management Console

1. Go to **Settings** **Security** page

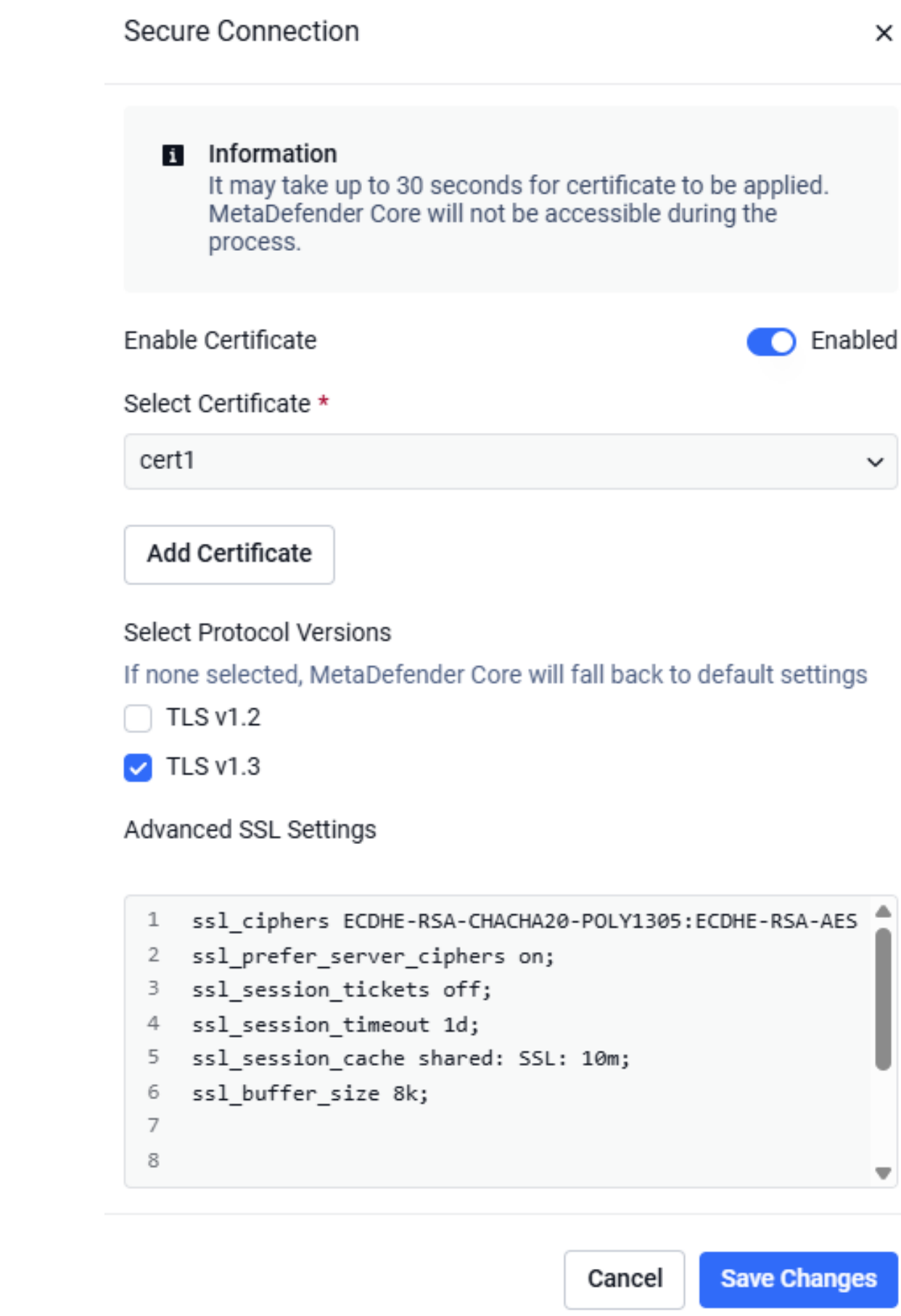
2. If there's no certificate-key pair added to the Library, please go to **Library** Certificates add one that is desired to use for securing HTTP connections.





3. Check ***Enable certificate*** *connection* button and choose a certificate-key pair.
4. Users can define advanced SSL settings in the "**Advanced SSL Settings**" section. These settings follow Nginx syntax.

4. As clicking on *Save changes*, you will be warned that Management Console is going to be restarted and this will take some time.
5. Approximately 30 seconds after confirming saving of configuration the Management Console will be reloaded via HTTPS.


Secure Connection
Information
It may take up to 30 seconds for certificate to be applied. MetaDefender Core will not be accessible during the process.
Enable Certificate
Enabled
Select Certificate *
cert1
Add Certificate
Select Protocol Versions
If none selected, MetaDefender Core will fall back to default settings
TLS v1.2
TLS v1.3
Advanced SSL Settings
1 ssl_ciphers ECDHE-RSA-CHACHA20-POLY1305:ECDHE-RSA-AES
2 ssl_prefer_server_ciphers on;
3 ssl_session_tickets off;
4 ssl_session_timeout 1d;
5 ssl_session_cache shared: SSL: 10m;
6 ssl_buffer_size 8k;
7
8
Cancel
Save Changes

Enabling HTTPS via configuration files

> **Info**
>
> Starting from version **5.14.0**, advanced SSL settings can be configured directly through the UI. We highly recommend using the UI for SSL configuration to ensure better management and ease of use.

For config HTTPS via configuration file, please use this instruction: SSL Configurations

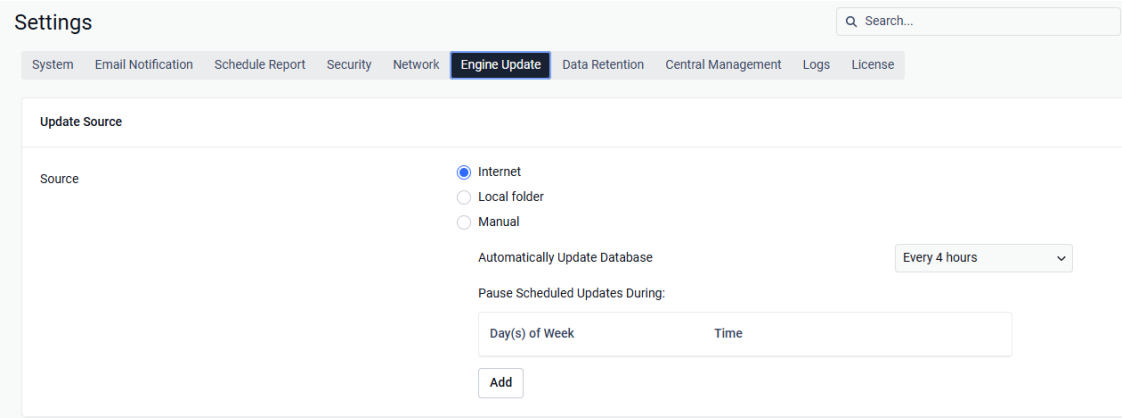
Engines Update Configuration

Update settings are accessible under **Settings > Engine Update** tab .

On this page the update mechanism can be chosen between three different methods

- Internet: automatic update downloading from the internet.
- Local folder: searching for updates in a specific folder.
- Manual: disable auto update

Internet update



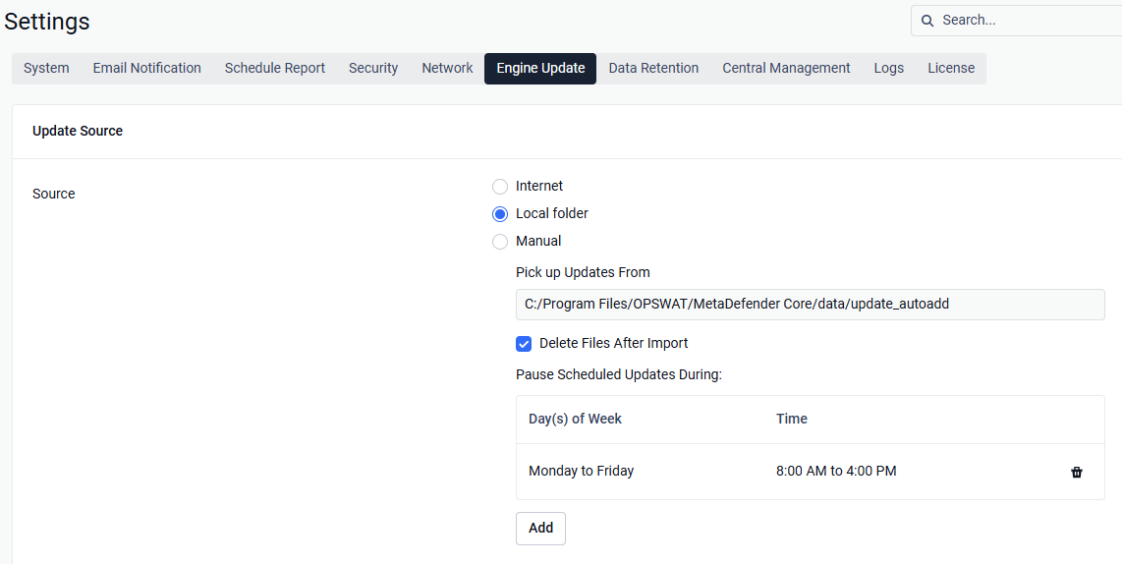
Choosing the Internet method means the product will typically perform automatic update downloads from the internet. In cases where the Core is managed by My OPSWAT / Central Management, the product will perform automatic update downloads from My OPSWAT / Central Management.

To set the frequency of these updates choose the corresponding value presented on the **Automatic database updates** scrollbar.

Setting the interval to off, means the update will only occur, when the **Update all** button is clicked on the engines page under Engines

With the **Updates are not applied during** field it is configurable when NOT to distribute update packages to scan nodes.

Local folder update



Choosing the **Folder** method will make the product searching for updates in a specific folder set in the **Pick up updates from** option.

> **Warning**
>
> The pick folder you select must grant read and write permissions to the MetaDefender Core process; without that access, the product cannot run as intended.

> **Pick up-updates-from setting**
>
> Network mapped drive and UNC path is also accepted.
>
> On Windows, make sure to grant MetaDefender Core (run as SYSTEM account) to access the location and execute files within where you map the network drive.

The product watches the folder for modification, whenever the content is modified it will try to pick up the files placed under the folder.

Another option of this method is **Delete files after import**, which means product will delete files after they were processed successfully. This means even if an update could not be applied, it will be removed because it was processed without any issue.

With the **Pause scheduled updates during** field, it should be defined to indicate when you do NOT want MetaDefender Core to perform the engine update activity.

Manual update

Choosing the **Manual** option will turn off any automatic update mechanism stated above and only accepts updates on the engines page under Engines

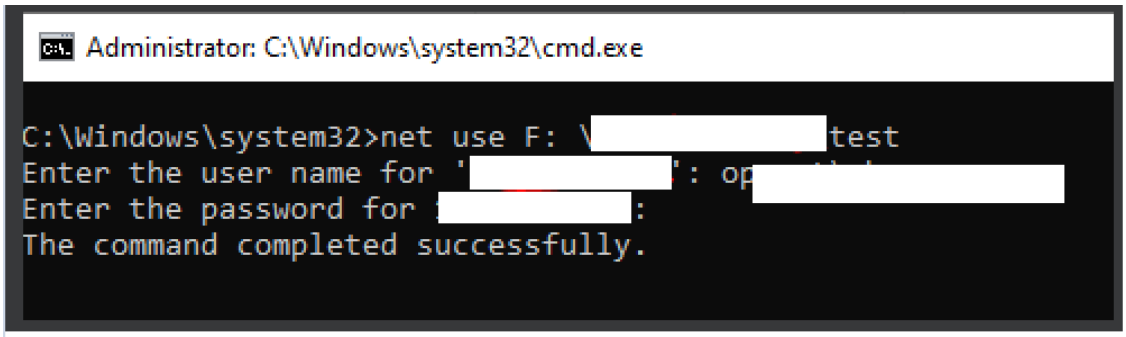
Advanced configuration for UNC path

Info

- On Windows, make sure the shared folder have permission for SYSTEM account

There are 2 options for input UNC path to pickup folder:

- Input directly UNC path to the configuration, for example: \ABC-D1111\test or \192.168.0.1\test
  - But to use this format, MetaDefender Core server and shared folder's machine need to be on the same domain
- Map network drive to local drive, here is the step by step:
  - Us PSTools to map network drive with local drive for SYSTEM account, for example we will map \192.168.0.1\test with F drive:
    - Download PSTools and extract it
    - Open CMD in Administrator
    - PsExec.exe -s net use F: \192.168.0.1\test



- After that, add F:\ in pickup folder

Info

Since the drive is mapped using the SYSTEM account, the mapping is removed upon restarting the machine. Therefore, it is necessary to re-run the command to remap the drive after each restart.

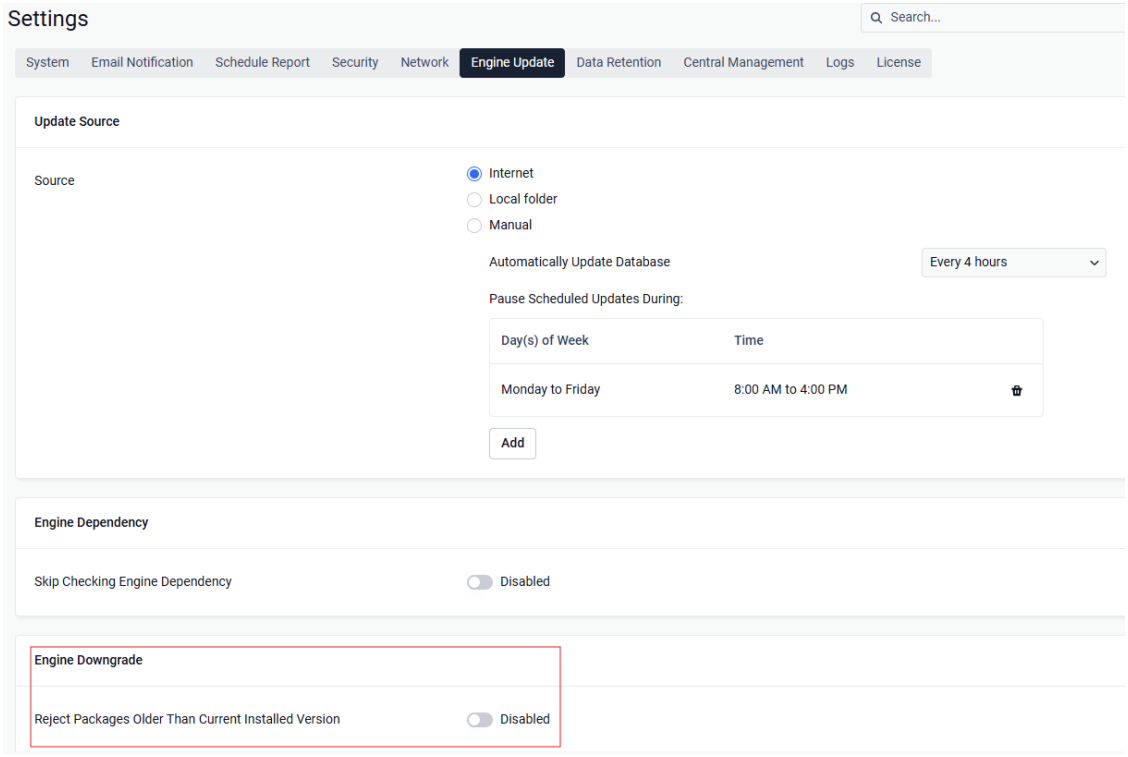
- PsExec.exe -s net use F: \192.168.0.1\test

Prevent Engine Downgrade

By default, MetaDefender Core installs any engine or database package whose version differs from the currently installed version, regardless of whether the package version is newer or older. This behavior is intentional and designed to provide flexibility for package management.

MetaDefender Core also provides a setting that blocks downgrades of engine and database packages. When enabled, an installed package cannot be replaced by an older one, which helps protect your deployment from

accidental rollbacks to outdated threat definitions.

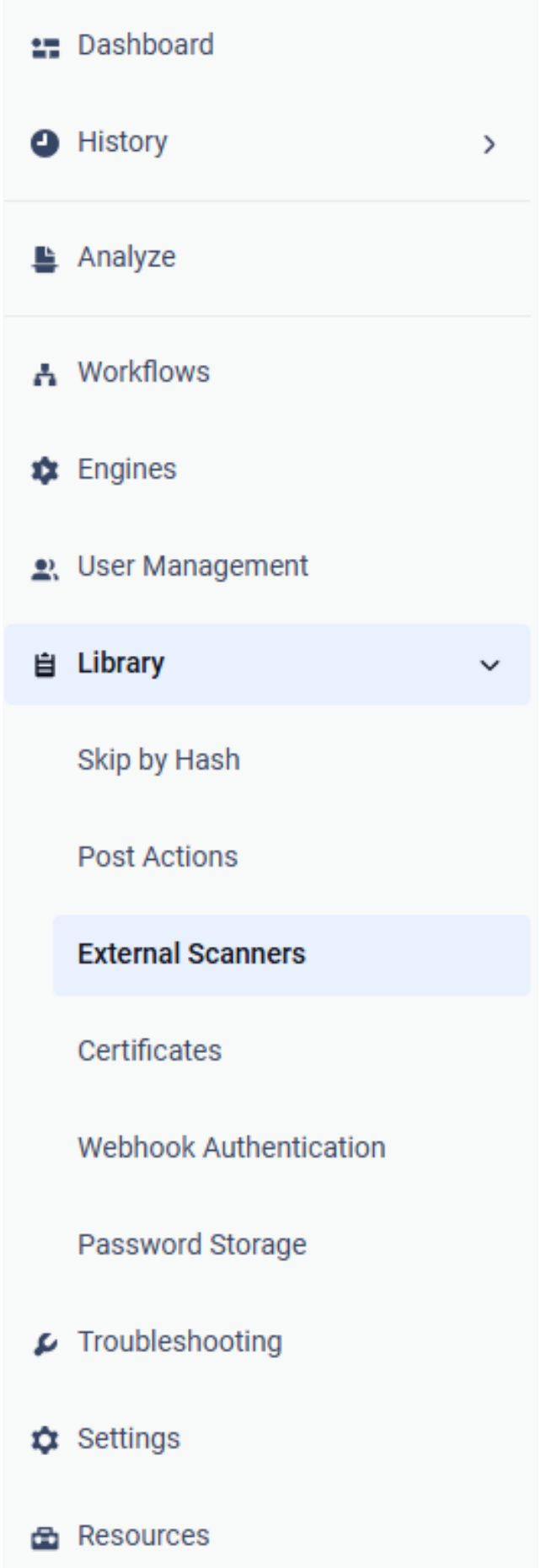


External Scanners and Post Actions

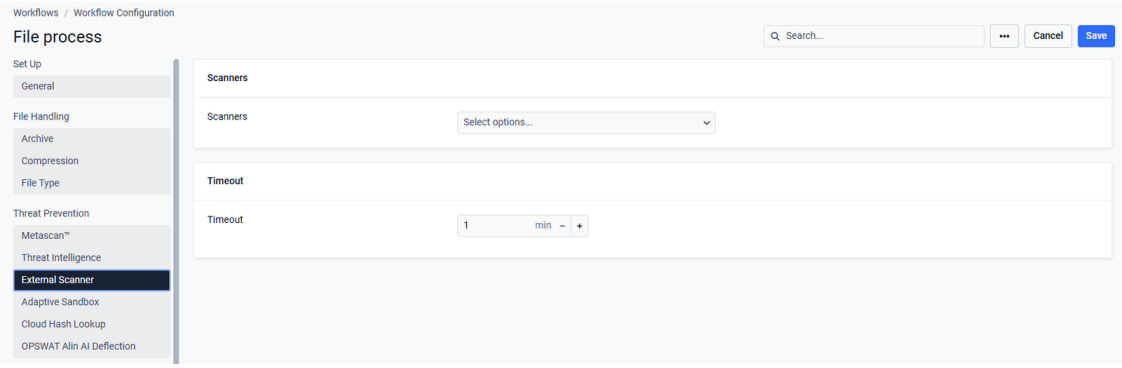
Under **Library** menu it is possible to configure custom **External Scanners** and custom **Post Actions**.

For both these options we must enter two fields:

- a unique name (maximum 16 character ASCII only text)
- a full path to your executable/interpreter, that will be called by the processing node


Dashboard
History
Analyze
Workflows
Engines
User Management
Library
Skip by Hash
Post Actions
External Scanners
Certificates
Webhook Authentication
Password Storage
Troubleshooting
Settings
Resources

Note: Since MetaDefender Core 5.2.2, External Scanner and Post Action no longer apply to all the requests. It now binds to Workflow. Therefore, after creating either External Scanner or Post Action, we need to add them into our desired Workflow Rules or Workflow Templates.



External Scanners

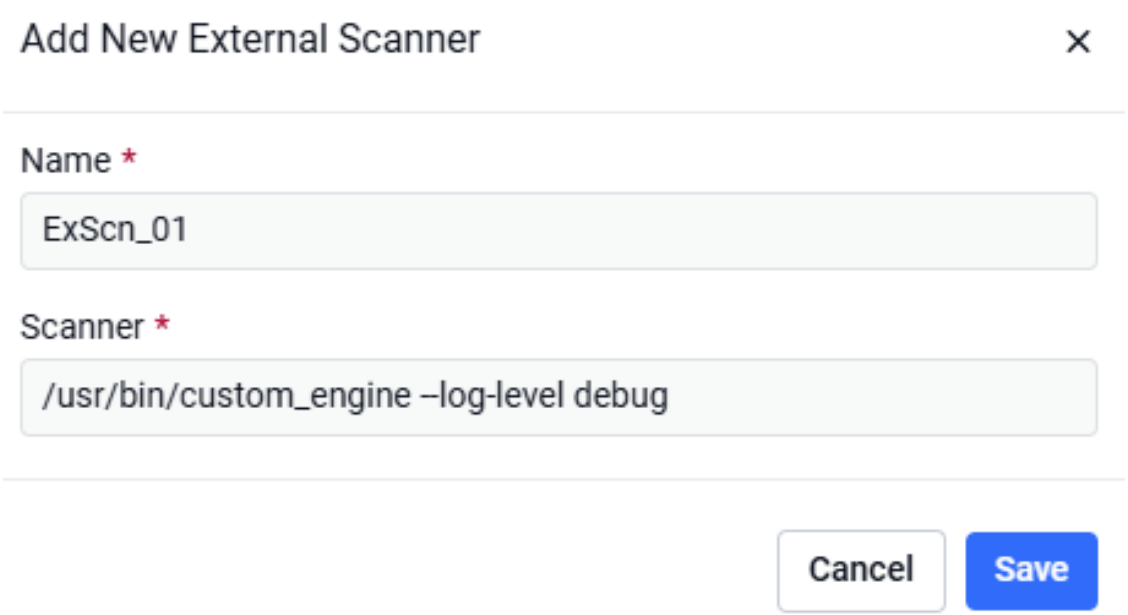
External Scanners are handled as scan engines from product side but are not updatable through the product.

Specification for external scanner process

- **INPUT**
  - On standard input it gets the currently available scan result JSON without the `extracted_files` field
  - As last argument on the command line it gets the absolute path for the file to scan
  - If the command path contains space character, then must nest the command path into double quote. For example: "C:\Test Space\Hello.exe"
- **OUTPUT**
  - If everything goes well, the return value must be 0; otherwise, to indicate that the scanner has **Failed**, a value in the range of 1 to 255 on Linux and 1 to 2,147,483,647 on Windows should be returned.
  - Scan result must be put on standard output in JSON format with the following fields
    - **def_time**: the definition time of this scanner in milliseconds since epoch that will be displayed by Metadefender Core
    - **scan_result_i**: the scan verdict for the file, see the GET Fetch Analysis Result > scan_all_result_i
    - **threat_found**: the found threat's description if any
    - If any of the above fields is missing or invalid, the result will automatically be **Failed** for this scanner

> **Info**
>
> Number of External Scanners is a separately licensed feature. If you plan to use this feature please contact your OPSWAT reseller.

Example for a Custom Scanner



Example input for a Custom Scanner

response.json · json


```
{
  "data_id": "091c07fe6203479983682f3b4a491ee6",
  "file_info": {
    "display_name": "archive.zip",
    "file_size": 2123967,
    "file_type": "application\/zip",
    "file_type_description": "ZIP compressed archive",
    "md5": "ec8fa3c2897c0956f0e9ed5c092310b9",
    "sha1": "0027fc18ed97063387bca9c518a02a6faba85c38",
    "sha256": "4fb0083cd3cd966817c1ee4fa3f02519d05eca0b57c2bf71109d3bd69acebd41",
    "upload_timestamp": "2017-04-27T13:05:20.435Z"
  },
  "process_info": {
    "blocked_reason": "Infected",
    "file_type_skipped_scan": false,
    "post_processing": {
      "actions_failed": "",
      "actions_ran": "",
      "converted_destination": "",
      "converted_to": "",
      "copy_move_destination": ""
    },
    "profile": "File scan",
    "progress_percentage": 100,
    "result": "Blocked",
    "user_agent": "webscan"
```

```
  },
  "scan_results": {
    "data_id": "091c07fe6203479983682f3b4a491ee6",
    "progress_percentage": 100,
    "scan_all_result_a": "Infected",
    "scan_all_result_i": 1,
    "scan_details": {
      "ClamAV": {
        "def_time": "2017-04-27T06:59:21.000Z",
        "location": "local",
        "scan_result_i": 1,
        "scan_time": 51,
        "threat_found": "Win.Trojan.Trojan-1082 FOUND"
      }
    },
    "start_time": "2017-04-27T13:05:20.471Z",
    "total_avs": 1,
    "total_time": 1444
  },
  "vulnerability_info": {}
}
```


Example valid output of a Custom Scanner

Code · json


```
{
"def_time": 1491288912392,
"scan_result_i": 0,
"threat_found": ""
}
```


Example scan result where External Scanner found the file to be clean

Code · json


```
...
  "scan_results": {
    "data_id": "091c07fe6203479983682f3b4a491ee6",
    "progress_percentage": 100,
    "scan_all_result_a": "Infected",
    "scan_all_result_i": 1,
    "scan_details": {
      "ClamAV": {
        "def_time": "2017-04-27T06:59:21.000Z",
        "location": "local",
        "scan_result_i": 1,
        "scan_time": 51,
```

```
        "threat_found": "Win.Trojan.Trojan-1082 FOUND"
      },
      "ExtScn_01": {
        "def_time": "2017-02-27T05:19:11.000Z",
        "location": "local",
        "scan_result_i": 0,
        "scan_time": 10,
        "threat_found": ""
      }
    },
    "start_time": "2017-04-27T13:05:20.471Z",
    "total_avs": 1,
    "total_time": 1444
...
```


Post Actions

Post Actions (PA) run after the scan of the file for any post functionality such as copying the file etc...

Specification for post action process

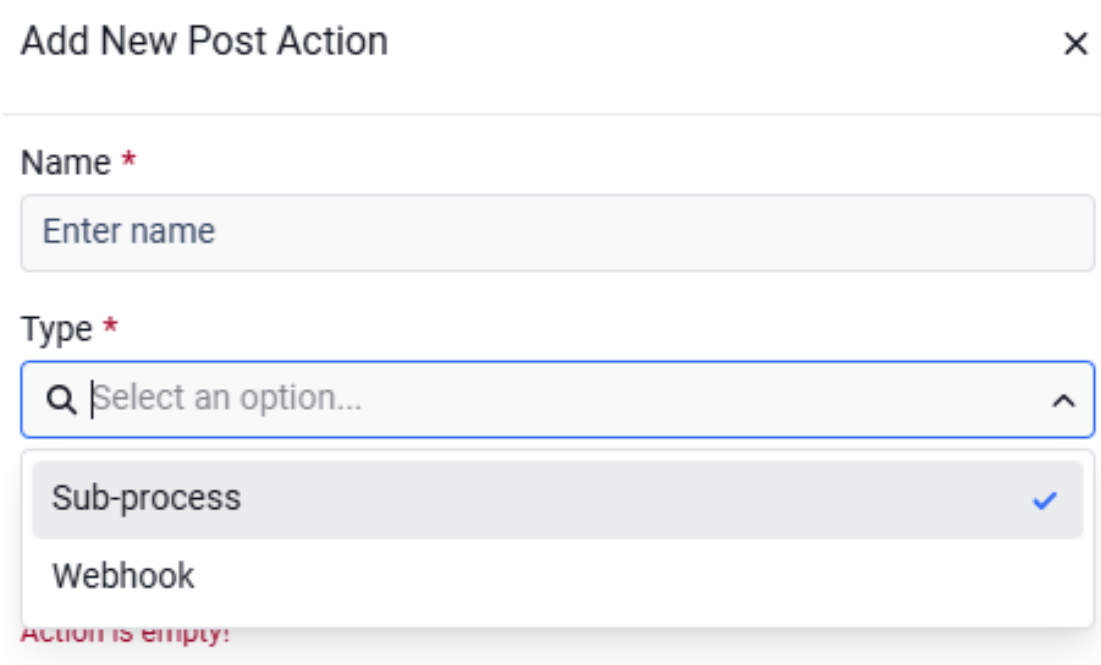
- **INPUT**
  - On standard input it gets the currently available scan result JSON without the `extracted_files` field
  - As last argument on the command line it gets the absolute path for the file
- **OUTPUT**
  - If everything goes well, the return value must be 0; otherwise, to indicate that the action has **Failed**, a value in the range of 1 to 255 on Linux and 1 to 2,147,483,647 on Windows should be returned**.**

Adding a Post Action is the same as in case of an External Scanner. The only difference is in the result handling.

All executed Post Action's result will be on the **process_info.post_processing** object of the scan result JSON. If the return value of an action is zero it will be shown in the **actions_ran** field, if the return value of the action is non-zero then it will be listed in the **actions_failed** field.

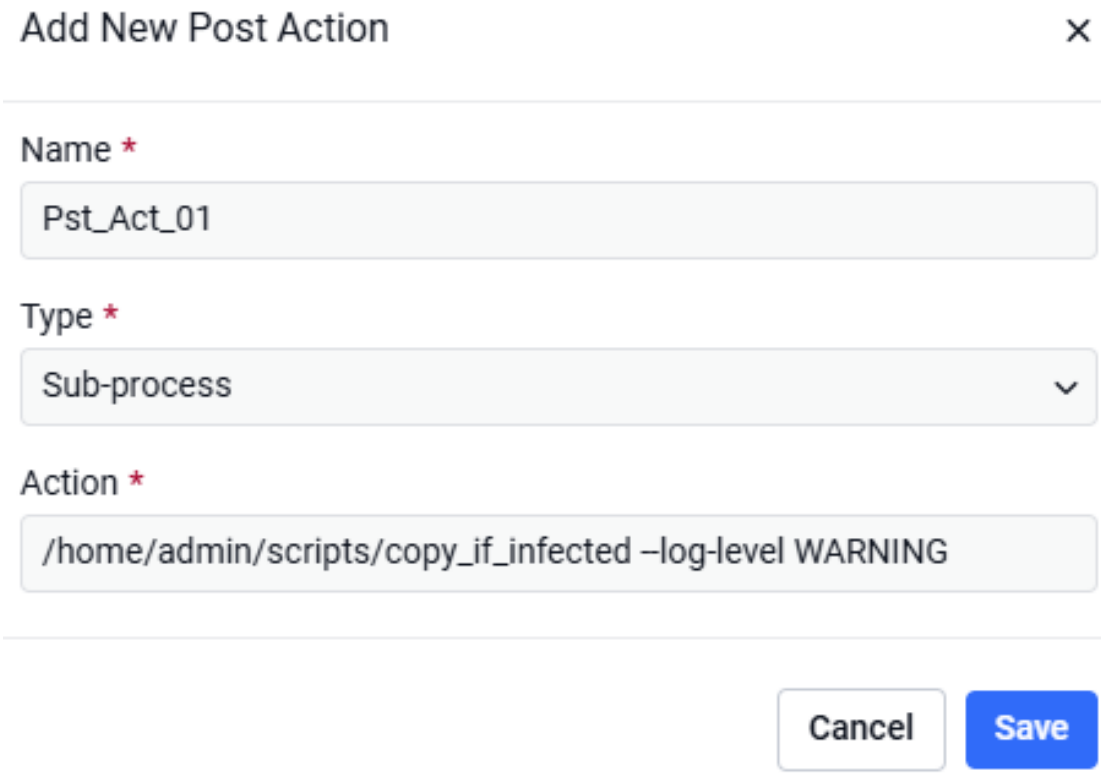
Two types of Post Action

Currently, MetaDefender Core supports two types of Post Action: Sub-process and Webhook.



- Sub-process: MetaDefender Core will call the PA as a sub-process
- Webhook: MetaDefender Core will call a `POST` request to customers' self-hosted server (customers needs to deploy and maintain this server)

Example of a Sub-process Post Action



The scan result JSON if the Post Action returns 0

The field `post_action` will look like this

Code · json

```
"post_action": {
        "details": [
            {
                "duration": 6,
                "name": "Test PA",
        "return_value": 0,
                "verdict": 35
            }
        ]
```

Code · json

```
{
    "data_id": "cf24c1b6a14b418eae59b0cc1db1a9d9",
    "dlp_info": {},
    "extracted_files": {
        "files_extracted_count": 2,
        "total_extracted_files": 5,
        "worst_data_id": "975d30e24c4243bb9d27645eb1dcd2b8"
    },
    "file_info": {
        "display_name": "test.zip",
        "file_size": 564,
        "file_type": "application/zip",
        "file_type_description": "ZIP Archive",
        "is_skip_hash": false,
        "md5": "eb63b20d6df576dd753da3312e71ae46",
        "receive_data_timestamp": "2021-12-31T06:56:22.104Z",
        "sha1": "77ec04c7190a7aa04a9c0666359bedddafada05e",
        "sha256": "9b98873f15075fe24ccc6ddc22058d8b6415a03f1349e688adcd19192c5a834f",
        "type_category": [
            "A"
        ],
        "upload_time": 2,
        "upload_timestamp": "2021-12-31T06:56:22.106Z"
    },
    "post_action": {
        "details": [
            {
                "duration": 6,
                "name": "Test PA",
        "return_value": 0,
                "verdict": 35
            }
```

```
        ],
        "total_blocked": 0,
        "total_executed": 1,
        "total_failed": 0,
        "total_post_action": 1,
        "total_success": 1
    },
    "process_info": {
        "blocked_reason": "",
        "blocked_reasons": [],
        "file_type_skipped_scan": false,
        "post_processing": {
            "actions_failed": "",
            "actions_ran": "Test PA",
            "converted_destination": "",
            "converted_to": "",
            "copy_move_destination": ""
        },
        "processing_time": 10,
        "profile": "File process",
        "progress_percentage": 100,
        "queue_time": 15,
        "result": "Allowed",
        "user_agent": "webscan",
        "username": "LOCAL/admin",
        "verdicts": [
            "No Threat Detected"
        ]
    },
    "scan_results": {
        "current_av_result_a": "No Threat Detected",
        "current_av_result_i": 0,
        "data_id": "cf24c1b6a14b418eae59b0cc1db1a9d9",
        "last_file_scanned": "test.zip",
        "progress_percentage": 95,
        "scan_all_result_a": "No Threat Detected",
        "scan_all_result_i": 0,
        "scan_details": {
            "Avira": {
                "def_time": "2020-09-13T00:00:00.000Z",
                "eng_id": "avira_1_windows",
                "location": "local",
                "scan_result_i": 0,
                "scan_time": 0,
                "threat_found": "",
                "wait_time": 1
            }
        },
        "start_time": "2021-12-31T06:56:22.121Z",
        "total_avs": 1,
        "total_time": 10
```

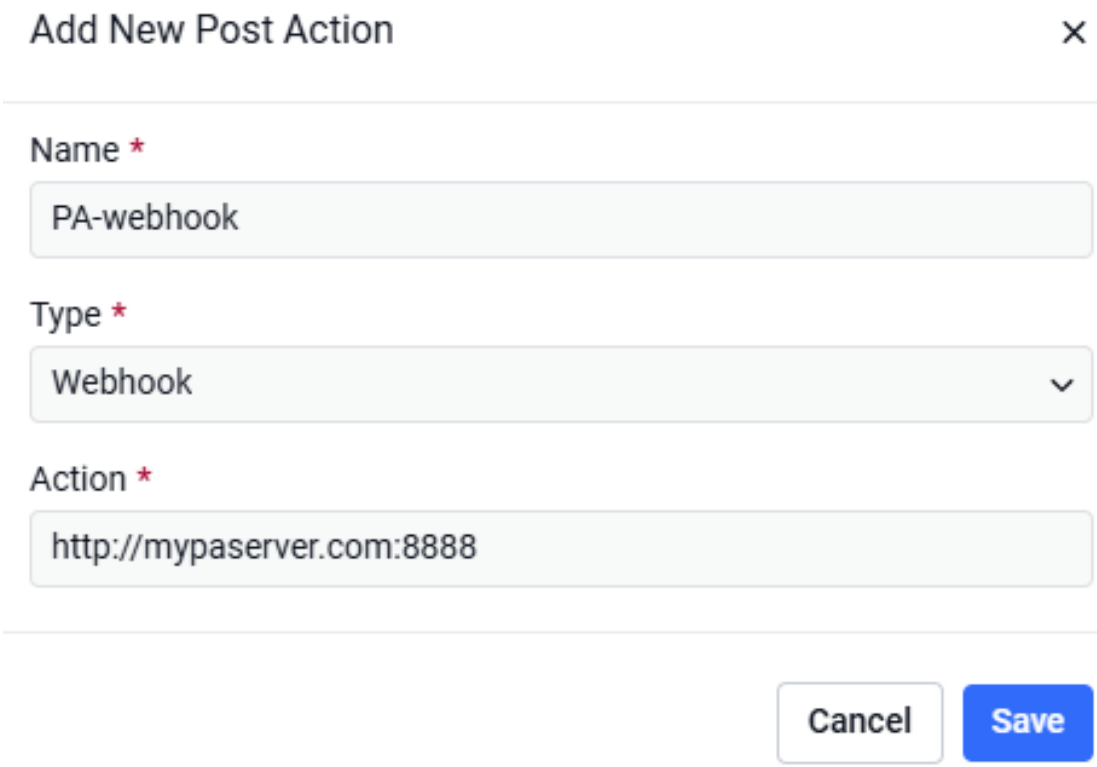
```
    },
    "vulnerability_info": {
        "verdict": 3
    },
    "yara_info": {}
}
```

The scan result JSON if the Post Action returns non-zero

Code · json

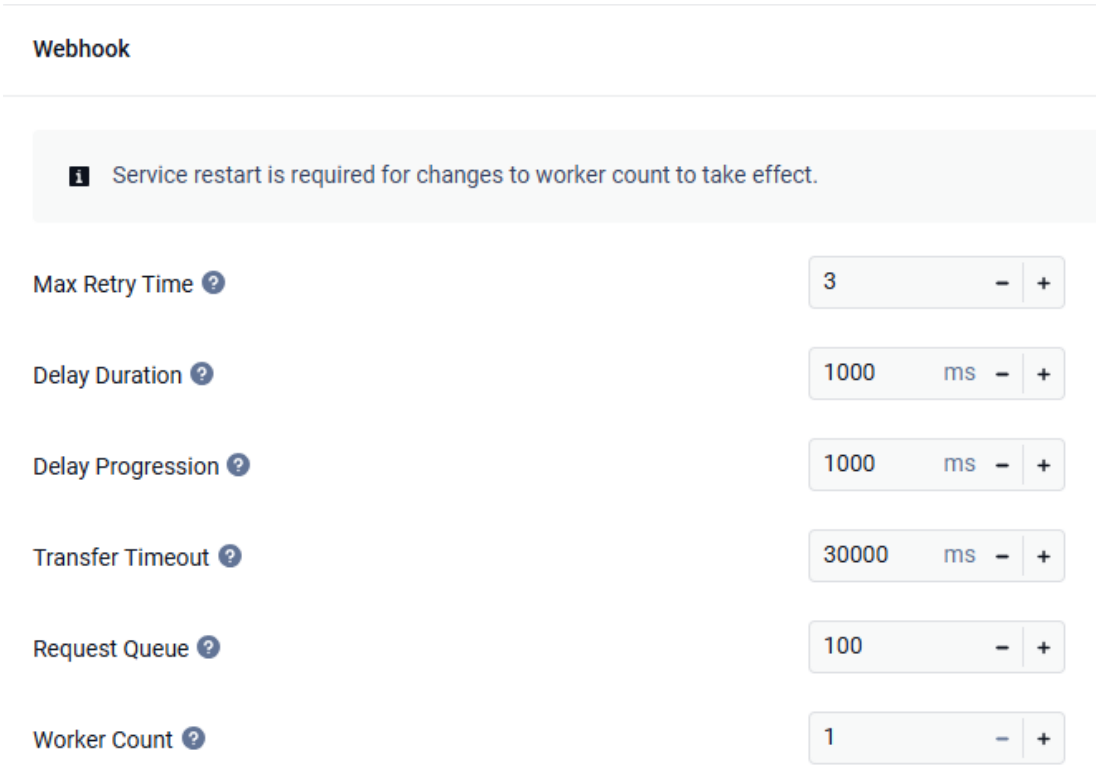
```
...
  "post_action": {
    "details": [
      {
        "duration": 6,
        "name": "Pst_Act_01",
        "return_value": 1,
        "verdict": 37
      }
    ],
    "total_blocked": 0,
    "total_executed": 1,
    "total_failed": 1,
    "total_post_action": 1,
    "total_success": 0
  },
  "process_info": {
    "blocked_reason": "Infected",
    "file_type_skipped_scan": false,
    "post_processing": {
      "actions_failed": "Pst_Act_01 failed",
      "actions_ran": "",
      "converted_destination": "",
      "converted_to": "",
      "copy_move_destination": ""
    },
    "profile": "File scan",
    "progress_percentage": 100,
    "result": "Blocked",
    "user_agent": "webscan"
  },
...
```

Example of a Webhook Post Action



When a file is done, MetaDefender Core will call a `POST` request to the Callback URL, in this case `http://mypaserver.com:8888`

MetaDefender Core will wait for a timeout, that can easily configure via each workflow settings, under the tab "Post Action"



Specification for post action webhook

INPUT

MetaDefender Core will send this body (json) to PA webhook server

Code · json

```
{
  "processing_result": "current_processing_result_of_file",
  "resource_path": "absolute_file_path"
}
```

Please note that `current_processing_result_of_file` (that webhook server received) is just the current result of the file (has not done yet), so the `processing_time` will be -1

If you want to read the file in resource_path the webhook server need to be on the same machine of MetaDefender Core

OUTPUT

If everything goes well, the return value must be 200 HTTP status code; other status code rather than 200 will be consider at Failed.

The value of `post_action.details.return_value` will be the status of return code of webhook server.

For example: MetaDefender Core call `POST`http://mypaserver.com:8888`` the server return 200 -> `post_action.details.return_value` = 200

Example details of a file when run with PA - webhook

Code · json

```
{
    "coo_info": {},
    "data_id": "414c854742bf40aab6e597a655fadd3f",
    "dlp_info": {},
    "extraction_info": {
        "is_encrypted_file": false
    },
    "file_info": {
        "display_name": "abc.log",
        "file_size": 1236169,
        "file_type": "text/plain",
        "file_type_description": "ASCII Text",
        "file_type_id": "TXT",
        "is_skip_hash": false,
        "md5": "b886c7dd2b11cea526aa30669d9505a0",
        "receive_data_timestamp": "2025-08-01T08:20:34.582Z",
        "sha1": "ebc763f5562d9008ba424aa4b51a0249b3b49ef9",
        "sha256": "9478e713eda06b654084f846735d7f6486fb737afa2d46a49b438efdbaf4c78a",
        "sha512": "",
        "signer_infos": [],
        "type_category": [
```

```
            "T"
        ],
        "upload_time": 23,
        "upload_timestamp": "2025-08-01T08:20:34.605Z"
    },
    "filetype_info": {
        "file_info": {
            "description": "ASCII Text",
            "encrypted": false,
            "extensions": "txt,dat,prn,css",
            "groupID": "T",
            "groupIDs": [
                "T"
            ],
            "type": "text/plain",
            "typeID": "TXT",
            "type_ids": [
                "TXT",
                "LOG"
            ]
        },
        "is_file_type_mismatch": false,
        "result_template_hash":
"126b24af8d08e8199d478b03bfaf629bc0756d87c20fb46ff5c6b49b9a843303"
    },
    "insightsti_info": {},
    "opswatfilescan_info": {},
    "post_action": {
        "details": [
            {
                "duration": 824,
                "name": "PA-webhook",
                "return_value": 200,
                "verdict": 35
            }
        ],
        "total_blocked": 0,
        "total_executed": 1,
        "total_failed": 0,
        "total_post_action": 1,
        "total_success": 1
    },
    "process_info": {
        "archive_handling_details": {},
        "blocked_reason": "Not Scanned",
        "blocked_reasons": [
            "Not Scanned"
        ],
        "file_type_skipped_scan": false,
        "hash_time": 5,
        "post_processing": {
```

```
            "actions_failed": "",
            "actions_ran": "PA-webhook",
            "converted_destination": "",
            "converted_to": "",
            "copy_move_destination": ""
        },
        "processing_time": 893,
        "processing_time_details": {
            "filetype_time": 54,
            "others_time": 13,
            "post_action_time": 826
        },
        "profile": "File process",
        "progress_percentage": 100,
        "queue_time": 33,
        "result": "Blocked",
        "user_agent": "webscan",
        "username": "LOCAL/admin",
        "verdicts": [
            "Not Scanned"
        ]
    },
    "reputation_info": {},
    "sbom_info": {},
    "scan_results": {
        "current_av_result_a": "Not Scanned",
        "current_av_result_i": 10,
        "data_id": "414c854742bf40aab6e597a655fadd3f",
        "progress_percentage": 100,
        "scan_all_result_a": "Not Scanned",
        "scan_all_result_i": 10,
        "scan_details": {},
        "start_time": "2025-08-01T08:20:34.643Z",
        "total_avs": 0,
        "total_time": 893
    },
    "vulnerability_info": {
        "verdict": 3
    },
    "yara_info": {}
}
```


Hardening External Scanner and Post Action feature

Since MetaDefender Core v5.14.1, we have introduced a security hardening feature that allows administrators to restrict which commands can be used in External Scanners and Post Actions. This guide will help you set up and configure that external command allowlist.

Benefits of Command Allowlisting

- **Enhanced Security**: Prevents the execution of unauthorized or potentially harmful commands
- **Controlled Environment**: Ensures only approved commands are used in your workflow
- **Compliance**: Helps meet security requirements by restricting command execution

Setting Up the External Allowlist

File Location

Create a new JSON file named `external_allowlist.json` in the following location based on your deployment method:

Deployment	File Path
Windows	<MD Core installation folder>\external_allowlist.json
Linux VM/Physical	/etc/ometascan/external_allowlist.json
Docker	/opt/ometascan/core_data/external_allowlist.json

> **Info**
>
> ***Note for Docker users**: The file is located in the `CORE_DATA_PATH` directory, which can be modified using environment variables.*

File Format

The file should contain a JSON array of strings, with each string representing an allowed command:

json · javascript

```
[
  "/usr/bin/clamscan",
  "/opt/scripts/custom_scanner.sh",
  "python3 /path/to/script.py",
  "C:\\Program Files\\CustomScanner\\scan.exe",
  "/usr/local/bin/sanitize.sh"
]
```

Configuration Rules

1. Each entry must be a complete command string (including parameters if needed)
2. Include the full path to the executable

3. For Windows paths, use double backslashes () in your JSON

4. Commands are case-sensitive

5. Duplicate or empty entries will be skipped

File permissions

When the configuration has been done. Admin should change the permission of the file external_allowlist.json

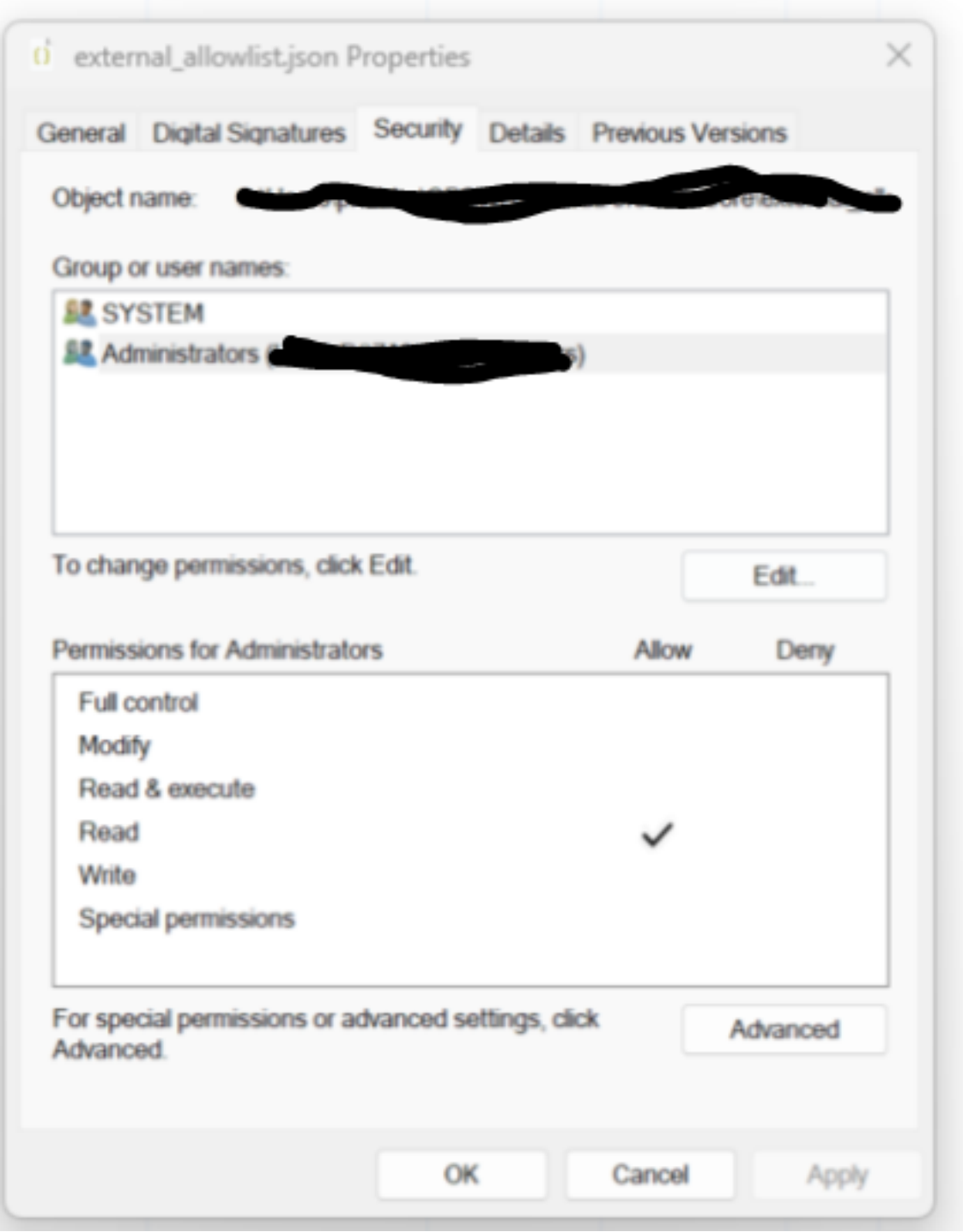
Linux

Shell · javascript

```
sudo chmod 400 /path/to/file/external_allowlist.json
sudo chown metascan:metascan /path/to/file/external_allowlist.json
```

Windows

We should only allow read for Admin and SYSTEM account



Enabling the Feature

The feature is automatically enabled when the `external_allowlist.json` file exists:

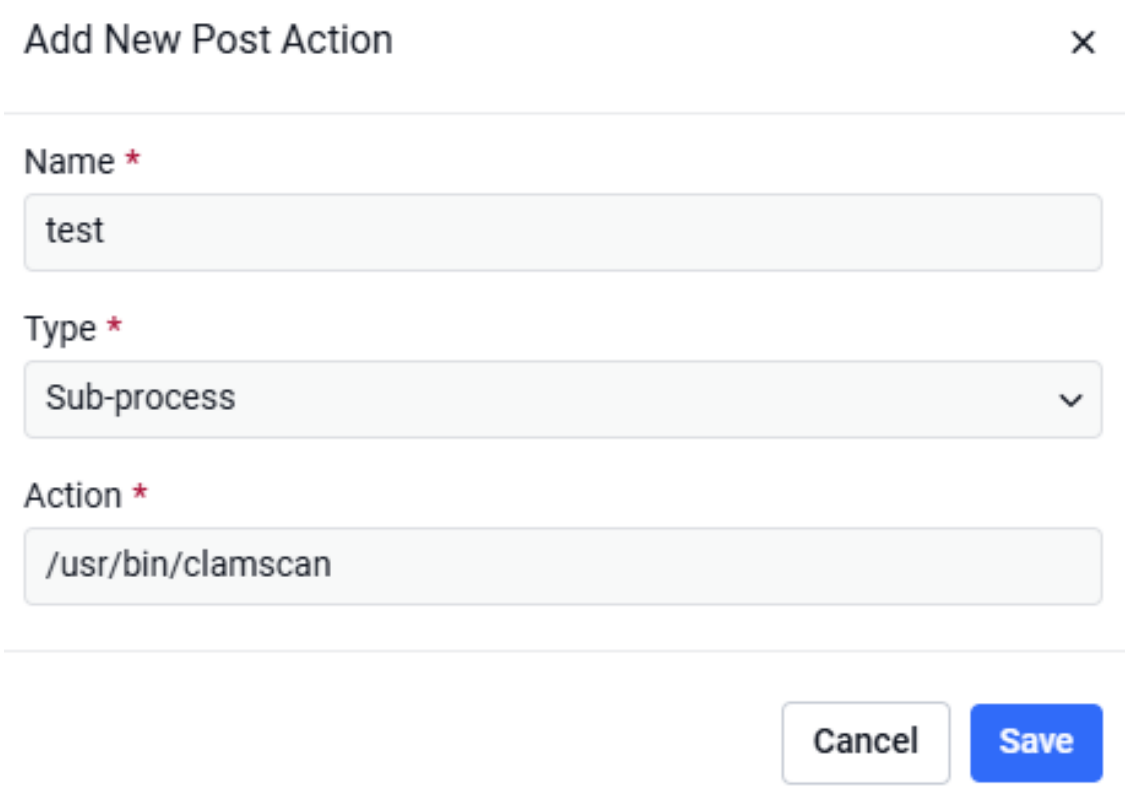
- If the content of external_allowlist.json is in right format -> allow the commands in the file
- If the content of external_allowlist.json is in wrong format or empty -> it means that **zero** commands has been configured -> MetaDefender Core's admin users cannot set any command to either external scanner nor Post-Actions

Using the Web Interface

After configuring the allowlist:

1. External Scanner and Post Action configuration screens will display a combo box with allowlisted commands
2. Commands that are not on the allowlist will be marked with a warning icon
3. When editing non-compliant configurations, warning messages will appear:

- For Post Actions: "Your action does not comply with security rules. Please choose an action from the approved list."
- For External Scanners: "Your scanner does not comply with security rules. Please choose a scanner from the approved list."



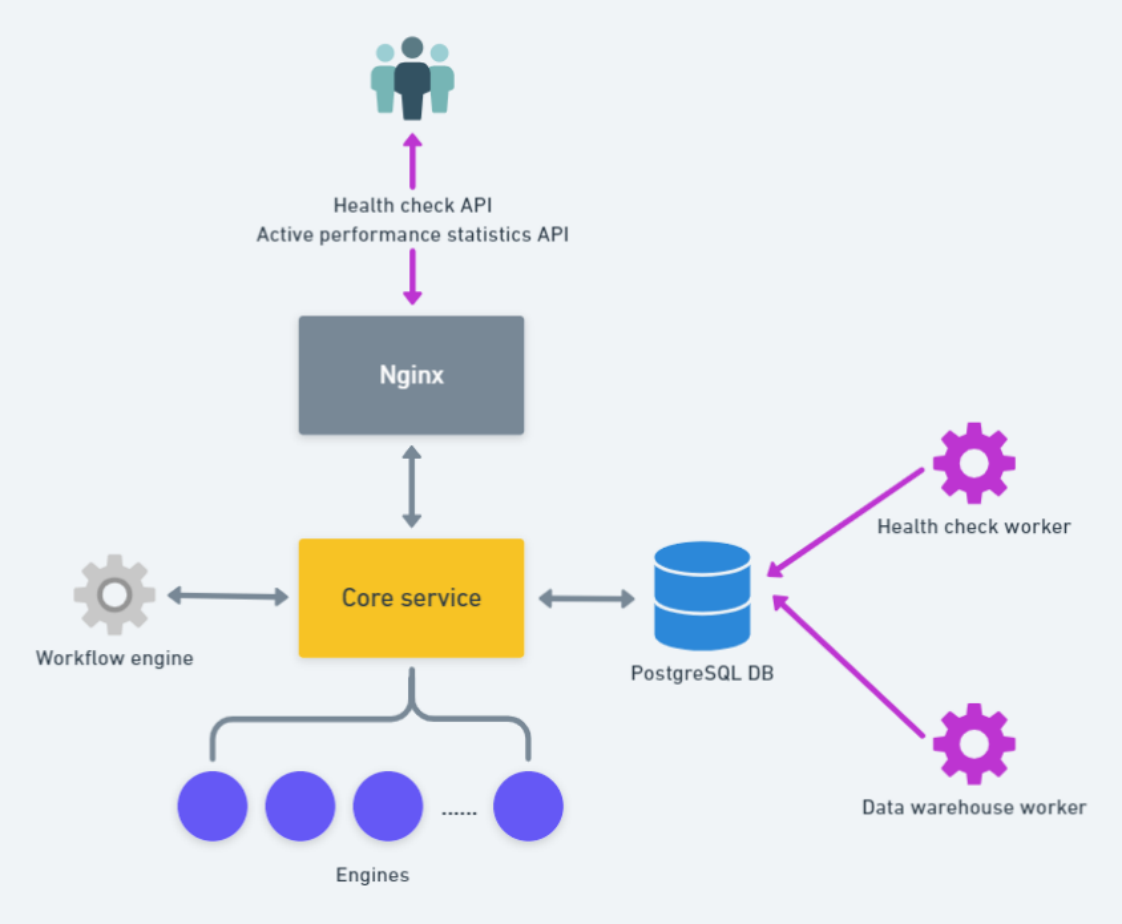
-

Health Check API Configuration

Product version supported

MetaDefender Core version 5.0.0 or newer

Health check API feature supports you to verify MetaDefender Core application readiness in term of file processing and licensing, before making a decision to submit files and avoid time and resources being wasted predictably.



Using the health check API

Health check API expects to return HTTP(S) 200 Successful only when all pre-configured conditions met.

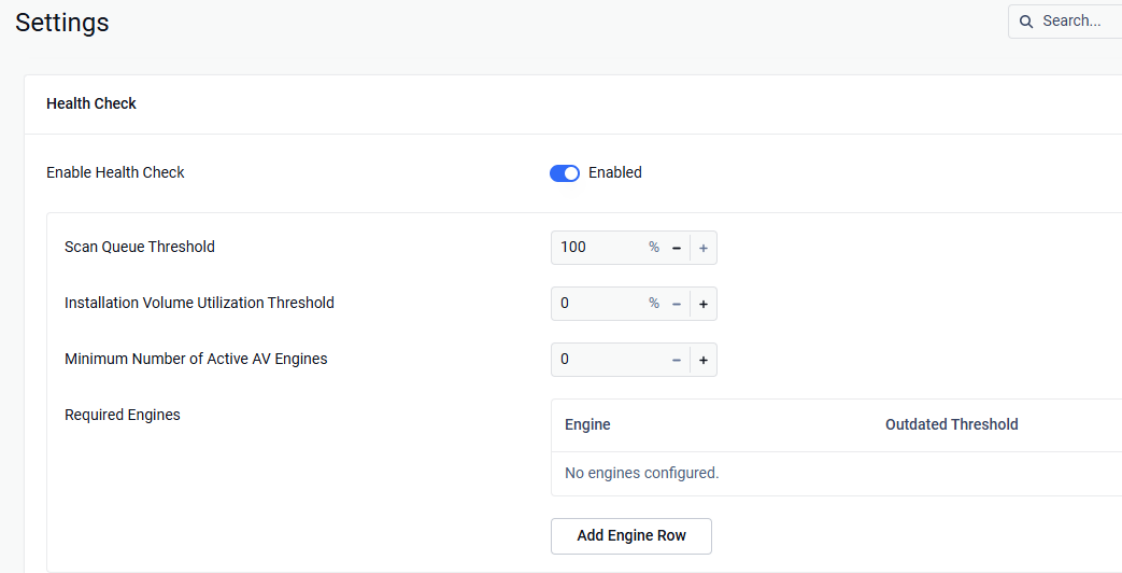
Learn more on how to use the heath check API at: GET - Get health check status

Configure health check policy

You are supported to adjust thresholds on vital criterias as your own policy tailored for heath check verification, via both REST API and MetaDefender Core management console.

Using UI

The relevant settings could be found under "Settings" page



When enabled, expecting all conditions in associated configurations met to determine MetaDefender Core's readiness for file processing.

	Value	Description
Enable health check	On / Off	Enabling this option will allow MetaDefender Core automatically run a background task to collect application operational data for health check logic.
Scan queue threshold	0% - 100%	MetaDefender Core's occupied percentage slots in queue must be equal or smaller than what configured.
Installation volume utilization threshold	0% - 100%	Monitor whether the free disk space on the MetaDefender Core installation drive falls below the configured threshold.
Minimum number of active engines	>= 0	Applicable to AV engines only. Making sure the pre-configured value is not greater than your current actual licensed AV engines.
Required engines		Select which engines are required for the health check to pass and configure their outdated threshold

	Value	Description
		in days.

Using administrative REST API

Learn more at: GET - Get health check

Hot Swap Backup Configuration

Product version supported

MetaDefender Core version 5.0.0 or newer

Hot backup mode supports you to register (activate) as many MetaDefender Core instances as you might need for backup purpose, and those backup instance(s) should be ready to take over with your active / production MetaDefender Core instance whenever applicable.

With engines and configurations ready to use, that helps mitigate any down time in scanning system in case of your recovery disaster plan needs to be triggered.

MetaDefender Core backup instance(s), once registered properly, does not accumulate into your actual license activation slots, that means, you will not be required to purchase additional product license just for this backup purpose alone.

What is allowed

MetaDefender Core backup instance, upon activated in the backup mode, allowing you to:

- Download and deploy licensed engines (modules), proceed them all to active mode.
- Modify MetaDefender Core and its engines (modules) settings.
- Import MetaDefender Core settings.
- Access and administrate all other operations on MetaDefender Core management console.

What is not allowed

The only thing that MetaDefender Core backup instance(s) does not allow is accepting any scan requests coming, and processing them.

Register a backup instance

1. Online environment: Using following REST API: POST - Setup hot-backup instance

2. Offline environment:

- Obtain `license.yml` file that supposed to use to active your MetaDefender Core active / production instance before.
- Using following REST API: POST - Setup hot-backup instance with that yaml file content.

> **Note**
>
> If you try to activate a MetaDefender Core instance which is currently registered as a backup one (using POST - Activate license online) then expecting to hit error. Instead you should **either** do following:
>
> 1. Nominate that backup instance (see below)
> 2. Unregister that backup instance (see below), and then activate it using POST - Activate license online

> **Register backup on an active / production instance**
>
> You cannot register a MetaDefender Core as a backup instance when it is still running as an active / production one.
>
> Instead, you are required to deactivate that instance first.

Unregister a backup instance

This step will be required when following circumstances happen:

1. You want to activate that backup instance directly without using nomination method (see below) via POST - Activate license online
2. You want to shutdown that backup instance for some reasons, but avoid removing the container or system where that MetaDefender Core instance is installed on for possible usage later.

Using following REST API: DELETE - Remove a backup instance

Upon unregistered successfully, that MetaDefender Core instance will no longer allow you to:

- Download engines and their databases from OPSWAT update server source (online mode)
- Upload engines and their databases manually
- Deploy engines and their databases on MetaDefender Core

Basically that MetaDefender Core will become an unlicensed one as a fresh installation.

Nominate a backup instance

Nominating a backup instance is to make that instance become an active / production one. That means, your nominated instance will be able to process files as in production.

Online environment: Using following REST API: POST - Nominate a backup instance to a production instance

Expected behaviors:

When the backup instance is successfully nominated to be a new active / production one, then expecting following behaviors:

- The backup instance's actual deployment ID will be registered with relevant license key on our OPSWAT Licensing system, and it is then able to process processing requests right away.
- The former active / production instance (before the nomination took place), will be auto unregistered from OPSWAT Licensing system.

Offline environment: You are not allowed to directly nominate a MetaDefender Core backup instance which was previously registered in offline mode.

Nominating MetaDefender Core backup instance in offline mode

Instead, following actions needs to be made as alternative procedure:

- Manually deactivate your MetaDefender Core active / production instance first, and unregister it from OPSWAT Licensing system.
- Unregister your MetaDefender Core backup instance.
- Then, activate your MetaDefender Core backup instance as a production one.

Logging

Since Metadefender Core (MD Core) v5.15.1, we've supported advanced log configuration, you can configure the log settings directly through the MetaDefender Core UI or via My OPSWAT Central Management: Advanced Log Configurations. However, the traditional configuration method (via registry or configuration file) is still supported: MetaDefender Configuration. Logs settings configured through the MetaDefender Core UI, Endpoints or via My OPSWAT Central Management take precedence over those set via the registry or configuration file.

Note

Permissions to set:

- The directory that contains the logs: read, write and execute permissions for ometascan user or at least for ometascan group
- The other directories on the path to the logs: at least read and execute permissions for ometascan user and/or ometascan group

The new log settings will be used after a service restart or a HUP signal.

Remote Syslog

For SIEM integration such as Splunk, on MetaDefender Core side, enable syslog to send to Splunk syslog server via specific protocol and port. Then on that syslog server, configure to listen all incoming messages over the protocol and port.

Supported protocols are

- TCP. Example: `tcp://192.168.56.1:3586`
- UDP. Example: `udp://192.168.56.1:3585`
- TCPS - TLS over TCP. Example: `tcps://192.168.56.1:3586`

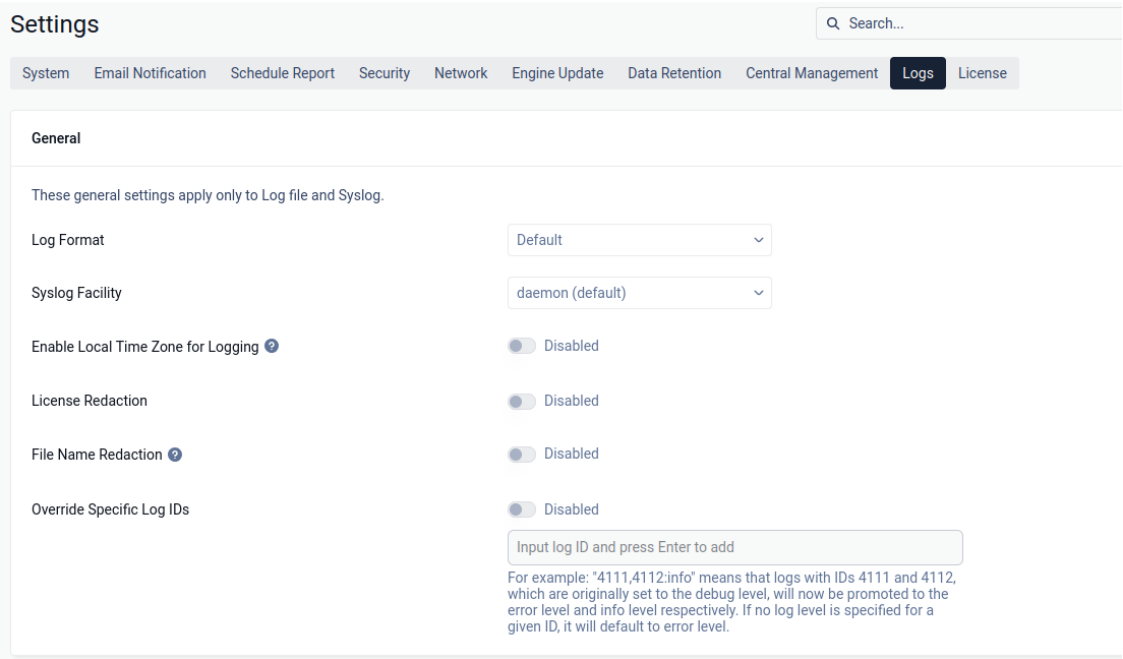
Info

When using TCPS - TLS over TCP, by default - MetaDefender Core will not allow for self-signed certs. If the users want to allow it, please change the option allow self signed cert with this Remote Syslog API

Advanced Log Configurations

Since Metadefender Core (MD Core) v5.15.1, we've supported advanced log configuration, you can configure the log settings directly through the MetaDefender Core UI, via My OPSWAT Central Management or via API endpoints: Logging configuration

General settings

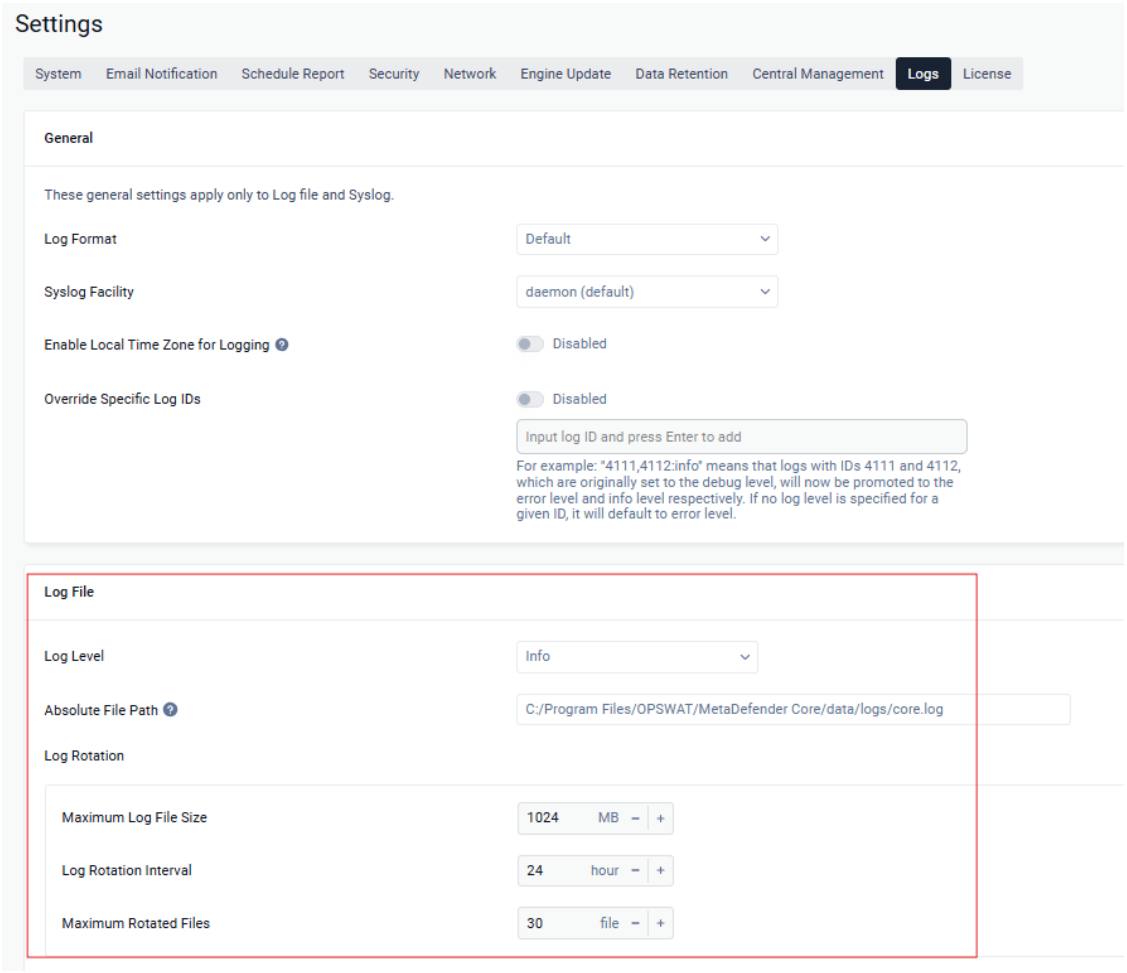


Log file settings

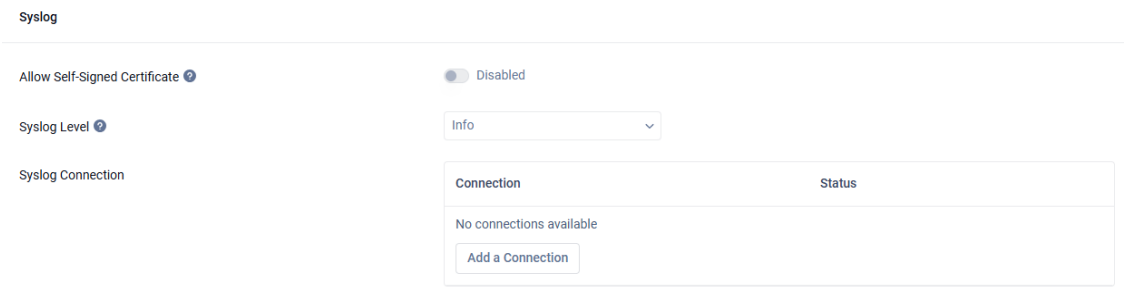
Warning

Changing the log file path is not supported when MetaDefender Core is running in a **container environment**.

When importing a MetaDefender Core configuration into a containerized instance, the log file path setting is not applied — only the log level is imported.



Syslog settings



Log Message Format

Format

In the log, each line represents a log message sent by the server or node. Depending on the log file, the format of the line is as follows:

log format · bash

```
[LEVEL] TIMESTAMP (COMPONENT) MESSAGE [msgid: MESSAGE ID]
```

Example:

log example · bash

```
[INFO] 2019.07.02 05:25:27.115: (core.workflow) Processing finished, node=':1076', user='LOCAL/
admin', workflow_id='lms::workflow::WorkflowExecutor(0x214b02a8f60)',
dataId='702a2230dd0d44de9bd773bccfe472a9', fileName='TermUtil.class',
sha256sum='07aca175cc8a9f40819a47f6b5f809404bae8d31cf16e70d0a182c413ab39c98', blocked='false',
blocked_reason='', overallResult='No Threat Detected', threatFoundCount='0',
embeddedObjectsWithThreat='0', totalResultCount='3', threatDetectedBy='', threatName='',
ruleName='File process', source='::1' [msgid: 82]
```

Where the different values are:

- **LEVEL** : the severity of the message
- **TIMESTAMP** : The date value when the log entry was sent
- **COMPONENT** : which component sent the entry
- **MESSAGE** : the verbose string of the entry's message
- **MESSAGE ID** : the unique ID of this log entry - Learn more at Error Message Description Table

Severity levels of log entries

Depending on the reason for the log entry, there are different types of severity levels.

Based on the configuration, the following levels are possible:

- **DEBUG** : Debuggers severity level, mostly used by support issues.
- **INFO** : Information from the software, such as scan results.
- **Extended INFO:** Like `INFO` level but having more information about every (engine) stage
- **WARNING** : An unexpected event occurred; however, the product is supposed to be operational.
- **ERROR** : Software error happened, please contact support if the issue persists. Software functionality may be downgraded in these cases.

Log Levels recommendation

- In most cases, the **INFO** log level is sufficient for general system monitoring and troubleshooting. It provides the essential information needed to observe normal operations without generating excessive log data.
- The **Extended INFO** log level is useful when users need detailed traceability of each processing stage within the engine. This level records additional events, such as which engine step a file is currently being processed at. Please note that enabling **Extended INFO** will increase log volume and file size compared to

the standard **INFO** level.

- The **DEBUG** log level should only be enabled when investigating issues or under the guidance of OPSWAT Support. Once enabled, users can collect the logs and provide them to OPSWAT for analysis. This level produces a significant amount of log data, which can increase storage usage and negatively impact system performance. Therefore, it is **not recommended for regular production use**.

Syslog Message Format

MetaDefender Core supports to send CEF (Common Event Format) syslog message style

Remote Syslog

log format · bash

```
[Local Timestamp] [Source IP Address] [UTC Timestamp]  [Hostname] [CEF:Version|Device Vendor|
Device Product|Device Version|Signature ID|Name|Severity|Extension]
```

For example:

log example · bash

```
Jun 24 14:33:18 192.168.200.223 2019-06-24T14:33:19+07:00 OPSWATPC CEF:0|OPSWAT|MSCL|4.16.0|
core.network|MSCL[7548] New maximum agent count is set|2|maxAgentCount='1' msgid=665
```

Prefix field	Sample value	Description
Local timestamp	Jun 24 14:33:18	
IP address	192.168.200.223	Source IP address ver. 4
UTC timestamp	2019-06-24T14:33:19+07:00	
Hostname	OPSWATPC	
CEF:Version	CEF:0	Version 0
Device Vendor	OPSWAT	
Device Product	MSCL	MSCL = MetaDefender Core on Linux MSCW = MetaDefender Core on Windows

Prefix field	Sample value	Description
Device Version	4.16.0	MetaDefender Core version
Signature ID	core.network	For example: • core.network: Component "network" on "Core" module • agent.engines: Component "engines" on "Node" • common.update: Component "update" on common module shared by all modules
Name	MSCL[7548] New maximum agent count is set	Subject of log message • MSCL[7548] = MetaDefender Core on Linux ["ometascan" process id = 7548] • ometascan-node[455] = MetaDefender Core Node ["ometascan-node" process id = 455]
Severity	2	Log level • **DUMP (0)**: The most verbose severity level, these entries are for debuggers only. • **DEBUG (1)**: Debuggers severity level, mostly used by support issues. • **INFO (2)**: Information from the software, such as scan results. • **WARNING (3)**: A problem occurred needs investigation and OPSWAT support must be contacted, however the product is supposed to be operational. • **ERROR (4)**: Software error happened, please contact support if the issue is persist. Software functionality may be

Prefix field	Sample value	Description
		downgraded in these cases.
Extension	maxAgentCount='1' msgid=665	To learn more about msgid (message ID): Error Message Description Table

Local Syslog

log format · bash

```
[Local Timestamp] [Hostname] [CEF:Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension]
```

For example:

log example · bash

```
Jun 24 14:33:18 OPSWATPC CEF:0|OPSWAT|MSCL|4.16.0|core.network|MSCL[7548] New maximum agent count is set|2|maxAgentCount='1' msgid=665
```

Prefix field	Sample value	Description
Timestamp	Jun 24 14:33:18	
Hostname	OPSWATPC	
CEF:Version	CEF:0	Version 0
Device Vendor	OPSWAT	
Device Product	MSCL	MSCL = MetaDefender Core on Linux MSCW = MetaDefender Core on Windows
Device Version	4.16.0	MetaDefender Core version
Signature ID	core.network	For example: • core.network: Component "network" on "Core" module

Prefix field	Sample value	Description
		• agent.engines: Component "engines" on "Node" • common.update: Component "update" on common module shared by all modules
Name	MSCL[7548] New maximum agent count is set	Subject of log message • MSCL[7548] = MetaDefender Core on Linux ["ometascan" process id = 7548] • ometascan-node[455] = MetaDefender Core Node ["ometascan-node" process id = 455]
Severity	2	Log level • **DUMP (0)**: The most verbose severity level, these entries are for debuggers only. • **DEBUG (1)**: Debuggers severity level, mostly used by support issues. • **INFO (2)**: Information from the software, such as scan results. • **WARNING (3)**: A problem occurred needs investigation and OPSWAT support must be contacted, however the product is supposed to be operational. • **ERROR (4)**: Software error happened, please contact support if the issue is persist. Software functionality may be downgraded in these cases.
Extension	maxAgentCount='1' msgid=665	To learn more about msgid (message ID): Error Message Description Table

Error Message Description Table

Every log entry on MetaDefender Core always comes with a message ID number which is an unique identifier for that log entry (Log Message Format)

For example:

log example · bash

```
[WARNING] 2019.07.02 05:25:27.749: (core.workflow) Can not send extract task,
id='lms::workflow::WorkflowExecutor(0x214af499d00)', node=':1076', block='ExtractBlock' [msgid:
3059]
```

Then message ID associated to that message would be 3059.

For warning/error message type, besides the message itself (e.g. `Can not send extract task, id='lms::workflow::WorkflowExecutor(0x214af499d00)', node=':1076', block='ExtractBlock'`), using message ID (e.g. 3059) can help reveal more information about that warning/error.

Following is the list of error message description (only applicable to warning/error message type).

Info

Disclaim:

- This list is auto generated and should be updated accordingly with MetaDefender Core upgrade.
- Should be used as reference only, for a complete issue troubleshooting guidance, please reach out to OPSWAT Support.

Error messages need to be monitored

All error and warning messages

Syslog facility

Do I need to change this?

Most users can leave this setting alone. MetaDefender Core works out of the box with the default value (`daemon`), which is the same behavior as before this feature existed.

Change Syslog facility only if your security or operations team has asked you to tag MetaDefender Core logs with a specific syslog category — for example, so a SIEM, Splunk, or rsyslog server can route or filter them separately from other applications.

What is a syslog facility?

When MetaDefender Core sends logs via syslog (to a remote syslog server or to the local Linux syslog), each message includes a facility — a label that tells the receiving system *what kind of source* produced the log.

Think of it as a category tag on every syslog message. Your syslog or SIEM server can use that tag to:

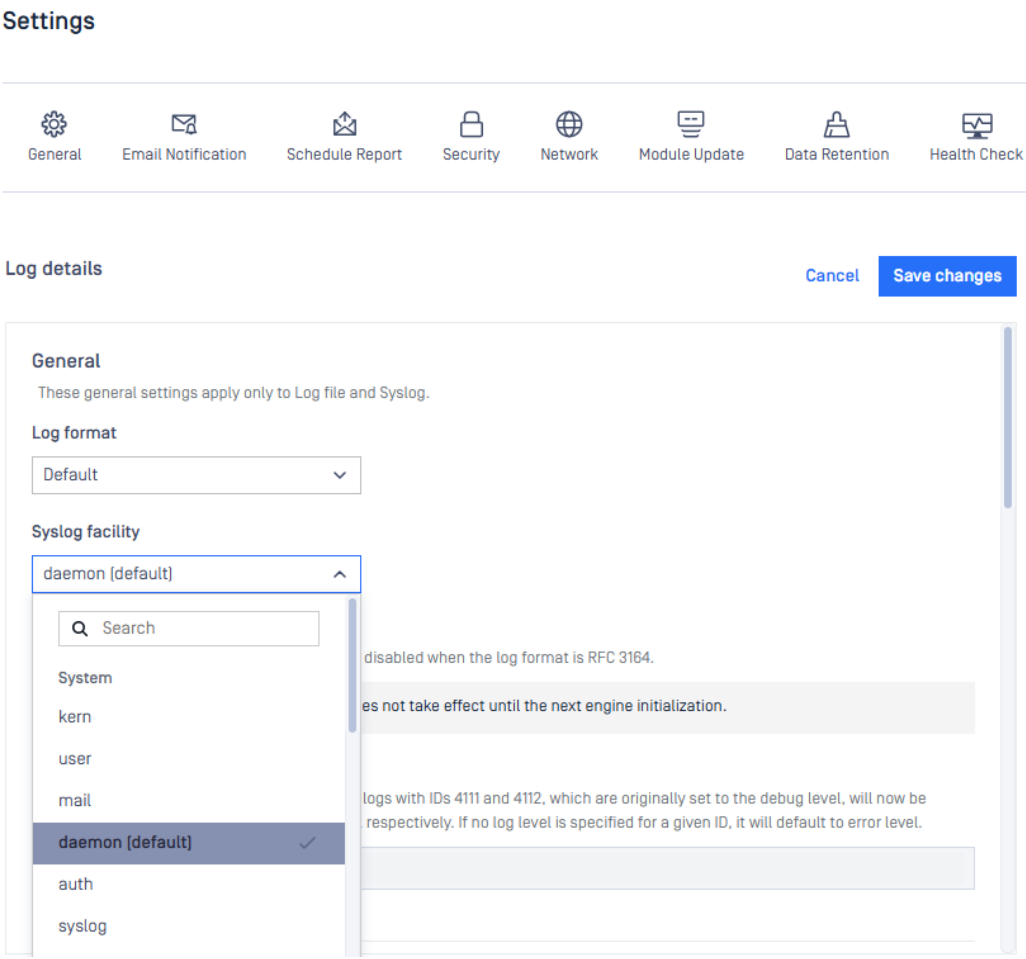
- Route logs to different files or indexes (e.g. all `local0` messages go to a dedicated Splunk index)
- Apply different retention or alerting rules
- Separate MetaDefender Core logs from OS or other application logs

A typical syslog message is read as `<facility>.<level>`. For example:

The level (info, warning, error, etc.) is controlled separately by your existing syslog log-level settings. The facility setting only changes the category prefix.

How it works in MetaDefender Core

- Where: Settings Logging General, under Log format
- Default: `daemon` — no action required for normal use
- Applies to: Both remote syslog (UDP/TCP/TLS) and local syslog (Linux)
- Takes effect immediately after you save — no MetaDefender Core service restart is needed
- One value for all syslog output — the same facility is used for every syslog message Core sends



Common values

Other standard options (`kern`, `user`, `mail`, `auth`, etc.) are available but rarely needed for Core. Use them only if your syslog administrator specifies one.

When should I customize it?

Set a non-default facility when:

1. Your SIEM or syslog administrator gives you a required facility name (commonly `local0`–`local7`)
2. Your syslog server rules expect MetaDefender Core logs under a specific category (e.g. `local0.*`) instead of `daemon.*`
3. You need to separate MetaDefender Core logs from other `daemon` traffic on a shared syslog collector

Example: Your Splunk admin asks MetaDefender Core to send logs as `local0` so they can index them under `mdcore`. In Settings Logging General, set Syslog facility to `local0` and save. New syslog messages will appear as `local0.info`, `local0.warning`, etc.

What stays the same if I ignore it?

- Upgrades from older MetaDefender Core versions keep `daemon` automatically
- Settings export/import without this field still work — they default to `daemon`
- Log content, format, and log levels are unchanged; only the syslog category tag differs when you customize it

Log Rotation and Retention

Do I need to change this?

Most users can leave these settings at their defaults. MetaDefender Core already rotates and manages log files automatically. The default values match the behavior that has been in place for years:

- Rotate when the log file reaches 1 GB or 24 hours have passed (whichever comes first)
- Keep up to 30 rotated (compressed) log files

Adjust these settings only if you need to control disk usage more tightly, meet a retention policy, or align with your operations team's logging standards.

What is log rotation and retention?

MetaDefender Core writes diagnostic logs to a file on disk (the Log file path in Settings Logs). Over time that file grows. Log rotation automatically:

1. Closes the current log file when it gets too large or too old
2. Compresses it and starts a fresh log file
3. Deletes the oldest rotated files when the count exceeds your retention limit

Retention here means how many old rotated log files are kept — not how many days logs are stored. If you set Maximum rotated files to `10`, MetaDefender Core keeps the active log plus up to 10 compressed archives; older ones are removed.

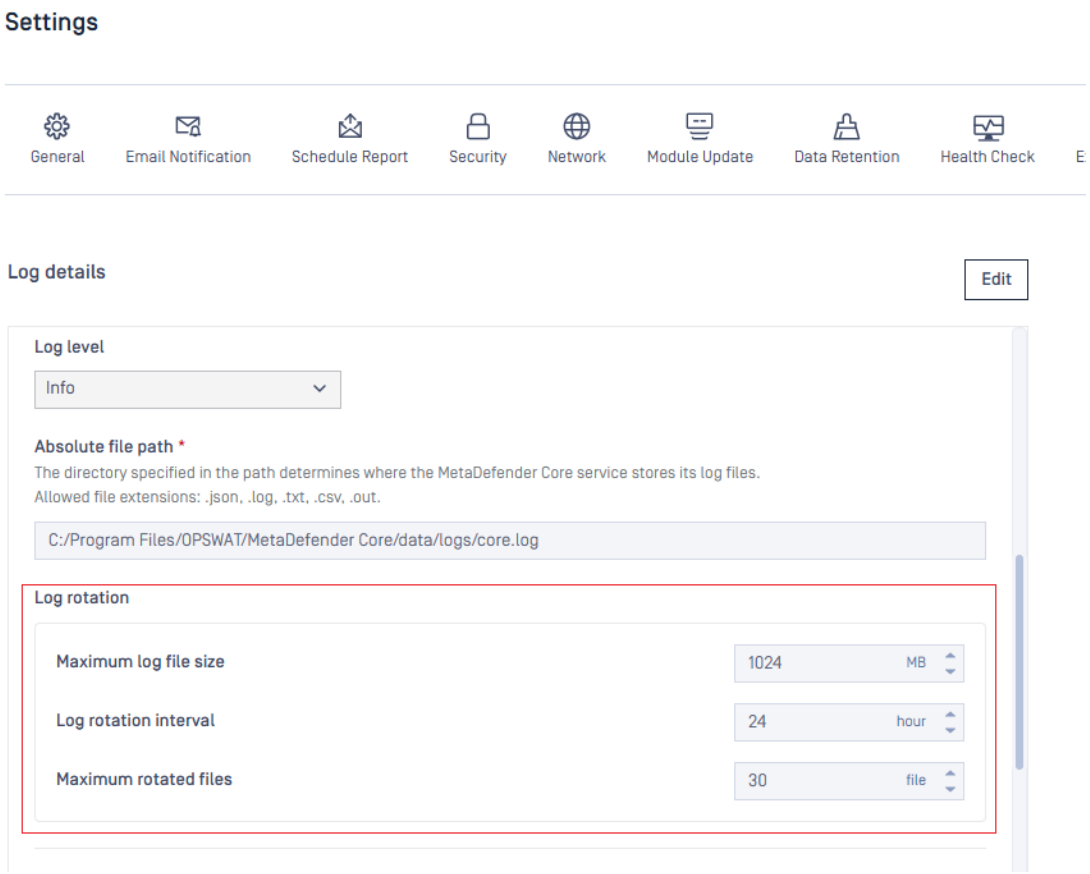
Where to find the settings

Settings Logs Log file (below Absolute file path)

Rotation runs when either the size limit or the time limit is reached — whichever happens first.

Setting	What it controls	Default	Allowed range
Maximum log file size (MB)	Rotate when the active log reaches this size	1024 (1 GB)	1–2048
Log rotation interval	Rotate when this many	24	1–48

Setting	What it controls	Default	Allowed range
(hours)	hours pass since the last rotation		
Maximum rotated files	How many compressed rotated files to keep	30	1-30



How it works in practice

Example with defaults (1024 MB / 24 hours / 30 files):

- MetaDefender Core writes to `ometascan.log`
- When the file hits 1 GB or 24 hours pass, Core compresses it (e.g. `core.log-20260628.gz`) and starts a new `ometascan.log`
- Up to 30 compressed files are kept; the oldest are deleted when a new rotation occurs

After you save changes:

- Changing only the three rotation/retention fields takes effect immediately — no MetaDefender Core restart is needed

- Changing the Absolute file path still requires a Core restart (same as before)

When should I customize it?

Consider changing these values when:

Goal	Example adjustment
Reduce disk usage	Lower Maximum log file size (e.g. 512 MB) or Maximum rotated files (e.g. 10)
Keep more history	Increase Maximum rotated files (up to 30)
Rotate more often	Lower Maximum log file size or Log rotation interval (e.g. 12 hours)
Compliance / ops policy	Set values your team specifies for file size, rotation frequency, or archive count

What these settings affect — and what they do not

Affected:

- MetaDefender Core main log file (the path configured under Log file)
- Engine logs — the Maximum rotated files value is applied when engine processes start (engine log cleanup on startup)

Not affected:

- NGINX web server logs
- OCM data-transporter logs
- Other log types outside the MetaDefender Core log file path

Upgrades and settings backup

- Upgrading from an older MetaDefender Core version automatically applies the default values (1024 / 24 / 30) if they were not previously set
- Importing an older settings backup without these fields also receives the same defaults
- Export/import of settings includes all three fields when present

My OPSWAT Central Management

On this session, we will introduce how to enroll MetaDefender Core (MD Core) into My OPSWAT Central Management (MOCM) version 10, configure MD Core on MOCM, and the settings related to the connection between MD Core and MOCM.

Supported modes

✅ **Standalone DB mode**: since MD Core v5.14.0.

Shared DB mode: starting from MD Core v5.18.0 with a few limitations.

Limitations when enrolling shared-mode MD Core with MOCM v10

Certificate Management Limitation

Per MOCM enrollment protocols, certificates synchronized from MOCM should remain non-deletable within MD Core. However, in certain shared-mode deployments where multiple MD Core instances are enrolled with MOCM, unenrolling a single instance may inadvertently allow those certificates to be deleted from the remaining enrolled MD Cores.

Group Configuration Limitation

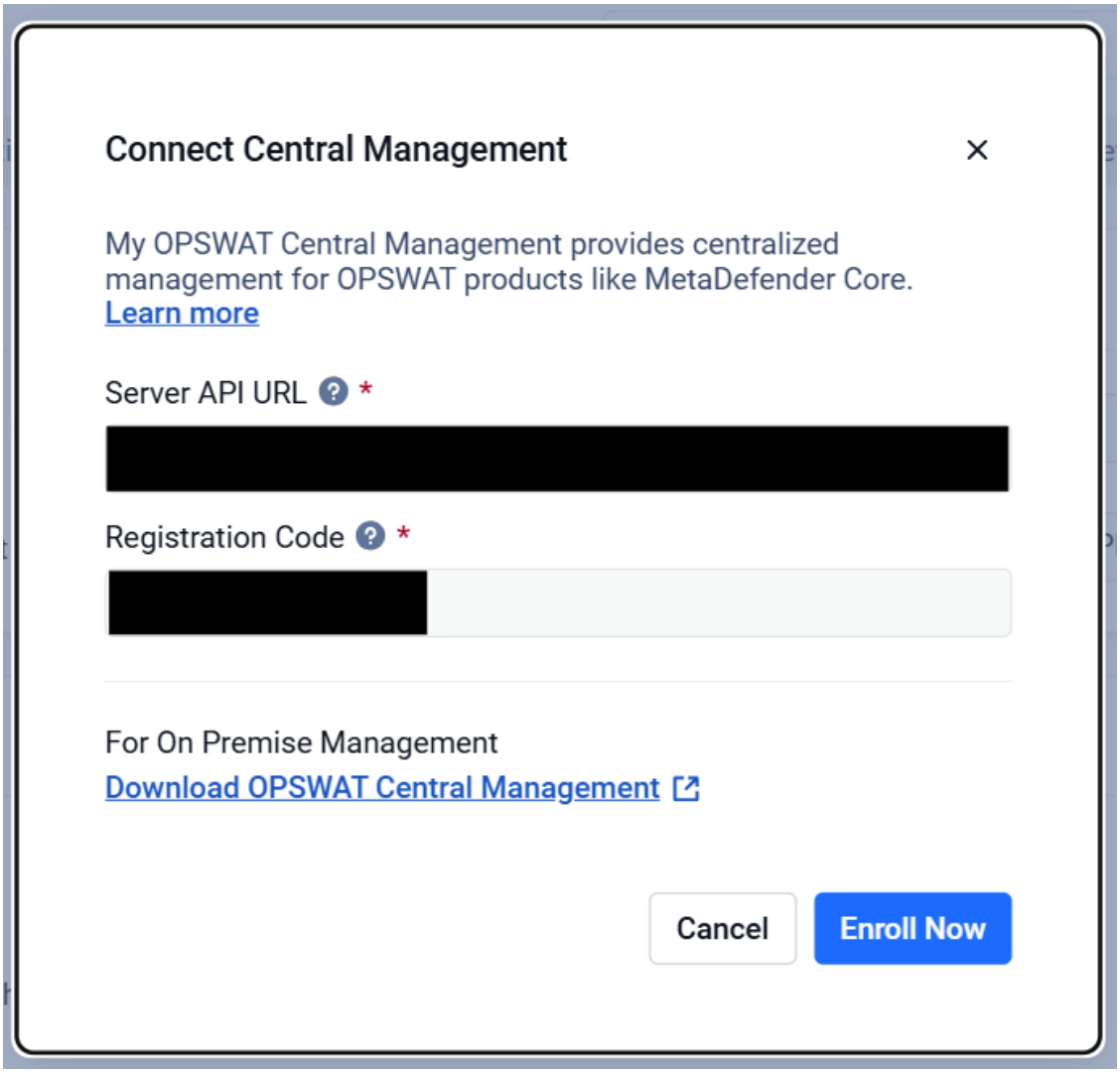
Shared-mode and standalone-mode MD Cores cannot coexist within the same MOCM Group Policy configuration.

Enrollment and MetaDefender Core Configuration on My OPSWAT Central Management

Enrollment

To enroll MetaDefender Core (MD Core) into My OPSWAT Central Management (MOCM), follow these steps:

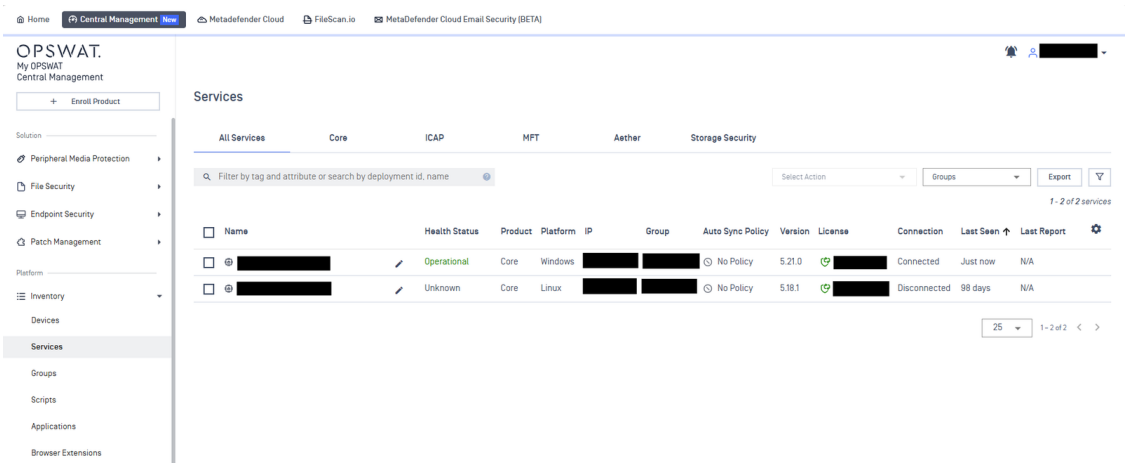
1. Obtain the `Server URL` and `Registration Code`. Please refer to this instruction for more details: Enroll other Products - My OPSWAT Central Management
2. Click the `Unmanaged` button, enter the `Server URL` and `Registration Code` then click `Enroll now.`

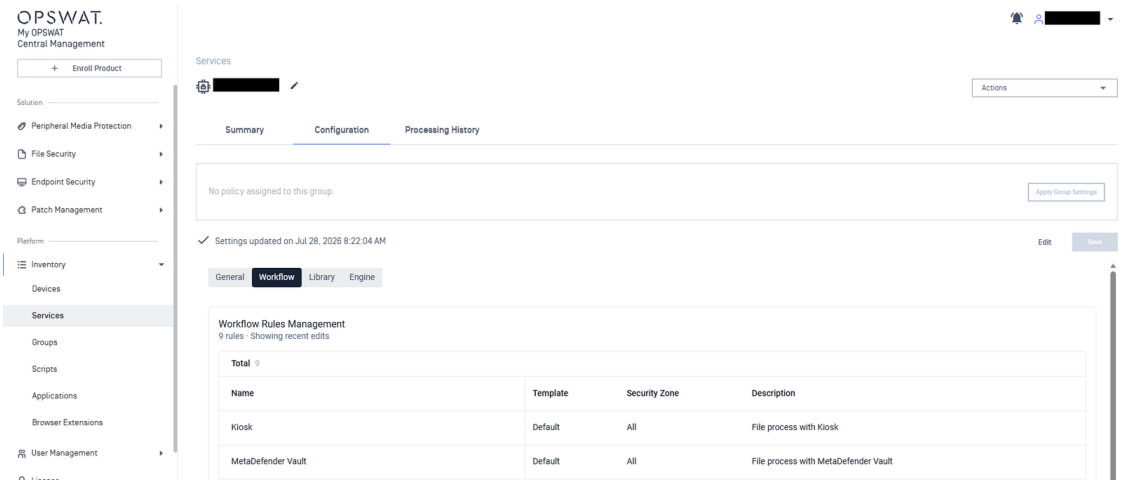


Configuring MetaDefender Core within My OPSWAT Central Management

Configuring each MetaDefender Core instance

To access the MD Core configuration in the MOCM console, navigate to **Inventory > Services** and select **<Instance Name> > Configuration**.





For more information about other functionalities in the **Inventory** menu of the MOCM console, please refer to the following link: Inventory - My OPSWAT Central Management

Configuring a group of MetaDefender Core instances

In addition to supporting the configuration of individual MD Core instances, MOCM also supports configuring a group of MD Core instances simultaneously. Please refer to this link for more details about group configuration in MOCM: Groups - My OPSWAT Central Management

What MetaDefender Core configurations are supported on My OPSWAT Central Management?

Legend

✅ Supported

◧ Partially supported

General

◧ Security

- Secure Connection

✅ Module Update

✅ Data Retention

✅ Health Check

✅ Central Management

Workflow

✅ Workflow Rules

✅ Workflow Templates

✅ Security Zones

Engines

- Configuration
- Lock/Unlock engine package
- Disable/Enable engine
- Delete engine

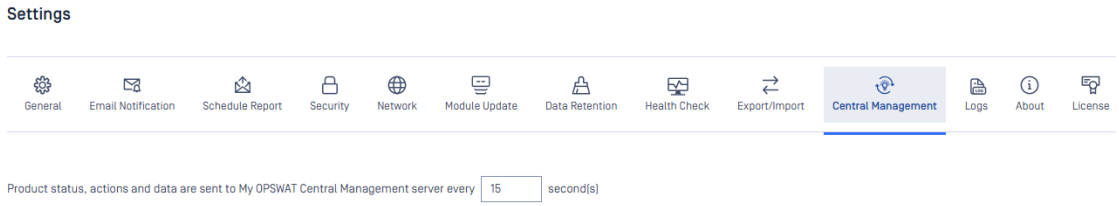
Library

- External Scanners
- Webhook Authentication
- Post Actions
- Password Storage

Settings for My OPSWAT Central Management

Synchronized configuration interval

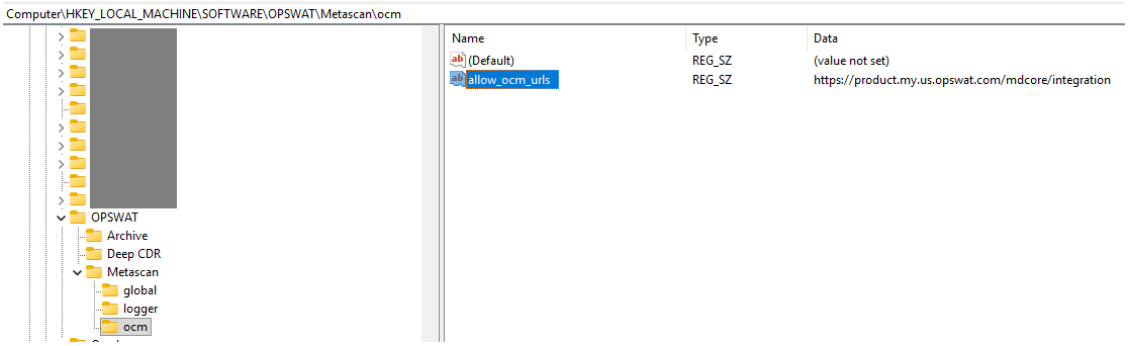
Administrator can adjust the synchronization interval between MetaDefender Core (MD Core) and My OPSWAT Central Management (MOCM) in order to reduce aggressive traffic to MOCM and lower resource usage on MD Core.



Enrollment URL Allowlist

To enhance security, the system administrator can configure an allowlist of enrollment URLs. Once configured, MD Core will allow enrollment in MOCM only through URLs that are included in the allowlist.

For example:



Please refer to: MetaDefender Configuration - OCM for more details about this setting.

NGINX Configuration

Info

Hardening guideline for Nginx web server are recommended by the vendor, and optional steps to MetaDefender Core product. Please only follow them when applicable.

Those guidelines are supported since MetaDefender Core version 4.19.0 or above.

Restrictions

Only allow access to our domain only

Code

```
if ($host !~ ^(metadefendercore.in|www.metadefendercore.in|images.metadefendercore.in)$ ) {
    return 444;
}
```

Deny certain user-agents

Blocking user-agents i.e. scanners, bots, and spammers who may be abusing your server.

Code

```
## Block download agents ##
if ($http_user_agent ~* LWP::Simple|BBBike|wget) {
    return 403;
}
##

## Block robots ##
if ($http_user_agent ~* msnbot|scrapbot) {
    return 403;
}
##
```

Block referral spam

Only direct access is allowed

Code

```
## Deny certain Referers ###
if ( $http_referer ~* (babes|forsale|girl|jewelry|love|nudit|organic|poker|porn|sex|teen) ) {
    return 403;
}
```

```
##
```

Block particular APIs

Code

```
## Block particular api ###
location ~ "^/admin/config/session$" {
  return 403;
}
##
```

How to configure

Step 1: Create a .conf file (create “built-in” folder if not existed)

- On Windows, under ***<Installation Directory>\nginx\built-in***
- On Linux, under **/etc/ometascan/nginx.d/built-in/**

Here is sample .conf file. Choose what meets to your scenario and update .conf file

Code

```
if ($host !~ ^(metadefendercore.in|www.metadefendercore.in|images.metadefendercore.in)$ ) {
    return 444;
}

if ($http_user_agent ~* LWP::Simple|BBBike|wget) {
    return 403;
}

if ($http_user_agent ~* msnbot|scrapbot) {
    return 403;
}

if ( $http_referer ~* (babes|forsale|girl|jewelry|love|nudit|organic|poker|porn|sex|teen) ) {
    return 403;
}
```

Step 2: A restart of the “OPSWAT MetaDefender Core” service is required.

OSCP - Online Certificate Status Protocol

1.) Modify “ssl.conf” file (create new if not existed)

- On Windows, under ***<Installation Directory>\nginx***
- On Linux, under **/etc/ometascan/nginx.d/**

Modify ssl.conf file with following recommended settings

Code

```
# Enable OCSP stapling, optimize session cache
ssl_ecdh_curve secp384r1;
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;

# verify chain of trust of OCSP response using Root CA and Intermediate certs
ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

# replace with the IP address of your resolver
resolver 127.0.0.1;
```

2.) A restart of the “OPSWAT MetaDefender Core” service is required.

API Rate Limiting

Warning

Since MetaDefender Core 5.13.0, a new setting for API rate limiting is officially supported in MetaDefender Core UI. Although it is still functional, setting the rate limit via `nginx_rate_limit.ini` file will be deprecated soon , and users are advised to follow `Settings` > `Security` > `Request Rate Limit` (Request Rate Limit) to set up the rate limit for the APIs mentioned on this page.

If both are used, the settings from the UI will be applied instead of those in the `ini` file.

By default MetaDefender Core does not have any hard limit on the number of API requests coming to Nginx web server. However, in order to secure more your MetaDefender Core server, users are supported to limit the number of API requests to better control their server load and prevent potential DOS (Deny of service) attack (this feature has been introduced since MetaDefender Core version 4.15.0).

This configuration support is applicable to two REST requests on MetaDefender Core:

- POST Login
- POST File Analysis

On Linux

1. Create file `nginx_rate_limit.ini`in the directory `/usr/lib/ometascan/`

Note

The configuration files should be readable for the user that runs MetaDefender Core service (On linux:

metascan, on Windows: service user).

2. Enter the following settings into the file:

nginx_rate_limit.ini · bash

```
max_scan_request=X;
max_login_request=Y;
```

Whereas X, Y > 0 (If X or Y is not valid then MetaDefener Core will ignore and remain unlimited as default behavior).

When these configurations are set, MetaDefender Core will allow users to send maximum X "/login" REST request per minute, and maximum Y "/file" REST request per minute.

3. Restart MetaDefender Core service (ometascan).

On Windows

1. Create file `nginx_rate_limit.ini` in the directory `<Installation Directory>\nginx`

Note

The configuration files should be readable for the user that runs MetaDefender Core service (On linux: metascan, on Windows: service user).

2. Enter the following settings into the file:

nginx_rate_limit.ini · bash

```
max_scan_request=X;
max_login_request=Y;
```

Whereas X, Y > 0 (If X or Y is not valid then MetaDefener Core will ignore and remain unlimited as default behavior).

When these configurations are set, MetaDefender Core will allow users to send maximum X "/login" REST request per minute, and maximum Y "/file" REST request per minute.

3. Restart MetaDefender Core service (ometascan).

How this feature works:

This feature fundamentally respects Nginx web server's rate limiting, learn more here: https://www.nginx.com/blog/rate-limiting-nginx/

For instance, users can set a limit for POST File Analysis by setting "max_scan_request" = 600, that means MetaDefender Core only allows serving maximum 600 file process requests per minute. However due to the fact that

NGINX mechanism tracks request at millisecond granularity, this limit means 1 request per 100 milliseconds, and thus users should not be able to send all 600 process requests at once (In this particular circumstance, every request coming after the allowed one will be rejected, and result in HTTP 503 response error code)

SSL Configurations

Warning

Starting from version 5.13.0, the `ssl on` directive is deprecated and can no longer be used for configuring SSL. Please utilize the Web UI console to configure HTTPS connections.

When upgrading from an older version to 5.13.0, MetaDefender Core will automatically migrate the SSL configuration to a new file named `ssl.conf.mdcore` to manage SSL settings. To configure SSL, please use this file but we recommend to use the Web UI console instead manually modify this file.

If the customer is already using `ssl.conf` for SSL configuration, there is no need to worry as it will be updated automatically during the upgrade.

Before upgrading, ensure that the file `ssl.conf` has permission set to `777`. After the upgrade, remember to adjust the file permissions accordingly.

Info

The `ssl.mdcore.conf` file can only use for 3 directives

- `ssl on`
- `ssl_certificate`
- `ssl_certificate_key`

`ssl_protocols` and `ssl_ciphers` (and other directives) MUST added in a conf file in `built-in` folder (please check **Advanced SSL configurations** section)

1.) Create a file called `ssl.conf.mdcore` (**MetaDefender Core only accept this name**)

- On Windows, under ***<Installation Directory>\nginx***

ssl.conf.mdcore

```
ssl on;
ssl_certificate "C:/Program Files/OPSWAT/Metadefender Core/nginx/your.crt";
ssl_certificate_key "C:/Program Files/OPSWAT/Metadefender Core/nginx/your.key";
```

- On Linux:
- Create folder `/etc/ometascan/nginx.d` if not existed, and make sure to grant read and write access for metascan user on this folder and all files inside.
- Create file `ssl.conf.mdcore` (MetaDefender Core only accept this name) in the directory `/etc/ometascan/ngin`

`x.d`

- Enter SSL-configuration:

ssl.conf.mdcore

```
ssl on;
ssl_certificate /etc/ometascan/nginx.d/your.crt;
ssl_certificate_key /etc/ometascan/nginx.d/your.key;
```

2.) A restart of the “OPSWAT Metadefender Core” service is required.

Advanced SSL configurations

Warning

For advanced SSL configurations like: TLS version and SSL ciphers, please create a conf file in built-in folder, for example:

- Windows: `<installation folder>/nginx/built-in/advanced_ssl.conf`
- Linux: `/etc/ometascan/nginx.d/built-in/advanced_ssl.conf`

1.) Explicitly allow specific TLS versions, optionally with preferred ciphers. For example:

built-in/advanced_ssl.conf

```
ssl_protocols tlsv1.2;
ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256;
```

2.) Use SSL private key and(or) certificate which is encrypted with a passphrase. Strongly recommended to put the passphrase file(s) into a secured vault where only MetaDefender Core can access.

A reference for typical practice: https://www.nginx.com/blog/protecting-ssl-private-keys-nginx-hashicorp-vault/

built-in/advanced_ssl.conf

```
ssl_password_file "C:/keys/private.pass";
```

For more SSL-options please consult Nginx documentation.

Warning

'\n' sequences in paths

Using the standard Windows path separator backslash " may give unexpected results if directory or file names start with 'n'. The reason is that the sequence '\n' is interpreted as a new line by nginx.

For example the following directive

`ssl_certificate "C:\Program Files\OPSWAT\Metadefender Core\nginx\your.crt";`

will appear at nginx as

`ssl_certificate "C:\Program Files\OPSWAT\Metadefender Core`

`ginx\your.crt";`

As a workaround instead of backslash " use

1. Forward slash '/' or

2. Double backslash ".

Note that certificate and key files are to provided by the user who can store them whenever it is convenient. Please adjust the paths accordingly.

Note: When choosing location for cert and key files, make sure the files are in a location which is readable to the service user.

Origin Client Source Identification

By default, when triggering scan from client through a load balancer or proxy server, MetaDefender Core's Nginx web server will not identify the original client source (but the load balancer or proxy server's address instead).

In order to force Nginx web server to trace back the origin client source:

1.) Create a "extra.conf" file

- On Windows, under ***<Installation Directory>\nginx***
- On Linux, under **/etc/ometascan/nginx.d/**

Code

```
set_real_ip_from 192.168.1.1;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

Make sure to change "192.168.1.1" to your actual load balancer or proxy server address.

In cases involving multiple load balancers, you can specify all the load balancer IP addresses in the `set_real_ip_from` directive.

Code

```
set_real_ip_from 192.168.1.1;
set_real_ip_from 192.168.1.2;
set_real_ip_from 192.168.1.3;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

If you use a virtual IP (VIP) for a load balancer cluster, you can specify that VIP instead.

Code

```
set_real_ip_from <VIP_Address>;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
```

2.) A restart of the “OPSWAT Metadefender Core” service is required.

Web Server Processing Statistics Query

Configure Nginx web server

1.) Create a “status.conf” file

- On Windows, under ***<Installation Directory>\nginx\built-in\status.conf***
- On Linux, under **/etc/ometascan/nginx.d/built-in/status.conf**

Code

```
location = /nginx_status {
  stub_status;
}
```

2.) A restart of the “OPSWAT Metadefender Core” service is required.

Query Nginx processing stats infomation

Property	Value
DESCRIPTION	Query Nginx processing stats information
URL	http://<server>:8008/nginx_status
HTTP METHOD	GET

Response

Result code	Value	Description
200	`Active connections: <n>``server accepts handled requests``<a1> <a2> <a3>``Reading: <b1> Writing: <b2> Waiting: <`	Request processed successfully. • Active connections <n> The current number of active

Result code	Value	Description
	`b3>` For example: `Active connections:``1``server accepts handled requests``144``144``358``Reading:``0``Writing:``1``Waiting:``0`	client connections including Waiting <b3> connections. • accepts <a1> The total number of accepted client connections. • handled <a2> The total number of handled connections. Generally, the parameter value is the same as accepts <a1> unless some resource limits have been reached (for example, the worker_connections limit). • requests <a3> The total number of client requests. • Reading <b1> The current number of connections where nginx is reading the request header. • Writing <b2> The current number of connections where nginx is writing the response back to the client. • Waiting <b3> The current number of idle client connections waiting for a request.

Password Policy

Password Policy settings are accessible under **Settings > Security** tab .

Info

These password policies changes only apply to new user creations and future password changes. Existing users' passwords are unaffected.

Local users' password can be enforced to meet requirements set by administrators, which includes following constraints:

- **Enforce password policy:**
  - Determines the number of unique new passwords that must be associated with a user account before an old password can be reused
  - Range: [0-24]
  - Default: 0 (to disable enforcement)
- **Minimum password length:**
  - The least number of characters that can make up a password for a user account
  - Range: [0-30]
  - Default: 0 (to disable enforcement)
- **Password must meet complexity requirements:**
  - Determines whether passwords must meet a series of guidelines that are considered important for a strong password.
  - Default: unchecked

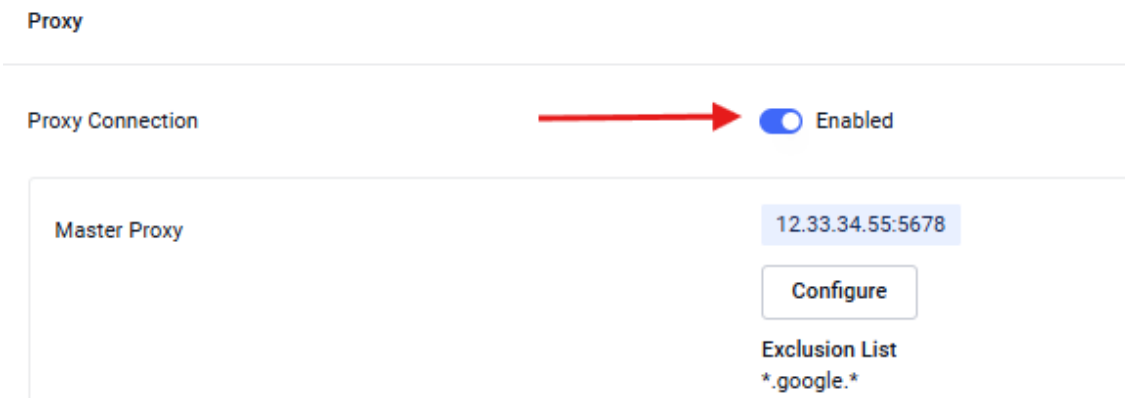
Complexity requirements

- At least 4 characters in length
- At least 1 uppercase letter of European languages (A through Z).
- At least 1 lowercase letter of European languages (a through z).
- At least 1 base 10 digits (0 through 9).
- At least 1 non-alphanumeric characters (special characters): (~!@#$%^&*_-+=`|(){}[]:;'"<>,.?/)



Proxy Configuration

Proxy configuration is possible via management console UI, and with proxy authentication support. MetaDefender Core no longer relies on system configurations to obtain the proxy setting, but it will control the thing itself.



Administrators can also test the proxy settings to verify connectivity to the updater and activation servers before applying changes.



Server configuration

Properties	Description	Option
SERVER	Proxy server host address	IP address / Domain all accepted
PORT	Proxy server port	Some default common ports: 3128 (Squid), 49152 (ProxySG)

User authentication

Properties	Description
USERNAME	Username for proxy authentication
PASSWORD	Password for proxy authentication

Hit "SAVE SETTINGS" button once configuration is done.

Exclusion List

Define a list of destination addresses to be excluded from going through pre-configured proxy server (use

semicolons ; to separate entries).

Applied to connections triggered by MetaDefender Core for following activities:

- Callback (webhook)
- Engine update
- Scan by download link
- Single Sign On integration (Fetch metadata from URL and login)
- Remote PostgreSQL database connection

Supported formats:

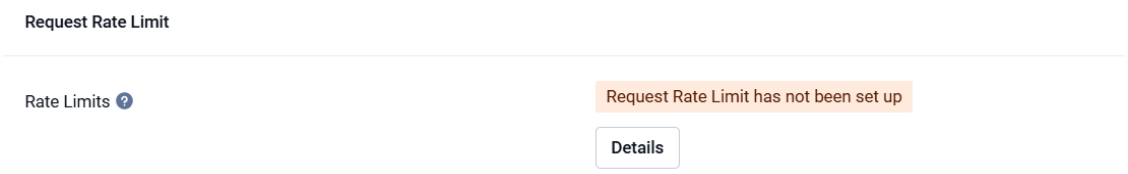
- Lowercase-only
- IP range (CIDR notation). For example: 192.168.203.1/16
- Valid IP address or host name. Wildcard character * is supported as prefix and/or suffix.
  - For example: 192.168.*; *.opswat.*

Request Rate Limit

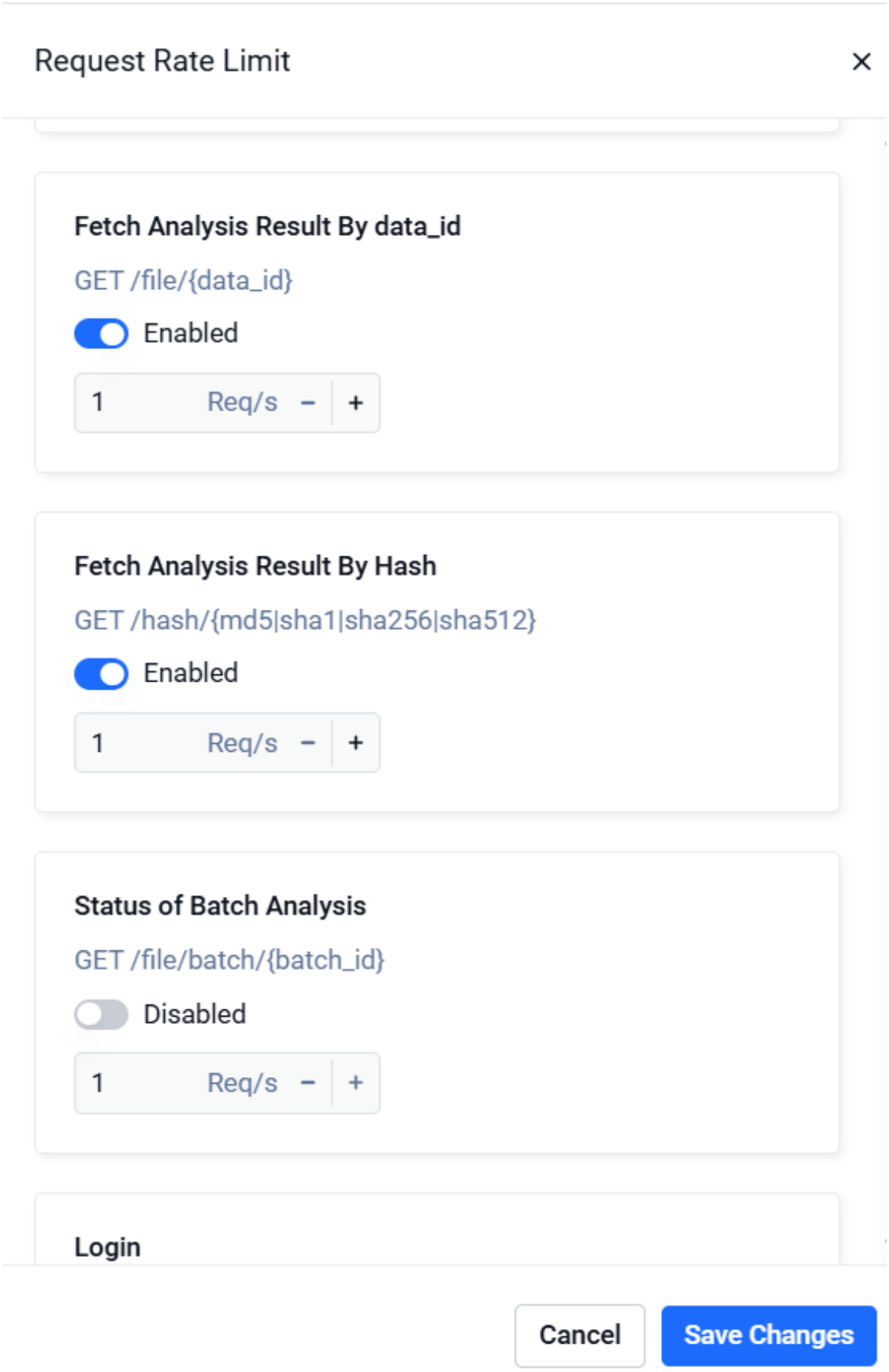
Since version 5.13.0, MetaDefender Core supports rate limiting for more API calls. This feature is not enabled by default.

To enable Request Rate Limit,

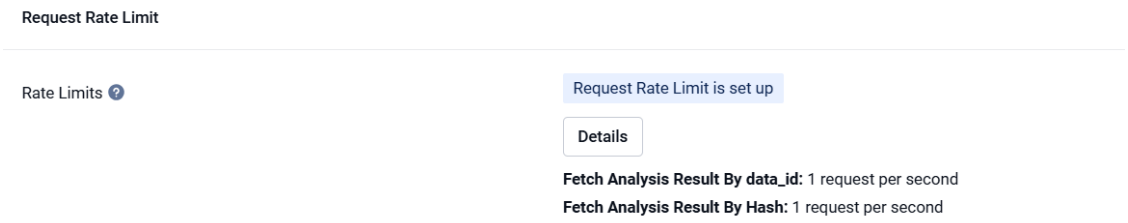
1. Go to **Settings** -> **Security** page



2. Hit **Details** to setup



3. Hit **Save** to complete. It takes MetaDefender Core a few seconds to apply the new settings.

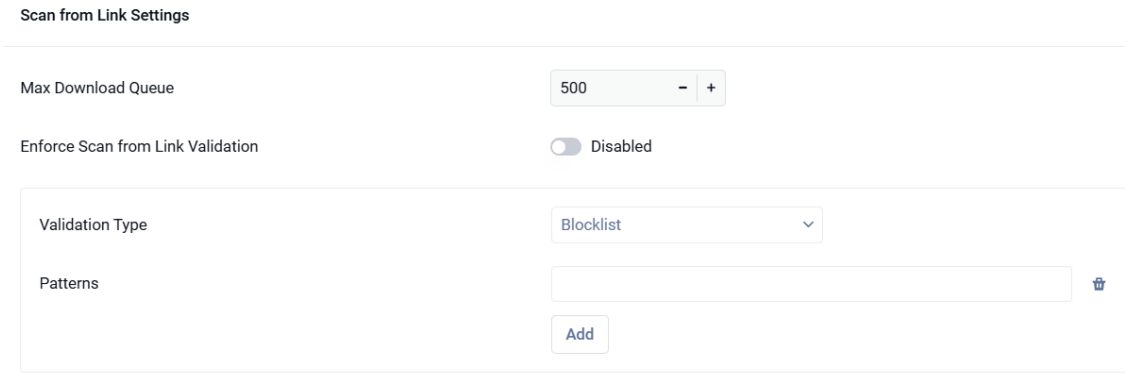


Warning

It may take MetaDefender Core up to 30 seconds to apply the settings. During this time, MetaDefender Core may not respond.

Scan-from-link Settings

Scan from link settings are accessible under **Settings > Security** tab .



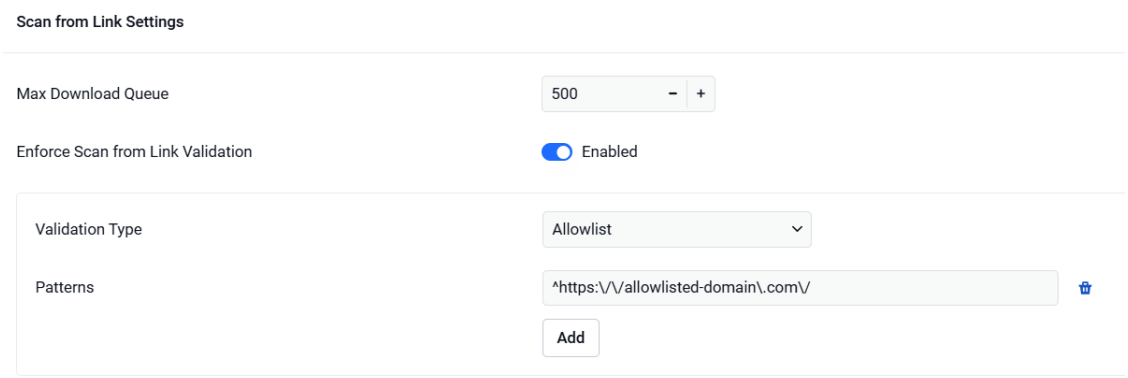
This setting is applied for the header `downloadfrom` when scanning file via API POST /file

- **Max download queue:**
  - Determines the number of total concurrent requests that can be downloaded at a time.
  - Range: [1-1000]
  - Default: 500
  - Info: If your system has poor internet speed, you should decrease this number or if you have a decent internet speed, you might want to increase the setting. This setting depends on your network.
  - Note: this setting only applies to total concurrent download requests, and it is different from the total scan queue in **Setting** -> **General** > **Queue size**
- **Enforce scan from link validation:**
  - Enable this setting will make Metadefender Core validate the links from "downloadfrom" header.
  - Default: disable
  - Validation type: Blocklist or Allowlist
  - Patterns: Regular Expression patterns

Info

For example, if we would like to validate the links in "downloadfrom" header, and ONLY allow for the site `https://allowlisted-domain.com` we should use this pattern:

`^https:\/\/allowlisted-domain\.com\/`



Enforce scan from link validate: use Allowlist Over Blocklist

We **strongly recommend** using an **allowlist approach** rather than a blocklist for the following reasons:

- **Positive security model**: Only explicitly permitted URLs are allowed
- **Reduced attack surface**: Unknown or new malicious domains are automatically blocked
- **Better maintainability**: Easier to manage a list of trusted sources

Configure Specific URL Patterns

To prevent URL manipulation attacks and bypass attempts, configure your `allowlist` patterns with maximum specificity:

Vulnerable Configuration:

`allowlisted-domain.com`

Secure Configuration:

`^https:\/\/allowlisted-domain\.com\/`

Why Specific Patterns Matter

Attackers can exploit loose patterns using various techniques:

- **Subdomain spoofing**: `https://allowlisted-domain.com.malicious-site.com/`
- **Path injection**: `https://evil-site.com/allowlisted-domain.com/malware.exe`
- **Query parameter manipulation**: `https://malicious-site.com/?redirect=allowlisted-domain.com`

Pattern Configuration Best Practices

1. **Always use anchors**: Start patterns with `^` and end with appropriate boundaries
2. **Specify protocol**: Include `https:\/\/` to enforce secure connections
3. **Escape special characters**: Use `\.` instead of `.` for literal dots
4. **Include path separators**: End with `\/` to prevent subdomain spoofing

Example Configurations

For a specific domain:

```
^https:\/\/downloads\.yourcompany\.com\/
```

For a subdirectory:

```
^https:\/\/cdn\.vendor\.com\/updates\/
```

Testing Your Configuration

After configuring your `allowlist` patterns, test with various URL formats to ensure they work as expected:

- Verify legitimate URLs are accepted
- **Confirm malicious variations are rejected**
- Test edge cases and potential bypass attempts

Following these guidelines will provide robust protection against URL-based attacks while maintaining legitimate functionality.

Schedule Report

The scheduled report will generate and deliver periodic updates tailored for the administrator. These reports are designed to provide comprehensive insights and relevant data, enabling informed decision-making and efficient management of ongoing tasks or operations.

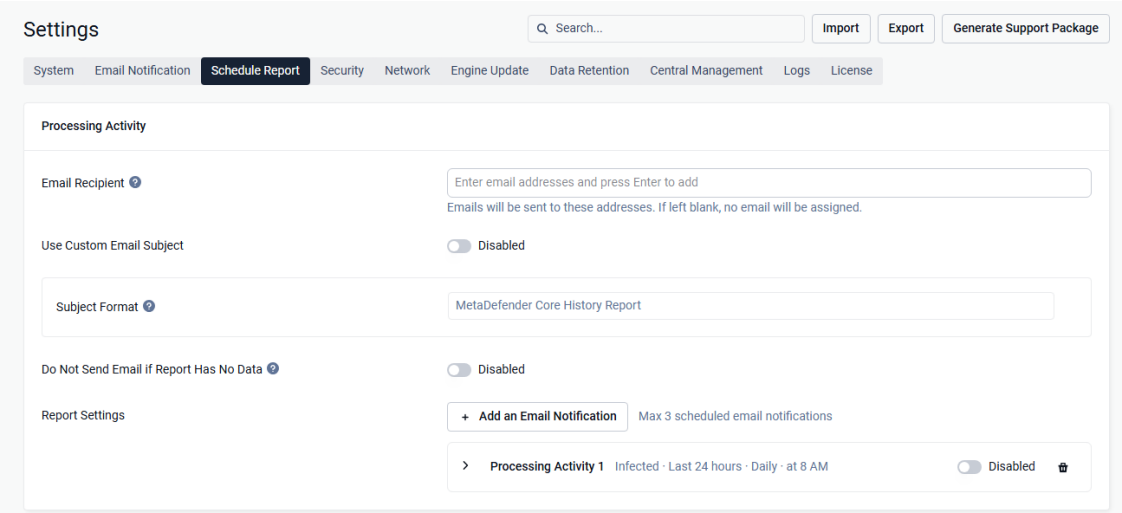
Processing Activity

The system generates a detailed report in CSV format, for file processing activities, offering comprehensive insights similar to the processing history. However, these reports are scheduled to be produced periodically and delivered automatically, ensuring consistent and timely updates without manual intervention.

- **Result**: Allows users to specify a predefined outcome to filter data when generating reports for processing activities. This ensures that only relevant information is included in the report.
- **Report Time Range**: Defines the specific time period for which the report will collect data. This helps in focusing on activities within a particular timeframe.
- **Schedule Time**: Configures the periodic schedule for sending reports. The following options are available for selection:
  - Daily

- Weekly
- Monthly
- Quarterly
- Onetime - send the report only one time

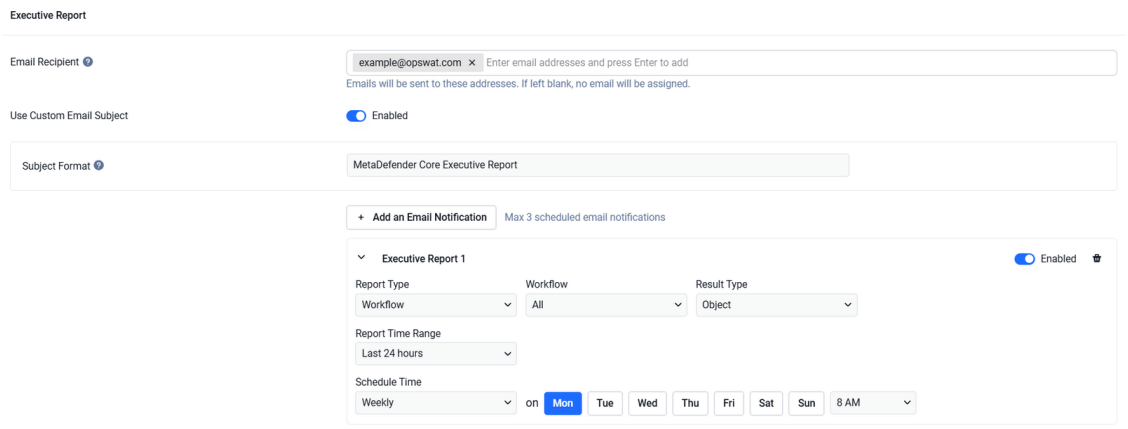
Tips: The users can enable the option "Do Not Send Email If Report Has No Data" to skip the reports that have no data.



Executive Report

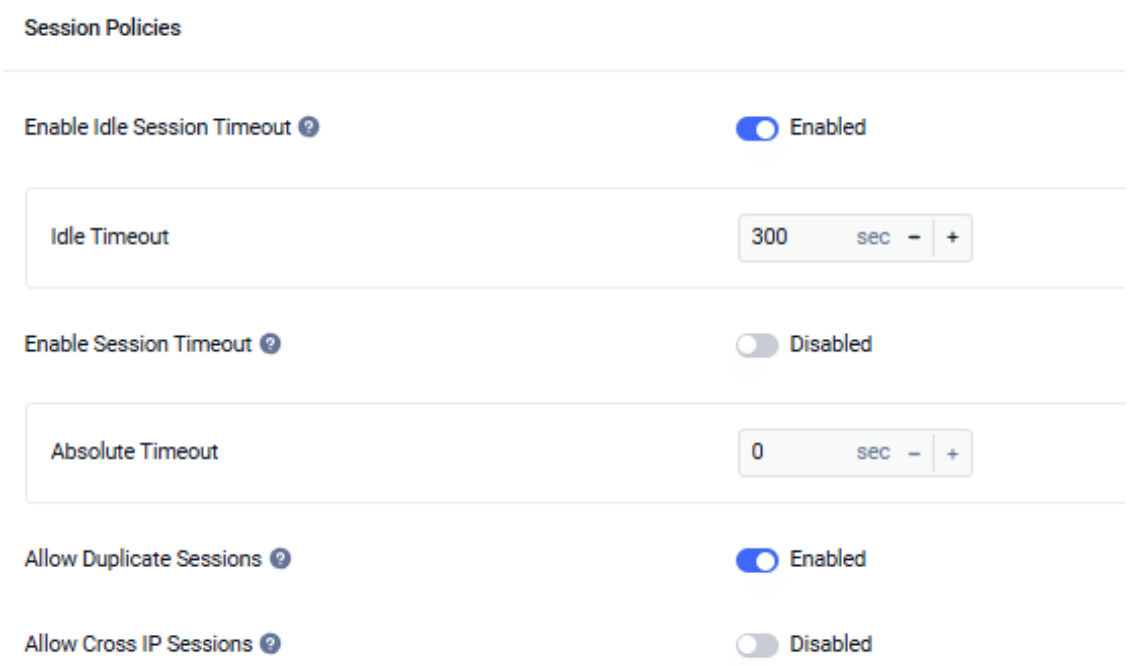
The system offers the ability to generate scheduled reports in PDF format, similar to the "Executive Report" available under the Dashboard. These reports are designed to provide a concise and professional summary of key metrics and insights, delivered automatically at predefined intervals.

- **Report type**: Enables users to select the type of report they want to generate. Options include filtering by `Workflow`, by `Users`, by `Sources`, or a combination of both `Users and Sources`.
- **Workflow:** Allows users to specify a particular workflow rule to focus the report on relevant processing activities.
- **Result Type**: Object or File
- **Report Time Range**: Defines the specific time period for which the report will collect data.
- **Schedule Time**: Configures the periodic schedule for sending reports. The following options are available for selection:
  - Daily
  - Weekly
  - Monthly
  - Quarterly
  - Onetime - send the report only one time



Session Policy

Session Policy settings are accessible under **Settings > Security** tab



- Idle session timeout: Idle timeout to invalidate individual user's session based on that user last activity.
- Session timeout: Absolute timeout to invalidate individual user's session regardless of that user activities.
- Allow Duplicate Sessions: Allow same user to have multiple active sessions.
- Allow Cross IP Sessions: Allow requests coming from sources different from the authenticated origin.

REST API for session timeout

Learn more at: PUT - Session settings

SSL Connection for bundled PostgreSQL

Note

The SSL connection configuration is possible thanks to the support made by PostgreSQL. For reference: https://www.postgresql.org/docs/14/ssl-tcp.html

MetaDefender Core is tested against this mode on particular version 4.21.1

Step 1: Generate SSL certificate for PostgreSQL server

Note

Following instructions are for self-sign certificate. For production environment, making sure to use the certificate signed by your trusted CA.

1) Generate a private key

Code · bash

```
openssl genrsa -aes256 -out server.key 2048
```

Sample output:

Code · bash

```
Generating RSA private key, 1024 bit long modulus ........++++++
..............++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
```

Type your desired password / pass phrase to encrypt the private key

2) Remove the pass phrase to automatically start up the PostgreSQL server

Code · bash

```
openssl rsa -in server.key -out server.key
```

Note

On Linux, make sure to set permission on the server.key file

Code · bash

```
chmod og-rwx server.key
```

3) Create a self-signed certificate

Code · bash

```
openssl req -new -key server.key -days 3650 -out server.crt -x509
```

You will be prompted to enter detailed information which is incorporated into your self-signed certificate request.

Code

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:CA
Locality Name (eg, city) [Default City]:San Francisco
Organization Name (eg, company) [Default Company Ltd]:OPSWAT
Unit Name (eg, section) []:OPSWAT
Common Name (eg, your name or your server's hostname) []:localhost
Email Address []:dummy_email@opswat.com
```

For self-signed certificate, use the server ceriticate as the trusted root certificate:

Code · bash

```
cp server.crt root.crt
```

Copy **server.key**, **server.crt**, **root.crt** to PostgreSQL data folder. For bundled local MetaDefender Core's PostgreSQL:

- Windows: **<Installation folder>\data\pg_data**
- Linux: **/var/lib/ometascan/pg_data/**

Step 2: Configure PostgreSQL server for SSL authentication connection

1) Create a custom config file (e.g. ssl.conf) for PostgreSQL.

For bundled local MetaDefender Core's PostgreSQL:

- Windows: **<Installation folder>\postgres\ssl.conf**
- Linux: **/usr/lib/ometascan/postgres/ssl.conf**

2) Content of ssl.conf:

Code

```
ssl = on
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_cert_file = '<abs_path_to_server.crt>'
```

```
ssl_key_file = '<abs_path_to_server.key>'
ssl_ca_file = '<<abs_path_to_root.crt>'
```

3) Modify pg_hba.conf

For bundled local MetaDefender Core's PostgreSQL:

- Windows: **<Installation folder>\data\pg_data\pg_hba.conf**
- Linux: **/var/lib/ometascan/pg_data/pg_hba.conf**

Add following (modify **::1/128** if needed to match with your environment requirement)

Code

```
hostssl all all      ::1/128      scram-sha-256
```

After the modify, the content of pg_hba.conf should be like this (example):

> **Note**
>
> In a Linux environment, a local and host connection is required to upgrade from version 5.5.1 or earlier to version 5.6.0 or later.

Code

```
# PostgreSQL Client Authentication Configuration File
# ===================================================
#
# Refer to the "Client Authentication" section in the PostgreSQL
# documentation for a complete description of this file.  A short
# synopsis follows.
#
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access.  Records take one of these forms:
#
# local         DATABASE  USER  METHOD  [OPTIONS]
# host          DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostssl       DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostnossl     DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostgssenc    DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostnogssenc  DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
local    all all                    password/scram-sha-256
host     all all      ::1/128       password/scram-sha-256
hostssl  all all      ::1/128       password/scram-sha-256

#More content.......
```

4) Reload the services

- Standalone DB mode: Restart **ometascan** service (MetaDefender Core service)
- Shared DB mode: Restart **ometascan-pg**/**ometascan-postgresql** service (MetaDefender Core PostgreSQL service) and then **ometascan** service (MetaDefender Core service)

Test your SSL database connection:

Code · bash

```
psql.exe "sslmode=require dbname=postgres user=postgres host=localhost port=5432"
Password for user postgres:
psql (12.7)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.
```

This message indicate that you are now using SSL connection:

Code · bash

```
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
```

Usage Report

MetaDefender Core Server optionally reports anonymous aggregated data about general usage to help OPSWAT improve its products.

Note: Participation in this program is optional. You may disable Usage Report at any time if you would prefer not to report any Aggregated Data.

Why is Usage Report collected?

Usage Report is used solely to help OPSWAT improve its products, including product performance, reliability, and the features customers rely on most. Understanding how features are used in aggregate allows OPSWAT to prioritize improvements and resolve issues faster. For full details, see the OPSWAT Terms of Service.

What is being collected?

OPSWAT takes privacy and security seriously. It tracks general, anonymized usage information, which may include:

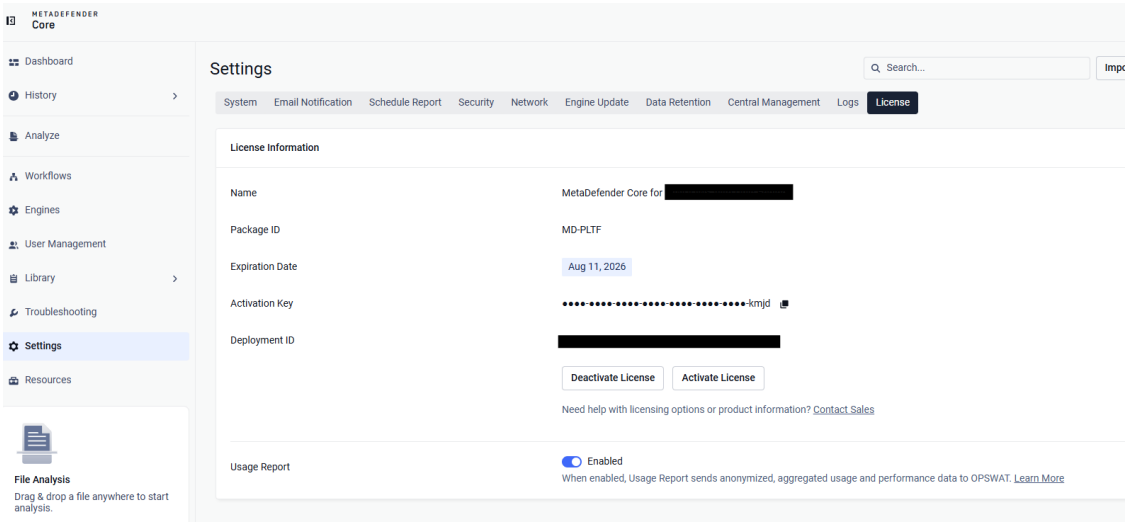
- Product and component version information
- Feature and configuration usage in aggregate
- General environment information (e.g., operating system, deployment type)
- Performance and reliability measures (e.g., processing throughput, error rates)

What about sensitive data?

Usage Report **does not** collect any information that could reasonably identify a person or entity, including but not limited to: credentials, file contents, scanned data, file paths, environment variables, or personally identifiable information

How do I disable Usage Report?

Setting > License > switch off Usage Report



Process files with MetaDefender Core

There are several ways to scan files with MetaDefender Core:

Process Files via REST API

The MetaDefender Core server also provides a REST API to interface with the application. To process a file even the user interface uses this API.

Raw binary file content must be packed in a single payload and streamed over to MetaDefender Core.

Chunked transfer encoding is not supported to upload files for processing.

All the responses from the server are in JSON format for easy parsing.

How to Interact with MetaDefender Core using REST

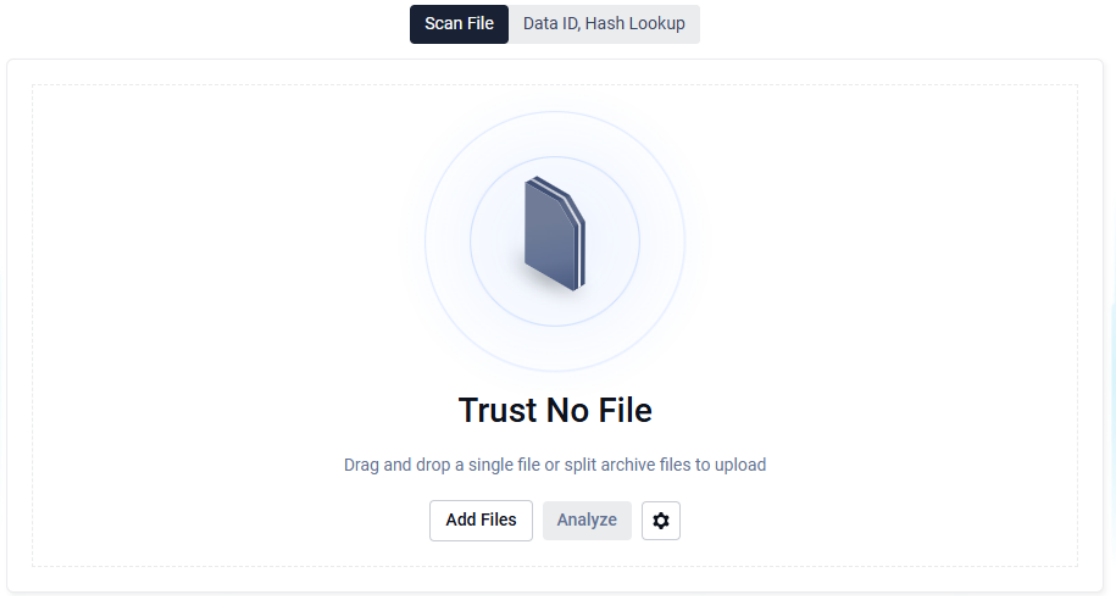
Beginning with MetaDefender Core 4.x, OPSWAT recommends using the JSON-based REST API. The available methods are documented (Full REST API documentation).

File analysis process

1. Upload a file to scan (`POST /file`), then receive data_id from response: (Scan File). Note: The performance depends on:
   - number of engines
   - type of file to be scanned
   - MetaDefender Core's hardware
2. Fetch the result with previously received data_id (`GET /file/{data_id}`) until scan result belonging to data_id doesn't reach the 100 percent progress_percentage: (GET Fetch Analysis Result). Note: Too many data_id requests can reduce performance. It is enough to just check every few hundred milliseconds.
3. Retrieve the scan results anytime after the scan is completed with hash for files (md5, sha1, sha256). (The hash can be found in the scan results) (GET Fetch Analysis Result)

Analyze Files via Web Interface

Once you open your browser and go to the MetaDefender Core server's URL the public file processing interface will be displayed.



Choose what to process and how

There are two option fields in the middle of the page. Next to them there is the **PROCESS FILE** button. With the leftmost option you can select between the available workflows for the public file processing. These workflows are determined by the MetaDefender Core administrators, so it is possible that only one workflow will be available for public scanning, or even none.

The next option is where you choose the file to scan. Click on the **SELECT A FILE** button and browse to the file to be scanned.

Dashboard

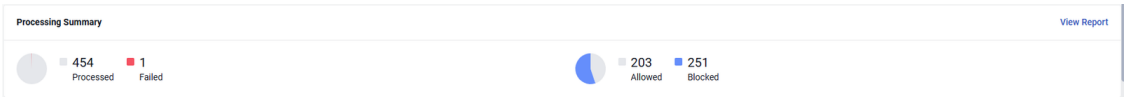
MetaDefender Core provides a Web-based user interface (default port is 8008) that gives a general overview of MetaDefender Core status and allows you to configure its options.

Security page

The Security page shows information on

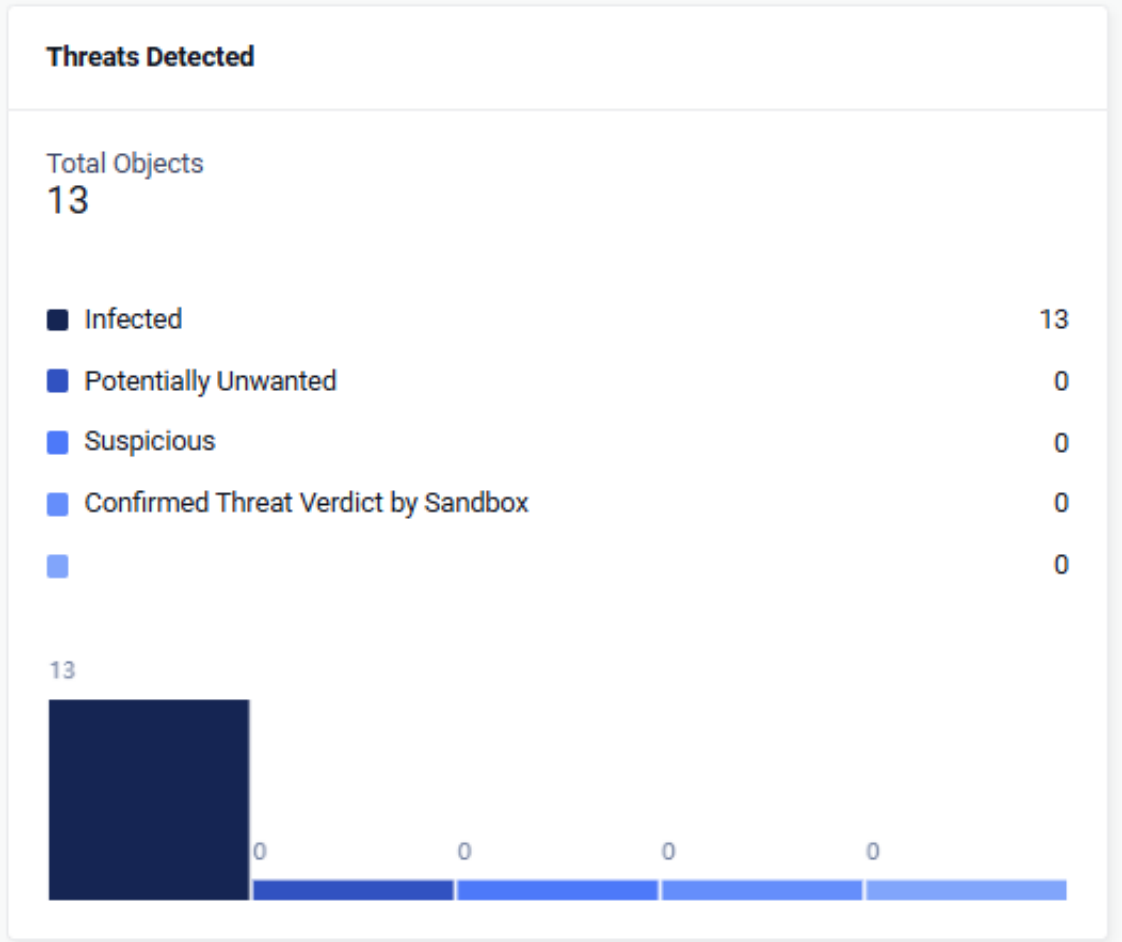
Processing Summary

Displays a summary for the selected instance and time range: total files scanned, allowed, blocked, and failed.

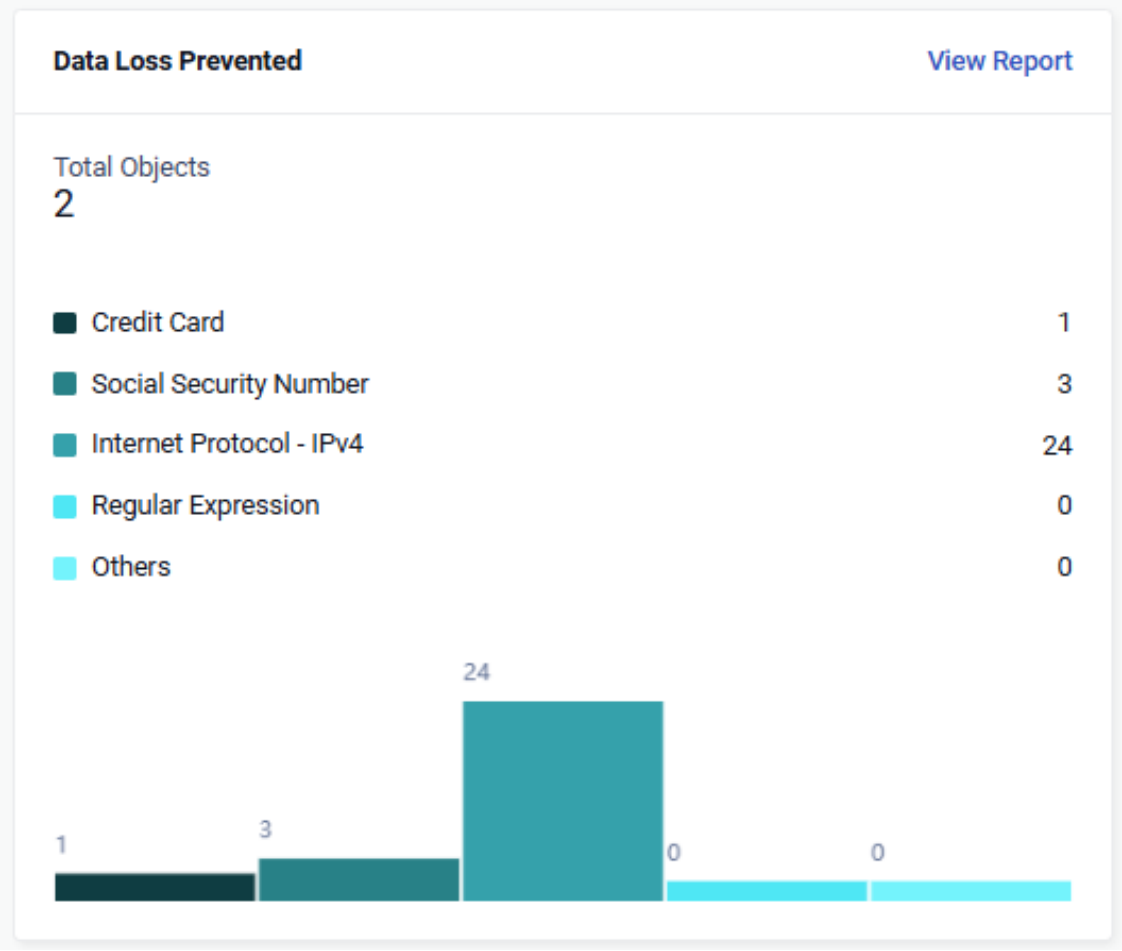


Threats Detected

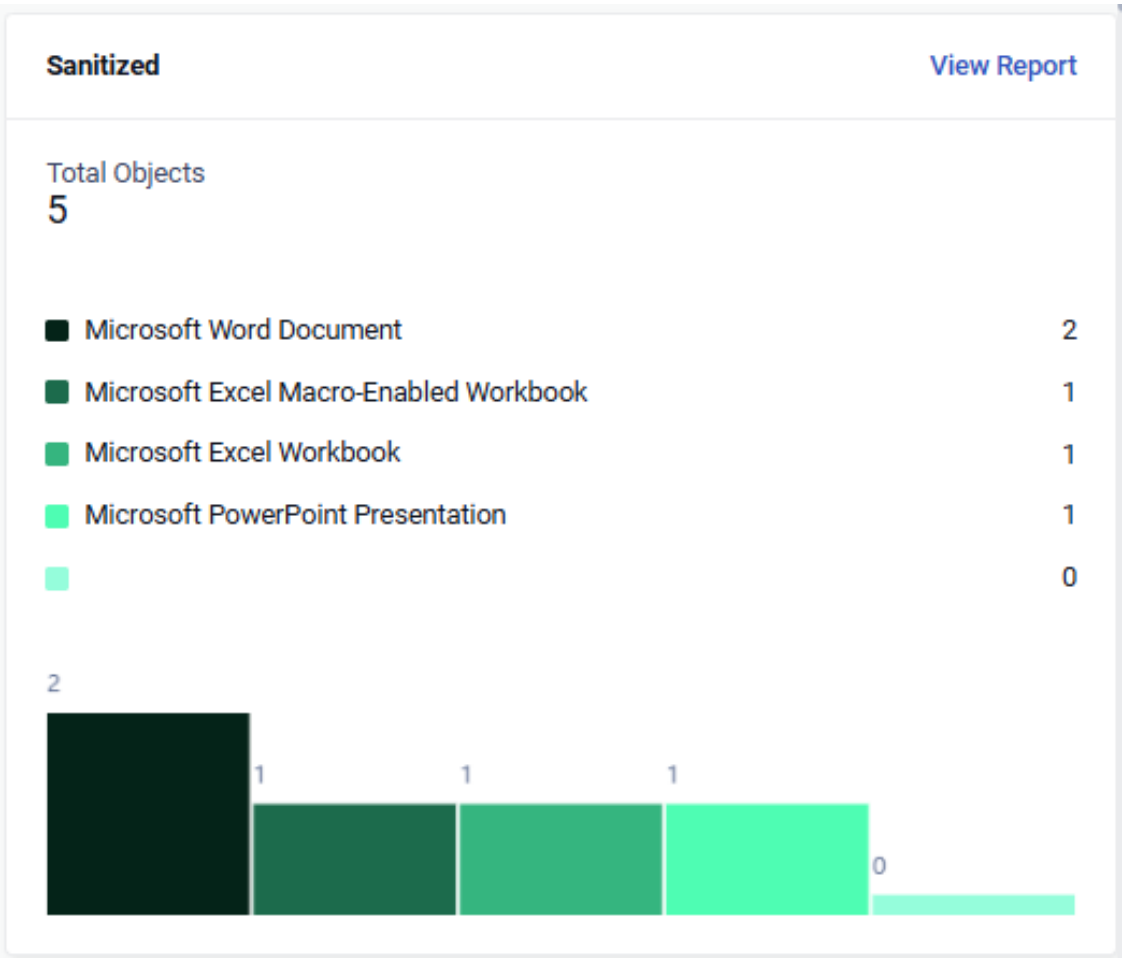
Displays the top threat verdicts detected and top file-blocking verdicts within the selected time range.



Data Loss Prevented

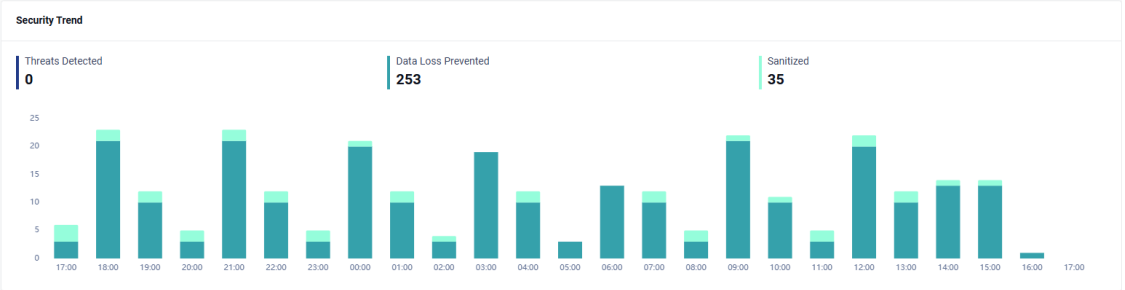


Number of Sanitized

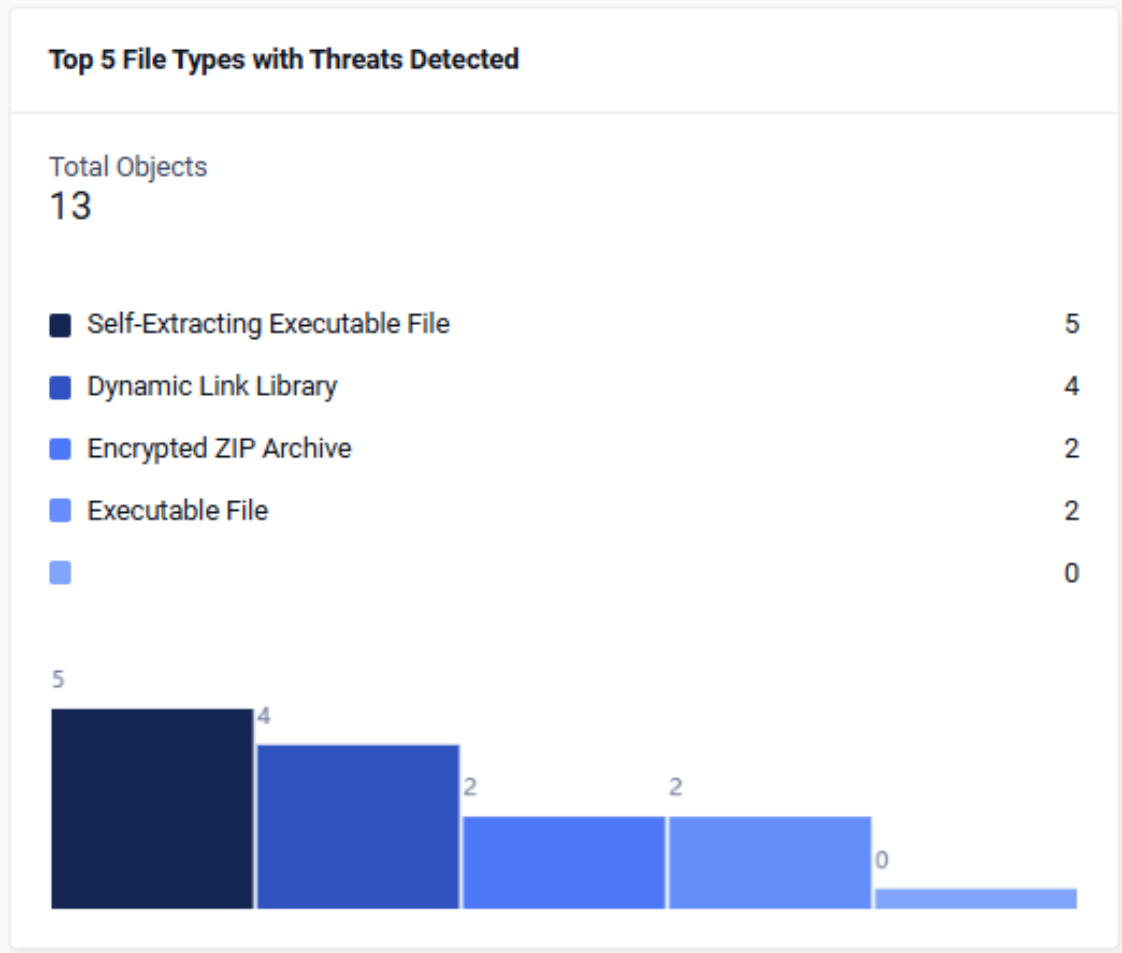


Security Trend

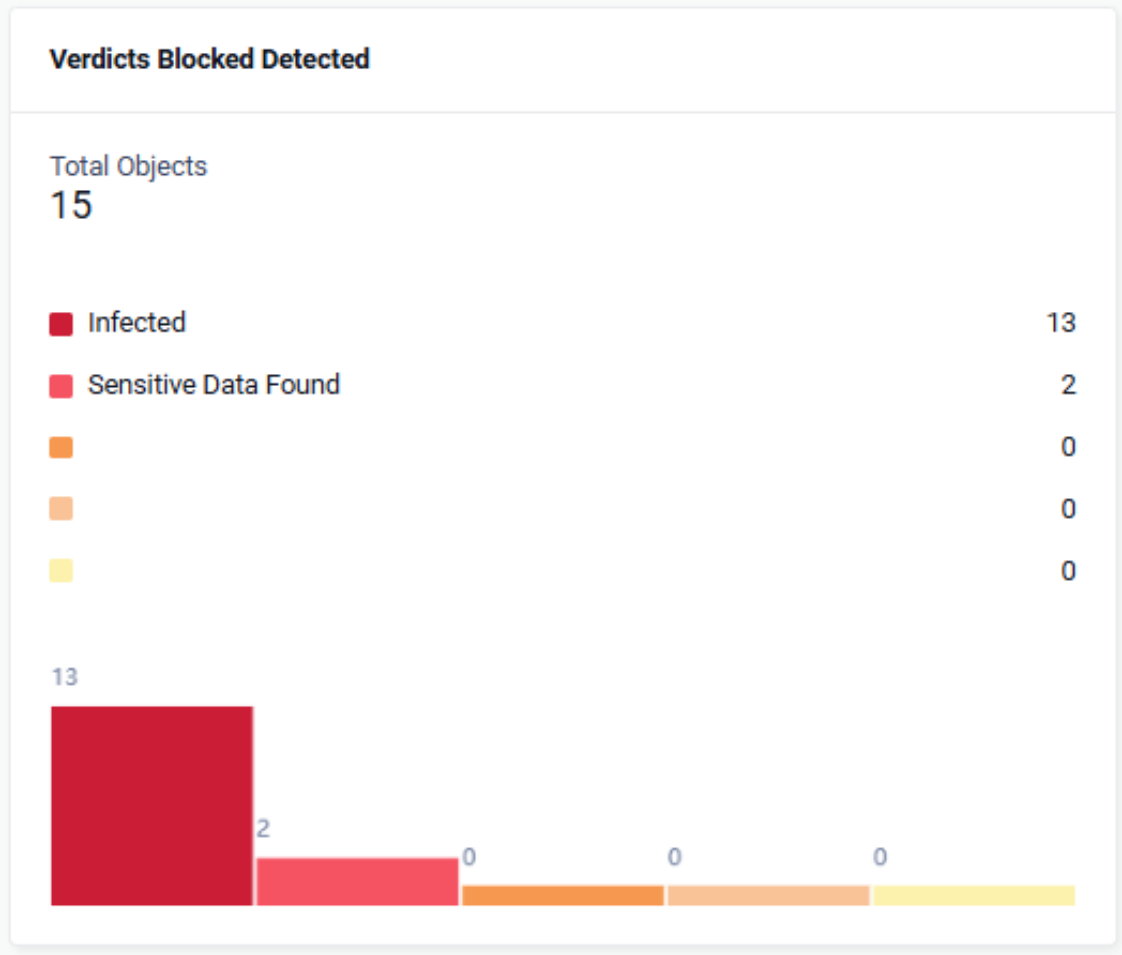
Displays Security Trend charts — threats detected, DLP hits, and sanitized files over time — for the selected range, grouped by hourly, daily, weekly, or monthly time buckets.



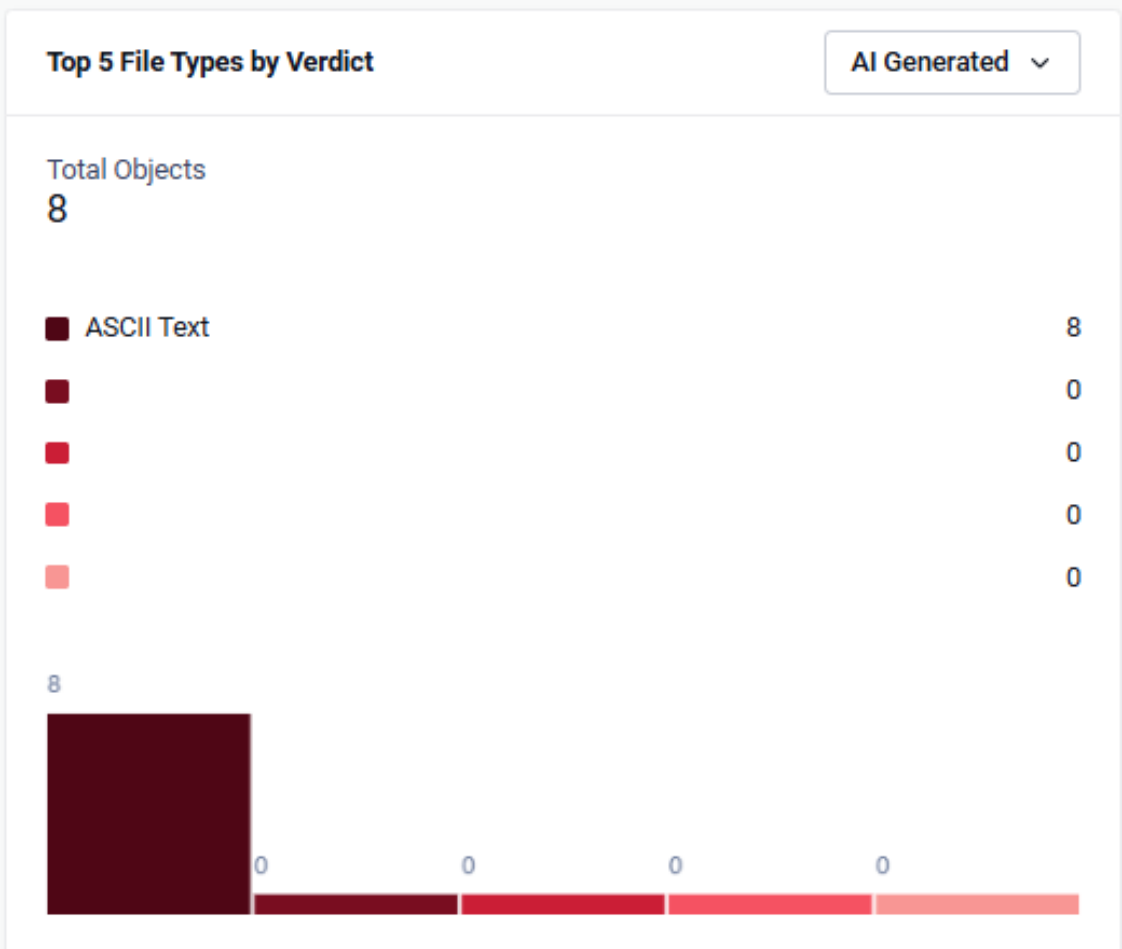
Top 5 File Types with Threats Detected



Verdicts Blocked Detected

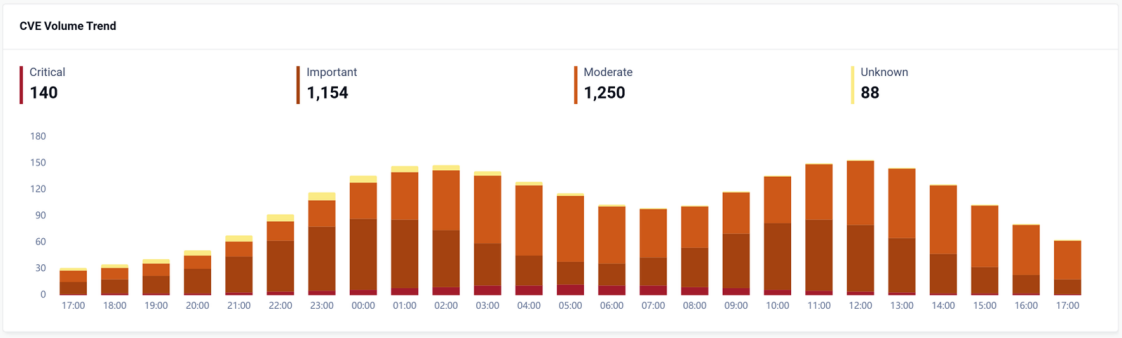


Top 5 File Types by Verdict



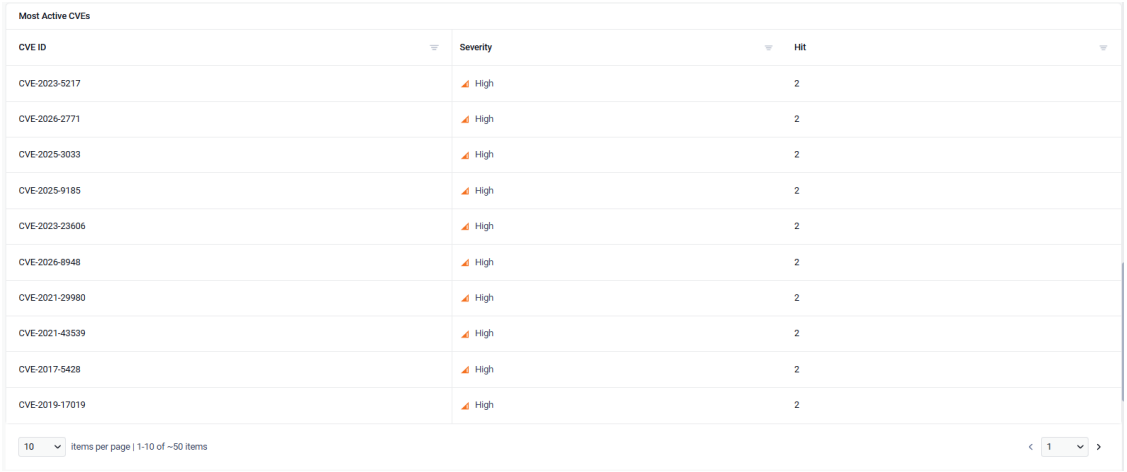
CVE Volume Trend

Displays CVE detections over time, broken down by severity (Critical, Important, Moderate, Low, or Unknown) for the selected time range and bucket size.



Most Active CVEs

Displays the most frequently detected CVEs for the selected time range, along with their severity and number of detections



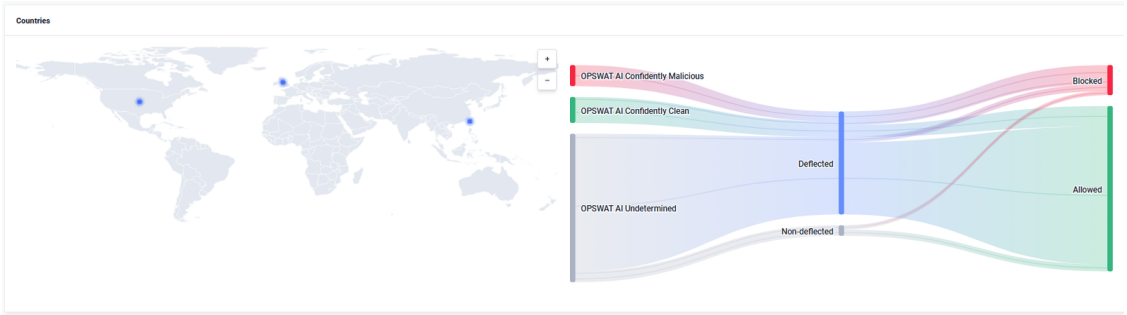
Most Active CVEs

CVE ID	Severity	Hit
CVE-2023-5217	High	2
CVE-2026-2771	High	2
CVE-2025-3033	High	2
CVE-2025-9185	High	2
CVE-2023-23606	High	2
CVE-2026-8948	High	2
CVE-2021-29980	High	2
CVE-2021-43539	High	2
CVE-2017-5428	High	2
CVE-2019-17019	High	2

10 items per page | 1-10 of ~50 items

1

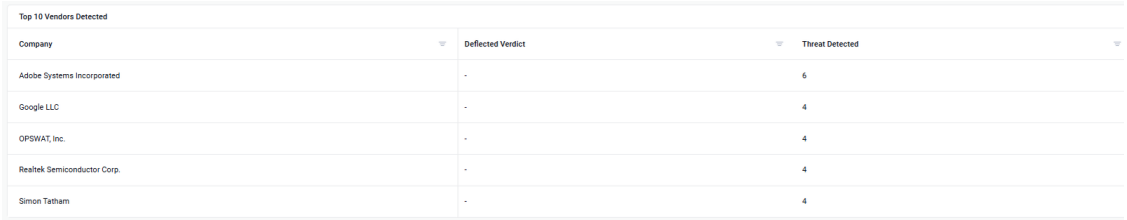
Top 10 Countries Detected



Top 10 Countries Detected

Country	Deflected Verdict	Threat Detected
-	Unknown	174
-	Clean	23
-	Malicious	17
United States	-	14
Chinese Taipei	-	4
United Kingdom	-	4

Top 10 Vendors Detected



Top 10 Vendors Detected

Company	Deflected Verdict	Threat Detected
Adobe Systems Incorporated	-	6
Google LLC	-	4
OPSWAT, Inc.	-	4
Realtek Semiconductor Corp.	-	4
Simon Tatham	-	4

System page

The System page page shows information on

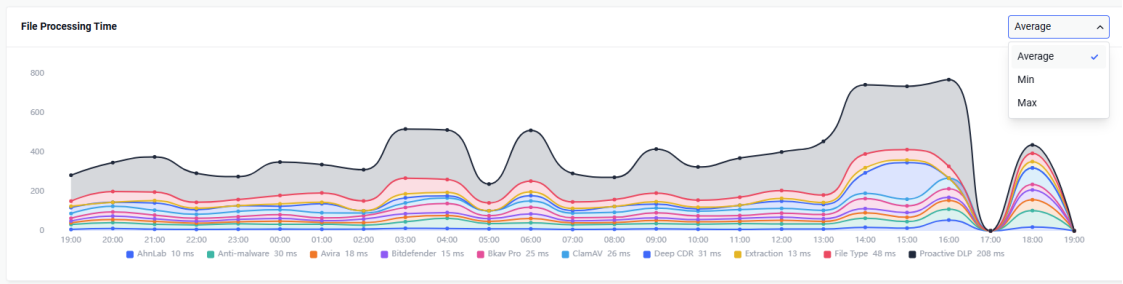
File Submission

Displays the number of submitted and blocked files over time for the selected time range, processing rule, and bucket size (hourly, daily, or monthly), counted either per submission or per object

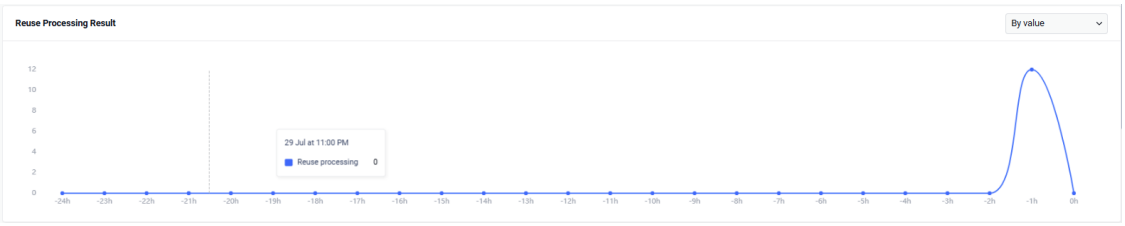


File Processing Time

Displays the File Processing Time widget, showing how long each engine spent processing files over time (in milliseconds) for the selected time range and bucket size (hourly, daily, or monthly), viewable as an average or total



Reuse Processing Result

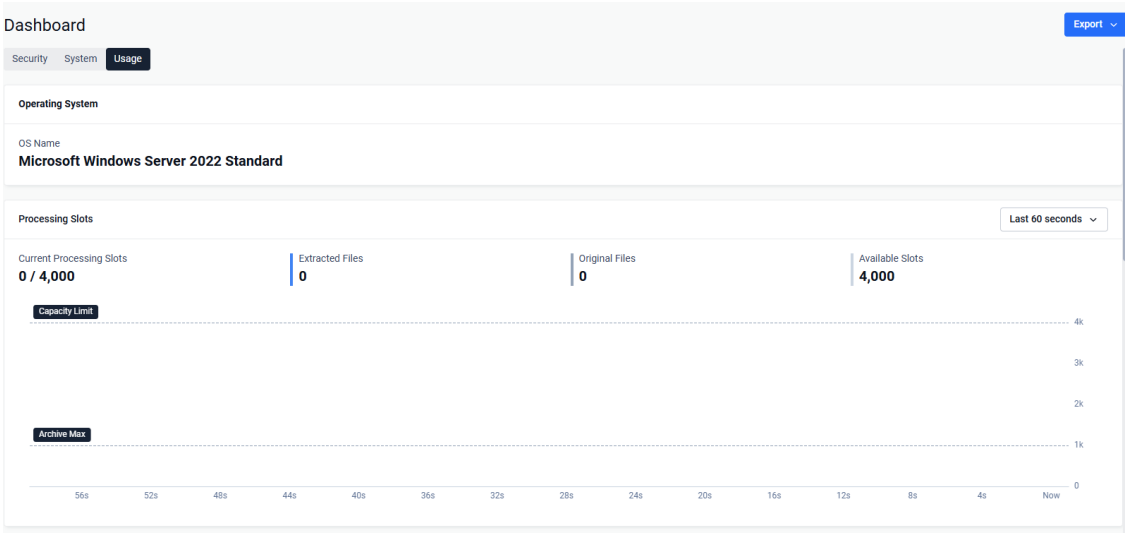


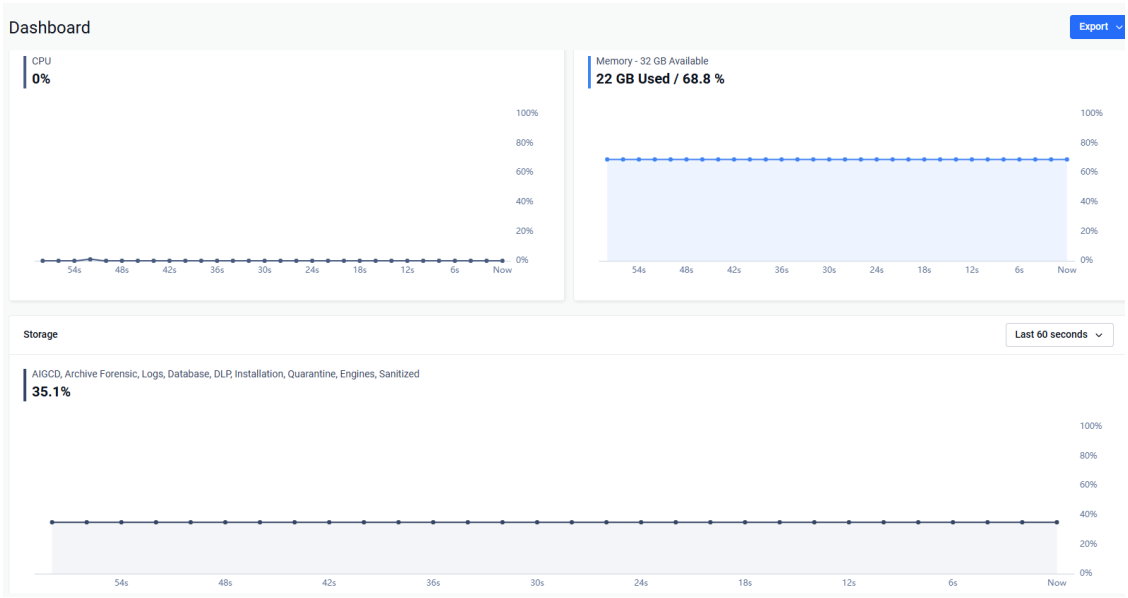
Total Objects Scanned



Usage page

The usage page is to provide the current system resource status (CPU, memory and disk) and processing performance (scan queue) of MetaDefender Core server.





Quarantine

The Quarantine page shows all scanned files which are copied to the quarantine. Each of them can be pinned to avoid removal on cleaning up. Also comments can be written to each quarantined file. Quarantine log can be searched for comment, file name and source of the scan request.

Update history

The Update history shows information on every update package related event.

On the Update history page you can also search for engine name, package type or message content. Also you can filter the list for severity.

Configuration history

The Configuration history shows information on user activity events regarding workflow rule changes, setting changes, engine setting changes, licensing.

You can search for text on the Configuration history page and filter the list for datetime.

Average processing time

Overview

The File Processing Time chart on the MetaDefender Core dashboard (Dashboard System tab) shows how long each processing engine takes to handle files over the selected time period. Use it to compare engines, spot slow ones, and follow processing-time trends across your deployment.

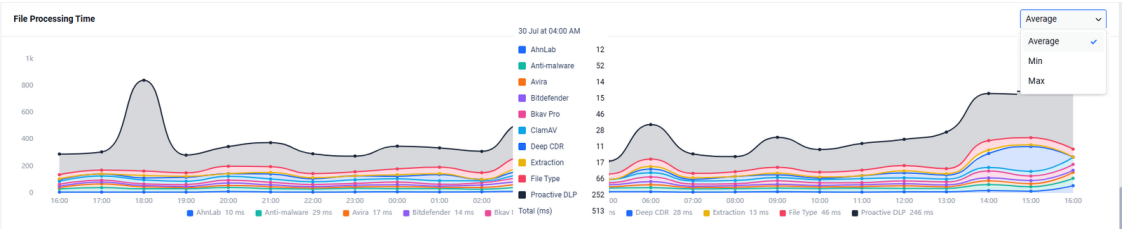


Chart elements

Element	Explanations
Metric selector	Drop-down at the top-right of the chart. Choose which value to plot: Average, Min, or Max processing time.
Engine series	One coloured line (with a shaded area) per processing engine — for example the av engines (AhnLab, Avira, ClamAV, ...), Deep CDR, Proactive DLP, File Type, and Extraction. Each line is that engine's processing time over time.
Legend	Below the chart. Lists every engine shown, its colour, and its value for the selected period.
Vertical axis	Processing time, in milliseconds (ms).
Horizontal axis	Time. Points are grouped by hour, day, or month to match the dashboard time range you select.
Tooltip	Hover over any point to see each engine's processing time at that moment.

Metric Values

Definition	Explanations
Average	The arithmetic mean of processing times for all files during the reporting period.
Max	The longest time any single file spent in a particular processing stage during the reporting period.
Min	The shortest time any file spent in a particular processing stage during the reporting period.

How to calculate

Average = Total processing time ÷ Number of files processed

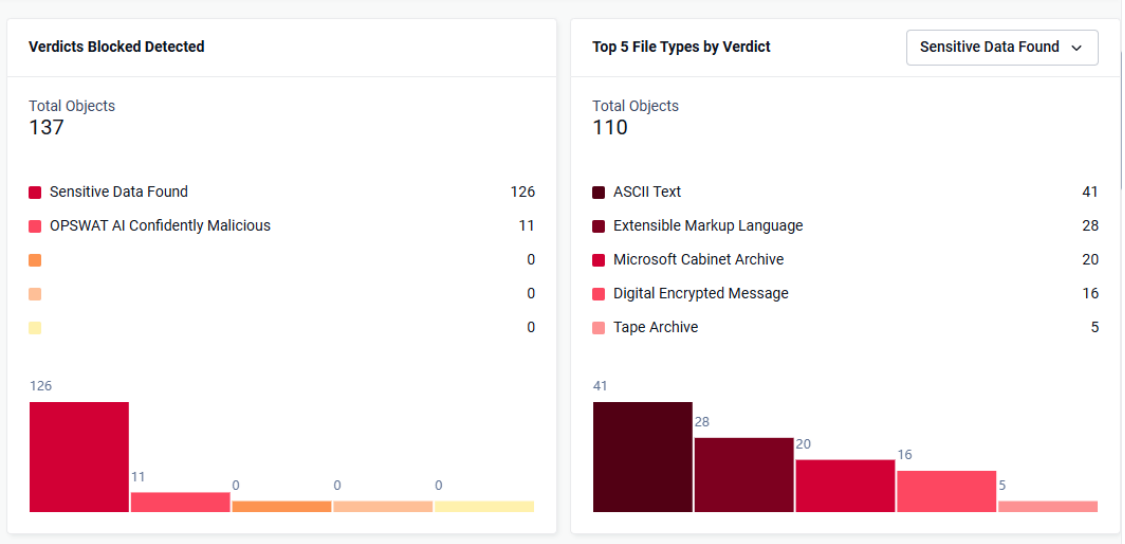
Example — an engine processes three files with processing times of 5 ms, 10 ms, and 12 ms:

- Average = (5 + 10 + 12) / 3 = 9 ms
- Max = 12 ms
- Min = 5 ms

Understanding Block Reason Statistics

Overview

When files are blocked in the OPSWAT MetaDefender Core system, they can be blocked for multiple reasons simultaneously (e.g., a file might be both "Infected" and contain "Sensitive Data Found"). To make statistics clearer and more actionable, our system reports each blocked file under only one block reason in statistical reports - the highest priority reason.



How Block Reasons Work

Files with Multiple Block Reasons

When a file is scanned, it may trigger multiple block verdicts, for example:

- Malware detection ("Infected")
- Sensitive Data Found
- File type restrictions (Blocklisted by File Type)
- Size limitations (Blocklisted by File Size)
- Failed sanitization
- And many others

All applicable block reasons are recorded for each file in the detailed scan results. However, for statistical reporting, only the highest priority reason is counted.

Archive Files

Archive files (ZIP, RAR, etc.) present a special case:

- An archive itself might not directly trigger any block reasons
- However, if any file within the archive is blocked, the archive inherits block reasons from its contents
- If multiple files within an archive are blocked for different reasons, the archive will have multiple block reasons

Block Reason Priority

When counting files for statistical purposes, we use a priority-based system where each file is counted only once under its highest priority block reason.

The general priority order (from highest to lowest) is:

1. Blocklist configure in workflow rule: Block by file type, file size, etc.
2. Security threats (Infected, Suspicious).
3. Policy violations (Sensitive Data Found, Country of Origin).
4. Processing issues (Failed Sanitization, Password Protected).
5. Workflow rule configuration (File Size Exceeded, Timeout).

Example: Understanding Block Statistics

If our system reports:

- 5 files blocked due to "Infected"
- 4 files blocked due to "Blocklisted by File Size"
- 1 file blocked due to "Sensitive Data Found"

This means:

- 5 files had "Infected" as their highest priority block reason (they may have had other reasons too)
- 4 files had "Blocklisted by file size" as their highest priority reason
- 1 file had "Sensitive Information Found" as its highest priority reason

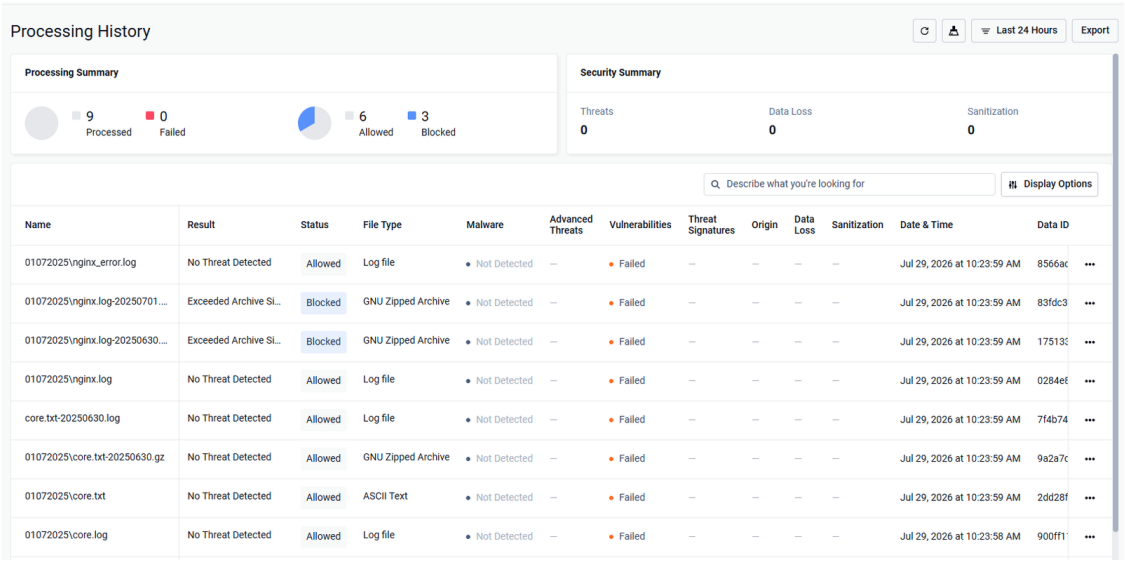
Important Notes

1. **Complete details are available**: While statistics show only the highest priority reason, complete block information for each file is available in detailed scan results.
2. **Archive handling**: An archive containing multiple blocked files is counted as a single blocked file in statistics, using the highest priority block reason among all its contents.
3. **Accuracy**: This prioritization system ensures each file is counted exactly once in statistics, preventing inflated numbers while highlighting the most critical issues.

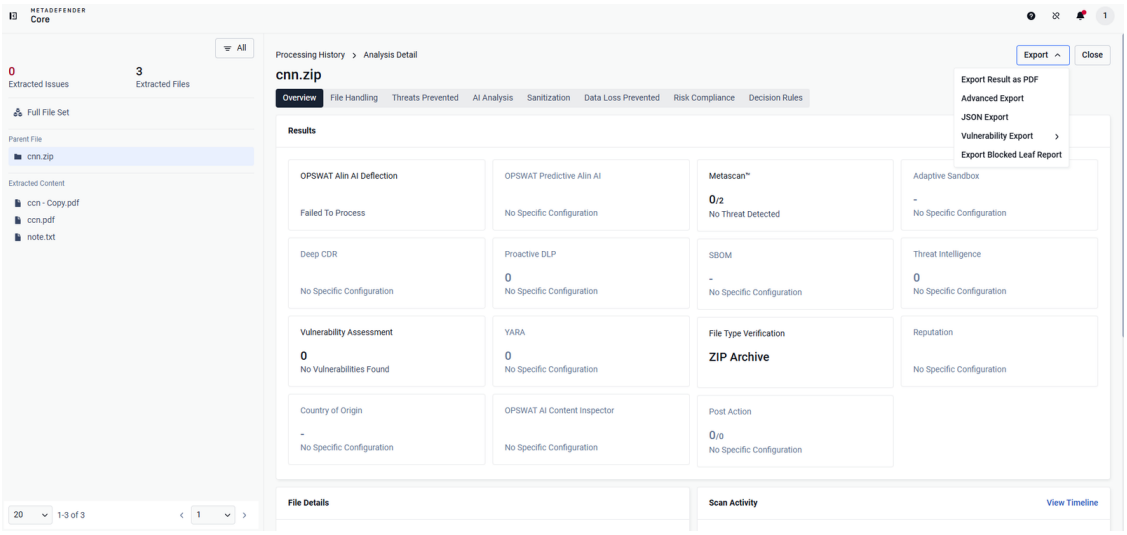
By understanding how block reasons are prioritized and counted, you can better interpret security statistics and focus remediation efforts on the most critical issues first.

Processing history

The Scan History page shows information on all scans made on the MetaDefender Core. Search and filter are also supported against each scan result attribute.



If an archive was scanned, its details popup will include tabs for the original files scan details and also a list with the results of the contained files.



On the Processing history page you can also search for:

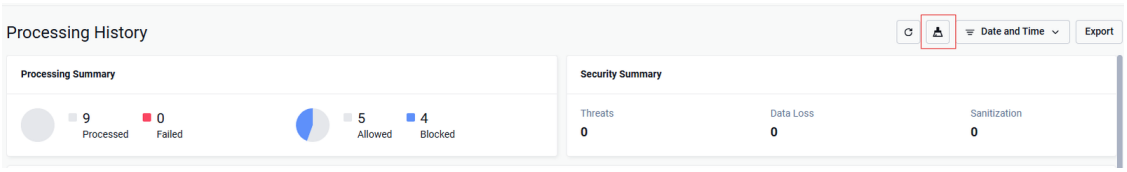
- MD5, SHA1, SHA256, SHA512 hashes.
- File name (and you can limit search result for a specific scan result, and for specific username who submitted files)
- Source (IP address)
- User name

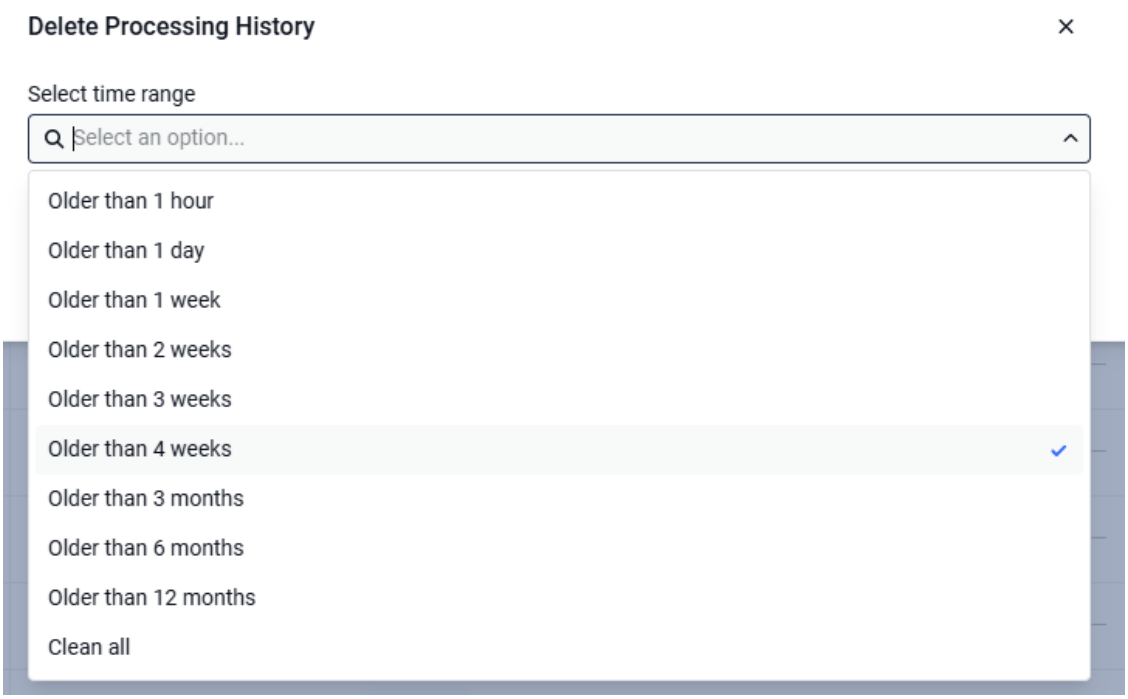
There is an option to export scan history in CSV or STIX format. For the export, the Result filter can also be applied.

- The user can export STIX file by clicking on STIX export button. In addition to set Result filter, STIX file will contain only blocked scans. After the desired time range selected, the download will be started by clicking on the OK button.
- CSV file is accessible by clicking on the CSV export button, and pressing OK after the desired time range selected.

Cleanup processing history

The processing history data will be automatically cleaned up based on the configured settings under **Settings > Data Retention**. Additionally, users can manually clean up the processing history by navigating to **History > Processing** and selecting the **Cleanup** button.





Warning

When the "**Clean All**" option is selected, please be aware that all data will be permanently removed.

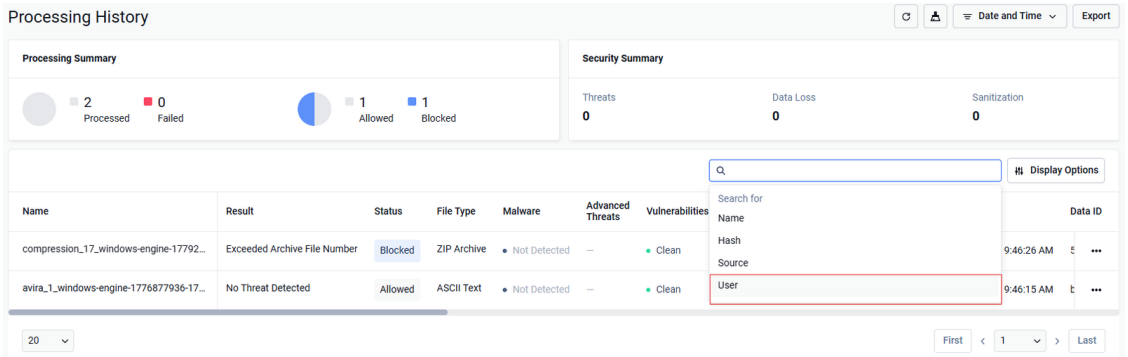
This action can only be performed when no files are being scanned.

This action may take some time to complete. During this period, please refrain from adding any files for processing.

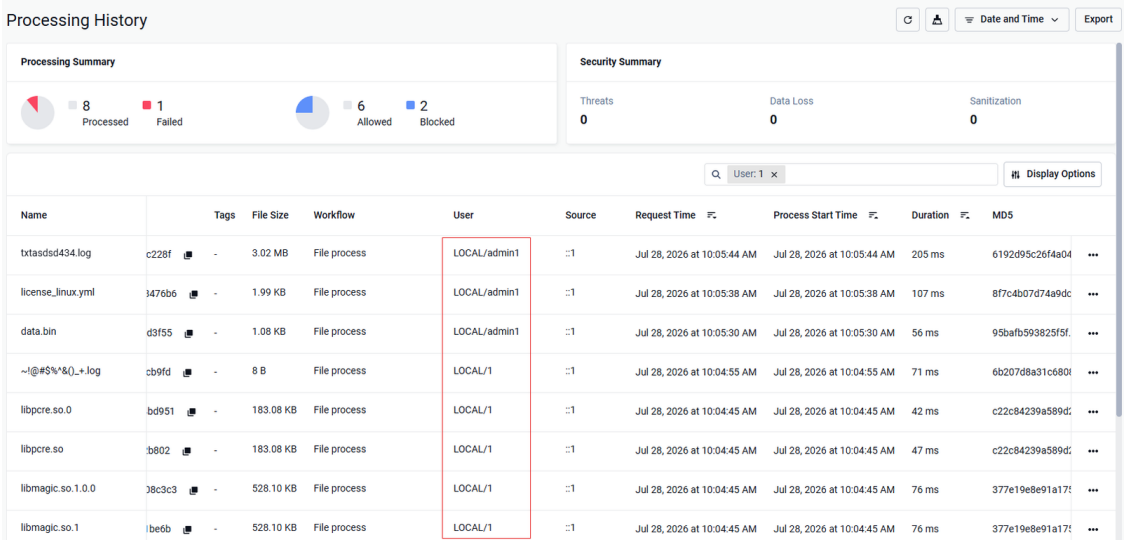
Processing history - search by user

Processing history supported for search by user.

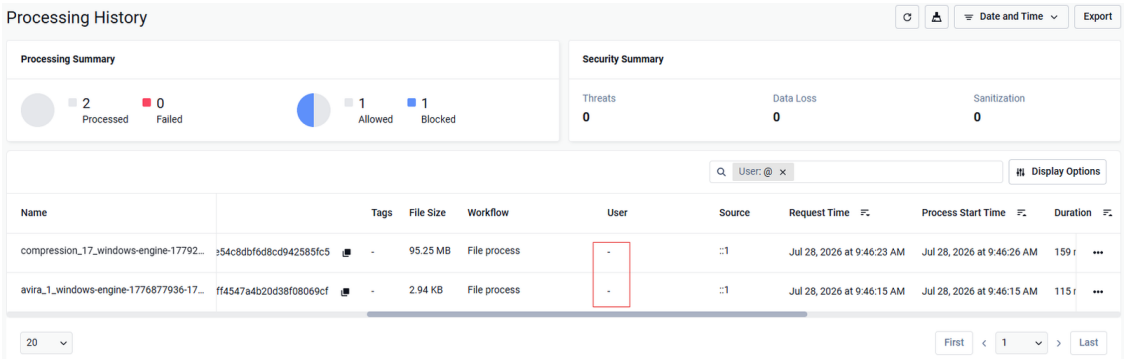
Please select Search box -> User



After that, please type user-name in the search box and hit Enter



For searching the scan request that has been scanned by "Anonymous" user, please type "@" for searching them



Processing history - search by extension

Since version 5.18.0, MetaDefender Core supports searching by file extension.

Please note that this search is based on the file name extension, not the actual file type detected by the file type engine.

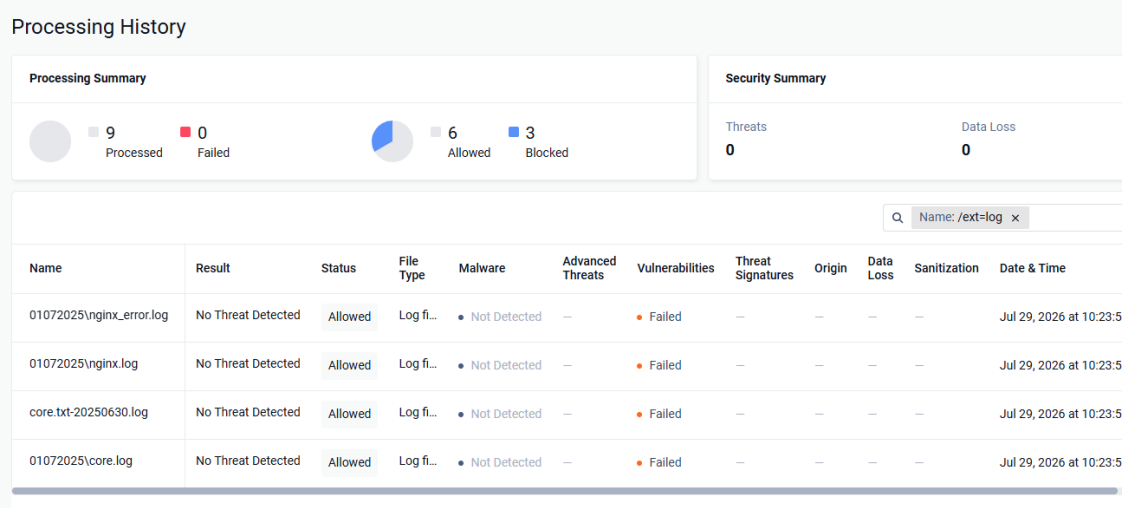
For example, a file named *test.pdf* will have an extension of “pdf” based on its name, even if it is detected as a different file type (e.g., TXT) by the file type engine.

Users can search by extension using the following format: `/ext=<your_extension>`

The extension is case insensitive - which means it can be JSON or json or even JsON

For example: `/ext=log`

The Processing History will then display all files whose names contain the specified extension.



Advanced Export Result Templates

What is Advanced Export Result ?

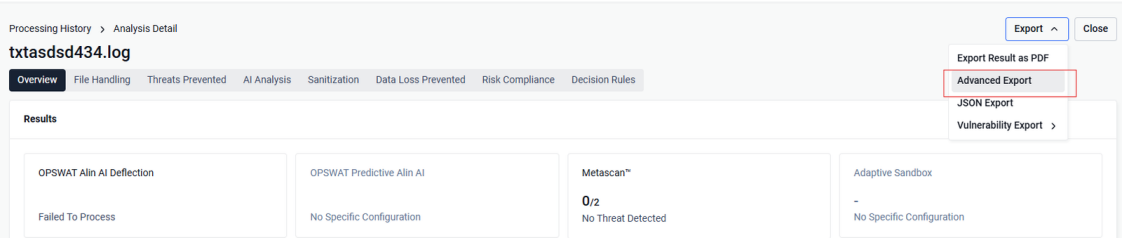
When you open a scan result in History Processing, you can export it as PDF or JSON. Advanced Export lets you choose exactly which sections appear in the report, such as:

- File Details, Process Result, Metascan / AV Engines
- Deep CDR, Proactive DLP, Threat Intelligence, SBOM, and others

Previously, every export started with all sections enabled, and you had to turn options off manually each time. Export templates let you save those choices under a name and reuse them later.

Where to find it

1. Go to History Processing
2. Open a scan result
3. Click Export Advanced Export



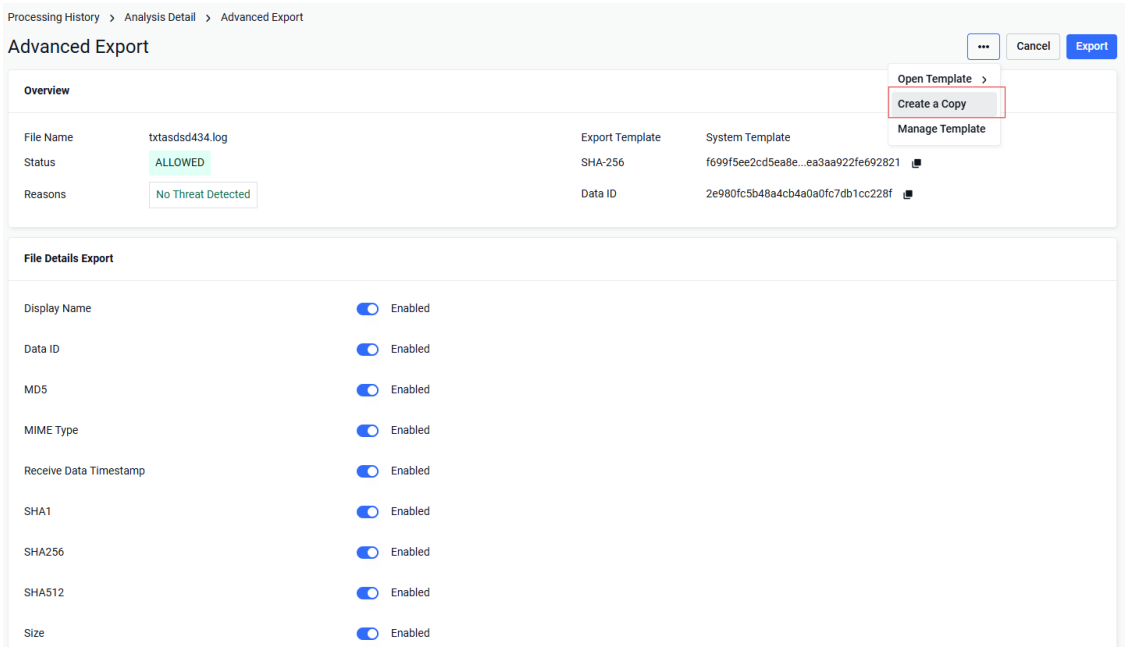
The Advanced Export dialog shows:

- Left panel — section categories (File Details, Metascan, etc.)
- Right panel — fields you can include or exclude

- Toolbar — the template currently in use (if any)

If you have a default template, it is applied automatically when you open Advanced Export.

Click `Create a Copy` for creating a new template



NOTE: Please make sure to check "Set as a default template" - so you don't need to select the template in the next time



Key concepts

Concept	What it means
Template	A saved set of export field selections under a name (e.g. "Summary only")

Concept	What it means
Default template	The template that loads automatically each time you open Advanced Export
System default	All sections and fields enabled — the original behavior with no template applied
Per-user	Templates belong to your login account. Other users on the same Core have their own templates

Restriction and notes

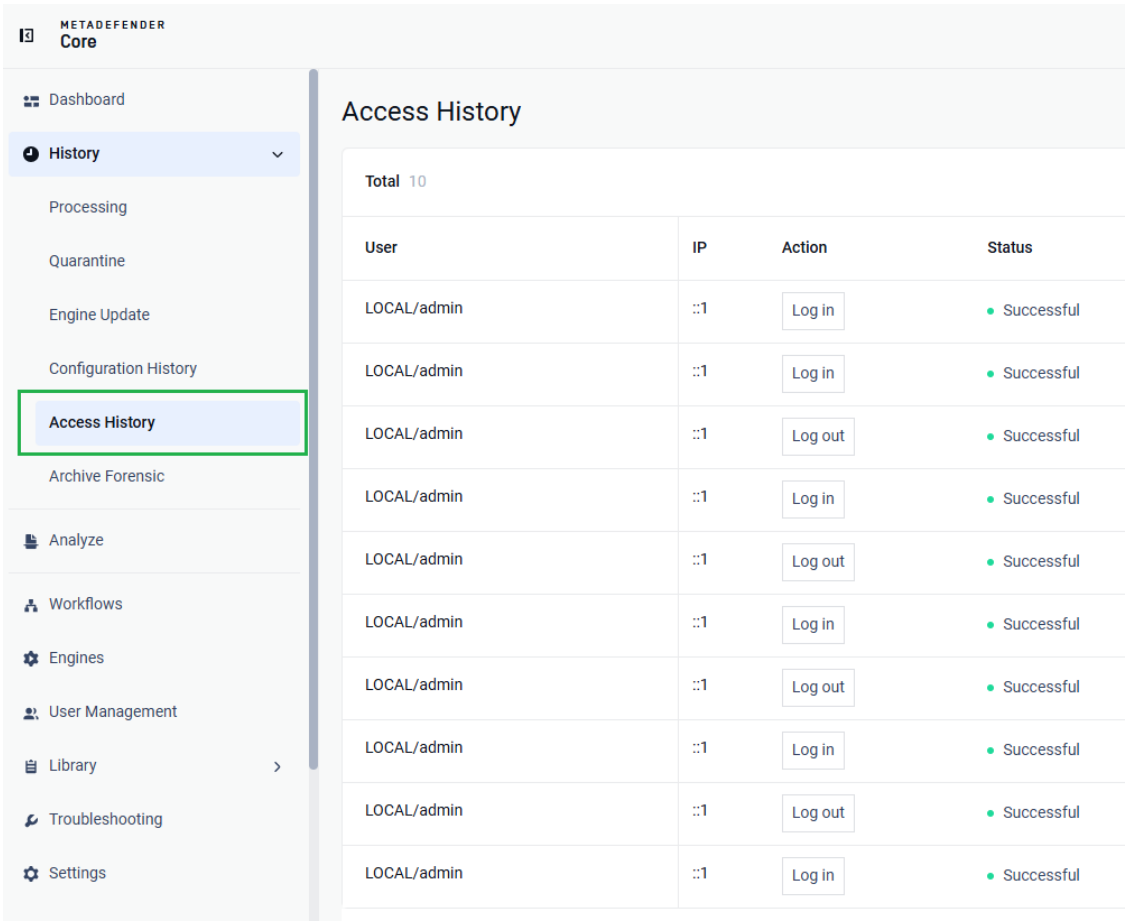
- Up to 10 templates per user account
- Template names must be unique for your account, non-empty, and at most 32 characters
- Templates are private — each user manages their own; admins cannot see or edit another user's templates through this feature
- Guest / anonymous users can still use Advanced Export, but cannot create or use templates — they must select fields manually each time

Access history

The Access history page shows all the information about the user's access activities:

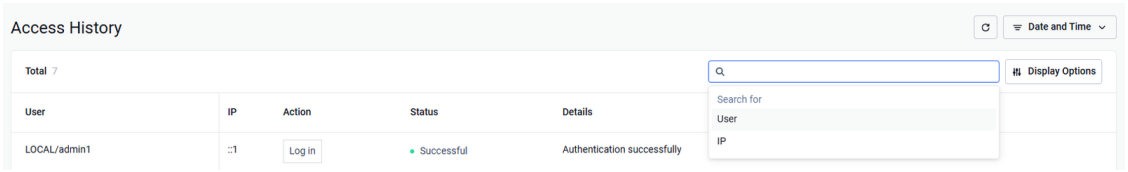
- User login
- User logout
- Change password
- Reset password
- Forget Password
- Change user role
- Session timeout (only supports standalone mode)

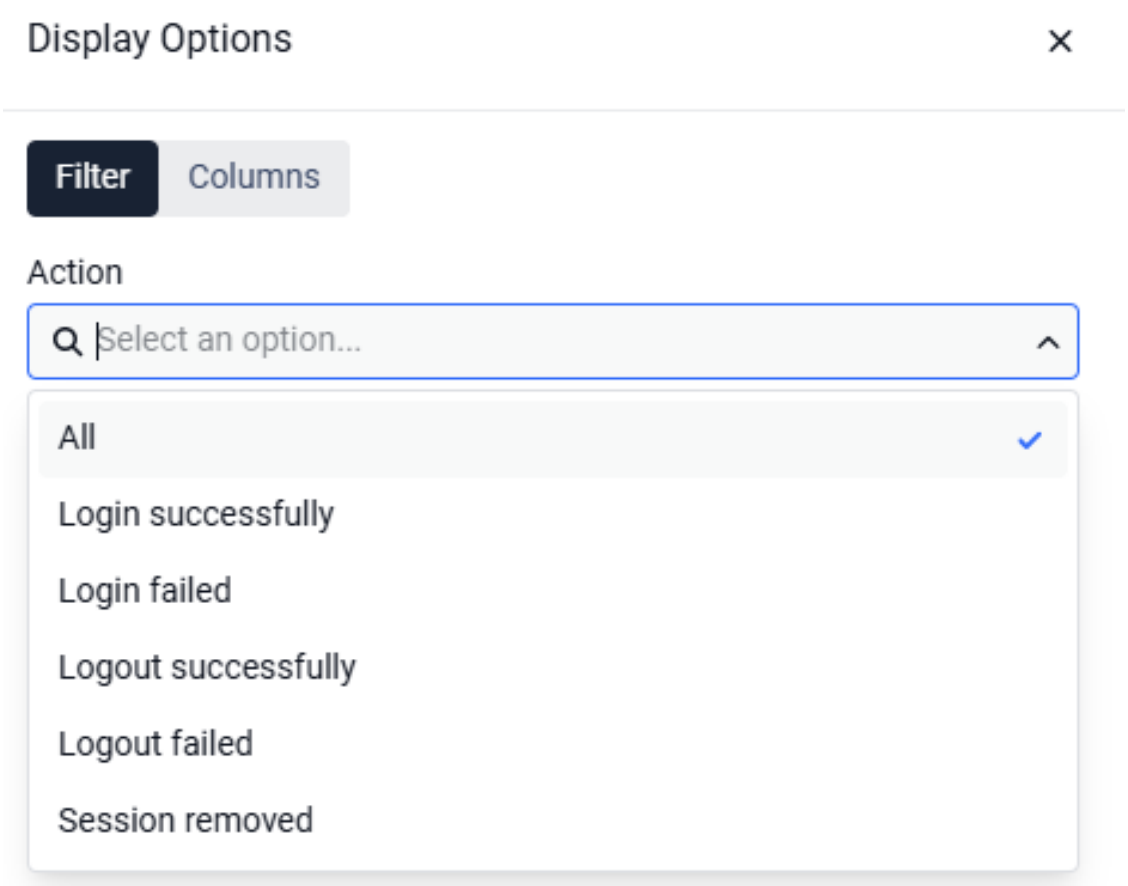
Users can access the page via History > Access history



Users can filter/search for several items:

- IP address
- User name
- Instance (shareddb)
- Action



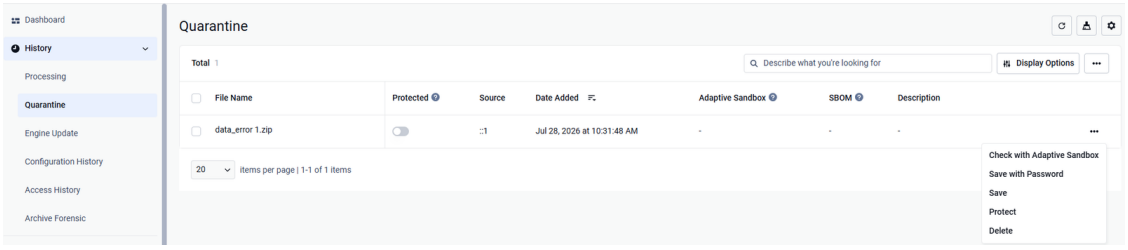


Download quarantine files

Please refer to this page for an overview of the Quarantine feature

This section provides guidance on how to save or download quarantined files for further investigation.

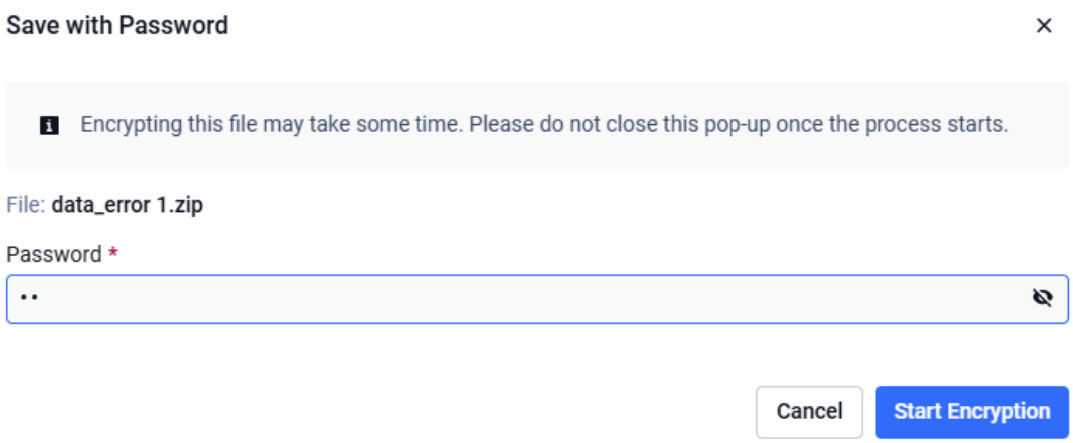
Navigate to **History > Quarantine**, select the file you wish to download, and click the **"..."** button. You will be presented with the following options:



- **Save:** The file is downloaded directly in its original format without password protection. For example, if the file is a PDF, it will be downloaded as a standard PDF file.
- **Save with Password:** The file is downloaded as a password-protected ZIP archive. You will be required to enter a password during the download process and use the same password to decrypt the file later. This option helps reduce the risk of accidental execution on the user's machine and prevents unintended actions.

Info

Password only accept these characters (same as export settings) **Alphanumeric**: A-Z, a-z, 0-9 **Special characters**: `~`!@#$%^&*()-_=+[]{}':<>,".;?/\|` and space.



Warning

Please note that when using the **"Save with Password"** option, the file may take several minutes to complete the encryption process. During this time, please do not close the pop-up window. If the window is closed, the process will be interrupted, and you will need to restart it from the beginning when attempting to download the file again.

Engines

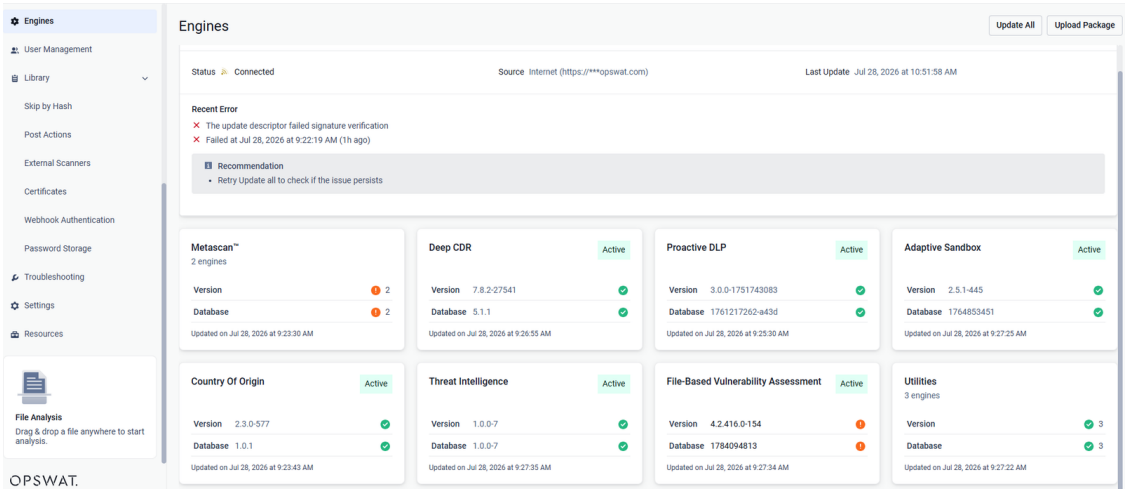
Engine type details

Under the **Modules** menu all the installed engines are listed with their details such as

- Type of engine. Possible types are
  - Anti-malware engine
  - Archive engine
  - Proactive DLP engine
  - Deep CDR engine
  - Adaptive Sandbox engine
  - Software Bill of Materials engine
  - Filetype detection engine
  - Reputation engine
  - Country of Origin engine

- Utility engine
- Vulnerability detection engine

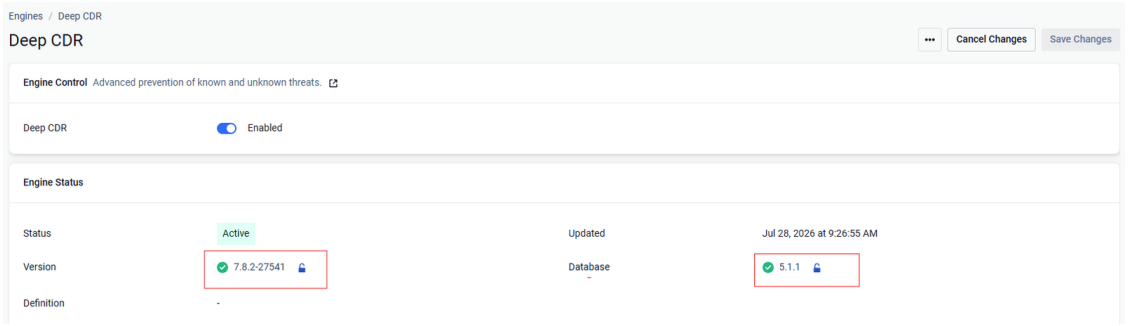
- Elapsed time since last update
- Proportion of active and non-active engines of a particular type
- Engine version
- Version of database the engine is crrently using
- Engine status (Active/Non-Active)



Engines can be disabled (and re-enabled afterwards) by clicking on the switch at the end of the line that belongs to that particular engine. When an engine is disabled neither the engine nor the corresponding database package is updated and it will be removed from every node.

Lock & Unlock engines (for auto-update prevention)

Engine and its database can be locked (previous called pin and unpin) to prevent it from being applied new updates when you allow auto update on Core.

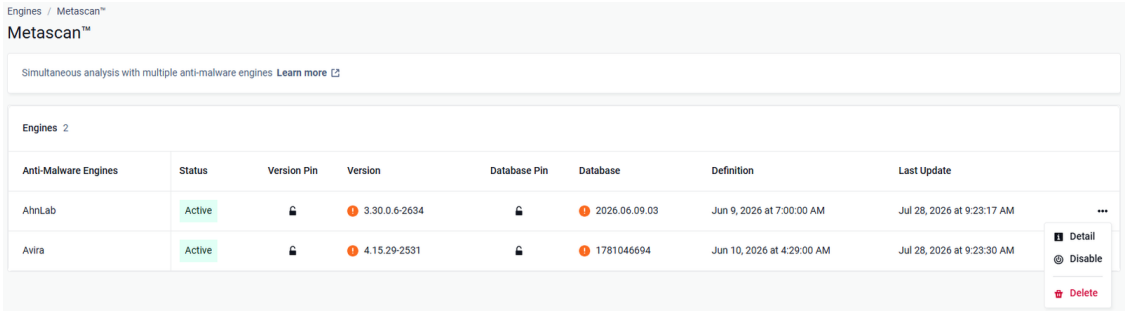


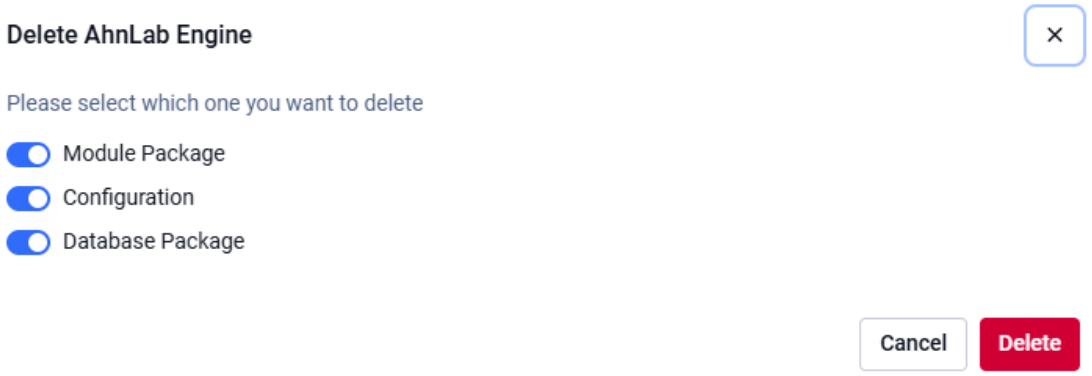
Note

- You can lock engine and database individually on same engine
- When locked, that means no auto update can be applied on that part (engine / database), even when user triggers "Update All" button

Remove engines (to start over engine download and deployment)

At some points, when engines become problematic and comes to the state of permanent failed, then you may need to remove engine.

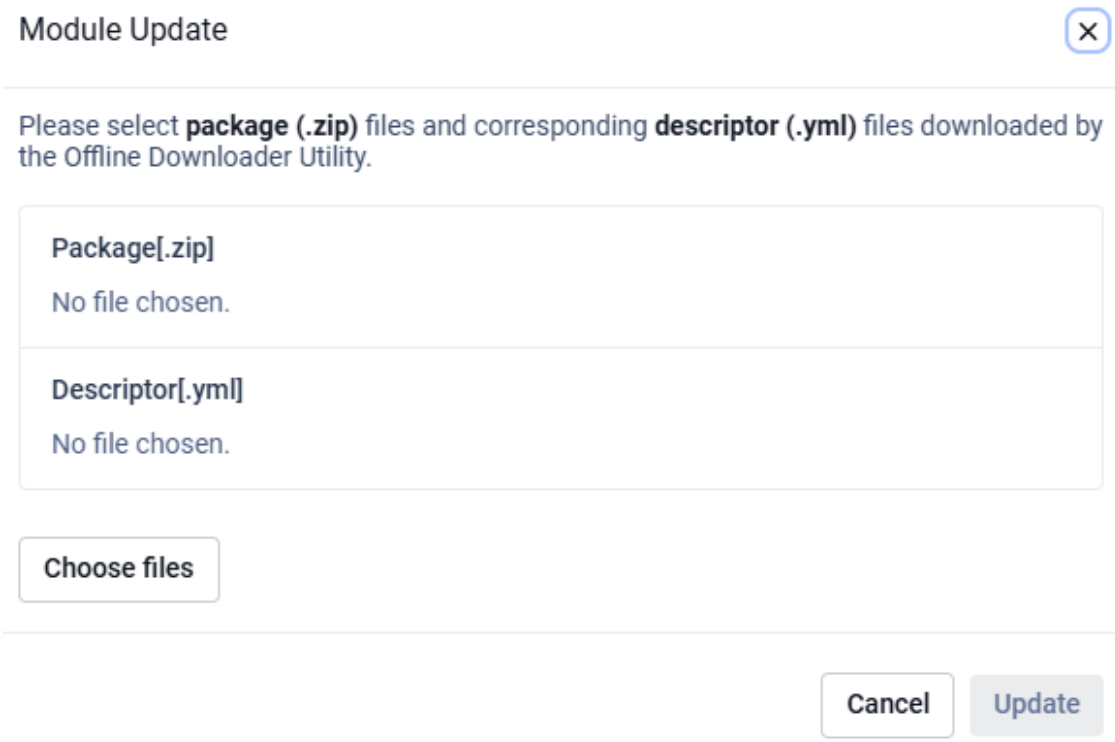




Manual updates

To manually trigger update of scan engine and database packages, click on the **Update now** button.

To provide engine or database packages on your own, select the **Upload package** option.



Upload packages

The package should be a ZIP and the descriptor YML file, which can be downloaded with the Update Downloader. Multiple files can be selected.

> **Info**
> Engine or database versions that have ever been used on a system won't be accepted as updates.

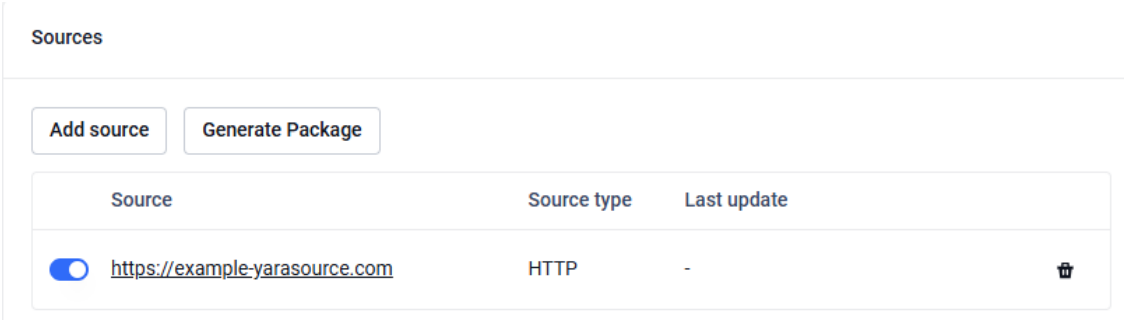
YARA Rule Package Generation

MetaDefender Core supports generating YARA packages from two types of sources:

- HTTP source
- Local directory

To generate a YARA package:

1. Go to Engines > Utilities > YARA.
2. Click Add Source.
3. Select the source type, enter the path, then click Add Source.
4. Repeat step 3 if you need multiple sources.
5. Click Generate Package.



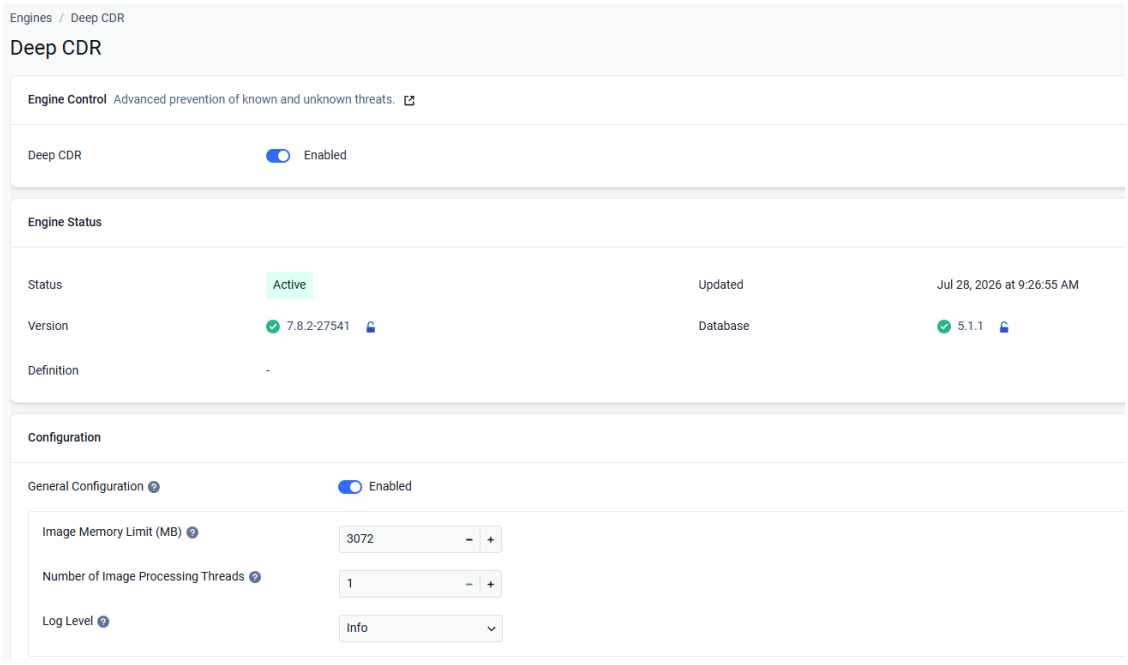
You can also generate a YARA package via MD Core API endpoints: POST - Generate YARA package

For HTTP sources, to protect MD Core from several attack methods, YARA packages must meet certain constraints. Packages that violate these constraints are rejected. The required constraints include:

Constraint	Limit	Description
Archive Size	50 MB	Maximum size of the compressed YARA package file
File Entry Count	1,000 files	Maximum number of individual files allowed within the archive
Nested Folder Depth	50 levels	Maximum recursive depth for nested directories within the archive
File Path Length	255 - engine DB path length	Maximum allowed path length for each file entry (calculated dynamically based on engine database folder location)
Extracted Content Size	50 MB	Maximum total size of all files after extraction from the archive

Configuring engine advanced settings

Some engines advanced settings can be configured.



Choose your desired configurations and click **Save changes**, then Close.

Library Management

Metadefender Core displays detailed information on scan nodes and the status of engine updates including anti-malware engines, archive engines, etc.

Certificates

On this page, path to certificates and private keys for signing scan batches or HTTPS configuration can be given.

> **Note**
>
> - Certificate should be in a Base64-encoded X.509 certificate file (.crt, .cer) format.
> - Private key should be a privacy-enhanced electronic mail file (.pem) format and it should not be locked by password.

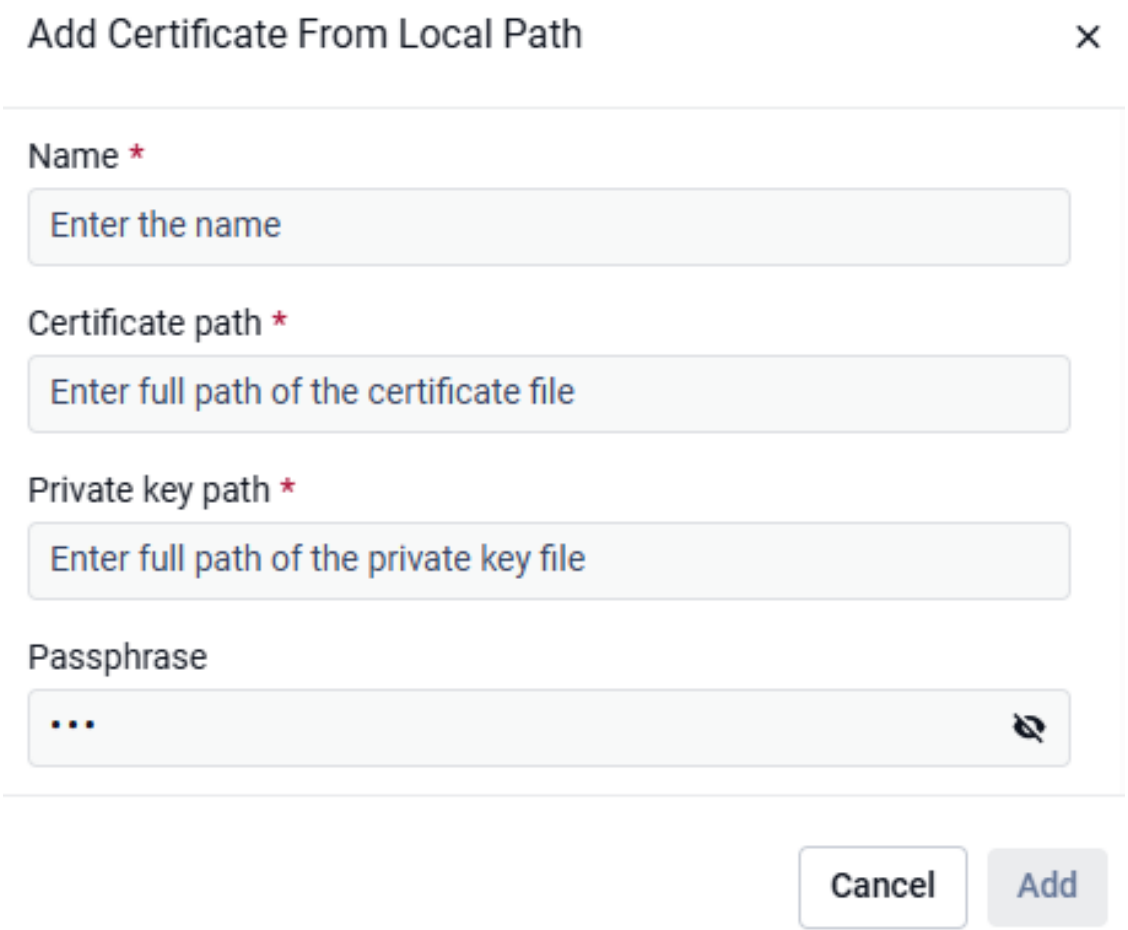
Adding certificate-key pair to the library

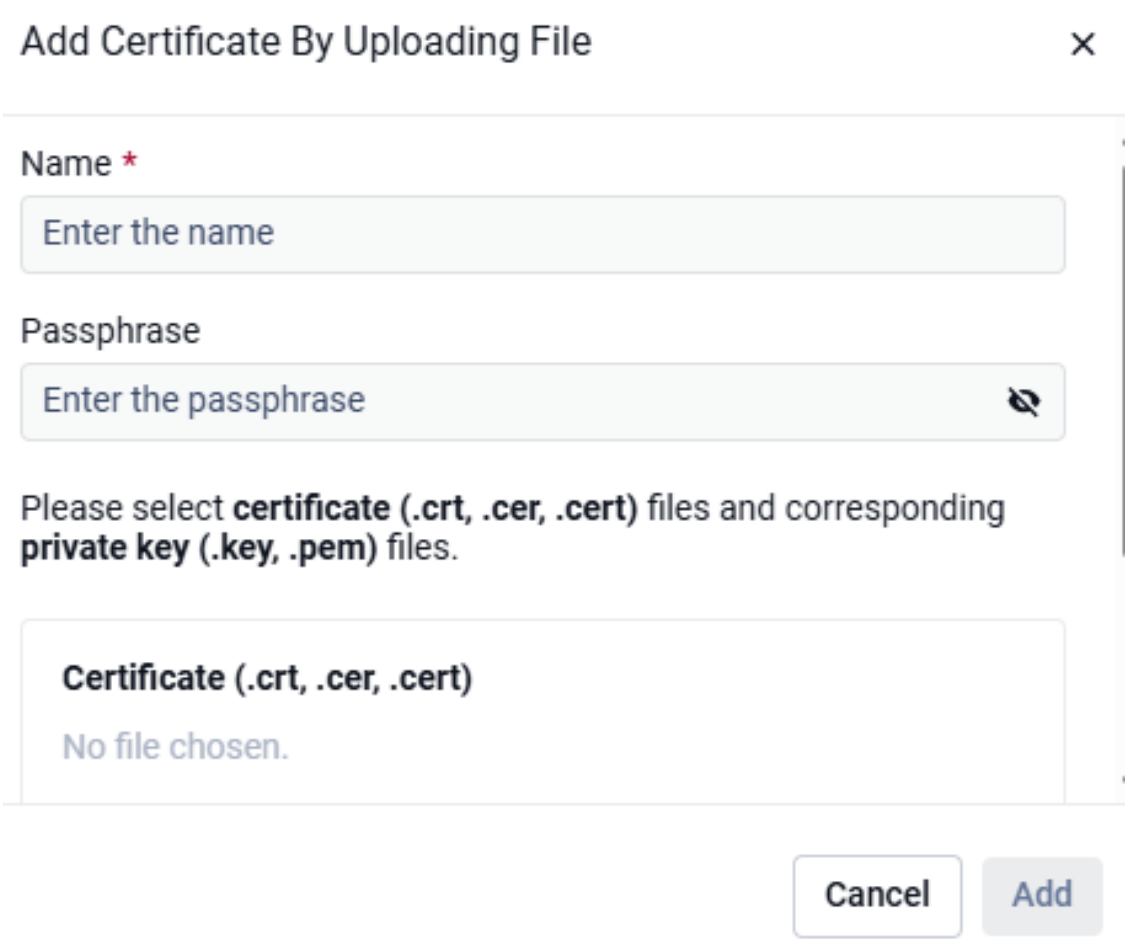
1. Go to **Library** → **Certificates** page
2. Click on **Add Certificate** button
3. Fill out the **Add new certificate** form using one of the following methods: **Method 1: Add From Local Path**
   - **Name**: Certificate name
   - **Certificate Path**: Absolute path to the certificate file
   - **Private Key Path**: Absolute path to the private key file
   - **Passphrase**: Enter the passphrase for the private key (if required)

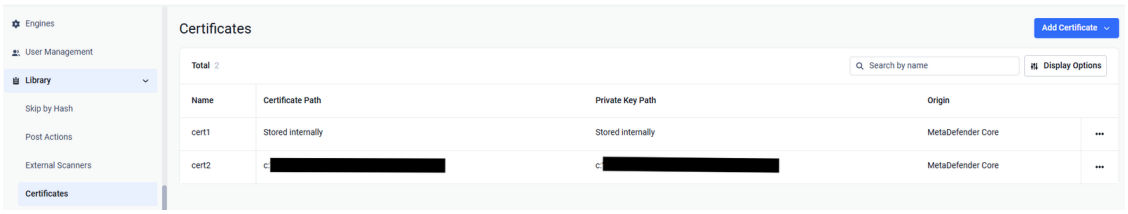
Method 2: Add By Uploading File

- **Name**: Certificate name - **Certificate**: Upload the certificate file - **Private Key**: Upload the private key file - **Passphrase**: Enter the passphrase for the private key (if required)

4. Click **Add** button







Checking the details of a certificate

By clicking the line of the certificate, the "Certificate viewer" pops up and shows the details of the certificate.



Modifying the name, passphrase or the paths of a certification

Hover the mouse cursor over the line that is to be modified and click on the pen picto. The **Modify certificate** modal pops up and the fields can be edited.

> **Note**
> The certificate file and the key file should be readable by the user who owns the ometascan process.

Webhook Authentication

While leveraging webhook on MetaDefender Core, the callback connection stays at un-secured by default. To harden the security, MetaDefender Core supports a mutual authentication mode between the server (MetaDefender Core) and client (where the callback is sent to), and prevents the man-in-the-middle attack (scan results could be possibly altered unauthorizedly).

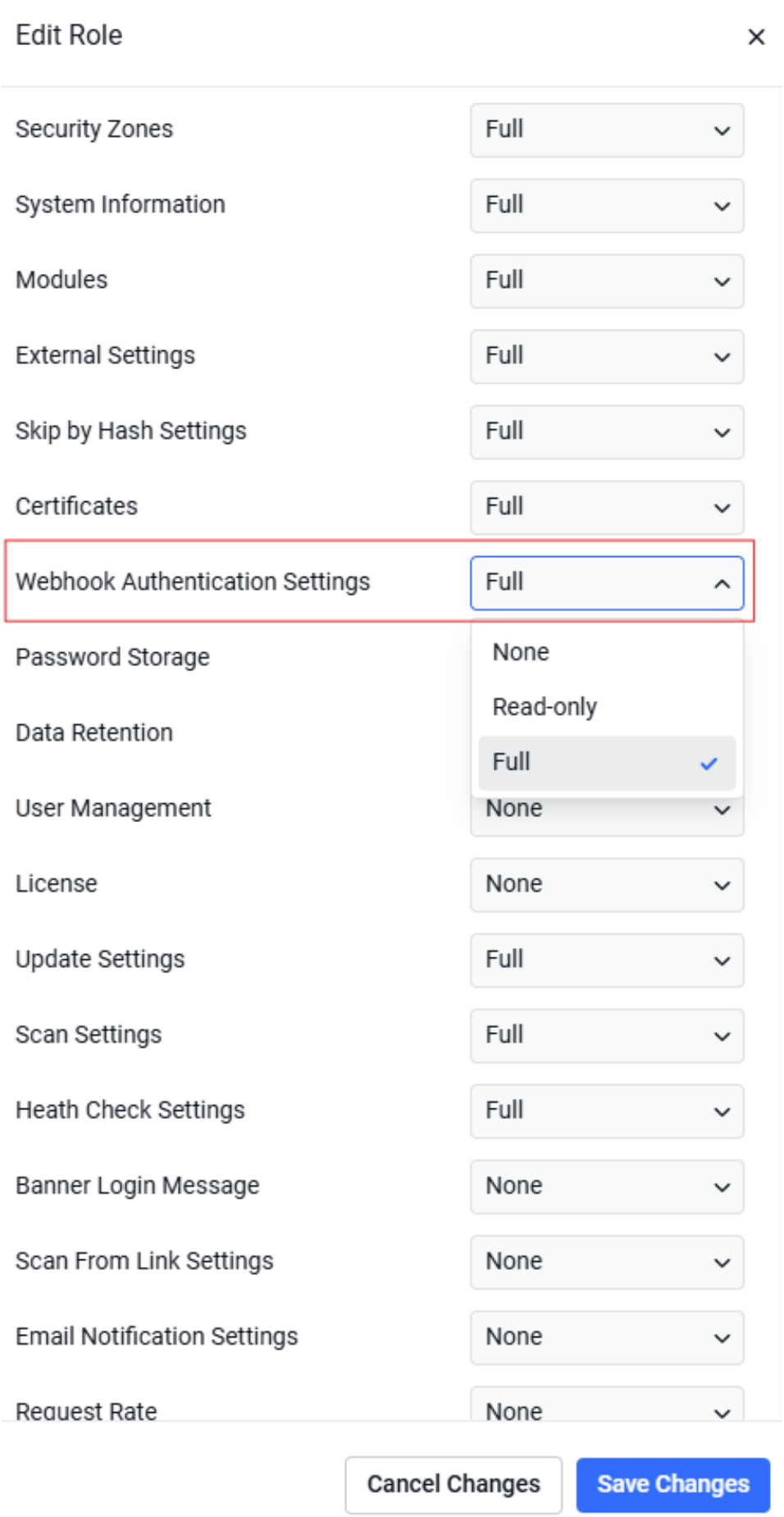
How this security feature works

- Admin user to request MetaDefender Core to generate a private-public key pair (the private key is encrypted within the product, and could not be seen or readable by anyone)
- Admin user to download and keep that public key in their secret vault on the client side (where the callback is sent to).

- Admin user to configure MetaDefender Core to use the private key in a desired workflow rule, for each scan request:
  - MetaDefender Core to calculate the signature of scan result using the selected private key.
  - MetaDefender Core to include the calculated signature into the callback response header before sending back to client.
  - Client side (where the callback is sent to) to use the stored public key to:
    - Decrypt the signature included in the callback response header.
    - Calculate SHA256 hash checksum from the callback reponse body (which is the entire scan result).
    - Compare the calculated SHA256 hash checksum to the hash value decrypted from the callback response header.

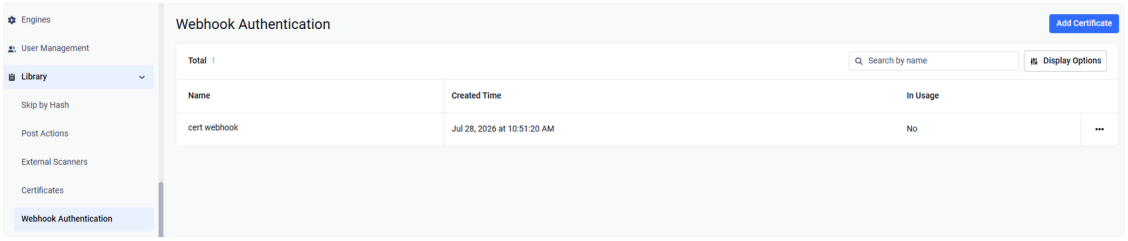
Note

MetaDefender Core also supports admin users to configure the user role for whom allowed to see and modify webhook authentication settings.


Edit Role
Security Zones
Full
System Information
Full
Modules
Full
External Settings
Full
Skip by Hash Settings
Full
Certificates
Full
Webhook Authentication Settings
Full
Password Storage
Data Retention
None
Read-only
Full
User Management
None
License
None
Update Settings
Full
Scan Settings
Full
Heath Check Settings
Full
Banner Login Message
None
Scan From Link Settings
None
Email Notification Settings
None
Request Rate
None
Cancel Changes
Save Changes

Create a new certificate-key pair for webhook authentication

1. Go to **Library** → **Webhook Authentication** page
2. Click on **Add certificate** button



Skip by hash

This page contains three lists which belong to similar but different features. Note that rules listed in these there features are globally applied, not per workflow.

- Skip engines
- Allowlist
- Blocklist

Hash can be either MD5, SHA1, SHA256 or SHA512.

Skip engines

On this page, users can define rules on what files should be skipped by what engines. That is, a file with the given hash will not be processed by the listed engines.

Hash and at least one engine are mandatory to give, comment is optional.

Allowlist

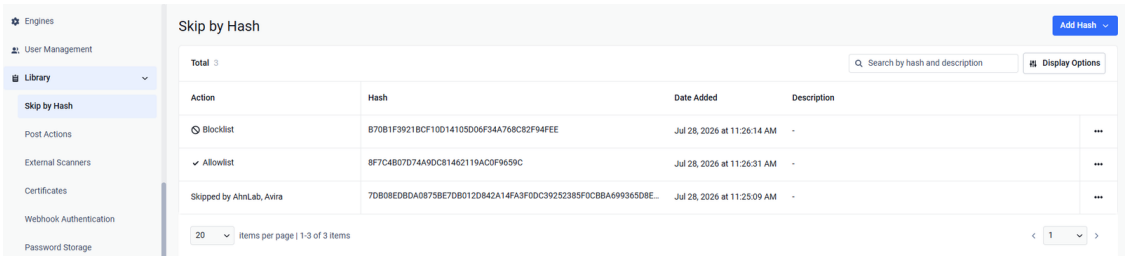
Files whose hashes listed here will be globally allowlisted, so they won't be processed in any workflow and will be allowed.

Hash is mandatory to give, comment is optional.

Blocklist

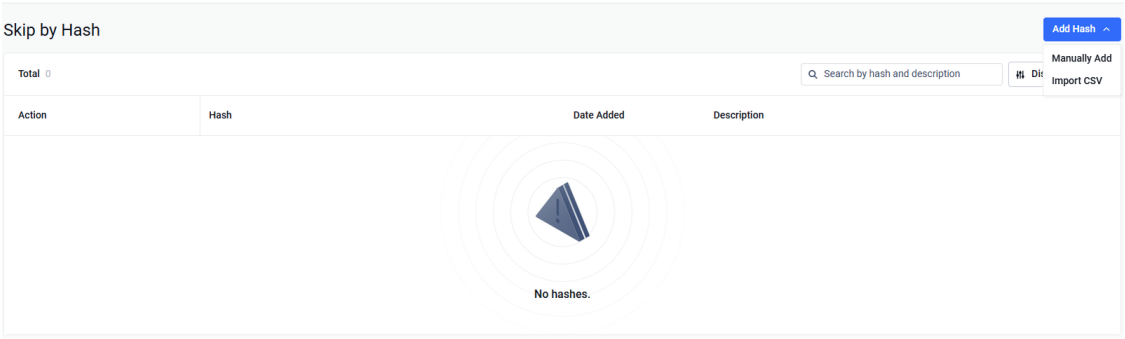
Files whose hashes listed here will be globally blocklisted, so they won't be processed in any workflow and will be blocked.

Hash is mandatory to give, comment is optional.



Adding a hash to skip list

On the **Skip by Hash** page click the "Add hash" button on the top right. There are 2 ways for Administrators to add hashes to the application.

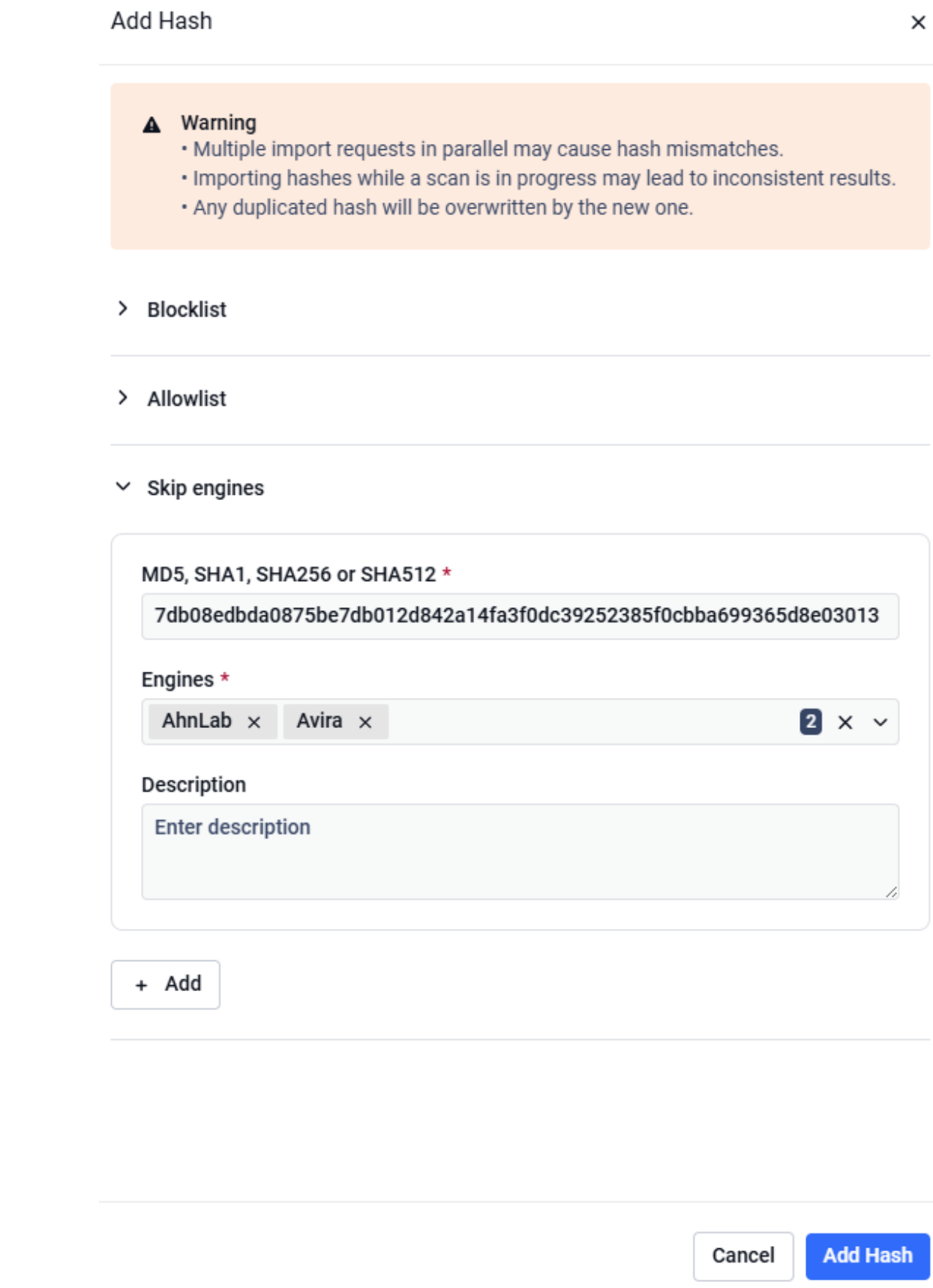


- Manually Add:
  - Allow you to input hashes and information manually.
- Import CSV:
  - Enable you to prepare a list of hashes and upload all at once to the application.
  - A CSV template is downloadable for easy content input.



Select your desired way, then "Add hash" page appears.

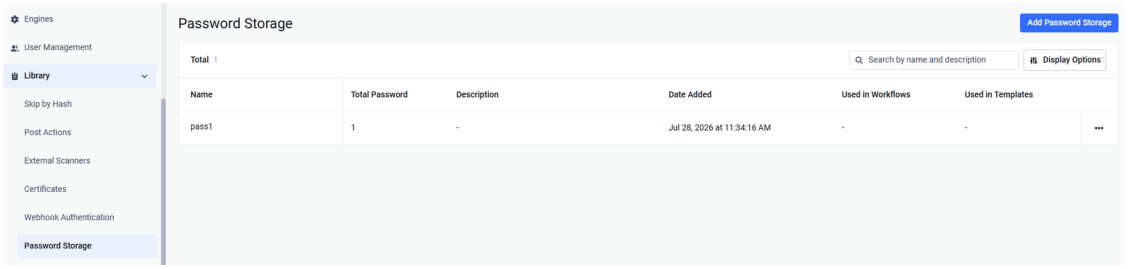
After giving the necessary information, click on the **Add** button.



Password Storage

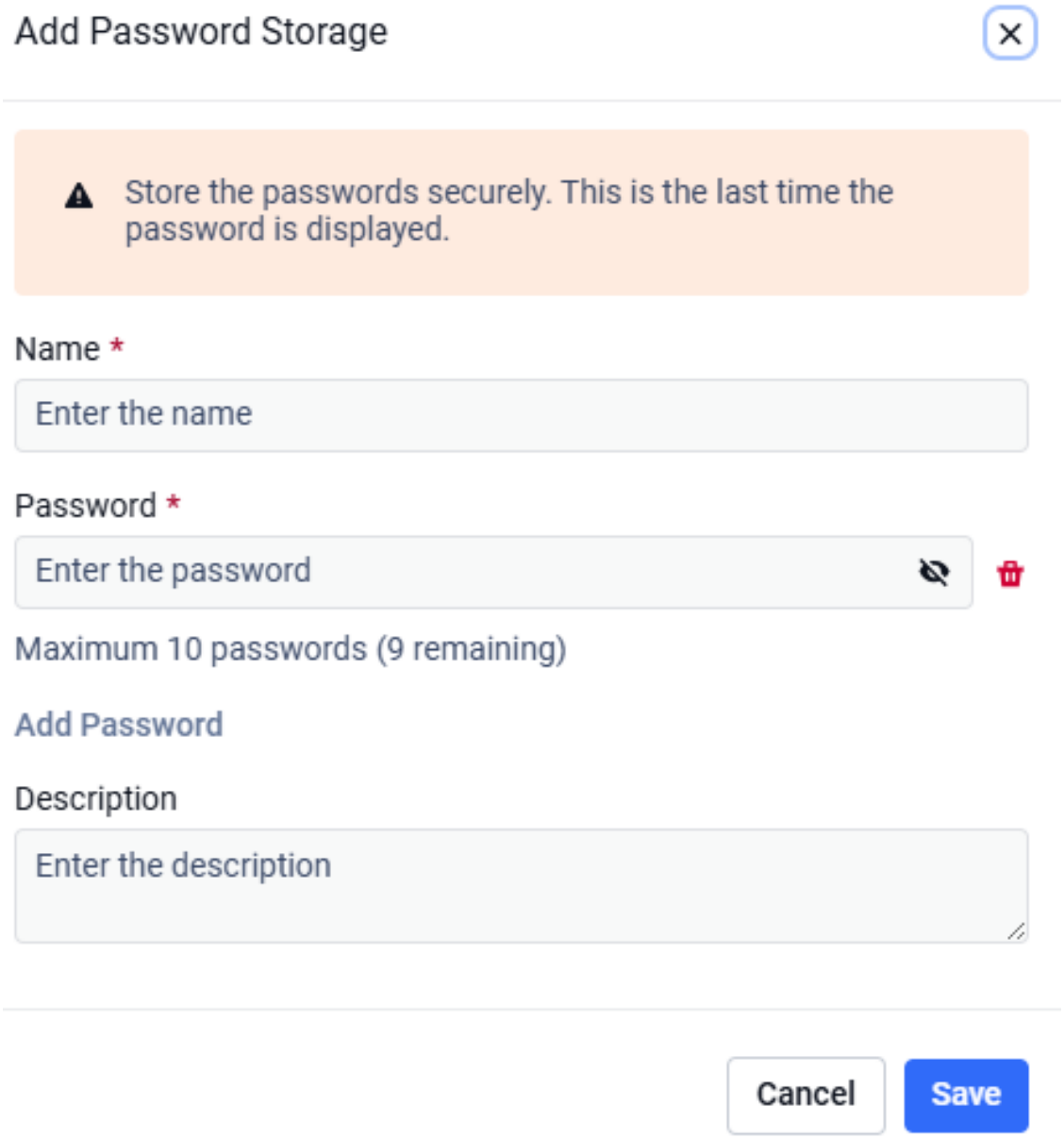
Users can define and manage lists of passwords used for file decryption, streamlining the process when handling

encrypted or password-protected files.

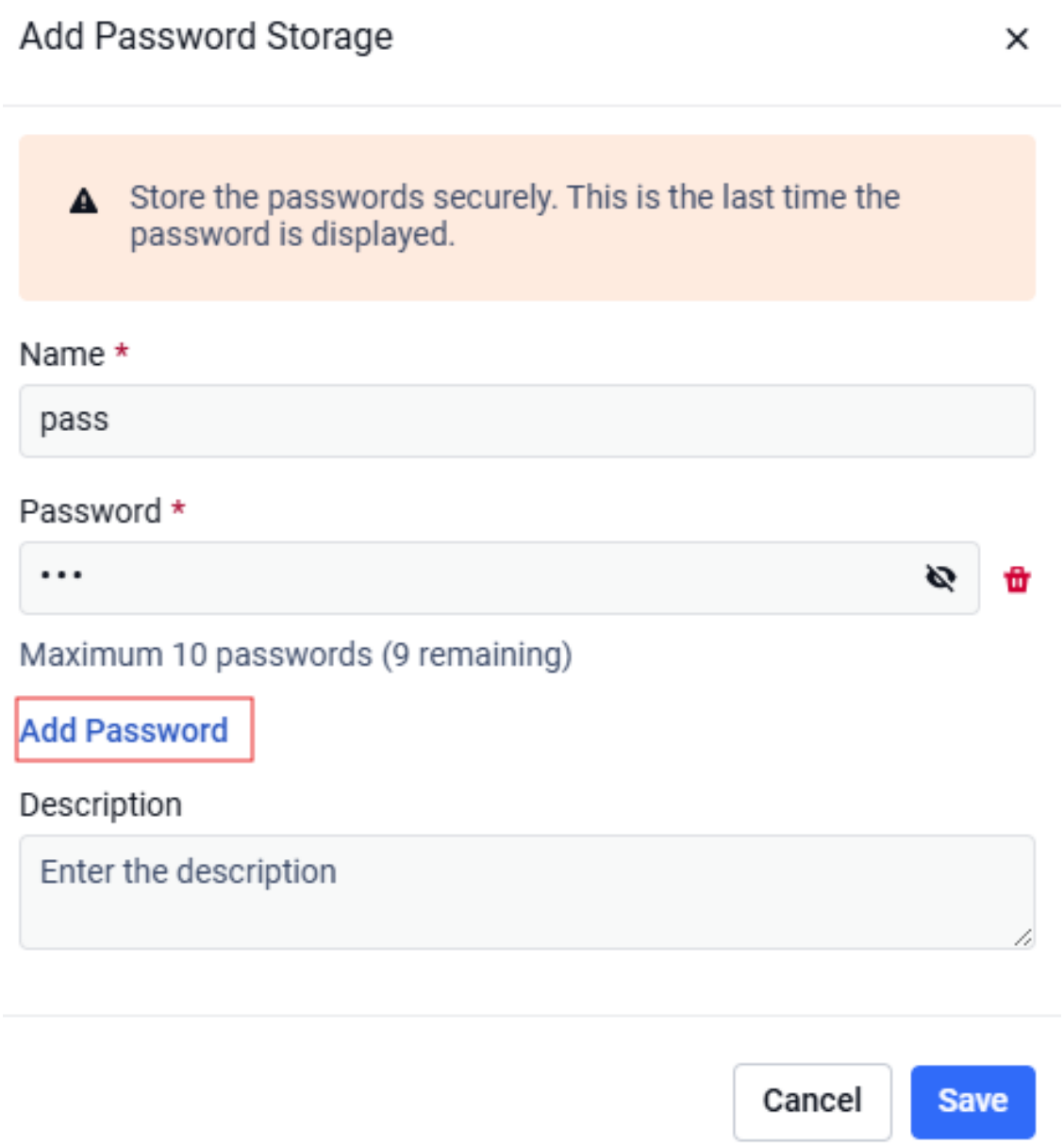


We can add up to 50 password storage and each password storage can contains up to 10 passwords

Click the "Add Password Storage" button for adding new password storage



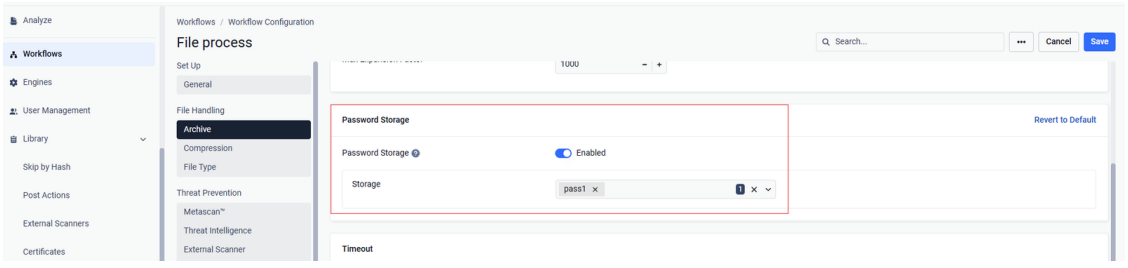
We also can add more passwords in the list for later, click to any password storage to modify



We can only add more passwords to the list, we cannot modify the old passwords or remove them from the list. To remove the password lists, we need remove the whole password storage.

After creating the password storages, we need to go to the workflow rule setting to select the password storages that we want to use for a specific workflow rule.

Workflows > Workflow Configuration > Select a workflow rule > Archive tab > Scroll down and check Password storage



Regular Maintenance

Checking for Upgrades

Metadefender Core checks for available database updates and scan engine updates for the installed anti-malware engines on a regular basis. To manually update a scan engine or its database, click on the update now button or the upload package link on the **Inventory > Engines** page.

Checking Engines / Databases Health

Metadefender Core regularly checks for available database updates and scan engine updates for the installed anti-malware engines. Both database and engine upgrades are based on a mechanism that checks for authenticity of the origin of the upgrade package. If the authenticity is confirmed, the upgrade package is downloaded. As an extra stability measure each downloaded upgrade package is tested locally to ensure that it is functioning properly. Only after successful testing will the upgrade package be distributed among MetaDefender Core nodes.

Database Defragmentation and Optimization

See more at:

- Database Maintenance | MetaDefender Core 4.19.0 or above
- Database Maintenance | MetaDefender Core 4.18.0

Database Maintenance | MetaDefender Core 4.19.0 or above

> **Info**
>
> This tool and instructions are only applicable to MetaDefender Core 4.19.0 or above that leverages PostgreSQL database.

PostgreSQL database could be more bulky over time with high scanning traffic. Even manually removing scan history or leaving it to data retention to clean up old records, it will not immediately remove actual data in the PostgreSQL database which is fully controlled by PostgreSQL.

The tool which is bundled with MetaDefender Core installation is to analyze database and vacuum database effectively, and simplify IT administration.

- Windows: `<MetaDefender Core installation folder>\ometascan-vacuum-db.exe`
- Linux: `/usr/sbin/ometascan-vacuum-db`

> **Warning**
>
> Even though it is not required to stop MetaDefender Core service while running, but still recommended to run the vacuum tool when the scanning service is at rest, or not in peak hours to avoid scanning performance impact.

How-to run instructions

Windows

1. Open cmd as Administrator
2. Navigate to MetaDefender Core installation folder (e.g. `C:\Program Files\OPSWAT\MetaDefender Core`)
3. Call: ometascan-vacuum-db.exe with no arguments.
4. Upon done, expected to have a new database analysis and vacuum outcome package auto generated in the same folder with format: `analyze_and``_cleanup_database``<timestamp>.zip`

Linux

1. Run as superuser privileges: `sudo export LD_LIBRARY_PATH=/usr/lib/ometascan`
2. Run as superuser privileges: `sudo /usr/sbin/ometascan-vacuum-db`
3. Upon done, expected to have a new database analysis and vacuum outcome package auto generated in `/usr/lib/ometascan/` with format: `/usr/lib/ometascan/analyze_and_cleanup_database_<timestamp>.tar.gz`

Kubernetes

To run the vacuum in a containerized environment, use this command:

Code · bash

```
kubectl exec <pod-name> -c <container_name> -n <namespace> -- "/opt/ometascan/usr/sbin/
ometascan-vacuum-db"
```

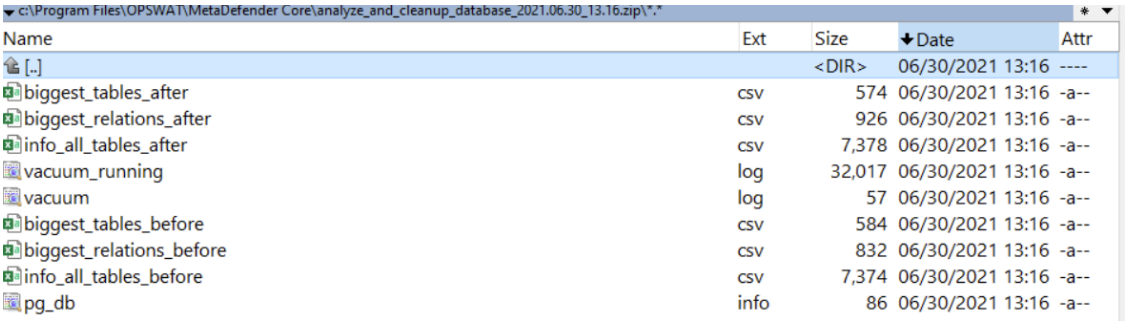
Example command and output:

Code

```
$ kubectl exec md-core-9d96d7cb8-g5dhg -c md-core -n <namespace> -- "/opt/ometascan/usr/sbin/
ometascan-vacuum-db"
Running vacuum with config file: /opt/ometascan/core_data/ometascan.conf
Postgres info: Temp directory: , /opt/ometascan/core_data/tmp
Postgres info: Current hba config version = 3, expected = 3,
Postgres info: Postgres private user is not defined,
Postgres info: Database name, metadefender_core
Postgres info: Postgres private password is not defined,
Postgres info: Generating private password for the first time, metadefender_core
Postgres info: The service and the config are mismatched, re-config the service..., /opt/
ometascan/core_data/var/lib/ometascan/pg_data
Postgres info: PG_DATA path in dat file = /opt/ometascan/core_data/var/lib/ometascan/pg_data,
Postgres info: PG_DATA path in conf file = /opt/ometascan/core_data/var/lib/ometascan/pg_data,
Postgres info: Saving /opt/ometascan/core_data/tmp/ometascan/ometascan-pg.service,
```

```
Postgres info: Opened /opt/ometascan/core_data/tmp/ometascan/ometascan-pg.service,
Postgres info: Replacing ometascan-pg.service...,
Postgres info: Restart services...,
Postgres info: Saving /opt/ometascan/core_data/tmp/ometascan/ometascan-pg.init,
Postgres info: Replacing /etc/init.d/ometascan-pg,
Postgres info: Checking for database upgrade,
Postgres info: Upgrade database status: 0,
Postgres service init done
Creating connection with: postgres-core
Creating connection done
Created out dir: /opt/ometascan/core_data/analyze_and_cleanup_database
Working dir: /opt/ometascan/usr/lib/ometascan/postgres
Running vacuum DONE
```

What's inside the generated package?



Including database object size analysis before and after running vacuum, the vacuum result and log, the new database size after vacuum. That helps us analyze and determine how effective the vacuum performs against that database, and investigate if any issue occurs.

Database Maintenance | MetaDefender Core 4.18.0

Info

Only applicable to MetaDefender Core 4.18.0 with SQLite database.

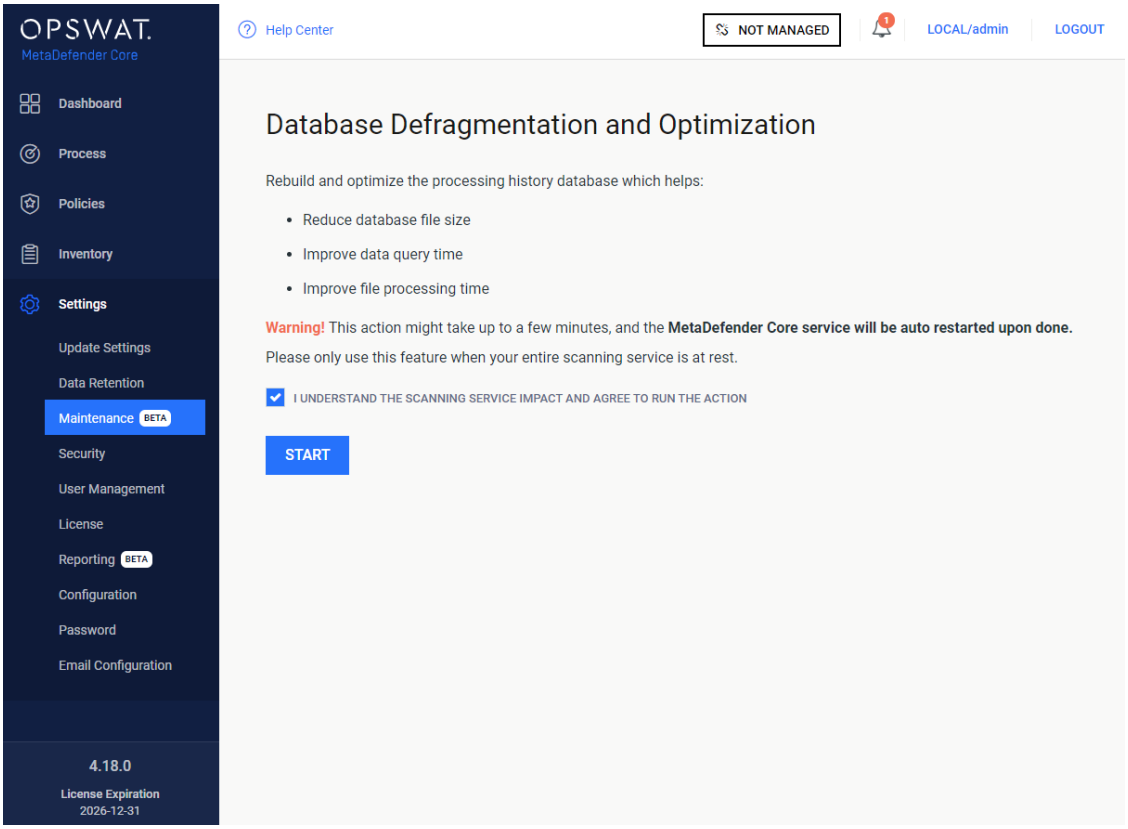
Starting with version 4.19.0, the product has changed to using PostgreSQL database. The applicable vacuum instructions could be found at MetaDefender Core 4.19.0 or above

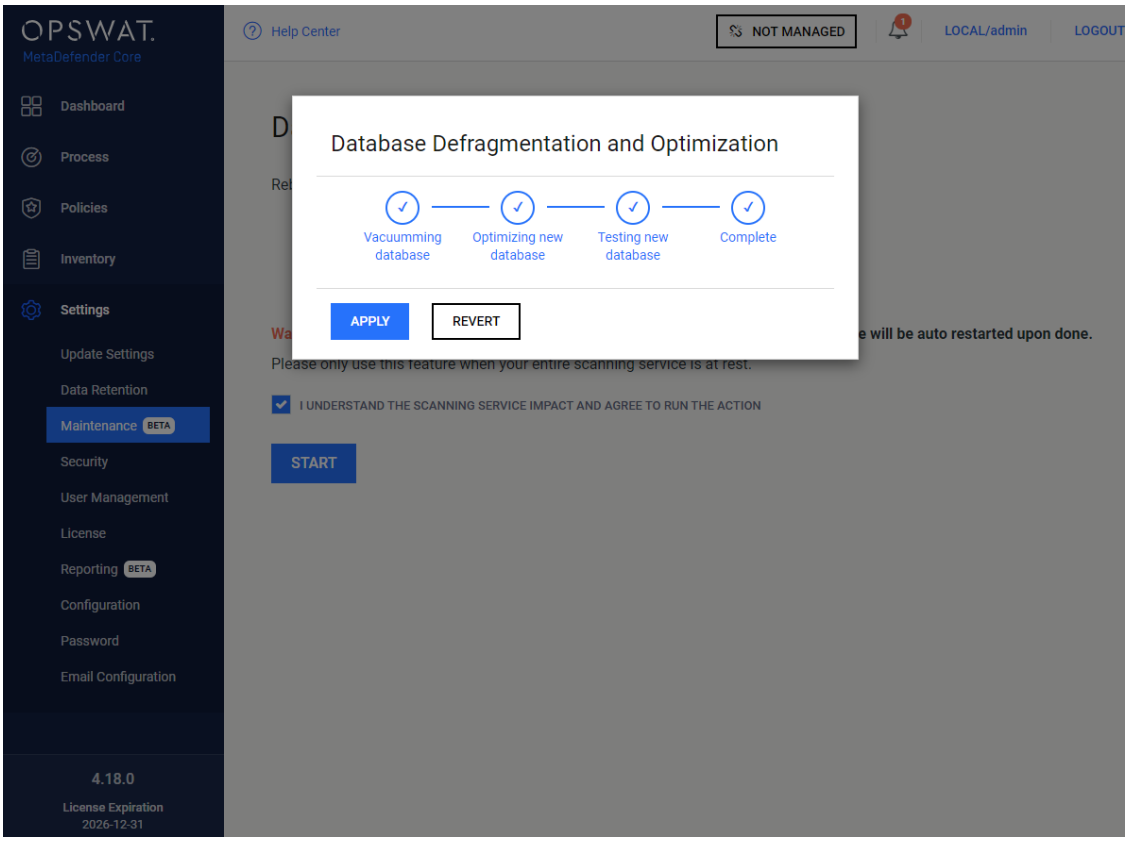
When your scan database grows big, it might cause performance degradation (e.g. timeout on client requests).

Since MetaDefender Core 4.18.0, admin user will be notified on the UI (also in product logs) when the processing history database file size exceeds over 10 GB.

Admin user is supported to perform database maintenance including defragmentation and optimization without loss of actual scan data under **Settings > Maintenance page**. As a result, your database file size could be reduced which helps boost processing performance tremendously over usage time.

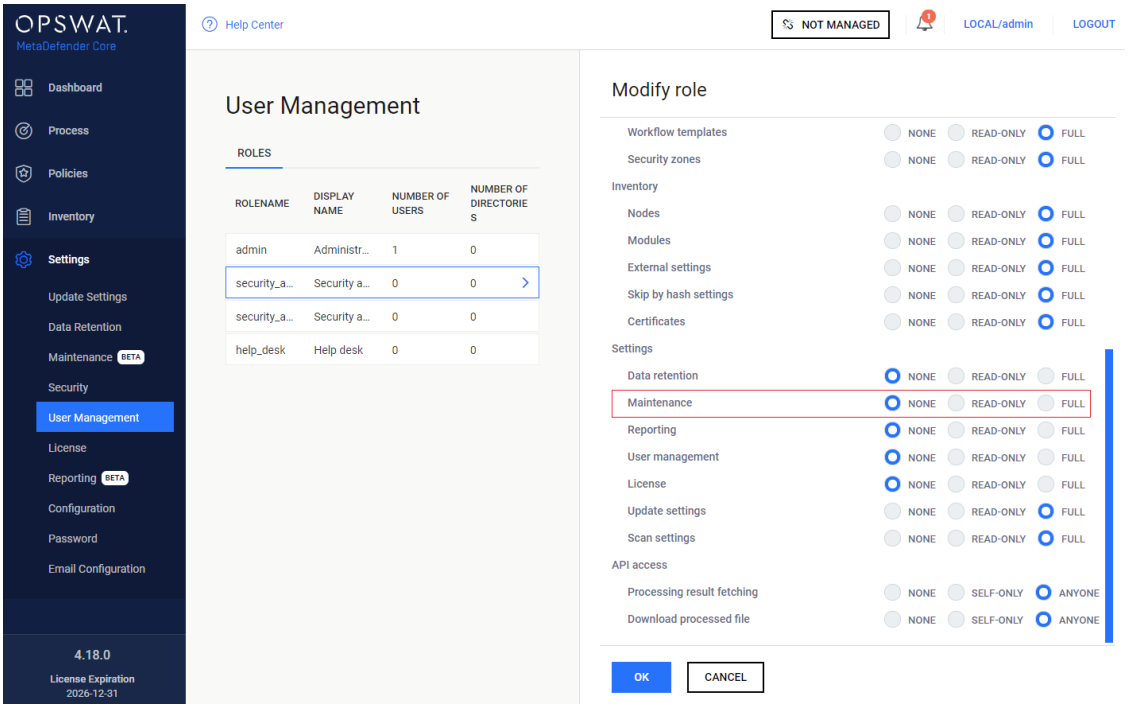
Caution: Please only use this feature when your entire scanning service is at rest.





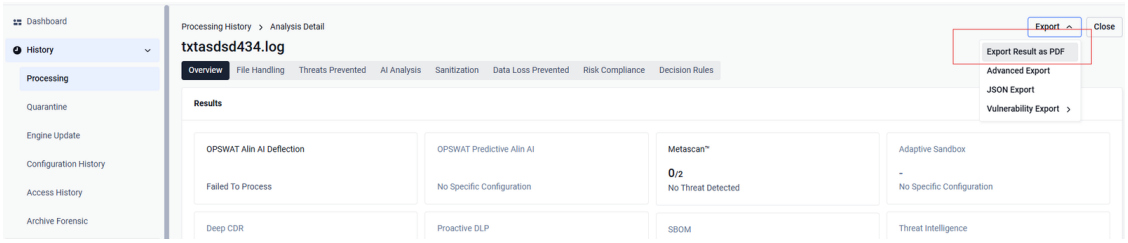
Hitting “APPLY” to apply all changes and restart MetaDefender Core service (or “REVERT” to roll back changes).

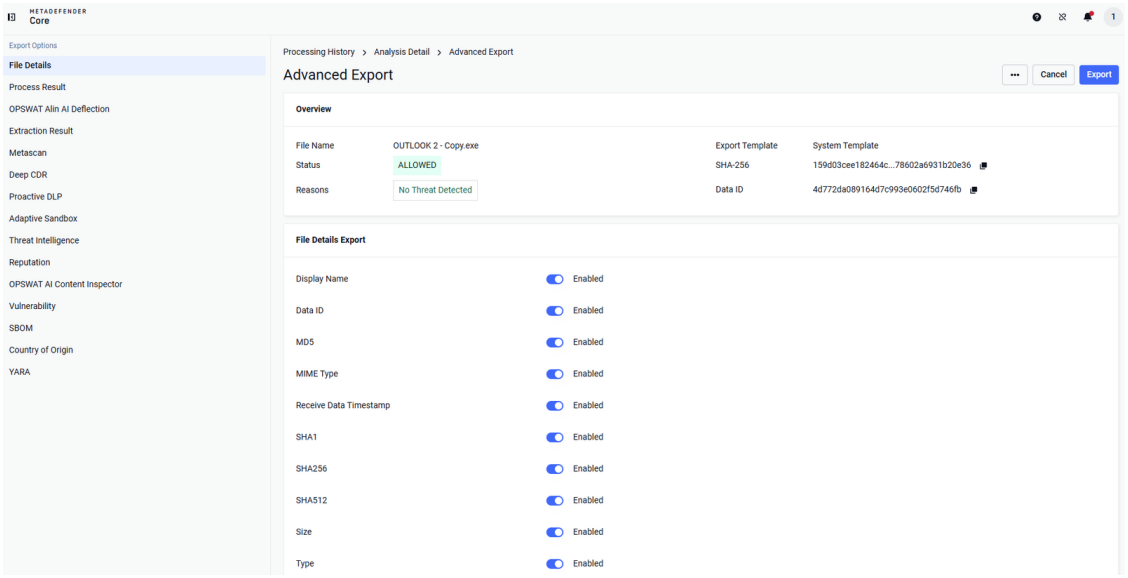
By default, only the “admin” user role can see this Maintenance page and make the changes, but you can configure to grant access to other user roles if desired.



Download single scan report as PDF

Starting with MetaDefender Core 4.19.1, admin users or scan owners are supported to download a detailed scan report in PDF format on any processed individual file.





A sample scan report:

OPSWAT.
MetaDefender Core

PROCESSING RESULT REPORT

File Details

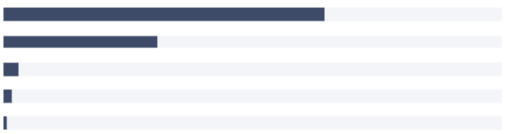
Data ID	2e980fc5b48a4cb4a0a0fc7db1cc228f
Display Name	txtasdsd434.log
Size	3019425 bytes
Type	Log file
Type ID	LOG
MIME Type	text/log
Receive Data Timestamp	July 28, 2026 at 10:05:44 AM
Upload Time	75 ms
MD5	6192d95c26f4a04b7160130ec092229a
SHA1	b70b1f3921bcf10d14105d06f34a768c82f94fee
SHA256	f699f5ee2cd5ea8efab5febe1a62d30acf65023b874cab34ea3aa922fe692821
SHA512	-

Process Result

Result	ALLOWED
Verdict	No Threat Detected
Processing Time	205 ms
Workflow Rule	File process
Hash Time	16 ms

Processing Time Details

Filetype	132 ms
Metascan™	63 ms
Vulnerability assessment	6 ms
Others	3 ms
Filetype waiting	1 ms



OPSWAT Alin AI Deflection

Result	OPSWAT AI Failed

METADEFENDER CORE
Version 5.21.0

1

OPSWAT.
MetaDefender Core

Metascan

Total AV Engines	2
Start Time	July 28, 2026 at 10:05:44 AM
Result	No Threat Detected

Engine Information	Scan Result	Definition Time	Scan Duration
AhnLab Version: 3.30.0.6-2634 Database: 2026.06.09.03	No Threat Detected	June 09, 2026 at 07:00:00 AM	2 ms
Avira Version: 4.15.29-2531 Database: 1781046694	No Threat Detected	June 10, 2026 at 04:29:00 AM	61 ms

Deep CDR

Result	Sanitization Not Configured

Proactive DLP

Result	Proactive DLP Not Configured

Adaptive Sandbox

Result	Adaptive Sandbox Not Configured

Threat Intelligence

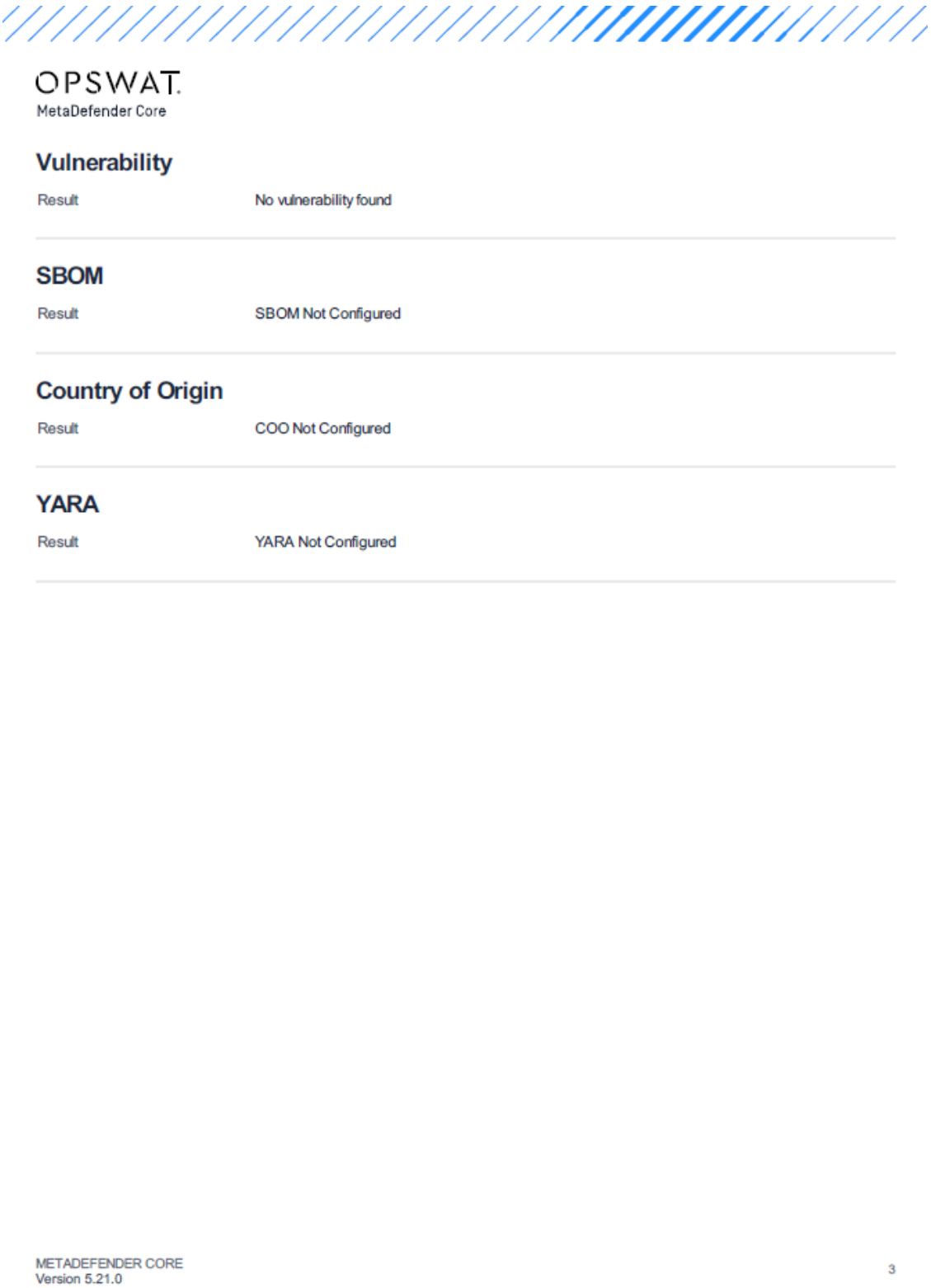
Result	Threat Intelligence Not Configured

Reputation

Result	Reputation Not Configured

OPSWAT AI Content Inspector

Result	OPSWAT AI Content Inspector Not Configured



OPSWAT.
MetaDefender Core

Vulnerability

Result	No vulnerability found

SBOM

Result	SBOM Not Configured

Country of Origin

Result	COO Not Configured

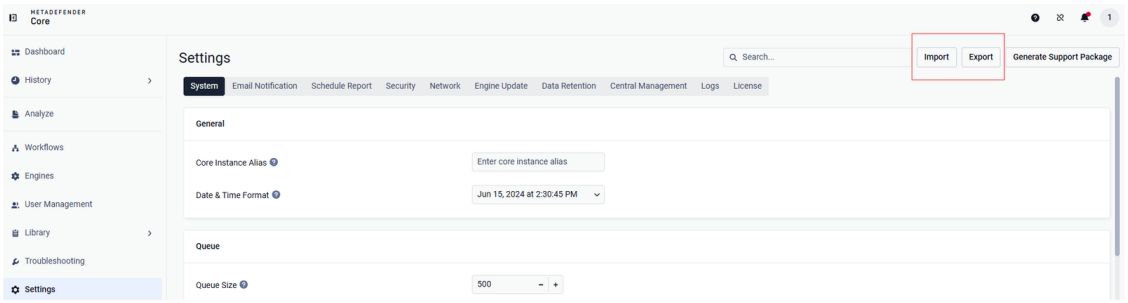
YARA

Result	YARA Not Configured

METADEFENDER CORE
Version 5.21.0 3

Import/Export configuration

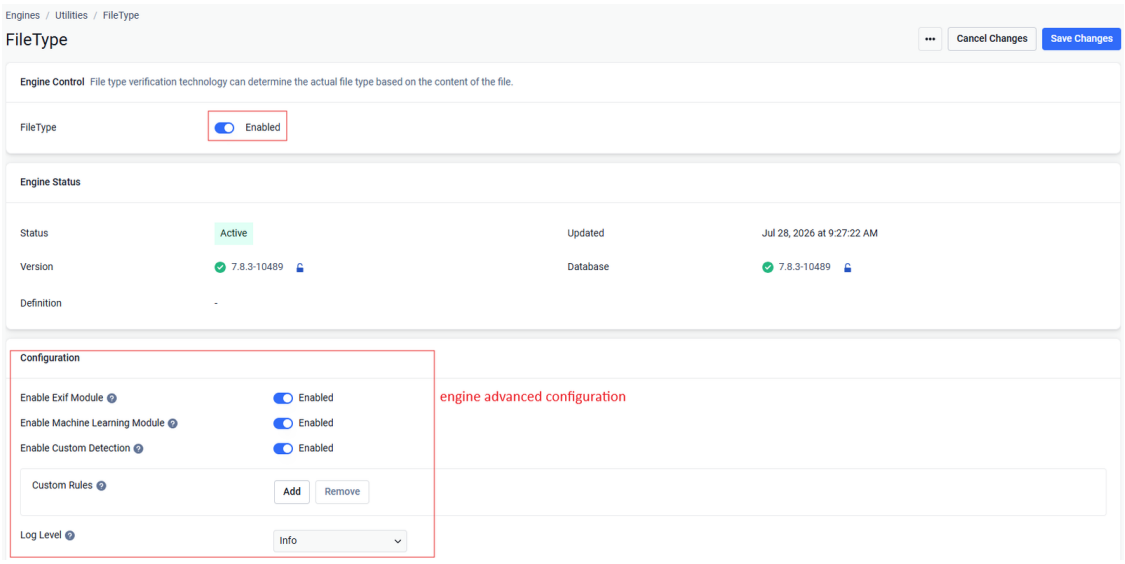
MetaDefender Core current configuration can be exported or a new one can be imported under the **Settings > Top Right Corner > Import or Export**

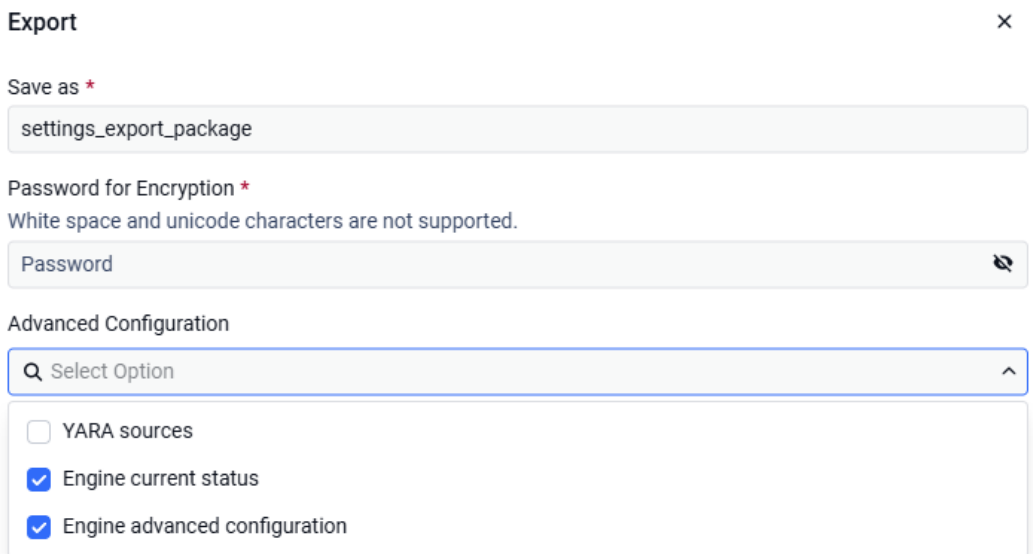


Export

Click the export button to set the password to encrypt the archive package for exported configurations. This package will contain the following configurations:

- Workflow rules and workflow templates and security zones
- Configuration in the **Library** menu
  - Skip by Hash
  - External Scanners
  - Certificates
  - Post Actions
  - Password storage
- Almost settings on **Settings**:
  - General: queue size and Login banner
  - Email notification: all
  - Security: Password policies, Session policies, Scan from link settings
  - Network: Email server, Proxy
  - Module update: Update source, Engine dependency
  - Data Retention: Automatic deletion
  - Health Check: all
  - Central Management: all
  - Usage Report
- Users management settings: users, role, directories
- Engine status and engine advanced configuration: these options are **OFF by default,** user need to select these options if want to export:
  - Engine current status: the current status of the engines: Enable or Disable
  - Engine advanced configuration: The advanced configuration of the engines - user can find it in the Engines > Select engine, for example FileType





Yara sources setting: **OFF by default**, user need to select YARA sources in Advanced configuration when export



Why password is required?

The exported archive package must be set with a password to encrypt because it might contain sensitive data including user information.



REST API for configuration export

Details: POST - Export configuration (new version)

Import

Choose the exported package and also input proper password to import.

JSON config file import supported

Besides package import support, you can also import JSON config file like in MetaDefender Core v4.

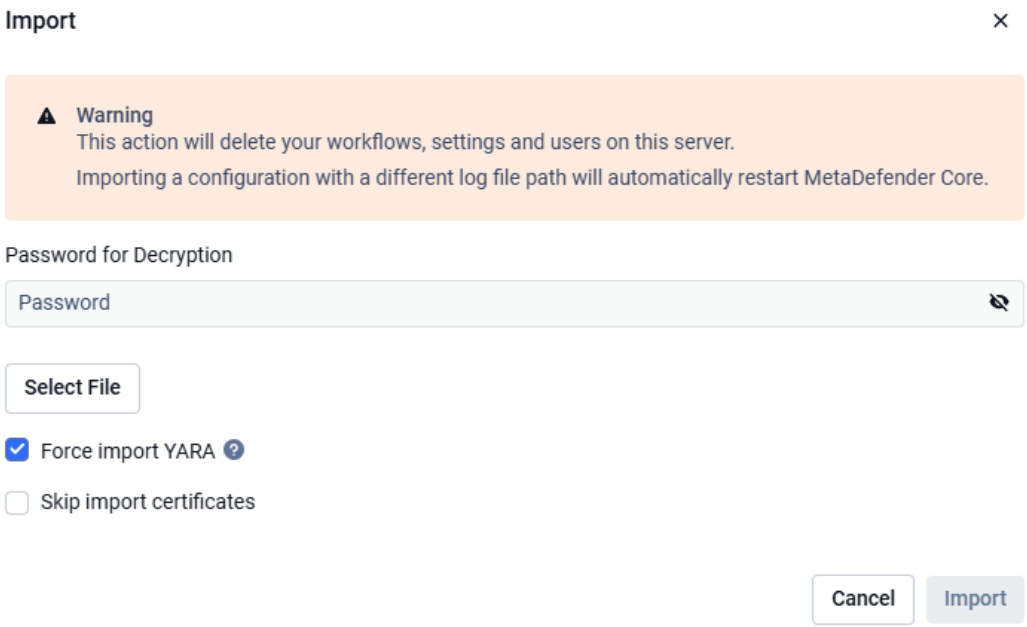
JSON config file is only for MetaDefender Core workflow settings.

After the successful import, the new configurations will replace the current one.

REST API for configuration import

Details: POST - Import configuration (new version)

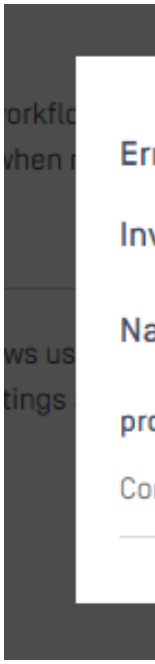
YARA sources will be not imported by default. User need to select `Force import YARA` to import it



Troubleshooting

Occasionally, errors may occur during the configuration import process. This section aims to provide a clear understanding of the issue.

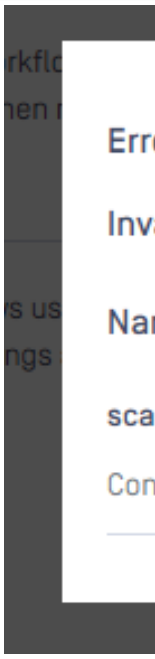
Mandatory engines are missing

Name	Description
Error details	
Root-cause	The importe

Name	Description
	General > **Ma** there is no su engines, Filet
Remediation	There are 2 c • Option 1: U • Option 2: T file" at the • Open the f and remov 

Excluded engines are missing

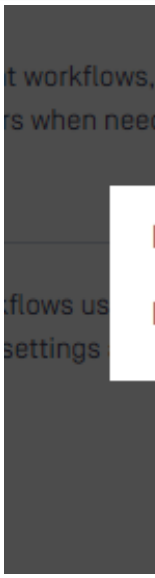
Name	Description
Error details	

Name	Description
	
Root-cause	The importe **Metascan** > *E* Core, there is engines.
Remediation	There are 2 c • Option 1: U • Option 2: T file" at the • Open the f remove the

Name	Description
	

The required engines in Health Check are missing

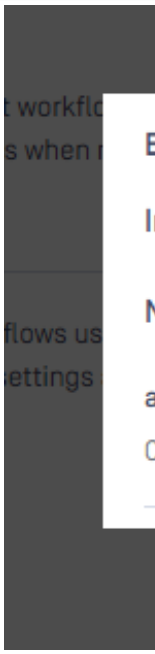
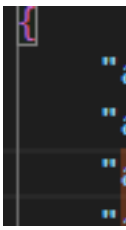
Name	Description
Error details	

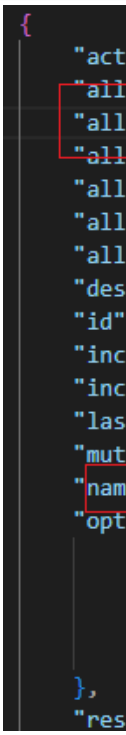
Name	Description
	
Root-cause	The importe example, in t Core, the lice
Remediation	There are 2 c • Option 1: U • Option 2: T file" at the • Open the f example:

Name	Description
	

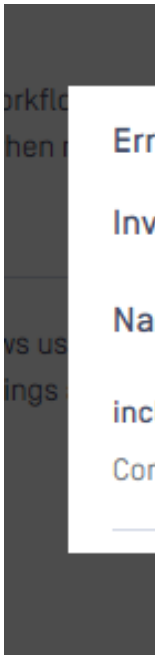
The batch cert that using in the workflow is missing

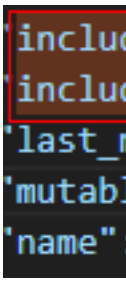
Name	Description
Error details	

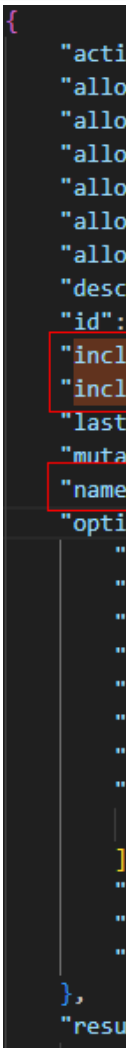
Name	Description
	
Root-cause	The importe **General** > **Ge** current MD C
Remediation	There are 2 c • Option 1: U configurati • Option 2: T file" at the • Open the f **allow_cert** 

Name	Description
	

The webhook signature using in the workflow is missing

Name	Description
Error details	

Name	Description
Root-cause	The importe **General** > **We** Library > We
Remediation	There are 2 o • Option 1: U • Option 2: T file" at the • Open the f to **false**, "**i** 

Name	Description
	

Manually modify the zip file

> **We do not recommend modify the zip file manually**
>
> Please get in touch with OPSWAT for support if you have any difficulty in modifying the zip file

1. Before do anything, please back up the current file.
2. Extract the zip file, it will contain 3 files inside:
    1. wf_config_export.json workflow rules/template related settings.
    2. settings_export.json general configuration.
    3. skip_list.data data from the 'skip by hash' list.

3. Open the JSON file that you want to modify
4. After the modification, save the file. **Re-zip the file again with password. We highly recommend using 7-Zip to compress the file**

Archive Forensic

When a scan detects a problematic file buried inside an archive, isolating that specific file can be difficult. Archive Forensic solves this by automatically capturing flagged sub-files during archive processing and storing them in a dedicated, secure location for later retrieval.

How it works

When turned on, MetaDefender Core selectively preserves child files that match configurable trigger verdicts during archive extraction — up to a configurable limit per archive, selected on a first-come-first-served basis.

> **Info**
>
> Office documents like DOCX and XLSX are internally structured as ZIP archives containing XML parts, media, and metadata. When the parent file is an Office document, all child files are intentionally skipped by Archive Forensics to avoid flooding the forensic store with document internals.

Archive Forensic vs. Quarantine

Archive Forensics and Quarantine are both storage features in MetaDefender Core, but they serve different purposes and operate at different levels. The table below highlights the key differences.

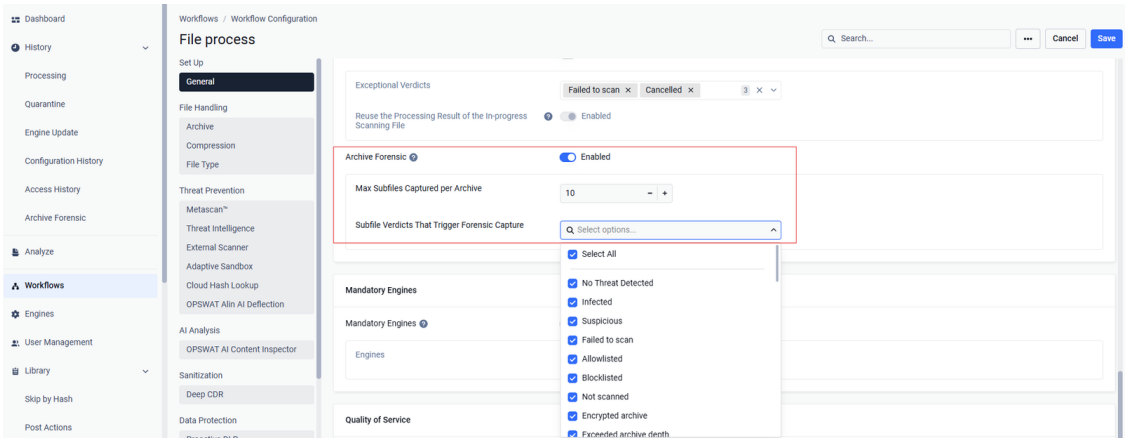
Aspect	Archive Forensic	Quarantine
What gets stored	Only the individual subfiles inside an archive that triggered a configured verdict — not the full archive.	The entire original file (including the whole archive if the archive is blocked).
Scope	Archives only (ZIP, RAR, 7z, ISO, etc.). Standalone files that aren't part of an archive are not captured, and child files within office documents are skipped.	Any file type — such as archives, emails, documents, and others.
Encryption at rest	Always encrypted with AES-256-GCM before writing to disk.	Configurable; by default, files are stored with a salt.
Download	Always a password-protected ZIP. No plain download is available.	Direct download or password-protected ZIP.

How to use

> **Warning**
>
> Enabling the feature may introduce additional resource consumption on the MetaDefender Core, and increased storage usage that grows with scan volume and retention window. In high-throughput environments, it may lead to noticeable performance degradation.

Enable Archive Forensic:

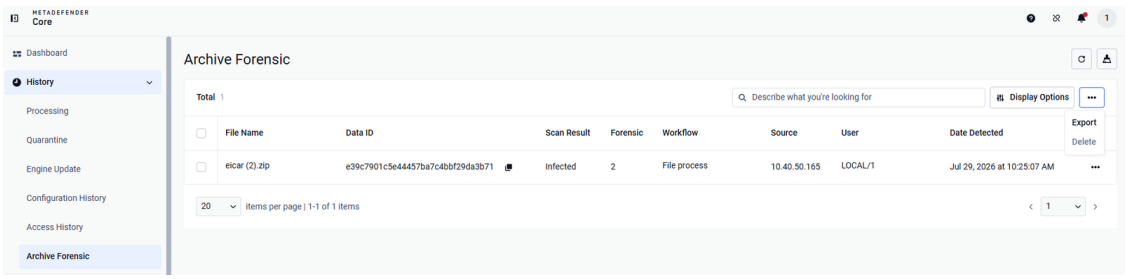
This feature is disabled by default. You can enable it per scanning rule by selecting the **Archive forensic** checkbox under the **General** tab.



View the forensic captures

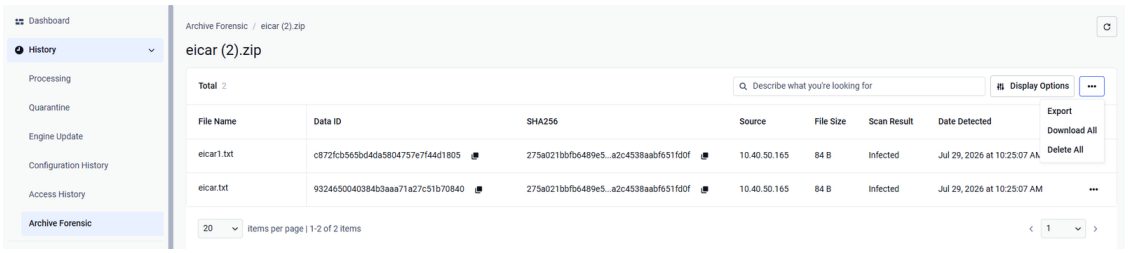
Navigate to **History** > **Archive Forensic** to view the list of captured root archives, along with it's data ID, scan result, workflow rule, and the user who triggered the scan. Actions support for this page:

- Search for a root archive by name.
- Search for a root archive or captured child file by data ID.
- Download all captured child files of a root archive.
- Trigger an on-demand cleanup action.
- Export results.



To view the contents of a root archive, hover over and click the archive's row. A new page will appear, listing all files captured inside that archive. Actions support for this page:

- Search for a child file by name or data ID.
- Download all captured child files.
- Download a single captured child file.
- Delete all data for the current archive.
- Export results.



Info

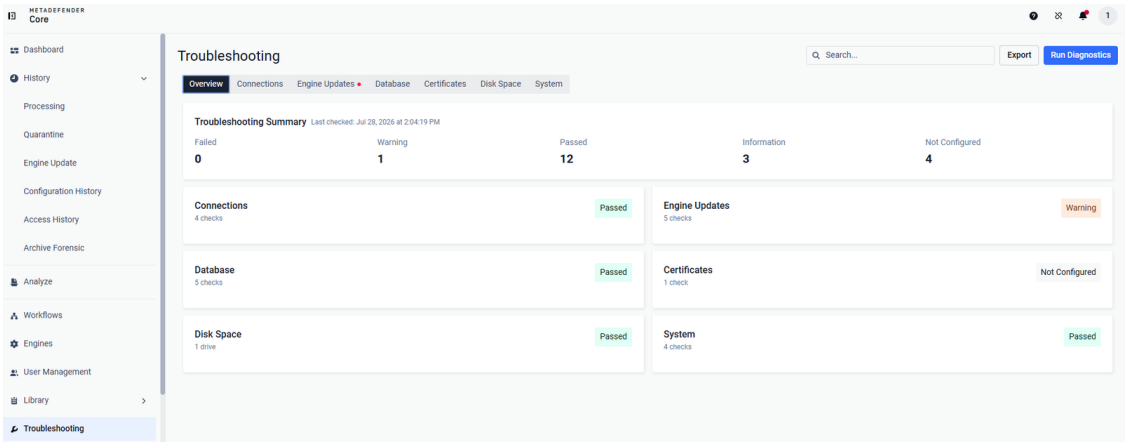
Downloading a file always requires a password. The file is compressed into a password-protected ZIP, which prevents local antivirus software from accidentally removing it.

Auto Troubleshooting

What is it?

Troubleshooting is a built-in health dashboard in MetaDefender Core. It runs automated checks on your deployment and shows the results in one place — so you can quickly see whether Core, its engines, database, and integrations are healthy, without digging through logs or running manual tests.

The page refreshes automatically in the background. You can also run a deeper on-demand scan when you need a full picture.



How to read the page

Overall status

At the top, a summary banner shows the overall health:

- All checks passing — nothing is failing or warning.
- Warning / Failed — at least one check needs attention.

The donut chart and counts break down results:

Status	Meaning
Passed	Check ran and the system looks healthy.
Warning	Something works but could be improved.
Failed	A real problem was found — review the message and remediation.
Not Configured	An optional feature is not set up (e.g. email server, OCM). This is not a failure.

> *For example, 11 Passed and 4 Not Configured with 0 Failed means the system is healthy. The gray "Not Configured" items are optional settings you have not enabled.*

Categories

Checks are grouped by area. Expand a category to see individual checks and details.

Category	What it covers
Connections	Update source, proxy, email server, OPSWAT Central Management (OCM)
Engine Updates	Active engines, definition freshness, update download errors, dependencies, failed packages
Database	PostgreSQL status, connection count (Full run adds size and maintenance checks)
System	CPU, memory, disk I/O, process memory (Core, engines, PostgreSQL)
Certs	Managed certificate expiration
Disk Space	Disk usage across Core data paths *(Deep Analysis*

Category	What it covers
	only)

Each check shows a message (what was found) and, when applicable, remediation steps to fix a failure.

Automatic vs. Deep Analysis

	Automatic checks	Deep Analysis
When	Every ~5 minutes in the background	On demand — click Deep Analysis
Scope	Quick/lightweight checks	Full set, including disk space and deeper database checks
Use when	Day-to-day monitoring	Investigating an issue, before/after changes, or for support

Scan result codes and explanations

These are the possible values returned for file scans. These values appear in *scan_all_result_i* and *scan_all_result_a*:

Info

This article explains how to obtain the possible scan results: How can I run tests to see the different scan results on MetaDefender Core?

Code	Scan result	Relation	Description
0	No Threat Detected	Metascan engine	No threat detection or the file is empty.
1	Infected	Metascan engine	One or more threats have been found.
2	Suspicious	Metascan engine	Classified as a potential threat without a specific identification.
3	Failed	Metascan engine / MetaDefender Core	The scan failed due to the system or application error.
7	Allowlisted	MetaDefender Core	The scan is skipped for good because one of file characteristics meet certain pre-configured conditions (based on hash / file type / file name etc.).
8	Blocklisted	MetaDefender Core	The scan is skipped for bad because one of file characteristics meet certain pre-configured conditions (based on hash / file type / file name etc.).
9	Exceeded Archive Depth	Archive Extraction engine	Reached archive handling limit for archive

Code	Scan result	Relation	Description
			depth level (Max recursion level).
10	Not Scanned	Metascan engine	The scan is not performed by Metascan engines either due to engine updates or other engine-specific failure reasons. If Metascan is disabled, this will be the final result.
12	Encrypted Archive	Archive Extraction engine	The scan is not performed because the file type is detected as an encrypted (password-protected) archive but without having a correct password to decrypt.
13	Exceeded Archive Size	Archive Extraction engine	Reached archive handling limit for archive size limit (Max total size of extracted files).
14	Exceeded Archive File Number	Archive Extraction engine	Reached archive handling limit for archive file size limit (maximum number of files extracted).
15	Password Protected Document	Archive Extraction engine	A document that is protected by a password [e.g., Office documents or PDFs that require a password to view its contents] but without having correct password to decrypt.
16	Exceeded Archive Timeout	Archive Extraction engine	The archive extraction process on a certain

Code	Scan result	Relation	Description
			archive file reached the given timeout value (Archive analysis timeout).
17	Mismatch	File type analysis engine	The file extension is not a part of the allowed extensions associated with the true file type.
18	Potentially Vulnerable File	File-Based Vulnerability Assessment	A Potentially Vulnerable File is a file associated with vulnerable components or applications identified by OPSWAT's File-Based Vulnerability Assessment technology.
19	Cancelled	MetaDefender Core	The scan is canceled because the client terminated the connection or explicitly submitted a cancellation request.
20	Sensitive Data Found	Proactive DLP	Sensitive Data Found file detected by Proactive DLP refers to a file that contains information classified as sensitive according to predefined criteria set by the organization's data security policies.
21	Yara Rule Matched	YARA engine	File content matched at least one of the pre-defined YARA rules.
22	Potentially Unwanted	Metascan engine	Potentially unwanted application (PUA) classified by Metascan

Code	Scan result	Relation	Description
			AV engines.
23	Unsupported File Type	Metascan engine	The engine does not support scanning this file type. Certain Metascan AV engines such as AI-based engines only scan specific file types such as executable files or documents. https://docs.opswat.com/mdcore/metascan-engines/supported-file-type-for-ai-engines.
24	Extraction Failed	Archive Extraction engine	Archive extraction failed due to some reasons which could be found under the archive processing section (e.g. insufficient disk space, file content data error, etc.).
26	Suspicious Verdict by Sandbox (displayed as **Low Risk Verdict by Adaptive Sandbox** on the MD Core UI starting from version 5.17.1)	Adaptive Sandbox	When Adaptive Sandbox labels a file with a "Suspicious" verdict, it means that the file exhibited behaviors or characteristics that are commonly associated with malware or other malicious activity but are not conclusively harmful. Starting from MD Core version 5.17.1, the Core UI will display **Low Risk Verdict by Adaptive Sandbox** instead of **Suspicious Verdict by Sandbox**. The verdict in

Code	Scan result	Relation	Description
			the JSON response will still remain **Suspicious Verdict by Sandbox**.
27	Likely Malicious Verdict by Sandbox (displayed as **High Risk Verdict by Adaptive Sandbox** on the MD Core UI starting from version 5.17.1)	Adaptive Sandbox	Adaptive Sandbox has identified behaviors or characteristics in the file that strongly suggest it is harmful, but the evidence is not entirely conclusive to label it as definitely malicious. Starting from MD Core version 5.17.1, the Core UI will display **High Risk Verdict by Adaptive Sandbox** instead of **Likely Malicious Verdict by Sandbox**. The verdict in the JSON response will still remain **Likely Malicious Verdict by Sandbox**.
28	Malicious Verdict by Sandbox (displayed as **Confirmed Threat Verdict by Adaptive Sandbox** on the MD Core UI starting from version 5.17.1)	Adaptive Sandbox	Adaptive Sandbox has definitively determined that the file is harmful. Starting from MD Core version 5.17.1, the Core UI will display **Confirmed Threat Verdict by Adaptive Sandbox** instead of **Malicious Verdict by Sandbox**. The verdict in the JSON response will still remain **Malicious Verdict by Sandbox**.
29	Blocked Verdict by Sandbox	Adaptive Sandbox	This indicates that the Adaptive Sandbox has determined that this file is dangerous and needs

Code	Scan result	Relation	Description
			to be blocked by the Core. This verdict appears when a file is blocked but does not fall under any predefined verdicts mentioned above. The result also depends on the configuration of the Adaptive Sandbox.
30	Blocked Verdict by Deep CDR	Deep CDR	It indicates that a file was blocked based on a decision by the Deep CDR engine. This occurs when the user has enabled the file blocking feature in the Deep CDR configuration. For example, users can enable the option to block files if Office documents contain hyperlinks, embedded objects, QR codes, etc.
31	Global Timeout Exceeded	MetaDefender Core	The global processing timeout takes precedence over any engine timeout and is disabled by default. Certain scans may be terminated when enabled if they exceed the pre-defined global processing timeout.
32	Vulnerable Verdict by SBOM	Bill of Materials	This indicates that software components contain known vulnerabilities that could potentially be exploited and therefore may require attention or

Code	Scan result	Relation	Description
			remediation. Please note that this case may also include a License Risk Found. You should check the SBOM details for more information.
33	Non-vulnerable Verdict by SBOM	Bill of Materials	This indicates that software components have been evaluated and found to be free of security weaknesses based on the current vulnerability data.
34	Blocked Verdict by SBOM	Bill of Materials	This indicates that the SBOM has determined that software components have security weaknesses or risks and need to be blocked by the Core. This verdict appears when the item is blocked but does not fall under any of the predefined verdicts. The result also depends on the configuration for the SBOM.
38	Known Bad	Reputation engine	The Reputation engine found the file as a known (recognizable) harmful one. For more details, the Reputation Engine cross-references file hashes with a database of known malicious files and utilizes advanced analysis to quickly address false positives.

Code	Scan result	Relation	Description
			For further information, please refer to this link: Metadefender Core v5.9.0: Powerful New Engines for Proactive Threat Detection. Please note that the Reputation Engine is not enabled by default.
39	Known Good	Reputation engine	The Reputation engine found the file as a known good one. For more details, the Reputation Engine cross-references file hashes with a database of known malicious files and utilizes advanced analysis to quickly address false positives. For further information, please refer to this link: Metadefender Core v5.9.0: Powerful New Engines for Proactive Threat Detection. Please note that the Reputation Engine is not enabled by default.
40	Unknown	Reputation engine	Reputation engine does not come to final conclusion to determine if the file is harmful or not (inconclusive).
41	Allowed Verdict by COO	Country of Origin	Based on the current configuration, the Country of Origin engine indicates that the file comes from an allowed source.

Code	Scan result	Relation	Description
42	Blocked Verdict by COO	Country of Origin	Based on the current configuration, the Country of Origin engine indicates that the file comes from an unallowed source.
45	License Risk Verdict by SBOM	Bill of Materials	This indicates that the software components present one or more license risks. Please note that this case may have Vulnerabilities Found, but they do not meet the pre-configured threshold to impact the decision to block the file. You should check the SBOM details for more information.
46	Blocked Verdict by File Type	File type analysis engine	Other scenarios, apart from Mismatch, where a file is flagged as blocked by the FileType engine. This is dependent on the specific configuration of the FileType engine.
47	Multipart Upload Timed Out	MetaDefender Core	Cases where multipart uploading is used to upload processed files to MetaDefender Core, and this upload action times out.
48	Sanitized	Deep CDR	When the `Block files if sanitized successfully` setting is enabled in the Deep CDR workflow settings, and the file is successfully sanitized, the file will be

Code	Scan result	Relation	Description
			blocked. This verdict has the lowest priority compared to other block verdicts.
51	Exceeded Limited Sandbox File Size	Adaptive Sandbox	When the `Block files that exceed the file size threshold` setting is enabled in the Adaptive Sandbox workflow settings, the file will be blocked if its size exceeds the configured threshold.
52	Blocked Verdict by Archive Extraction	Archive Extraction engine	The file has been blocked based on the recommendation from the Archive Extraction engine. The reason for the block is detailed in the Archive Extraction scan result.
53	Blocked Verdict by Archive Compression	Archive Compression engine	The file has been blocked based on the recommendation from the Archive Compression engine. The reason for the block is detailed in the Archive Compression scan result.
54	Sandbox Execution Limit Reached	Adaptive Sandbox	When the `Block files if execution limit reached` setting is enabled in the Adaptive Sandbox workflow settings, the file will be blocked if the license execution limit is reached.

Code	Scan result	Relation	Description
55	Rejected	MetaDefender Core	The request was rejected due to policy, validation, or configuration constraints.
56	Empty Split Archive	Archive Extraction engine	The reassembled archive contains no nested file.
57	Split Archive Upload Timed Out	Archive Extraction engine	The split-archive upload session exceeded the configured time window.
58	Unsupported Split Archive Type	Archive Extraction engine	The archive format of the split upload is not supported.
59	Split Archive Not Supported	Archive Extraction engine	The Archive Extraction engine version does not support split-archive.
60	Exceeded Archive Expansion Factor	Archive Extraction engine	Reached the archive handling limit due to the expansion factor limit (Max expansion factor).
61	Blocked Verdict by Threat Intelligence	Threat Intelligence	The file has been blocked based on the Threat Intelligence engine's recommendation. The reason for the block is detailed in the Threat Intelligence scan result.
62	Allowed Verdict by Threat Intelligence	Threat Intelligence	The file being allowed by Threat Intelligence is a rare verdict in practice, unless you explicitly configure the system to block files when this verdict occurs.

Code	Scan result	Relation	Description
63	AI Generated	OPSWAT AI Content Inspector	The analysis engine did find sufficient evidence that the content was generated by artificial intelligence.
64	Not AI Generated	OPSWAT AI Content Inspector	The analysis engine did not find sufficient evidence that the content was generated by artificial intelligence.
65	AI Detection Uncertain	OPSWAT AI Content Inspector	The analysis engine could not confidently determine whether the content was AI-generated or human-written.
66	RMS Protected Document	Archive Extraction engine	A document that is protected by Microsoft Rights Management Services (RMS) and can not be read for processing.
67	OPSWAT AI Confidently Clean	OPSWAT Alin AI Deflection	The AI deflection engine is confident that the file is clean. Depending on the workflow rule's *Deflect on* setting, MetaDefender Core skips the configured downstream engines for this file instead of running the full analysis.
68	OPSWAT AI Confidently Malicious	OPSWAT Alin AI Deflection	The AI deflection engine is confident that the file is malicious. Depending on the workflow rule's *Deflect on* setting,

Code	Scan result	Relation	Description
			MetaDefender Core skips the configured downstream engines and the file is treated as a threat.
69	OPSWAT AI Undetermined	OPSWAT Alin AI Deflection	The AI deflection engine could not reach a confident verdict for the file. The file continues through the normal analysis pipeline (no engines are deflected unless *Deflect on* explicitly includes "unknown").
70	Download Timeout	MetaDefender Core	The file was submitted by URL and MetaDefender Core could not finish downloading it within the configured download timeout. No engine ran on the file.
71	Download Failed	MetaDefender Core	The file was submitted by URL and the download failed (for example the remote server was unreachable, returned an error, or the link was invalid). No engine ran on the file.
72	Allowed Verdict by File Structure Validation	File Structure Validation	The file's internal structure was analyzed and no structural anomaly or policy violation was found; the file is allowed by the File Structure Validation engine.

Code	Scan result	Relation	Description
73	Blocked Verdict by File Structure Validation	File Structure Validation	The file's internal structure violates the File Structure Validation policy (malformed, non-conformant, or containing disallowed structural elements) and the file is blocked.
74	Exceeded Structural Depth	File Structure Validation	Embedded objects produced by File Structure Validation were nested deeper than the configured *Max nesting level* (default 5). Objects beyond this depth are not processed.
75	Exceeded Structural Object Count	File Structure Validation	A single parent object produced more embedded objects than the configured *Max number of output objects* (default 200). The limit applies per parent object at each nesting level, not to the whole file; the remaining objects of that parent are not processed.
76	Exceeded Structural Object Size	File Structure Validation	The total size of the embedded objects produced by a single parent object exceeds the configured *Max output size* (default 200 MB). The limit applies per parent object at each nesting level, not to the whole file; the remaining objects of that parent are not

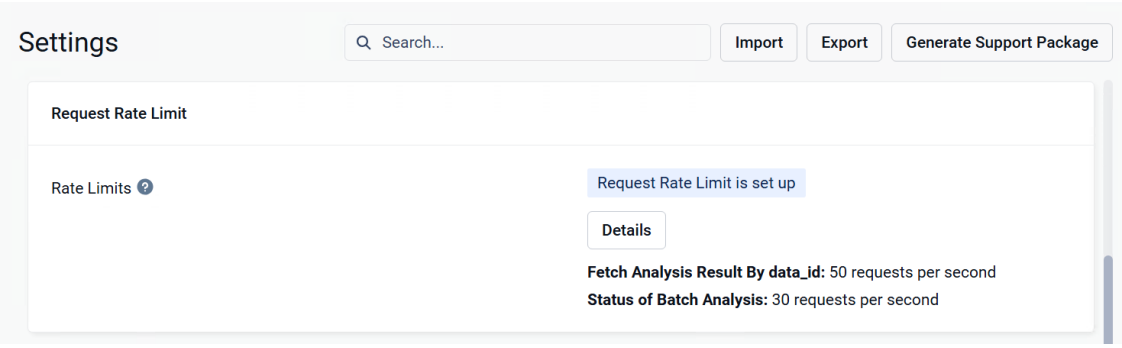
Code	Scan result	Relation	Description
			processed.
77	Component Risk Found	Bill of Materials	One or more software components identified by BOM pose a risk (such as known vulnerabilities, license risks, unsafe cryptography or AI components), and the workflow is configured to block on it.
78	No Component Risk Found	Bill of Materials	BOM analysis completed and no risk was found in the identified software components.
79	Processed	MetaDefender Core	No AV (Metascan) result is available, because Metascan was skipped or is not licensed, and "Exclude Metascan Not Scanned Verdict" is enabled. At least one other engine (e.g. Adaptive Sandbox, Deep CDR, Proactive DLP) processed the file successfully. This replaces "Not Scanned".
80	Not Processed	MetaDefender Core	"Exclude Metascan Not Scanned Verdict" is enabled, but no engine produced a usable result for the file.
81	Early Exit	MetaDefender Core	Processing stopped because another file extracted from the same archive was blocked by

Code	Scan result	Relation	Description
			a reason configured in Stop archive processing on Blocked Reasons. Files not yet extracted are not processed. The archive's final verdict is based on the worst result collected.
255	In Progress	MetaDefender Core	The scan is still in progress, not yet finished.
256	Multipart Uploading	MetaDefender Core	Parts of the file are being uploaded; processing has not started.
257	Split Archive Uploading	Archive Extraction engine	Split-archive parts are being uploaded; processing will begin after all parts are received.

Request rate limiting

MetaDefender Core has not limited the number of GET requests it serves per second thus far. Consequently, requests fetching results by data_id or batch_id can easily drain MetaDefender Core resources and directly impact the performance of scanning and processing files. OWASP considers the lack of rate limiting a type of vulnerability.

Since MetaDefender Core 5.13.0, request rate limiting is officially supported. Users can follow the steps here to set the request limits for APIs they concern.



To investigate the impact of the rate limit on system performance, the same dataset is tested against MetaDefender

Core 5.13.0 on the same environment for several scenarios.

Environment

Hardware	8vCPU, 24GB RAM, 150GB SSD
Operating system	Windows Server 2022

MetaDefender Core

Version	5.13.0
Max scan queue	4000
Engines	File type, Archive, AhnLab, Avira, Bitdefender, ESET, K7, Quick Heal, and Vir.IT Explorer

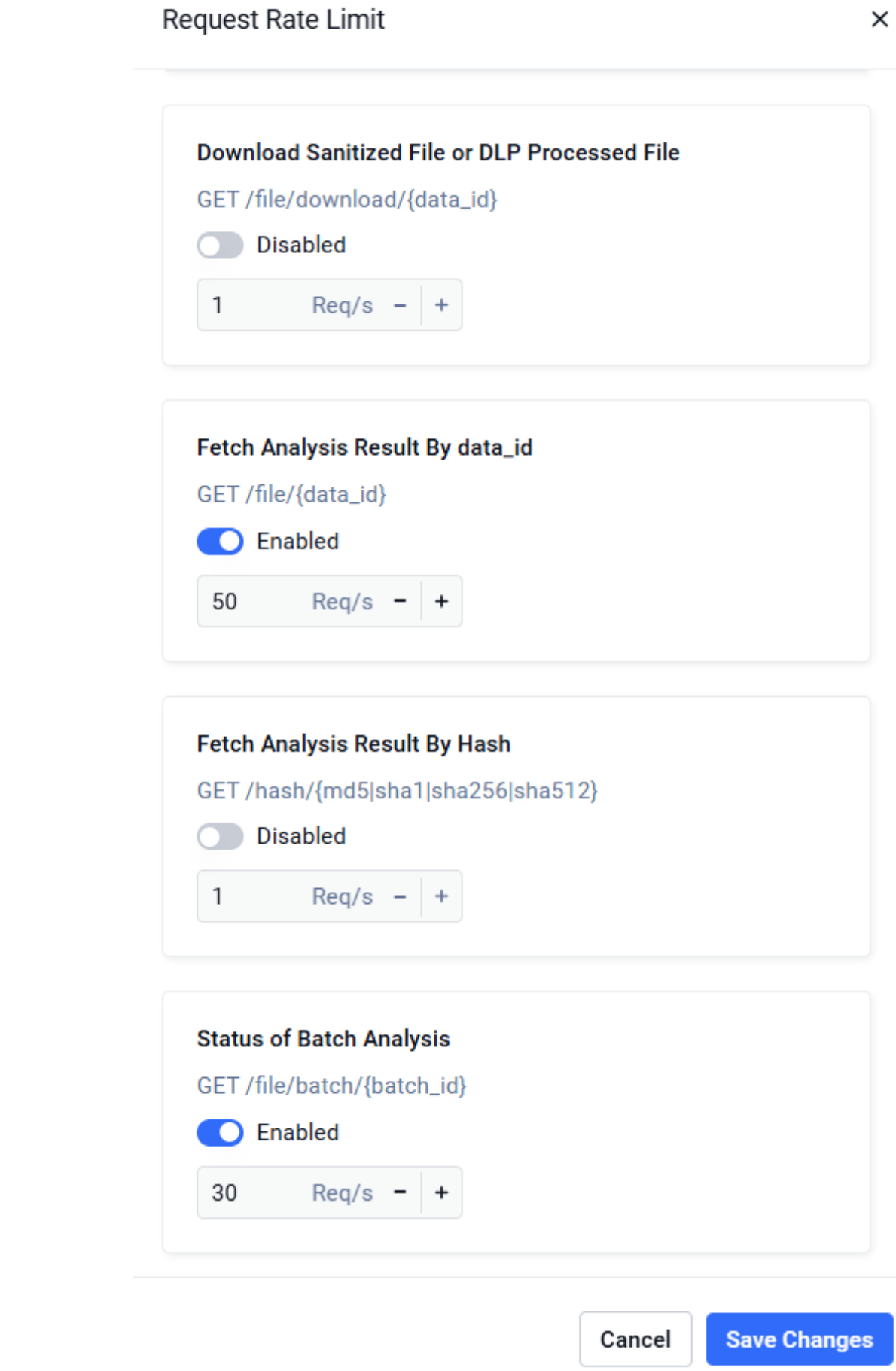
Data set

Total file	800
Total size	762 MB
File size	952 KB on average a minimum of 50KB a maximum of 27MB
File type	DOCX, XLSX, PPTX, XLTX

Scenarios

Case	Scenario
No-limit	No limit on fetching scan result
Rate-50	Only accept 50 fetching requests per second

Case	Scenario
Rate-20	Only accept 20 fetching requests per second
Rate-10	Only accept 10 fetching requests per second
Rate-1	Only accept 1 single fetching request per second
Webhook	Apply webhook for fetching result, no limit set


Request Rate Limit
Download Sanitized File or DLP Processed File
GET /file/download/{data_id}
Disabled
1 Req/s
Fetch Analysis Result By data_id
GET /file/{data_id}
Enabled
50 Req/s
Fetch Analysis Result By Hash
GET /hash/{md5|sha1|sha256|sha512}
Disabled
1 Req/s
Status of Batch Analysis
GET /file/batch/{batch_id}
Enabled
30 Req/s
Cancel
Save Changes

Info

The Webhook case includes header `callbackurl` in each submission request to MetaDefender Core. MetaDefender Core will respond with the full scan result to the server hosted at the location pointed to by `callbackurl` once the scan is complete, so no fetching mechanism is needed in this scenario.

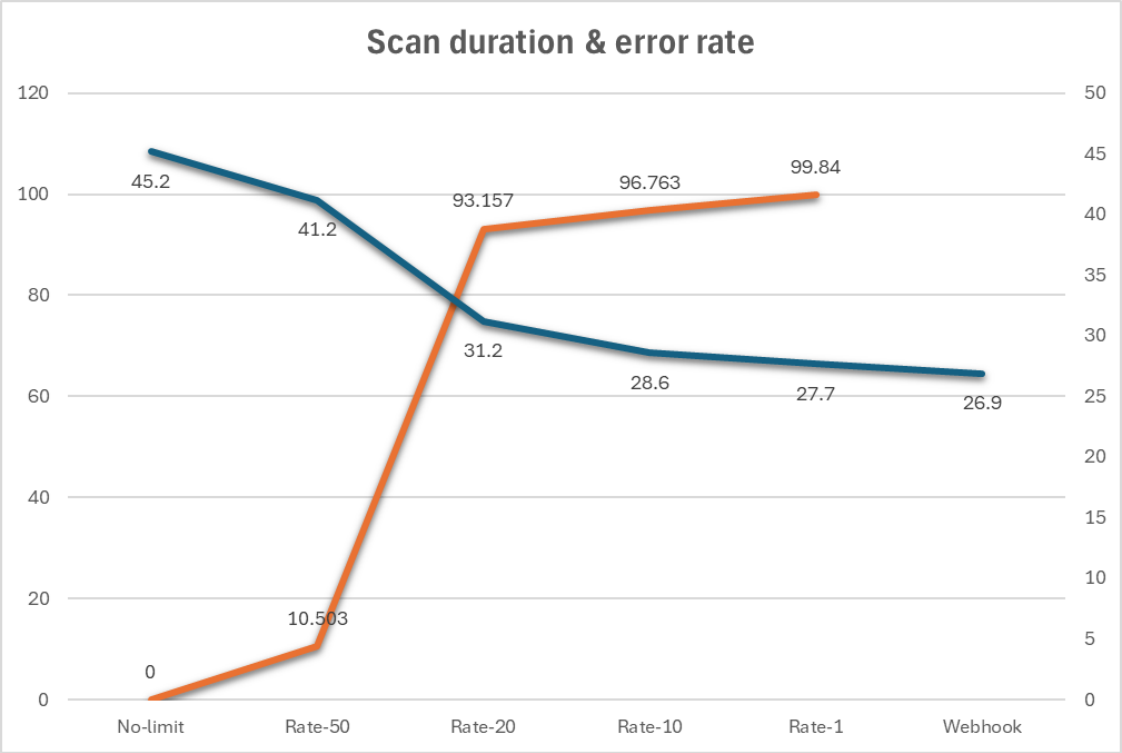
Reference here for more details about header `callbackurl`.

Test steps

- Install MetaDefender Core and its engines on a clean environment.
- Setup max scan queue and settings according to test scenario.
- Submit turns of 1 second apart separately to MetaDefender Core until 2000 files are submitted. Each turn includes 10 files submitted asynchronously to MetaDefender Core using API `POST /file`.
- For every 5 seconds, fetch results for all responded data_id asynchronously using `GET /file/{data_id}` until the scan completes or 503 code responded (except the webhook scenario).

Performance per scenario

Case	Duration (min)	Number of processed files per hour	Amount of fetching requests per second	Fetching error rate (%)
No-limit	45.2	173,895	51.002	0
Rate-50	41.2	190,822	35.897	10.503
Rate-20	31.2	251,536	11.038	93.157
Rate-10	28.6	274,913	6.141	96.763
Rate-1	27.7	283,113	0.799	99.84
Webhook	26.9	292,389		



The higher the request rate limit, the better performance of MetaDefender Core. However, it causes fetching scan result to hit HTTP code 503 more frequently. MetaDefender Core archieves the best performance if `callbackurl` header is used.

Recommendations

- Users should set appropriate rate limits for APIs of interest in order to balance the performance and fetching result frequencies.
- Callback mechanism is recommended for users whom system performance is the top priority. Certainly, they need another server to listen for results responded from MetaDefender Core.

Multipart upload guideline

Introduction

Since version 5.11.0, MetaDefender Core introduces a brand - new API that accepts individual requests containing parts split from a file and automatically combines them into the original file for scanning. The new API best fits one of the following use cases.

- If you have a stable, high-bandwidth network, you can split a large file into multiple parts on your end and call the new API to upload the parts simultaneously to maximize the available bandwidth.
- If there are size limits on uploaded files, you can divide a large file into smaller parts that fit within those limits and use the API to upload them separately to MetaDefender Core.
- If the network is unstable, a file upload can easily be interrupted in the middle. Instead of uploading entire file again, the API can be used to upload only the interrupted parts.

In all cases mentioned above, once the parts are fully uploaded, MetaDefender Core combines them into the original file and begins scanning it with the selected workflow.

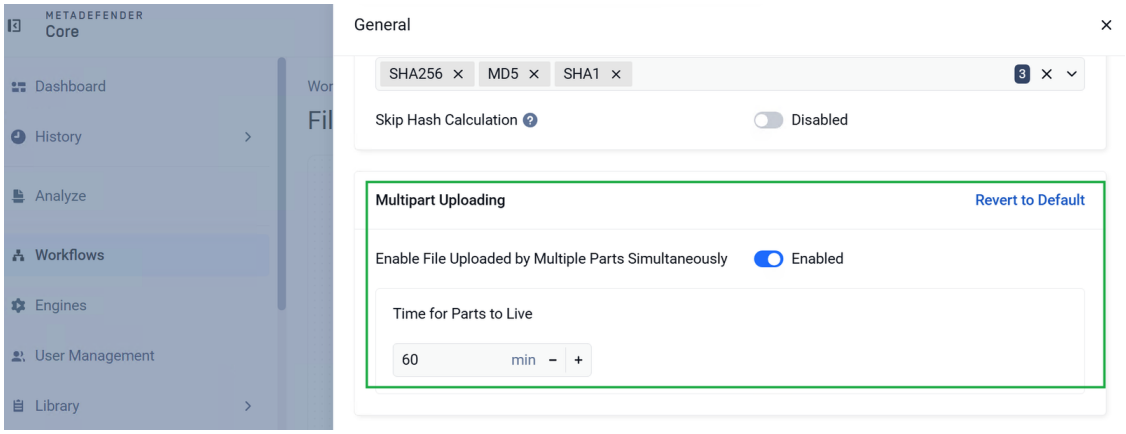
This feature composes a set of following APIs

Functionality	API
Initiate multipart upload session	POST `/file/multipart`
Upload individual part to multipart upload session	POST `/file/multipart/{data_id}`
Fetch status of multipart upload session	GET `/file/multipart/{data_id}`
Abort multipart upload session	DELETE `/file/multipart/{data_id}`

How to enable

Multipart upload is disabled by default. To enable it, please follow these steps

1. Log in to MetaDefender Core dashboard with your account
2. Expand `Workflow Management` --> `Workflows` in the left sidebar, and select the workflow to be affected.
3. In `General` tab, navigate to `Multipart Uploading` section and tick to `Enable file uploaded by multiple parts simultaneously`.
4. Hit `Save changes` to apply the new setting.



How to use

You can use the new APIs in your applications in three steps

1. Initiate a new multipart upload session in MetaDefender Core.
2. Split the file for scanning into multiple parts on your end.
3. Upload the parts into the created multipart upload session in MetaDefender Core.

For the first step, you need create a new multipart upload session by making a `POST` `/file/multipart` request to MetaDefender Core. The request contains no body but must include an essential header, `total-length`, which indicates the size of the entire uploaded file in bytes. For a full list of supported headers, please access here.

If successful, HTTP code `200 OK` and `data_id` will be returned. You should keep `data_id` for the next commands, such as uploading parts to the session, checking the upload status, aborting the upload session and checking the scan status.

In the second step, you should split the submitted file into multiple parts that fit your needs and size limitations.

In the final step, you should make several multipart upload requests to MetaDefender Core via `POST` `/file/multipart/{data_id}`. Each request carries one of the parts split from the original file. `data_id` term in the API is the output of the first step.

Info

API `POST` `/file/multipart/{data_id}` can only accept requests whose `content-type` header is set to `application/octet-stream` ; otherwise, the request body is ill-formatted and resulting in error HTTP `400 BAD REQUEST` .

The multipart upload request needs both the offset and length of the uploaded part, which must be provided by the `content-range` header in the form of `offset-length/total`, where the `offset` indicates the position from the beginning of the file to the first byte of the part. `length` and `total` are, respectively, the part size and the total size of the entire file, all in bytes .

Suppose you split a file of 10,485,760 bytes (10 MB) into 3 parts; then `content-range` header of each request should be 0-2000000/10485760, 2000000-2000000/10485760, and 4000000-6485760/10485760, respectively. You can find further details about the API here.

When all parts are fully uploaded, MetaDefender Core will automatically combine them into a complete file and begin scanning it with the selected workflow. You can later fetch the scan progress and result using the traditional API `POST` `/file/{data_id}`, or you can cancel file processing with `POST` `/file/{data_id}/cancel`.

If the network is broken while one of the parts is being uploaded, you can make another `POST` `/file/multipart/{data_id}` request to upload just the interrupted part again to MetaDefender Core. You can also apply the same approach if your application crashes while uploading parts or MetaDefender Core service is interrupted due to upgrade or maintenance.

With split parts, depending on your network bandwidth, you can choose to upload them one by one, several at a time, or all of them simultaneously to MetaDefender Core. Additionally, except for the first and the last parts, the content range of a part can overlap with its next or previous ones without causing any confusion to MetaDefender Core.

At any time, you can fetch the upload status with `GET` `/file/multipart/{data_id}` or abort part uploads totally with `DELETE` `/file/multipart/{data_id}`. As soon as the abort request is received, MetaDefender Core will stop all requests uploading parts and return HTTP code `422 CANCED`. The multipart upload session is then set to `CANCELED` verdict and all resources consumed by the session thus far are released.

Code · json

```
{
  "err": "Multipart uploading session cancelled"
}
```

How to configure

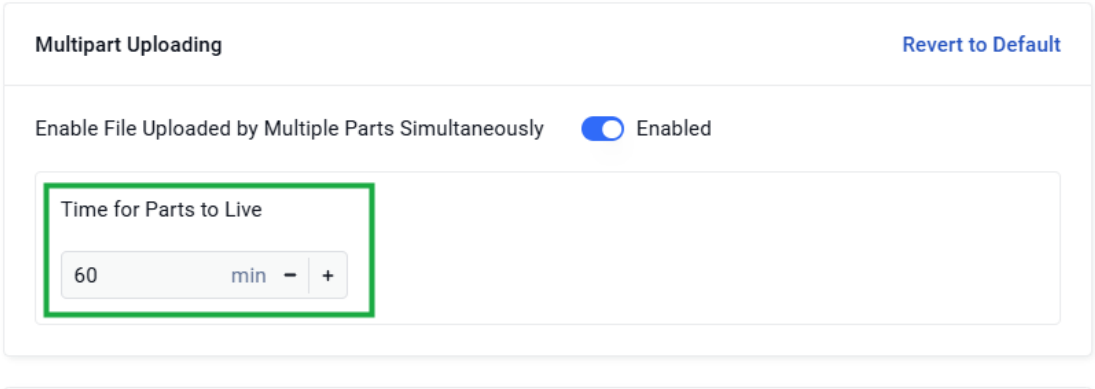
By default, a multipart upload session, once initiated successfully, will be available for parts uploaded within 60 minutes. Outside that duration, HTTP code `409 CONFLICT` is returned for each call to `POST /file/multipart/{data_id}`.

Code · json

```
{
  "err": "Multipart uploading session timed out"
}
```

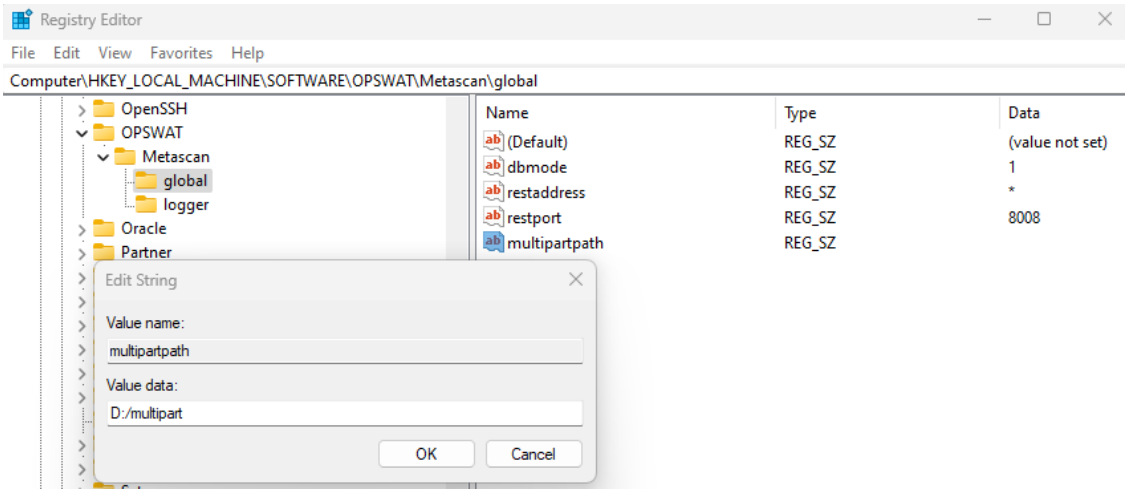
You can extend or collapse the time range as needed by following these steps:

1. Login to MetaDefender Core dashboard with your admin account.
2. Expand `Workflow Management` --> `Workflows` on the left sidebar.
3. Select the workflow of interest.
4. In `General` tab, navigate to `Multipart uploading` section.
5. Modify `Time for parts to live` to your desired value.



By definition, MetaDefender Core stores files that are submitted by multipart upload in `<Install-directory>/data/multipart` on Windows and `/var/lib/ometascan/multipart` on Linux. You can modify the location to another according to your needs.

- On Windows, run Registry Editor, navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global`, add a new string named `multipartpath`, fill in the path to new location, and hit OK to complete. Restart MetaDefender Core to apply the new setting.



- On Linux, open file `/etc/ometascan/ometascan.conf` in edit mode. Under `[global]` session, add a new item named `multipartpath`, fill in the path to new location, and save the file. Restart MetaDefender Core to apply the new setting.

> **Warning**
>
> Please ensure that MD Core has the necessary permission to access the files in the new folder.

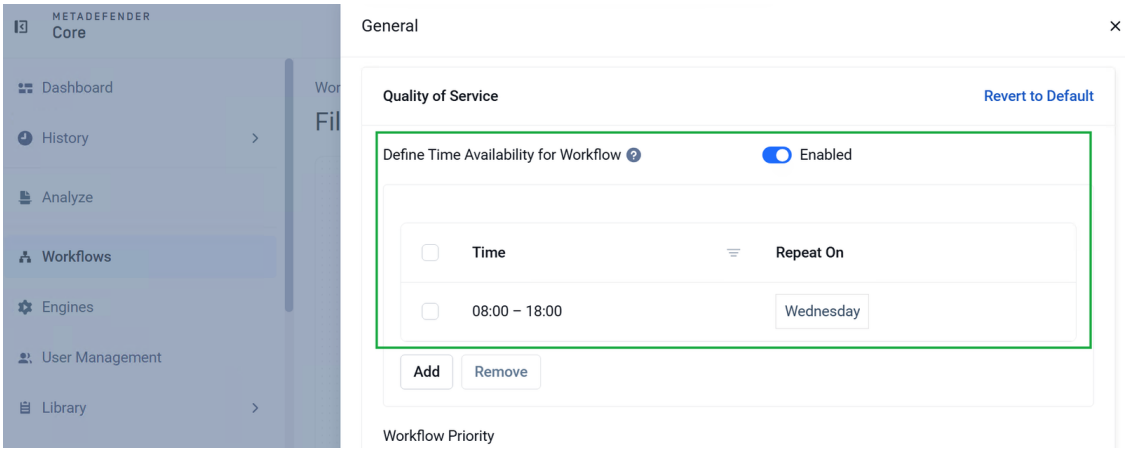
Multipart upload and other facilities

When all parts of a file are uploaded fully to MetaDefender Core, you can cancel the file processing with POST POST `/file/{data_id}/cancel`. Please do not be confused by the two APIs. DELETE `/file/multipart/{data_id}` is used when parts of the file are being uploaded, while POST `/file/{data_id}/cancel` is the best match for cases when the file content is fully received and the processing is underway.

API POST `/file/multipart` supports `callbackurl` header which specified the URL of an external web server to which MetaDefender Core sends the analysis result as soon as the file processing is complete. Header `sanitizedurl` is also supported. The header provides the URL of an external web server that will receive the file content sanitized and/or processed by MetaDefender Core from the original.

A file that is uploaded by multiple parts can be linked to a specific batch. To do this, please fill `batch_id` of the batch in `batch` header of POST `/file/multipart` request to MetaDefender Core. An HTTP code 200 OK and returned `data_id` implies that the file is successfully linked to the batch. The batch cannot be closed until all its linked files are fully uploaded.

Time availability feature under Quality of Service can be applied to files submitted via multipart upload. As long as the multipart upload session is initiated within the time availability, it remains valid for uploaded parts, event if they are uploaded outside the time range. Even so, those parts ust remain within the time-for-parts-tove.



Multipart upload and load balancing

In many deployment scenarios, a number of MetaDefender Core instances are placed behind a load balancer to share the workload. Multipart upload can also be utilized in these scenarios. For this feature to function correctly, a load balancer sticky session must be used in your application to ensure that all parts of a file are uploaded to the same session which is owned by one of the instances behind the load balancer. You can read more about sticky session for load balancing here.

From your application's perspective, you can follow these steps:

1. Make a `POST` `/file/multipart` request to the load balancer **without cookies**.
2. Keep the cookies and `data_id` returned by the load balancer for subsequent calls.
3. Make POST `/file/multipart/{data_id}` requests with the cookies received in step 2 for every uploaded part.

Considerations

- Multipart upload does not support `downloadfrom` header.
- MetaDefender Core can only recover the upload of parts but not the processing of a file if it has already been fully combined.
- Multipart upload recovery cannot be applied to files that are uploaded via multipart and also linked to a batch.
- Multipart upload is only available for asynchronous scans. Local scan, and synchronous scan are not supported.
- Multipart upload is not applicable for MetaDefender Core installed in non-persistence mode.

Split archive upload guideline

Introduction

Since 5.17.0, MetaDefender Core provides new API endpoints that accept individual requests with split-archive parts, use the Archive Extraction engine extract the archive from those parts, and then process the extracted files. These APIs are best suited for the following use cases:

- If your network is stable and high-bandwidth, split a large archive into multiple parts and use the new API

endpoints to upload the parts in parallel, maximizing throughput.

- If upload size limits apply, split the large archive file into smaller parts that meet those limits, then use the API to upload each part separately to MetaDefender Core.
- If the network is unstable and an upload is interrupted, you can use the API to resume by reuploading only the missing split-archive parts instead of the entire file.

Considerations

- Due to the nature of split-archives, only files extracted from the reassembled archive are processed; the combined archive file itself is not supported. Consequently, **Quarantine** and **PostAction** do not apply to the archive, only to its extracted files.
- Compression and splitting of the compressed archive after processing all extracted files is not yet supported. However, if you need a sanitized archive, you can use this workaround:
  - Compress the archive twice (nested compression).
  - Enable “Allow extracted files to be sanitized when configured” (see **Configurations**).
  - Configure the relevant sanitization settings under the Deep CDR or Compression tabs.
- Split-archive upload is only available for asynchronous scans. Local scan, and synchronous scan are not supported.
- Split-archive upload is not applicable for MetaDefender Core installed in non-persistence mode.
- Split-archive upload does not support `downloadfrom` header.
- MetaDefender Core can resume uploading parts, but it cannot resume processing once a file has already been fully received.
- Split-archive upload recovery does not apply to files uploaded via split-archive that are also linked to a batch.
- Track rejected requests does not support split-archive uploads

Prerequisite

This feature is only available when using MetaDefender Core version 5.17.0 with:

- Archive Extraction Engine version 7.7.0
- File Type Verification Engine version 7.7.0

Supported file types

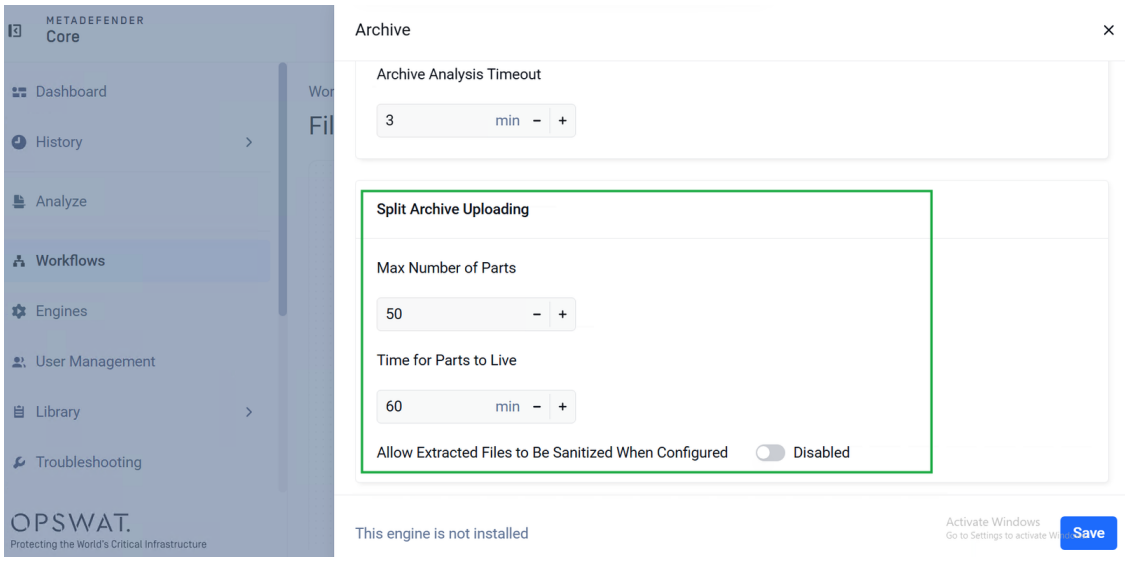
These archive formats are eligible for split-archive upload: ZIP, RAR, TAR, 7Z, XZ, GZ, WinZip, and split Clonezilla (partial support: Clonezilla with GZ compression only).

Configurations

Workflow configuration

To adjust the split-archive workflow configuration settings:

1. Sign in to the MetaDefender Core dashboard with an admin account.
2. Go to Workflow Management Workflows.
3. Select the target workflow.
4. Open the Archive tab and locate the “Split archive uploading” section.



Max number of parts

By default, the maximum number of allowed parts is 50. You can tailor this limit to your needs.

If the submitted part count exceeds the configured limit, each POST `/file/splitarchive/{data_id}` request will return HTTP 400 Bad Request.

Code · json

```
{

"err": "Failed to request scan. Total parts exceeded the maximum number %1 permitted by your configuration"

}
```

Time for parts to live

By default, a split-archive upload session remains open for **60 minutes** after initiation. After this window, each POST `/file/splitarchive/{data_id}` call returns HTTP 409 Conflict.

Code · json

```
{
```

```
"err": "Split archive upload session already timed out"

}
```

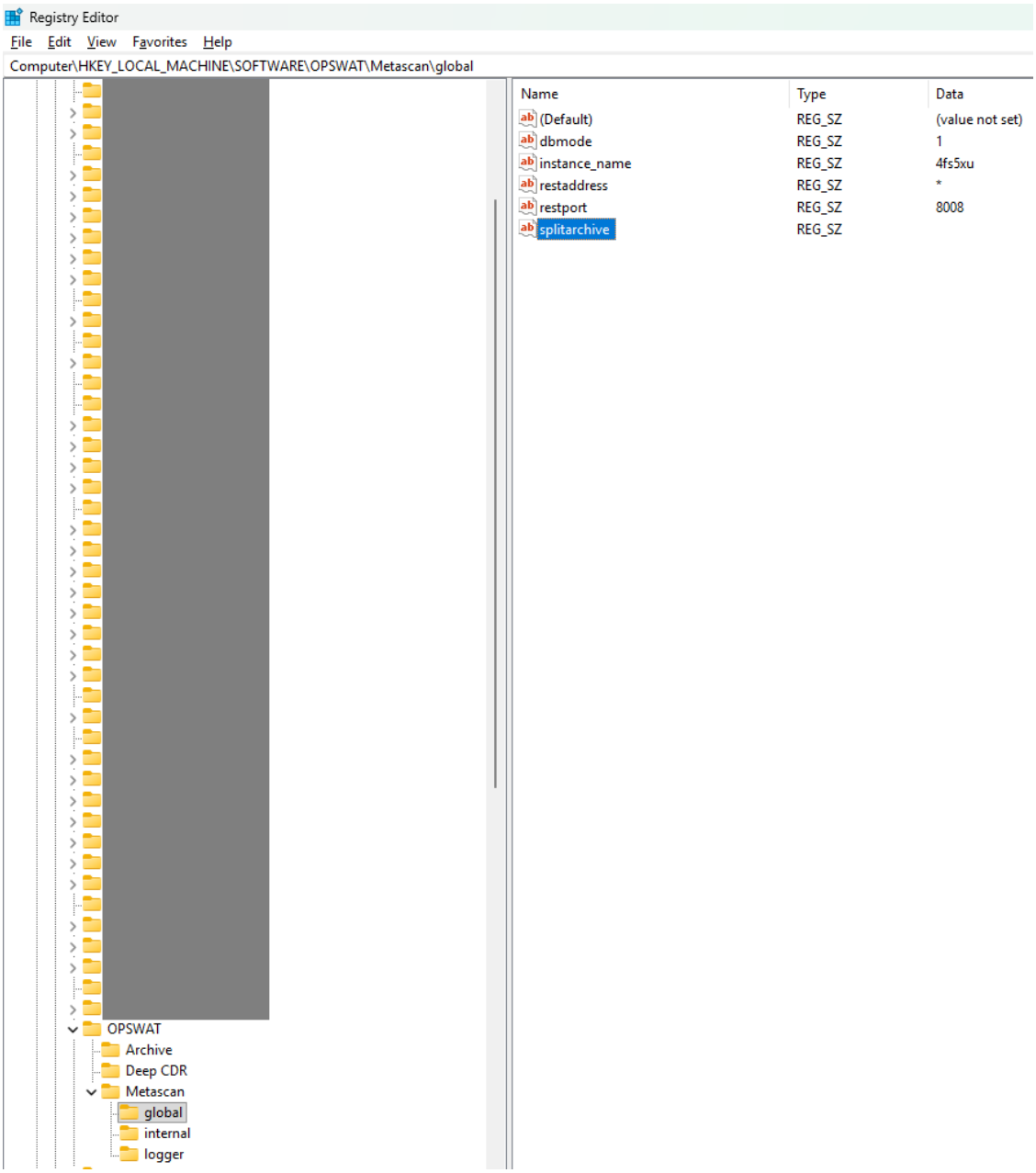
Allow extracted files to be sanitized when configured

As noted in the Considerations section, compressing and then splitting the archive after all extracted files are processed is not yet supported. However, extracted files can still be sanitized. To enable this, turn on this option and configure the sanitization settings under the Deep CDR or Compression tabs so extracted files are sanitized.

Storage settings for submitted parts

By definition, MetaDefender Core stores files that are submitted by split-archive upload in `<Install-directory>/data/splitarchive`on Windows and `/var/lib/ometascan/splitarchive` on Linux. You can modify the location to another according to your needs.

- On Windows, run Registry Editor, navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global`, add a new string named `splitarchive`, fill in the path to new location, and hit OK to complete. Restart MetaDefender Core to apply the new setting.



- On Linux, open file `/etc/ometascan/ometascan.conf` in edit mode. Under `[global]` session, add a new item named `splitarchive`, fill in the path to new location, and save the file. Restart MetaDefender Core to apply the new setting.

IMPORTANT

Please ensure that MD Core has the necessary permission to access the files in the new folder.

How to use

This feature composes a set of following API endpoints:

Functionality	API

Functionality	API
Initiate split-archive upload session	POST /file/splitarchive
Upload individual part to split-archive upload session	POST /file/splitarchive/{data_id}
Fetch status of split-archive upload session	GET /file/splitarchive/{data_id}
Abort split-archive upload session	DELETE /file/splitarchive/{data_id}

Please see the API documentation for detailed usage, parameters, and examples: POST - Create split archive upload session

You can use the new APIs in your applications in three steps:

1. Split the archive file for scanning into multiple parts on your end.
2. Initiate a new split-archive upload session in MetaDefender Core.
3. Upload the parts into the created split-archive upload session in MetaDefender Core.

Step 1:

Use an archive-splitting tool to divide the archive into multiple parts sized to your constraints (e.g., API upload limits, network bandwidth) before uploading.

Step 2: Create a split-archive upload session

- Send a POST request to `/file/splitarchive` to initialize a new upload session.
- Request body: none.
- Optional header: total-length (bytes), indicating the full size of the original archive.
- If total-length is not provided, MetaDefender Core will automatically sum the sizes of all uploaded parts to validate against configuration limits.

If successful, the response is HTTP 200 OK with a `data_id`. Persist this `data_id` and use it for subsequent operations: upload parts to the session, check upload status, abort the session, and query scan status.

Final step: Upload all parts

- Send multiple POST requests to `/file/splitarchive/{data_id}`, one per part from the original archive.
- Use the `data_id` returned in Step 1.
- Each request includes a single part in the body and appropriate headers (e.g., part index, size).
- Continue until all parts are uploaded, then finalize per API guidance.

Info

API POST `/file/splitarchive/{data_id}` can only accept requests whose `content-type` header is set to `application/octet-stream`; otherwise, the request body is ill-formatted and resulting in error HTTP 400 BAD REQUEST.

When all parts are uploaded, MetaDefender Core automatically forwards them to the Archive Extraction engine to reconstruct, extract, and begin processing the extracted files using the selected workflow. You can monitor progress and fetch results via GET `/file/{dataid}`, or cancel processing with POST `/file/{data_id}/cancel`

If the network fails during a part upload, resend only the interrupted part via POST `/file/splitarchive/{data_id}`. Use the same approach if your application crashes mid-upload or if MetaDefender Core is temporarily unavailable due to upgrades or maintenance.

With split parts, you can upload sequentially, in parallel batches, or all at once—choose based on your available bandwidth. Additionally, aside from the first and last parts, a part's content range may overlap with adjacent parts without causing issues for MetaDefender Core.

At any time, you can check upload status via GET `/file/splitarchive/{data``id}` or abort all part uploads with DELETE `/file/splitarchive/{data_id}`. Upon abort, MetaDefender Core stops in-flight part uploads and returns HTTP 422 Unprocessable Entity. The multipart session is set to CANCELED, and all session resources are released.

Code · json

```
{
  "err": "Split archive upload session already cancelled"
}
```

Split archive upload and other facilities

Once all parts have been fully uploaded to MetaDefender Core, you can cancel processing via POST `/file/{data_id}/cancel`.

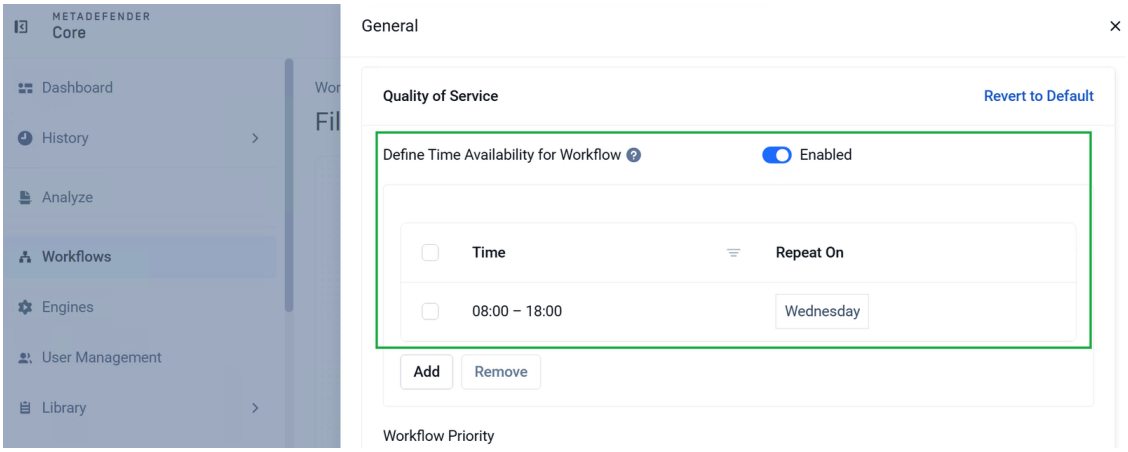
Don't confuse the two endpoints:

- DELETE `/file/splitarchive/{data_id}`: aborts part uploads while the file is still being uploaded.
- POST `/file/{data_id}/cancel`: cancels processing after the full file has been received and processing has

The POST `/file/splitarchive` API supports the callbackurl header, which specifies an external endpoint that MetaDefender Core will call with the analysis result once processing completes.

A file uploaded in multiple parts can be linked to a batch. Include the batchid in the batch header of each POST `/file/splitarchive`request to MetaDefender Core. A 200 OK response with a data id indicates the file was successfully linked. The batch cannot be closed until all linked files have been fully uploaded.

The Time Availability feature under Quality of Service applies to split-archive uploads. If the upload session is initiated within the allowed time window, the session remains valid for parts uploaded later—even outside that window—provided they are uploaded within the configured "Time for parts to live."



Multipart upload and load balancing

In many deployment scenarios, a number of MetaDefender Core instances are placed behind a load balancer to share the workload. Split-archive upload can also be utilized in these scenarios. For this feature to function correctly, a load balancer sticky session must be used in your application to ensure that all parts of a file are uploaded to the same session which is owned by one of the instances behind the load balancer. You can read more about sticky session for load balancing here.

From your application's perspective, follow these steps:

1. Send POST `/file/splitarchive` to the load balancer without cookies.
2. Capture and persist the cookies and `data_id` returned by the load balancer.
3. For each part upload, send POST `/file/splitarchive/{data_id}` and include the cookies from step 2.

Integrate MetaDefender Core syslog into Splunk Enterprise

Overview

Info

This page assumes MetaDefender Core and Splunk are **already installed and running**. It only covers wiring the two together.

Info

1. The Splunk information is used on this page:

- Splunk Enterprise
- Version: 9.4.3
- Deployment: self-hosted container

2. The MetaDefender Core version is any as long as it supports Syslog

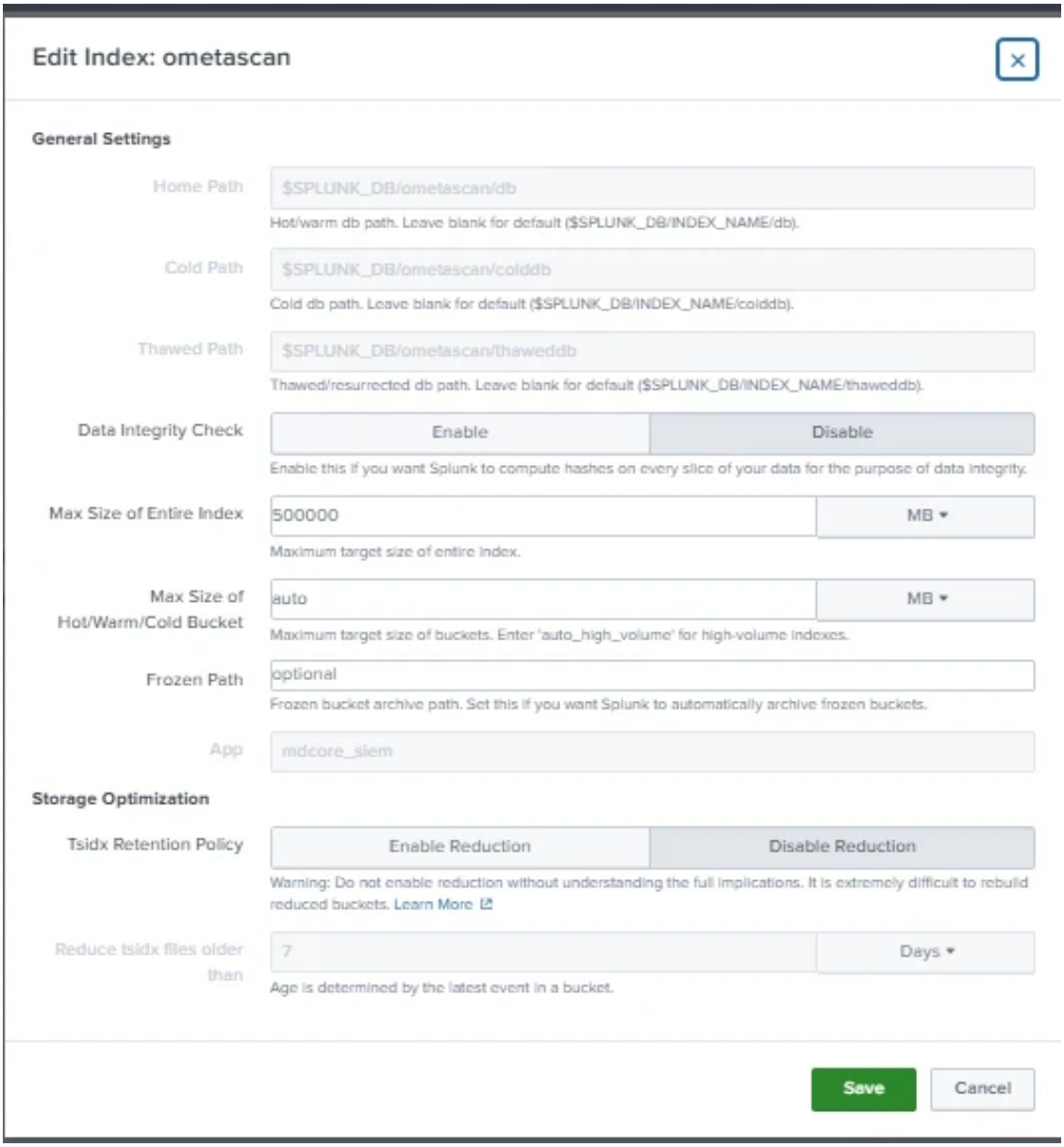
The integration has two sides:

Component	Action
Splunk	Open a syslog data input and route it into a dedicated index.
MetaDefender Core	Point its syslog logger at the Splunk host/port.

Configure Splunk to receive syslog

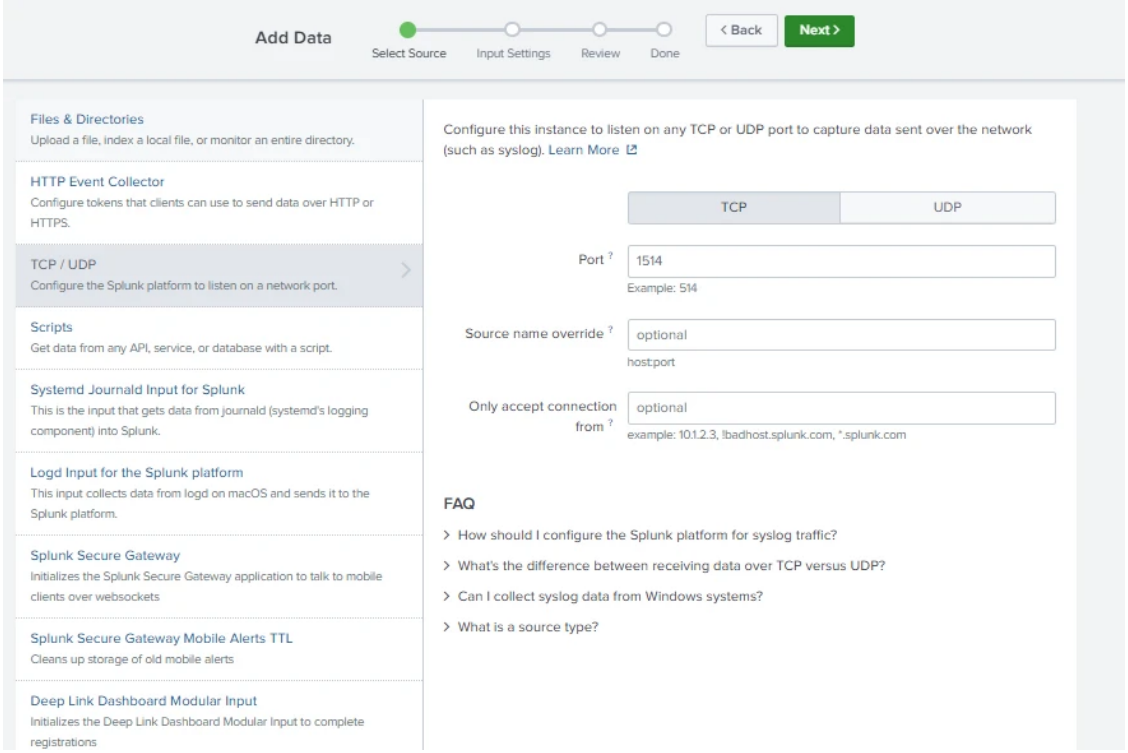
Create a dedicated index and a TCP (and/or UDP) data input that lands MD Core events in it. Add the following stanzas to your Splunk app's local config, or configure them via:

Settings > **Indexes** > **New Index** in the UI.

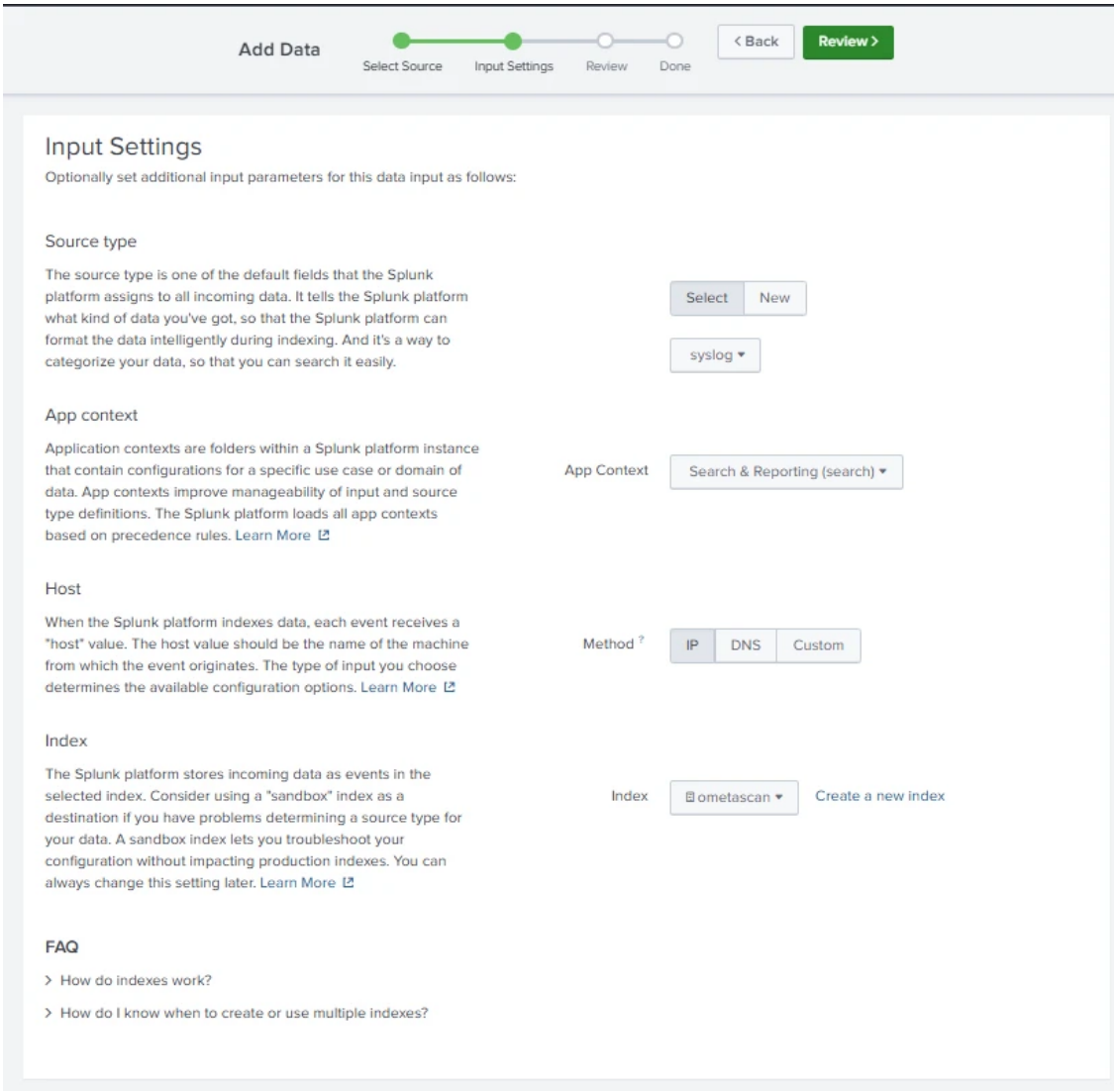


Settings > **Data Inputs** > **TCP** > **+ New** in the UI.

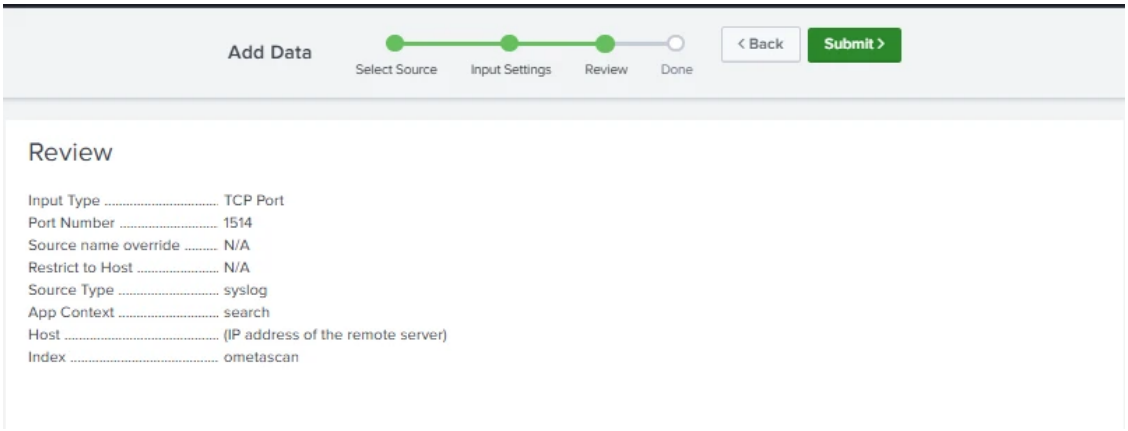
- Protocol: TCP or UDP (It is recommended that the user use TCP because it is more reliable)
- Port: 1514



- Source type: syslog
- Method: IP
- Index: use the index name that is created in the previous step



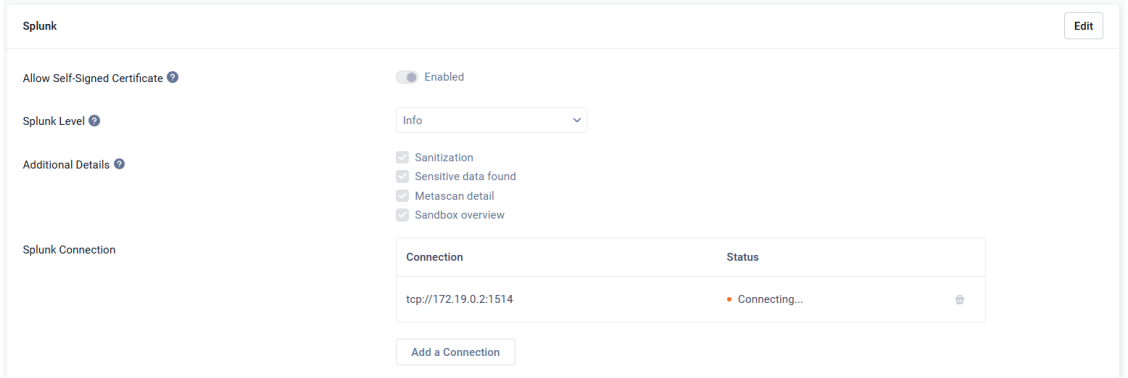
Summary:



Point MetaDefender Core at Splunk

From the MetaDefender Core UI, go to **Settings** > **Logs** > down to **Splunk** > **Edit** > **Add a Connection**

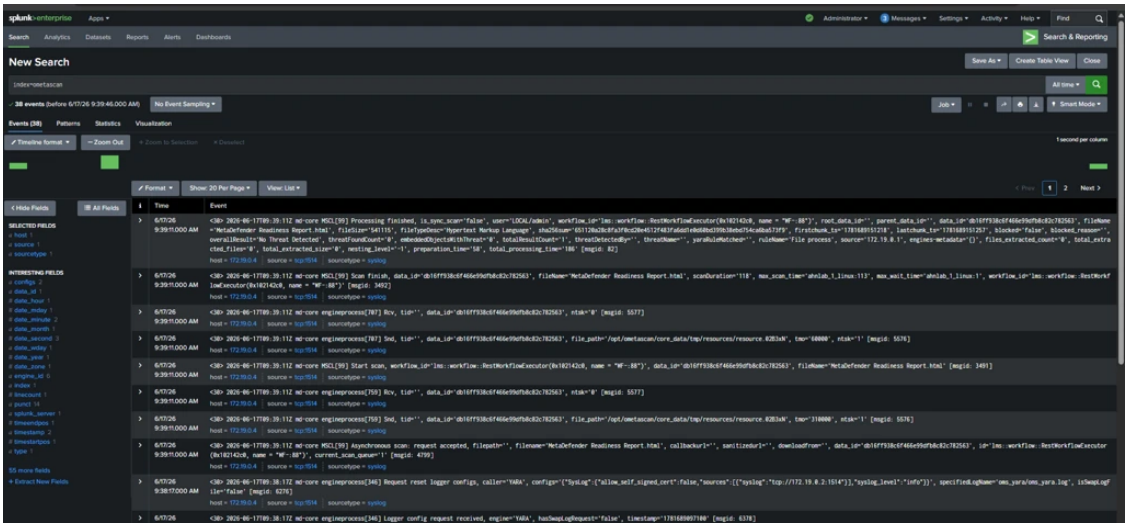
Provide the IP address of Splunk and the port:



This feature not supported for OPSWAT Central Management yet. When MetaDefender Core is managed by OCM, the option "Additional Details" is not available.

Verify

1. In Splunk, confirm the input and index exist: Settings > Data inputs shows the 1514 TCP input, and Settings > Indexes shows ometascan.
2. Since an idle MetaDefender Core sends nothing, run a scan in MetaDefender Core to generate events or restart the Ometascan service
3. Search in Splunk: `index=ometascan`



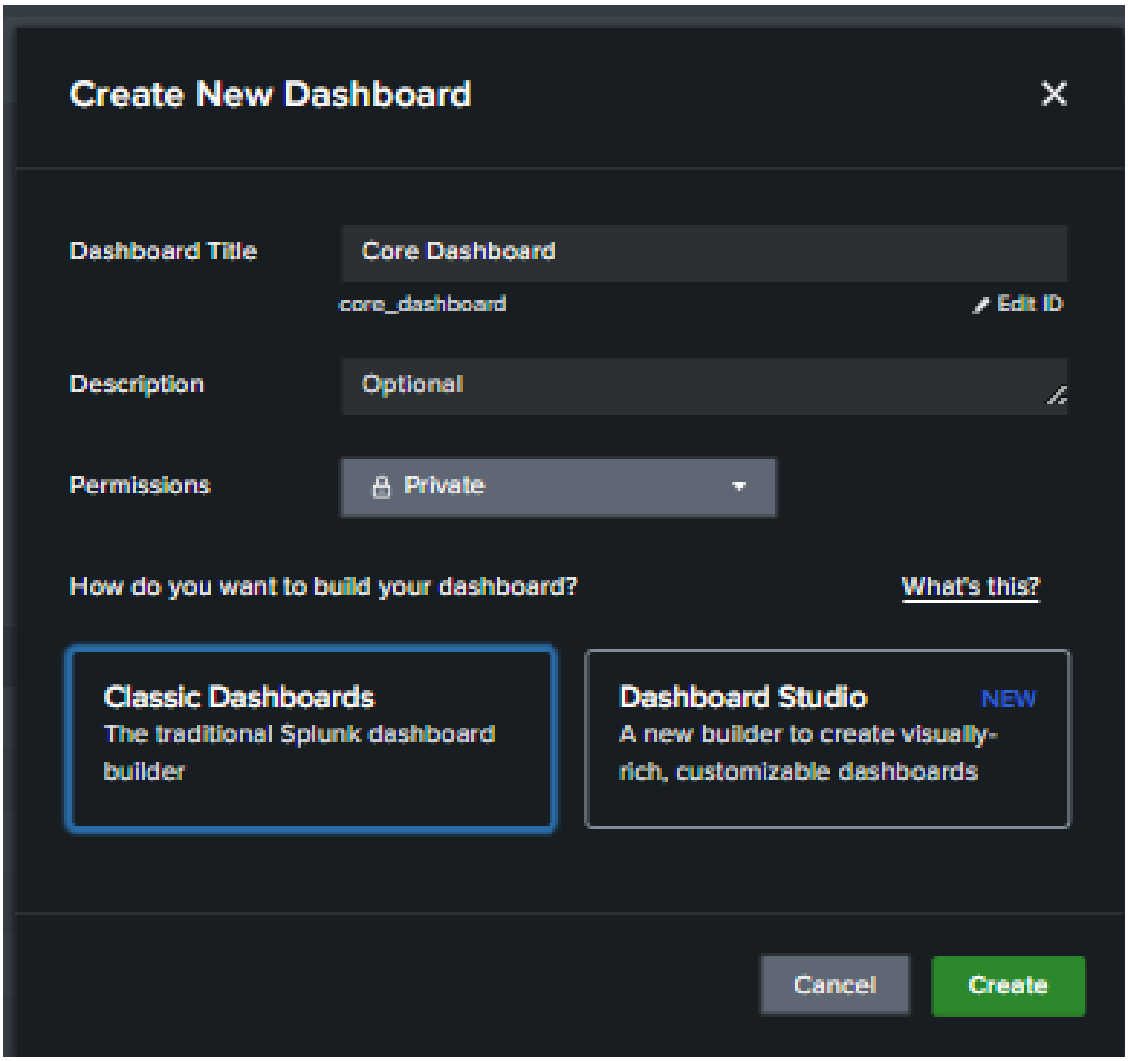
Splunk Dashboard

Info

The user can customize the Splunk Dashboard for themselves. The following steps are to add the suggested dashboard from OPSWAT

In the Splunk UI, go to **Apps** (top-left dropdown) > **Search & Reporting** > **Dashboards** > **Create New Dashboard**

Provide the dashboard name and choose **Classic Dashboards**



In **Edit Dashboard**, choose **Source** and use the following XML code, then click **Save**

Code · xml

```
<form version="1.1" theme="dark">
  <label>MetaDefender Core - Scan Overview</label>
  <description>Scan-history events forwarded from MetaDefender Core over syslog
(index=ometascan). Fields are extracted inline from the key='value' syslog payload.</
description>

  <!-- Shared time picker drives the base search -->
  <fieldset submitButton="false" autoRun="true">
    <input type="time" token="tr">
      <label>Time range</label>
      <default>
        <earliest>-24h@h</earliest>
        <latest>now</latest>
      </default>
    </input>
  </fieldset>
```

```
  <!-- One base search (filters to completed scans + cleans the key='value' fields);
       every panel post-processes it, so the raw events are read once. -->
  <search id="baseScans">
    <query>index=ometascan "Processing finished"
| rex field=_raw "overallResult='(?&lt;overallResult&gt;[^']*)'"
| rex field=_raw "source='(?&lt;src_ip&gt;[^']*)'"
| rex field=_raw "blocked='(?&lt;blocked&gt;[^']*)'"
| rex field=_raw "sha256sum='(?&lt;sha256sum&gt;[^']*)'"
| rex field=_raw "fileName='(?&lt;fileName&gt;[^']*)'"
| rex field=_raw "fileTypeDesc='(?&lt;fileTypeDesc&gt;[^']*)'"
| rex field=_raw "threatName='(?&lt;threatName&gt;[^']*)'"
| rex field=_raw "user='(?&lt;user&gt;[^']*)'"
| rex field=_raw "total_processing_time='(?&lt;total_processing_time&gt;[^']*)'"</query>
    <earliest>$tr.earliest$</earliest>
    <latest>$tr.latest$</latest>
  </search>

  <!-- Row 1: KPIs -->
  <row>
    <panel>
      <title>Files scanned</title>
      <single>
        <search base="baseScans"><query>| stats count</query></search>
        <option name="drilldown">none</option>
        <option name="colorBy">value</option>
      </single>
    </panel>
    <panel>
      <title>Threats found</title>
      <single>
        <search base="baseScans"><query>| where overallResult!="No Threat Detected" | stats
count</query></search>
        <option name="drilldown">none</option>
        <option name="rangeColors">["0x53a051","0xdc4e41"]</option>
        <option name="rangeValues">[0]</option>
        <option name="useColors">1</option>
        <option name="colorBy">value</option>
      </single>
    </panel>
    <panel>
      <title>Blocked files</title>
      <single>
        <search base="baseScans"><query>| where blocked="true" | stats count</query></search>
        <option name="drilldown">none</option>
        <option name="rangeColors">["0x53a051","0xf8be34"]</option>
        <option name="rangeValues">[0]</option>
        <option name="useColors">1</option>
        <option name="colorBy">value</option>
      </single>
    </panel>
```

```
    <panel>
      <title>Unique files (by SHA256)</title>
      <single>
        <search base="baseScans"><query>| stats dc(sha256sum)</query></search>
        <option name="drilldown">none</option>
      </single>
    </panel>
  </row>

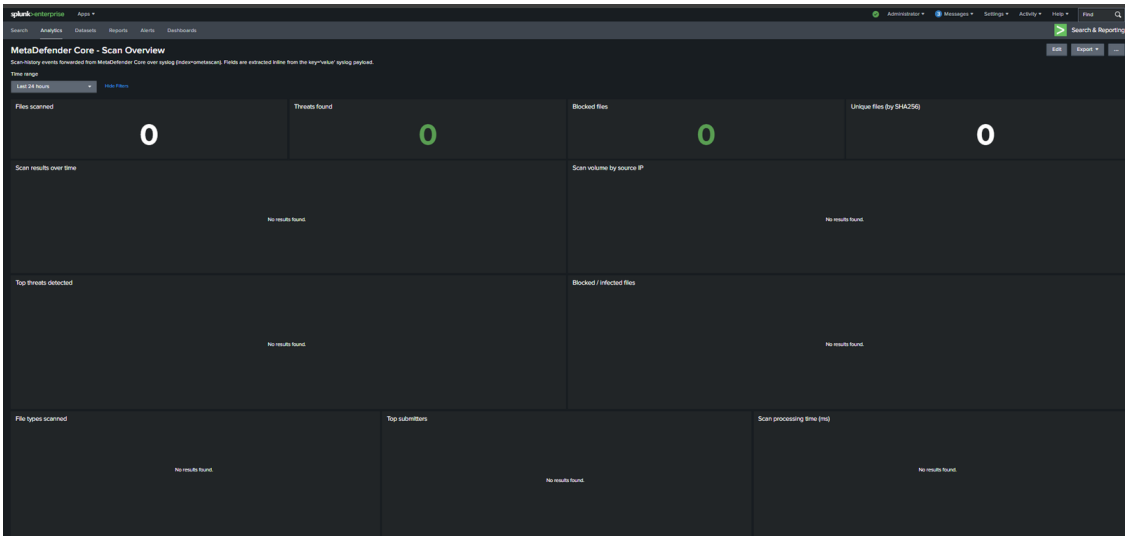
  <!-- Row 2: Trends -->
  <row>
    <panel>
      <title>Scan results over time</title>
      <chart>
        <search base="baseScans"><query>| timechart span=1h count by overallResult</query></
search>
        <option name="charting.chart">column</option>
        <option name="charting.chart.stackMode">stacked</option>
        <option name="charting.legend.placement">bottom</option>
      </chart>
    </panel>
    <panel>
      <title>Scan volume by source IP</title>
      <chart>
        <search base="baseScans"><query>| timechart span=1h count by src_ip</query></search>
        <option name="charting.chart">column</option>
        <option name="charting.chart.stackMode">stacked</option>
        <option name="charting.legend.placement">bottom</option>
      </chart>
    </panel>
  </row>

  <!-- Row 3: Threat focus -->
  <row>
    <panel>
      <title>Top threats detected</title>
      <table>
        <search base="baseScans"><query>| where overallResult!="No Threat Detected" | top
limit=20 threatName</query></search>
        <option name="drilldown">none</option>
        <option name="count">20</option>
      </table>
    </panel>
    <panel>
      <title>Blocked / infected files</title>
      <table>
        <search base="baseScans"><query>| where overallResult!="No Threat Detected" OR
blocked="true" | stats count latest(overallResult) as result values(threatName) as threats by
fileName sha256sum | sort - count</query></search>
        <option name="drilldown">none</option>
        <option name="count">20</option>
```

```
      </table>
    </panel>
  </row>

  <!-- Row 4: Context and performance -->
  <row>
    <panel>
      <title>File types scanned</title>
      <chart>
        <search base="baseScans"><query>| top limit=15 fileTypeDesc</query></search>
        <option name="charting.chart">bar</option>
        <option name="charting.legend.placement">none</option>
      </chart>
    </panel>
    <panel>
      <title>Top submitters</title>
      <table>
        <search base="baseScans"><query>| stats count by src_ip user | sort - count</query></
search>
        <option name="drilldown">none</option>
        <option name="count">20</option>
      </table>
    </panel>
    <panel>
      <title>Scan processing time (ms)</title>
      <chart>
        <search base="baseScans"><query>| timechart span=1h avg(total_processing_time) as "avg
ms" p95(total_processing_time) as "p95 ms"</query></search>
        <option name="charting.chart">line</option>
        <option name="charting.legend.placement">bottom</option>
      </chart>
    </panel>
  </row>
</form>
```

The following is the dashboard after being created:


splunk>enterprise
Apps
Administrator
Messages
Settings
Activity
Help
Find
Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reporting
MetaDefender Core - Scan Overview
Scan-history events forwarded from MetaDefender Core over syslog (index=metascan). Fields are extracted inline from the key="value" syslog payload.
Time range
Last 24 hours
Hide Filters
Edit
Export
Files scanned
0
Threats found
0
Blocked files
0
Unique files (by SHA256)
0
Scan results over time
No results found.
Scan volume by source IP
No results found.
Top threats detected
No results found.
Blocked / infected files
No results found.
File types scanned
No results found.
Top submitters
No results found.
Scan processing time (ms)
No results found.

Overview

This guide is for having MetaDefender Core running in a Cloud Service Provider (CSP). It contains details for AWS, Azure and GCP of the following topics:

- **Recommended Architectures**
  - AWS Recommended Architectures
  - Azure Recommended Architectures
  - GCP Recommended Architectures
- **How to deploy in different CSPs**
  - Kubernetes Cluster
  - Single Instance
  - Autoscaling Instance
- **Recommended database service**
- **How to monitor**
- **Troubleshooting and maintenance**

Recommended Architectures

This page explains the recommended architecture for running MetaDefender Core in an AWS EC2 instance or AWS EKS (Kubernetes)

Infrastructure Options for MetaDefender Core in AWS EC2

The possible infrastructure options vary depending on the number of EC2 instances where MetaDefender Core will be installed.

Options:

1. Single EC2
2. Autoscaling EC2

Infrastructure Options for MetaDefender Core in AWS EKS

The possible infrastructure options vary depending on the AWS services used to host MetaDefender Core and how the PostgreSQL database is provisioned.

Options:

1. EKS Cluster with **EC2 Instances + AWS RDS**
2. EKS Cluster with **EC2 Instances + PostgreSQL Pod**

3. EKS Cluster with **AWS Fargate + PostgreSQL Pod**

4. EKS Cluster with **AWS Fargate + AWS RDS**

The above options are explained in EKS Cluster Architecture

We offer a provisioning script that, depending on the selection, automatically provisions the cluster for you and installs the K8S components needed for the MetaDefender Core service. See EKS Cluster Deployment

EKS Cluster

The options vary depending on the AWS services used to host MetaDefender Core and how the PostgreSQL database is provisioned

Options:

1. EC2 Instances + AWS RDS

2. EC2 Instances + PostgreSQL Pod

3. AWS Fargate + PostgreSQL Pod

4. AWS Fargate + AWS RDS

We offer a provisioning script that, depending on the selection, we automatically provisions the cluster for you and installs the K8S components needed for the MetaDefender Core service.

MetaDefender PostgreSQL database details

Recommendations

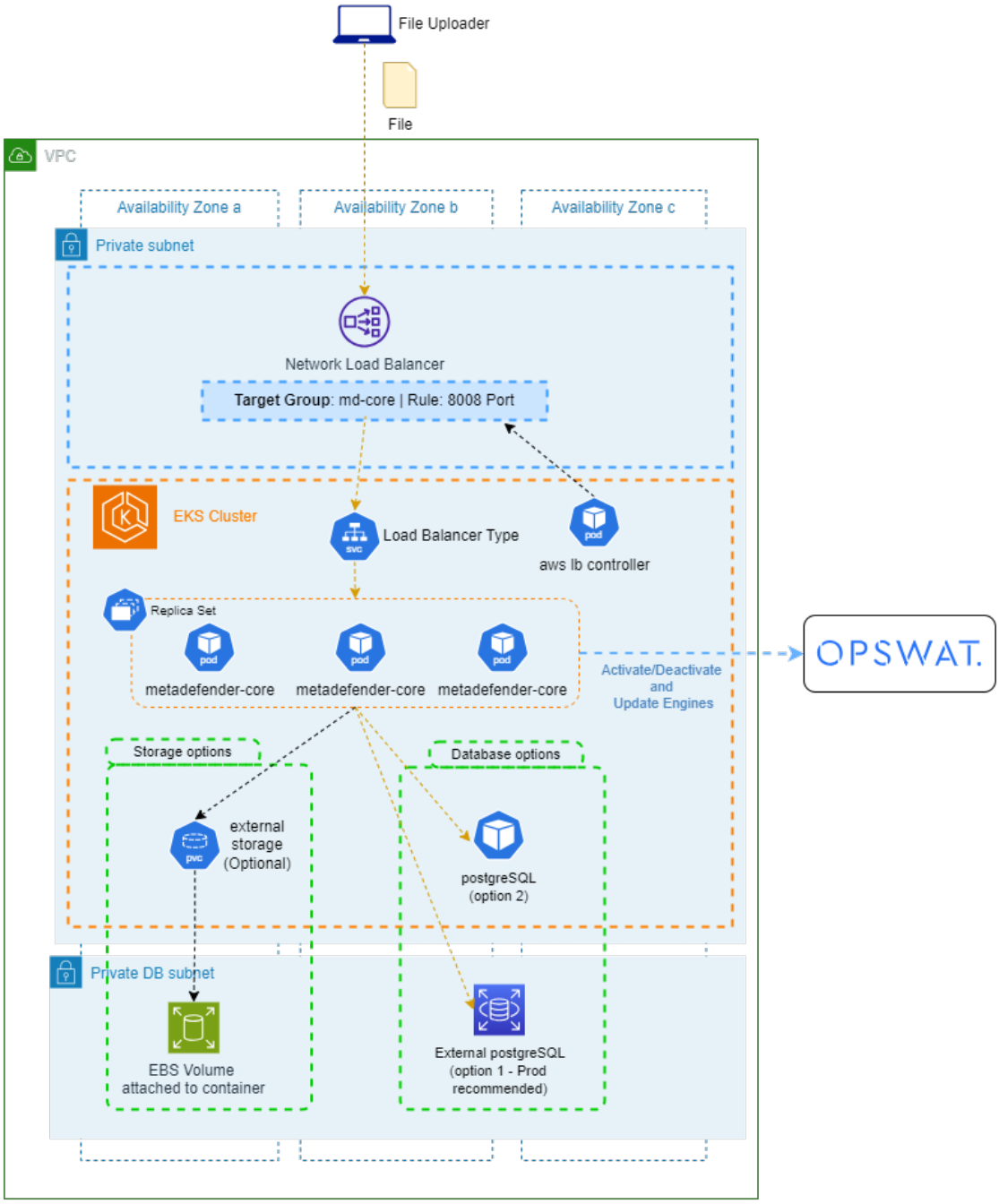
For running MetaDefender Core in a Kubernetes cluster we recommend to use, for production environments, an external database service for PostgreSQL. In case of **AWS** it is **AWS RDS** which the script will install and configure to be accessible from the cluster.

For non-production environments or for stateless use cases, we offer the option of installing the database within the cluster using a standard PostgreSQL image.

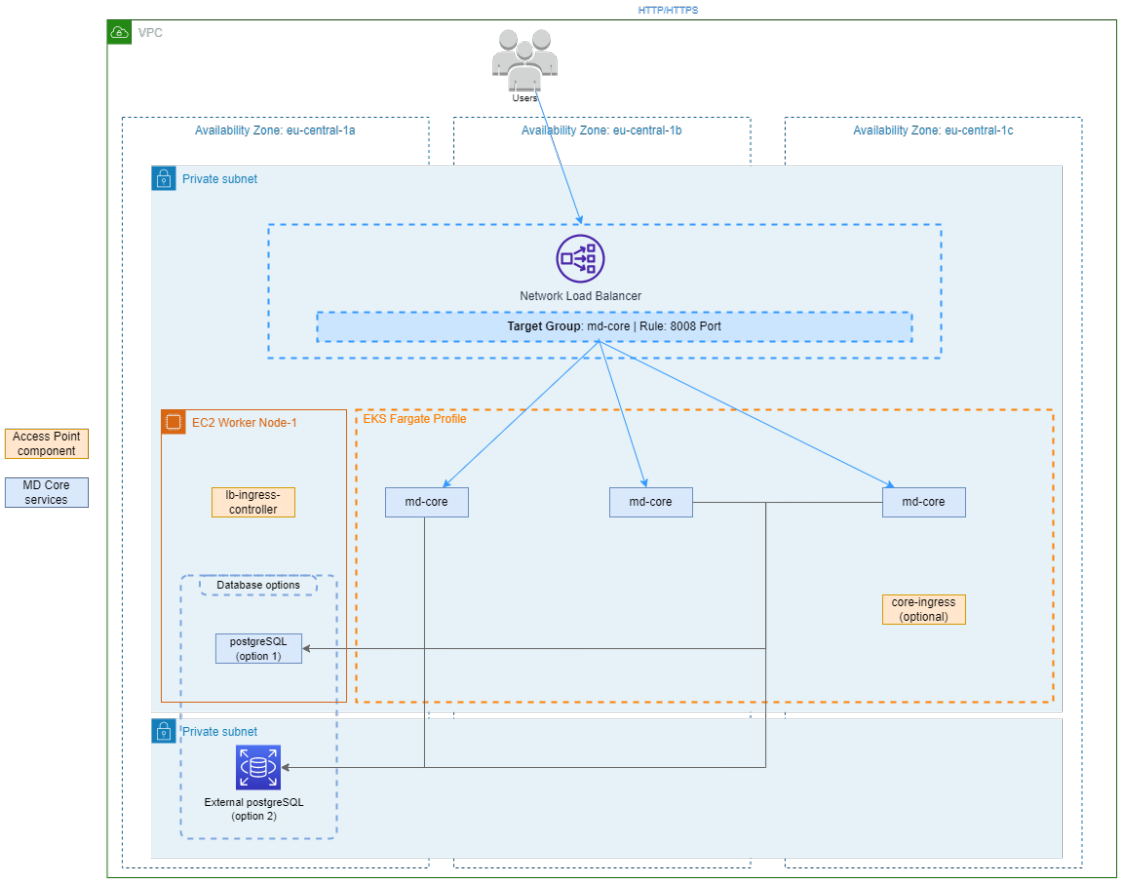
EC2 Instances + AWS RDS / PostgreSQL Pod

There are 2 routes to take a look in the diagram

- **Orange**: Flow to scan a file, hitting the network load balancer that is generated by the AWS Load Balancer Controller. It has a target group pointing the the Kubernetes Service to route the traffic to the MetaDefender Core Pods.
- **Blue**: Each pod communicates with OPSWAT to activate/deactivate and update the AV engines.



AWS Fargate + AWS RDS / PostgreSQL Pod



Resources Inventory

Access Management

Service Type	Resource Name	Description
IAM Role	terraform-<CLUSTER_NAME>-cluster-ServiceRole	IAM Role for the cluster to assume the policies attached to it
IAM Role	terraform-<CLUSTER_NAME>-cluster-NodeInstanceRole	IAM Role for the nodes to assume the policies attached to it
IAM Policy	terraform-<CLUSTER_NAME>-cluster-PolicyELBPermissions	Policy for the EKS cluster to be able to get networking information for configurations
IAM Policy	terraform-<CLUSTER_NAME>-cluster-PolicyCloudWatchMetrics	Policy for the EKS cluster to be able to put metrics in CloudWatch
IAM Policy	terraform-<CLUSTER_NAME>-cluster-PolicyAWSLoadBalancerController	Policy for the nodes to be able to modify the load balancer settings

Service Type	Resource Name	Description
TLS Private Key	NA	Generates a secure private key stored in the Terraform state file
AWS Key Pair	<CLUSTER_NAME>	Upload to AWS the OpenSSH public key format to logging-in to the Worker Nodes.
EKS Cluster Auth config	terraform-<CLUSTER_NAME>-cluster-	Generates the kubeconfig file for accessing to the cluster with Kubectl

Networking

Service Type	Resource Name	Description
AWS Network Load Balancer	AWS Autogenerated	Network load balancer (Not created from terraform, created using **eks_install_lb_controller.sh**)
VPC	terraform-<CLUSTER_NAME>-cluster/VPC	VPC for the whole architecture to be deployed in
3 Public Subnets	terraform-<CLUSTER_NAME>-cluster/SubnetPublicX	3 public subnets deployed in 3 different availability zones
3 Private Subnets	terraform-<CLUSTER_NAME>-cluster/SubnetPrivateX	3 private subnets deployed in 3 different availability zones
Internet Gateway	terraform-<CLUSTER_NAME>-cluster/InternetGateway	Internet Gateway service to allow the cluster to access to internet
Public NAT Gateway	terraform-<CLUSTER_NAME>-cluster/NATGateway	Public NAT gateway for the services in the private network to access to the public network and route the traffic to the internet gateway to access to internet
Elastic IP	terraform-<CLUSTER_NAME>-cluster/NATIP	Public IP attached to the NAT gateway for accessing to internet

Security

Service Type	Resource Name	Description
Security Group	terraform-<CLUSTER_NAME>-cluster-ClusterSharedNodeSecurityGroup	Communication between all nodes in the cluster
Security Group	terraform-<CLUSTER_NAME>-cluster-ControlPlaneSecurityGroup	Communication between the control plane and worker nodegroups
Security Group	terraform-<CLUSTER_NAME>-cluster-remoteAccess	Allow SSH access to the managed Worker Nodes in the Node Group
Security Group	terraform-<CLUSTER_NAME>-cluster-PostgreSQLSecurityGroup	Allow access to postgres db

Compute

Service Type	Resource Name	Description	Option
EKS Cluster	<CLUSTER_NAME>	EKS Cluster service for managing the Kubernetes Cluster	All
EKS Node Group	terraform-<CLUSTER_NAME>-nodegroup	Node group configuration of the Worker nodes of the EKS Cluster	EC2 Instances
RDS Instance	<CLUSTER_NAME>-postgres-db	PostgreSQL engine in RDS instance	AWS RDS
EKS Fargate Profile	fp-<CLUSTER_NAME>	Fargate Profile configuration	AWS Fargate

Single EC2

The single instance deployment is only recommended for small or test environments that do not need high availability. This deployment consist of a AWS EC2 instance using the MetaDefender Core AMI published in the marketplace that includes a Standalone MetaDefender Core where the PostgreSQL database is deployed in the same instance than MetaDefender Core.

Diagram

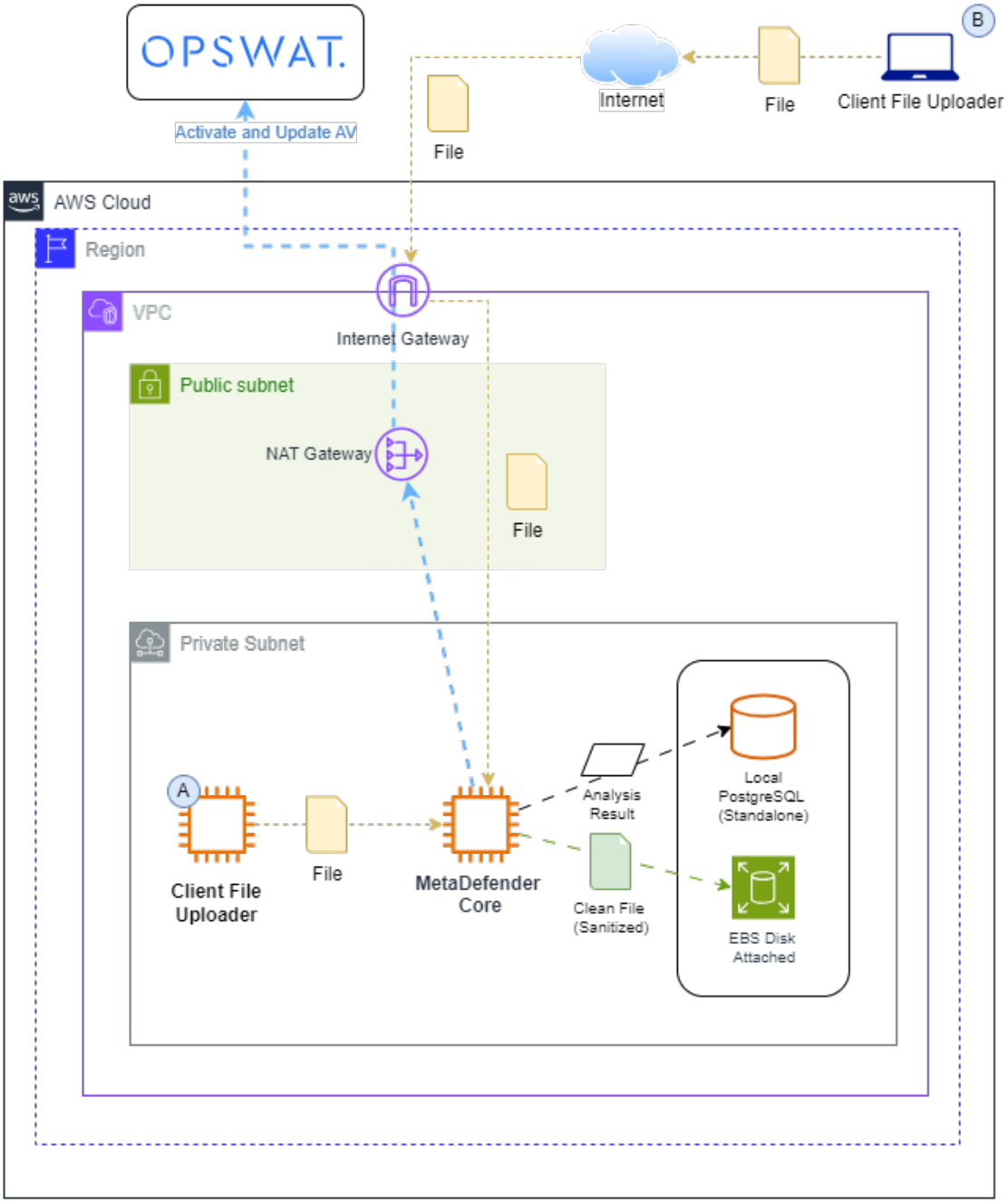
Orange flows

- **A**: From a private subnet, client is getting the files and uploading them to MetaDefender Core using the **private IP**
- **B**: From Internet, client is outside the VPC and is sending files to MetaDefender Core using the **public IP**

Blue flows

- MetaDefender Core get activated and updates the modules installed on it after activation

This diagram show the standalone database option where the analysis results are stored.



Resources Inventory

Access Management

Service Type	Resource Name	Description
IAM Role	LambdaAccessRole	Role for Lambdas
IAM Role	LambdaExecutionRole	Role with Policy for logging
IAM Instance Profile	LambdaInstanceProfile	IAM InstanceProfile that is attached to the EC2 instance
IAM Policy	lambdaaccess	Policy attached to LambdaAccessRole for allowing all lambda and and events actions.
AWS Lambda Permissions	PermissionForEventsToInvokeLambda	Lambda Permission needed to invoke the DeactivateLambda function

Networking

Service Type	Resource Name	Description
VPC	<VPC Name> (parameter)	Desired Virtual Private Cloud
Subnet	terraform-${var.ENV_NAME}/ SubnetPublicX	Desired Public Subnet
Subnet	terraform-${var.ENV_NAME}/ SubnetPrivateX	Desired PrivateSubnet
Internet Gateway	terraform-${var.ENV_NAME}/ InternetGateway	Internet Gateway for VPC
Elastic IP	terraform-${var.ENV_NAME}/ NATIP	Public IP for NAT Gateway
NAT Gateway	terraform-${var.ENV_NAME}/ NATGateway	NAT Gateway to give access to internet from private subnet
Route Table	terraform-${var.ENV_NAME}/	Route Table for Public Subnet

Service Type	Resource Name	Description
	PublicRouteTable	
Route Table	terraform-${var.ENV_NAME}/ PrivateRouteTable	Route Table for Private Subnet

Security

Service Type	Resource Name	Description
Security Group	MetaDefenderSecurityGroup	Generated security group to allow traffic to MetaDefender REST API.

Compute

Service Type	Resource Name	Description
EC2 Instance	${var.ENV__NAME}-${var.APP_ _NAME}-instance	Instance to run MetaDefender Core AMI
Launch Template	${var.APP_NAME}-template	Launch Template for MetaDefender Instance

Autoscaling EC2

The autoscaling deployment is the recommended deployment for production environments that need high availability. This deployment consist of an AWS Auto-Scaling Group (ASG in this page from now on) with instances using the MetaDefender Core AMI published in the AWS marketplace that includes a Standalone MetaDefender Core where the PostgreSQL database is deployed in the same instance than MetaDefender Core.

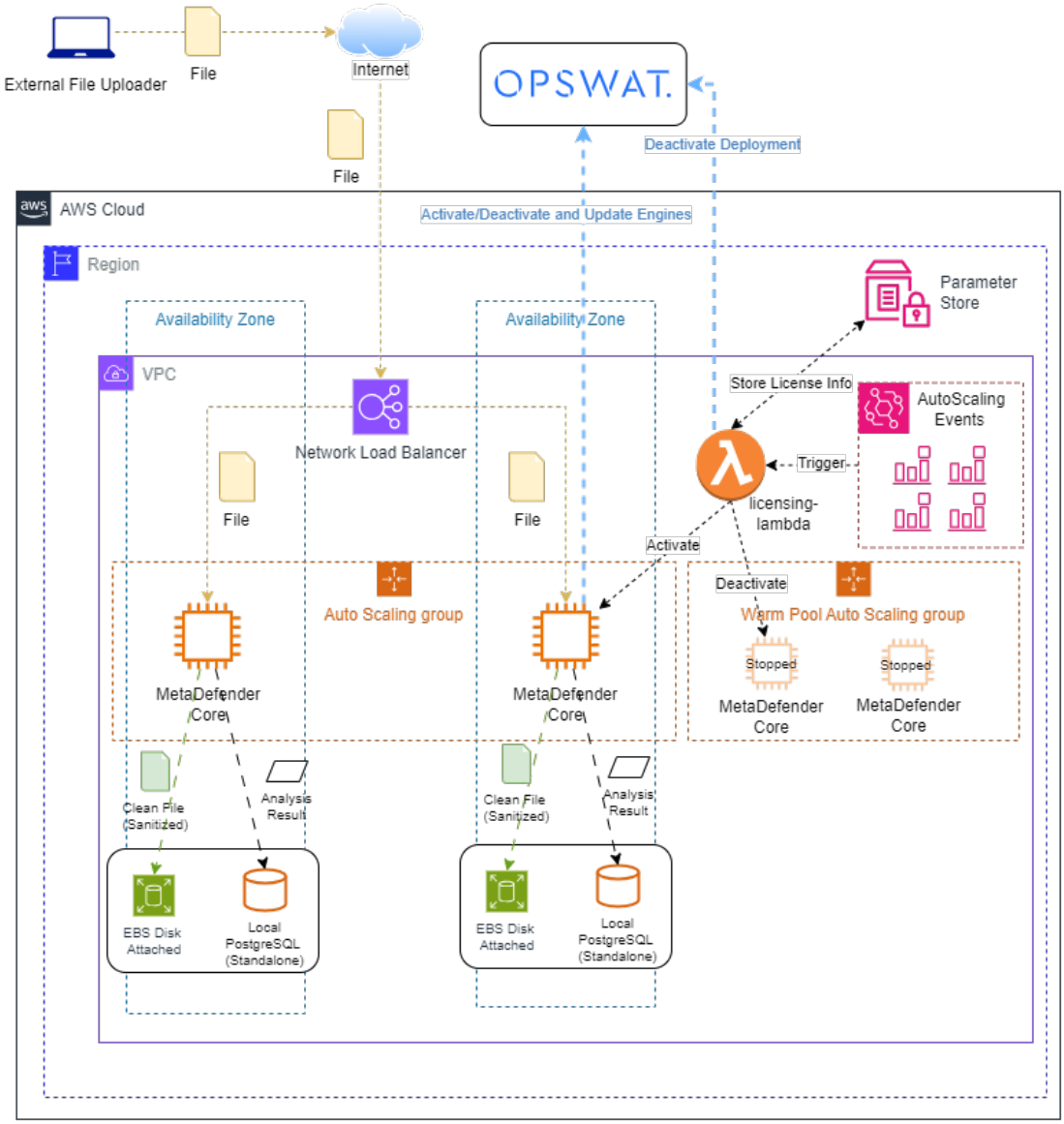
Only MetaDefender Core Diagram

There are 2 flows for MetaDefender Core to work

- **Orange Flow**: Flow to scan a file from internet hitting the network load balancer that is before the autoscaling group. There are other options for sending files to scan, like do it from a private subnet or through a VPN or any other mechanism to hit the network load balancer that is the point to send the request to the MetaDefender Core instances.
- To control the instances **activation**.
  - (**Default/Recommended**) Passing LICENSE_KEY and APIKEY as environment variables through the user-data. The MetaDefender Core instance will activate the instance on start and deactivate it on shutdown.

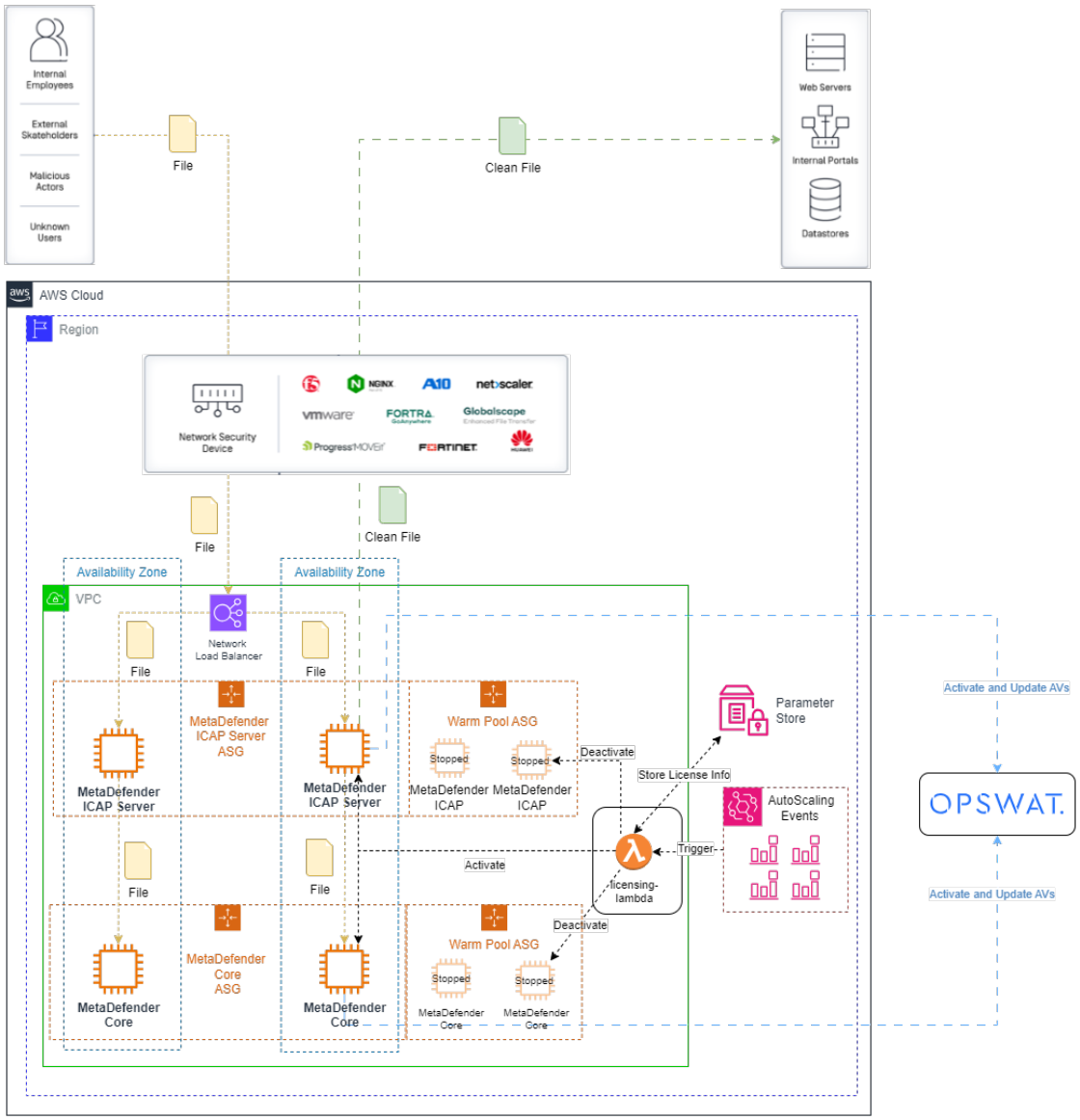
- **(Optional) Black Flow**: The lambda function is triggered based on the ASG events. On activation this flow joins with **blue flow,** the lambda function stores the licensing information in the parameter store.
  - When instance **new in ASG,** Core is activated and instance **started**
  - When instance moved from **ASG to Warm Pool**, Core is deactivated on Scale down and instance **stopped**
  - When instance **new in Warm Pool** ASG, Core is left deactivated and instance stopped
  - When instance move from **Warm Pool ASG to ASG**, instance is **started** and Core is activated

- **Blue Flow**: Each instance communicates with OPSWAT to activate/deactivate and update the AV engines. In case of using lambda function activation, the deactivation happens between the Lambda function and OPSWAT.



MetaDefender Core and MetaDefender ICAP Server Diagram

Example of the recommended architecture for MetaDefender Core and MetaDefender ICAP Server running in ASG.



Resources Inventory

Access Management

Service Type	Resource Name	Description
IAM Role	LambdaAccessRole	Role for Lambdas
IAM Role	LambdaExecutionRole	Role with Policy for logging
IAM Instance Profile	LambdaInstanceProfile	IAM InstanceProfile that is attached to the EC2 instance
IAM Policy	lambdaaccess	Policy attached to LambdaAccessRole for allowing all

Service Type	Resource Name	Description
		lambda and and events actions.
AWS Lambda Permissions	PermissionForEventsToInvokeLambda	Lambda Permission needed to invoke the DeactivateLambda function

Networking

Service Type	Resource Name	Description
VPC	<VPC Name> (parameter)	Desired Virtual Private Cloud
Subnet	terraform-${var.ENV_NAME}/SubnetPublicX	Desired Public Subnet
Subnet	terraform-${var.ENV_NAME}/SubnetPrivateX	Desired PrivateSubnet
Internet Gateway	terraform-${var.ENV_NAME}/InternetGateway	Internet Gateway for VPC
Elastic IP	terraform-${var.ENV_NAME}/NATIP	Public IP for NAT Gateway
NAT Gateway	terraform-${var.ENV_NAME}/NATGateway	NAT Gateway to give access to internet from private subnet
Route Table	terraform-${var.ENV_NAME}/PublicRouteTable	Route Table for Public Subnet
Route Table	terraform-${var.ENV_NAME}/PrivateRouteTable	Route Table for Private Subnet

Security

Service Type	Resource Name	Description
Security Group	MetaDefenderSecurityGroup	Generated security group to allow traffic to MetaDefender REST API.

Compute

Service Type	Resource Name	Description
EC2 Instance	${var.ENV__NAME}-${var.APP_ _NAME}-instance	Instance to run MetaDefender Core AMI
Launch Template	${var.APP_NAME}-template	Launch Template for MetaDefender Instance

AWS Database Services

Support AWS Services with PostgreSQL Engines

MetaDefender Core natively supports AWS RDS and Aurora with PostgreSQL engine v14/15/16 (Supported PostgreSQL versions). MetaDefender Core does not require any specific pre-installation steps to work with Amazon RDS - all required extensions are performed internally during installation of the product.

Regarding Aurora support, as it generates 2 endpoints, **Cluster endpoint** for administrate, insert and read. And the **reader endpoint** that is only for reading using the LB that get the information from any instance of the cluster. MetaDefender Core only supports connections to 1 endpoint, the cluster endpoint, that allow the application do all the actions needed. Info from AWS

Database Configuration

High Availability with AWS RDS

AWS RDS supports various multi-az deployment that can be configured following this documentation. You can compare the different configuration options available in this AWS documentation. For having MetaDefender Core installed using any of this approaches we recommend to follow the instructions indicated by AWS in case of having Core installed in a EC2 instance. For EKS Cluster deployment OPSWAT provide a terraform code to deploy it together with the K8S cluster, see EKS Cluster Provisioning

Database Sizing

It depends on how many requests or traffic is sent to MetaDefender Core, and how complex the files are. As average the minimum database size desired **for each 1,000,000 analysis reports** added is **4.5GB.**

The recommended AWS RDS Instance Type is **db.r6g.xlarge** and the storage type recommended is General Purpose SSD. For gp2 storage type **the minimum size is 20 GiB** so this is marking as the minimum storage needed for working with MetaDefender Core

Security Group Configuration

MetaDefender Core does not require that the RDS database is publicly accessible. For enhanced security, OPSWAT recommends the following:

- Do NOT deploy the database in a public subnet.
- Do NOT include ingress security group rules with 0.0.0.0/0 sources
- Do NOT set the Amazon RDS Publicly Accessible parameter to true in terraform project for Kubernetes deployment

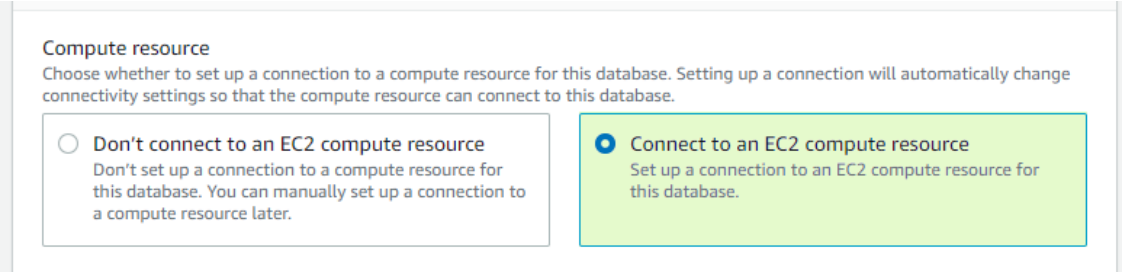
Database Connection

For general information of how to connect to the RDS depending on the scenarios it is recommended to read this docs from AWS

Connect to RDS in Private Subnet

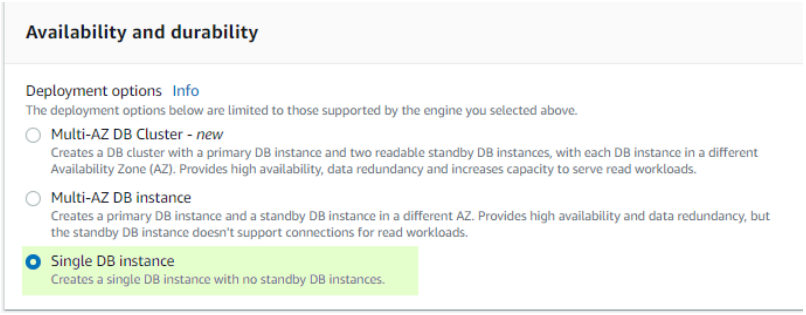
To connect MetaDefender Core with RDS hosted in a private subnet, the configuration will depend on the type of deployment.

For Single EC2 deployment, it is needed to indicate that the RDS will be only available from the EC2 instance where MetaDefender Core is running, configuring it from the Connectivity section when creating the RDS instance as following.

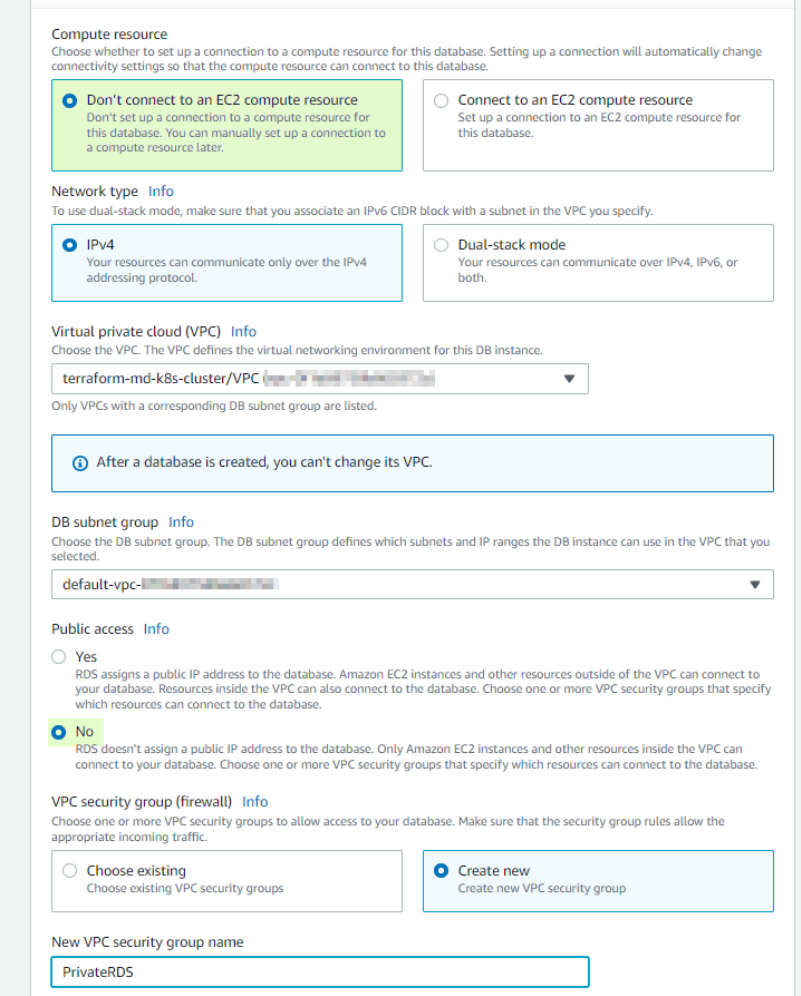


For EKS deployment, using the terraform project provided by OPSWAT it is deployed in a private subnet only allowing access from VPC CIDR where the EKS cluster is deployed. In case of creating the RDS instance from the console follow these steps

1. Indicate if you want to deploy it in Multi-AZ or Single DB



2. Indicate not to connect to an EC2 and the VPC public access "No". For multiple VPC, deploy it in a different VPC than EKS. This example is using the same VPC



Multiple VPC for MetaDefender Core and AWS RDS

When working with multiple VPCs where we have deployed MetaDefender Core either in a Single EC2 or with EKS in one VPC and the RDS instance in a different VPC, it will be needed to configure a VPC peering between both VPC for MetaDefender Core to use the database.

Steps to configure it

1. Create a Peering Connection between both VPCs
2. Modify the route table to route the traffic to the peering for the requests from both VPCs
3. Modify the security group to allow connections from both VPCs

Database Endpoint Management

The database connection endpoint is generated by AWS once the RDS instance is created you can access to this information on the **Connectivity & security** tab. Once you have this information it should be set up in the ConfigMap "**mdcore-env**", setting up the hostname of the database to DB_HOST and the port to DB_PORT

More detailed information from AWS here

Database Encryption

Database Encryption In Transit

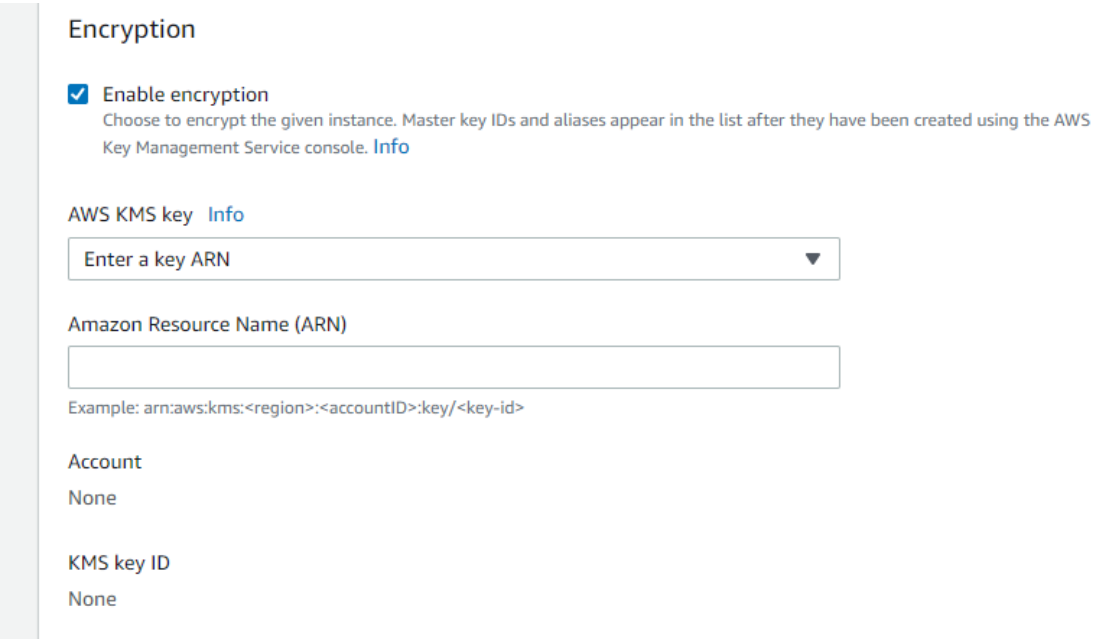
Network traffic processed by MetaDefender Core is SSL encrypted during transmission. The product does not collect or store any customer data outside of analyzing the contents of files submitted for scanning.

During the scanning process, files are stored unencrypted on disk to allow the scan process to complete. Once the scan completes, the file is deleted unless configured to place the file in quarantine. If you wish to encrypt these quarantined files, please use native OS encryption or EBS encrypted volumes.

Database Encryption At Rest

MetaDefender Core supports both native OS encryption or EBS encrypted volumes. The database is secured via credentials and can be configured to encrypt everything at rest. In general, OPSWAT recommends the use of a 3rd party database layer, e.g. Amazon Aurora or RDS, which should be encrypted. It is best practice to not share the database layer on the same hosts/VM/EC2 instance.

For encrypting RDS this is the recommended configuration



Licensing Automation on AWS

To activate the instances where the MetaDefender products are running, it is needed to activate these products providing the LICENSE_KEY.

There are 2 options to manage the licensing that will automate the activation and deactivation of the products. This automation applies for the MetaDefender Core and MetaDefender ICAP Server products.

Setting Up Environment Variables

This option is the **default and recommended mechanism** to automate the licensing management, as this is the easiest way to activate and deactivate the products.

When passing the LICENSE_KEY as environment variables, to the user-data, it will set up the automation that will activate the product when the instance is started and will deactivate the product when the instance is terminated or

stopped.

It is also recommended to indicate the APIKEY environment variable as this will set up the APIKEY to the admin user to be able to make requests to the MetaDefender Core API. In autoscaling deployment types it will help to manage all the instances with the same APIKEY when hitting the AWS Load Balancer.

Limitations

Deactivation with passwords other than instanceID

In case the user changes the password to anything other than `instanceID`, the auto deactivation function of the shutdown script will fail. A way to fix this is to add `APIKEY` as a parameter.

Windows images

Since the shutdown script on Windows platforms is configured using Local Group Policy, attaching Active Directory (AD) to a virtual machine (VM) created from our Windows images may cause the AD Group Policy to override the Local Group Policy, preventing the script from running.

Using Lambda Function

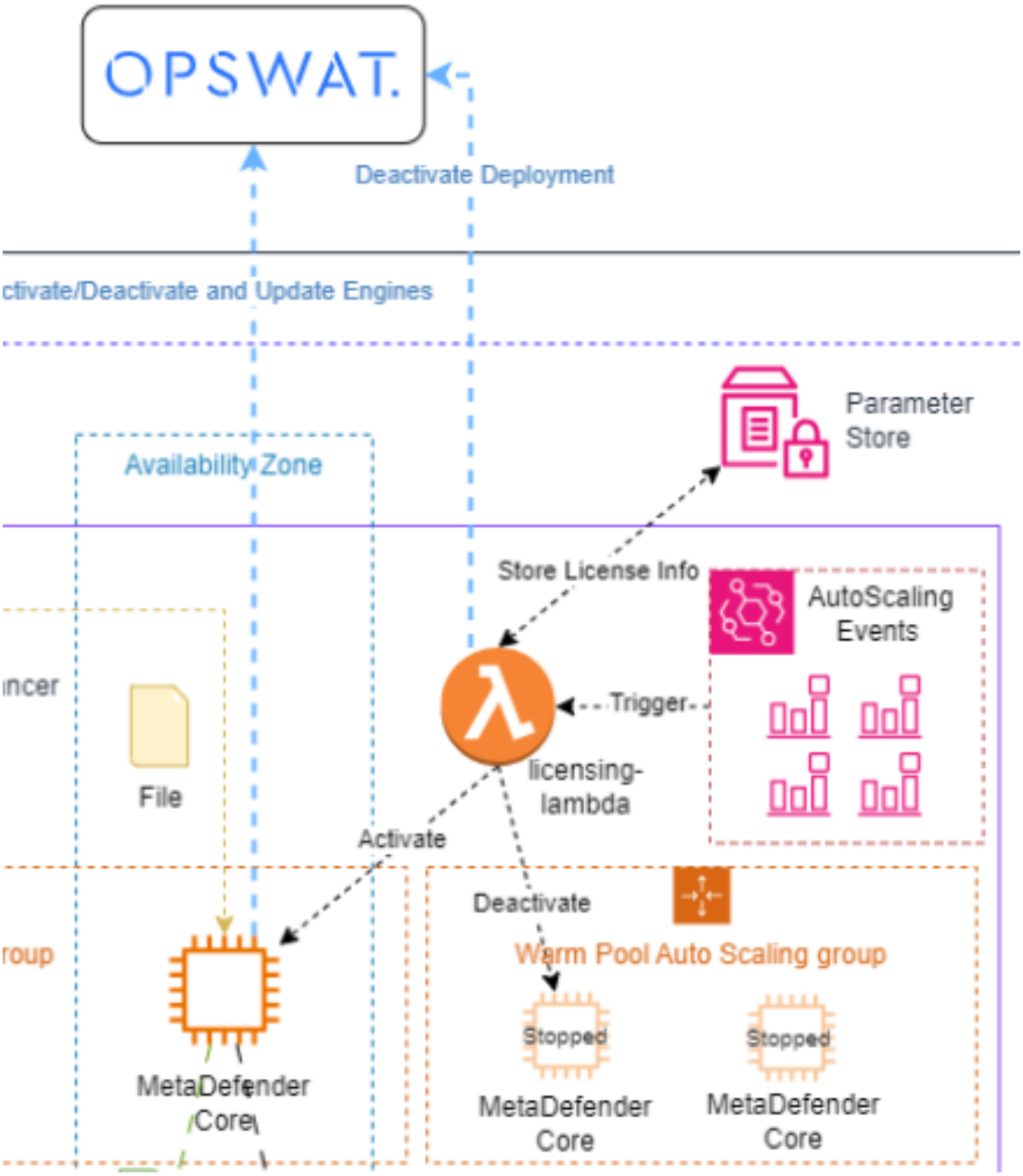
> **Warning**
>
> Do not enable the Lambda function automation and set up the environment variables from the user data of the instances, as this will generate a conflict and neither option will work.

This option is for handling the license automation in case there are any issues when the logic, that is included inside the instances, fails.

This option can be enabled from the terraform project, see How to deploy using Autoscaling EC2.

The automation consists of the following services:

- AWS Lambda function - to manage the activation/deactivation and store the deployment information.
- AWS Parameter Store - to store the deployment information
- Autoscaling Cloudwatch Events - Generated based on AWS ASG lifecycle hooks



Licensing when running in Kubernetes

The licensing automation options when the MetaDefender products are installed in a Kubernetes cluster, are explained in Licensing in K8S

How to deploy using AWS services

Introduction

This Deployment Guide provides step-by-step instructions for deploying MetaDefender Core on Amazon Web Services infrastructure.

Organizations interested in protecting their solutions deployed in AWS can leverage MetaDefender to analyze and sanitize files residing, or transitioning, their AWS deployment. MetaDefender can scan and either sanitize or check for known vulnerabilities, depending on the type of traffic it's seeing. The ideal use case would be an organization that allows files to be uploaded to AWS through an external facing web portal. Analyzing files before they are made accessible to the end-users is critical to ensure that no malicious content is allowed and distributed through the web application.

Advanced attacks are concealing the malicious payload and are relying on productivity files (documents, pdfs, images) as a distribution mechanism. Productivity files allow active content to be leveraged, but these features are frequently exploited to execute the malicious behavior.

This guide is for IT infrastructure architects, administrators and DevOps professionals who are seeking to prevent potential malicious traffic being allowed in their AWS Cloud deployment. Threat Prevention is ensured for both productivity files that might be uploaded and for known vulnerabilities that can be identified for all running services/ applications deployed in AWS. The vulnerability scanning is checking known vulnerabilities for unpatched OS and running applications.

MetaDefender is provided as:

- AMI through the AWS Marketplace (MetaDefender products)
- Packaged installer available for download through the My OPSWAT
- Helm Charts for MetaDefender products in GitHub together with the container images available in the OPSWAT Docker Hub Repository

There are currently no restrictions on AWS regions in which you can deploy MetaDefender. The only caveat is that MetaDefender is not available in AWS GovCloud at this time.

OPSWAT Contact Information

Sales:sales-inquiry@opswat.com

Support:https://www.opswat.com/support

Other:https://www.opswat.com/contact

For installing our solution, please review the guideline listed below:

- Single EC2
- Autoscaling EC2
- EKS Cluster

Database Configuration

MetaDefender Core supports Amazon Aurora or AWS RDS that are service that deploy PostgreSQL servers provided by AWS. This section information is extended in AWS Database Services

EC2 Key Management

The EC2 key pair is only used to SSH or RDP to the EC2 instance. This is a very uncommon scenario, used primarily for debugging. You can create a new key pair, or use an existing pair.

Costs

MetaDefender is made available as an annual subscription. The only mandatory service that needs to be purchase is MetaDefender Core. MetaDefender Core is a paid service and is currently licensed on the BYOL (Bring Your Own License) model. Contact our sales team via our Contact form, available here: https://www.opswat.com/contact

Considering that are over 60 different options to license MetaDefender, it is highly coupled to the use case and the advanced features that you are considering deploying. Note that the more functionality you are adding to MetaDefender, the more need will be of CPU from the EC2 instance.

Our recommendation would be a minimum 8 vCPU for our lower tiers.

We recommend General Purpose (m family) and Compute Optimized (c family) instance types, preferably that are using NVMe-based SSDs for high throughput expectations:

- m5.xlarge
- m5.2xlarge
- m5.4xlarge
- c5d.2xlarge
- c5d.4xlarge
- c5d.8xlarge

We recommend to go with Reserved Instances, considering that you are committing to an annual subscription for MetaDefender.

For pricing per instance, please refer to the official AWS pricelist: https://aws.amazon.com/ec2/pricing/

Sizing

MetaDefender needs an EC2 instance with minimum 8 vCPU, in order to have an optimal response rate to submitted files for analysis. However, depending on the use case and expected throughput (analysis SLA), higher tier instances are recommended.

The system requirements (hardware and supported operating system) are defined here: Recommended System Configuration

Operation Guidance for EC2 deployments

Monitoring

For monitoring indications please read the Monitoring in AWS page created for this topic.

Troubleshooting

For troubleshooting indications please read the troubleshooting page created for this topic.

Backup and Recovery

MetaDefender service in general does not store any persistent data that needs to be recovered or backed-up. Regardless if it's a hardware or service failure, MetaDefender will not recover the files submitted in the queue and those files will need to be resubmitted for analysis. Using OPSWAT Central Management (OCM) or a central database (such as AWS Aurora/RDS), there is no backup required per service. All configurations will automatically be

shared with the new instances when they are brought up (this is the recommended model to prevent data loss). In order to have a sustainable service, not affected by any hardware or software failures, the recommendations are to:

- Store the files in an S3 bucket before they are submitted to MetaDefender
  - In order to avoid any AZ or Region failure, would be recommended to use S3's Cross-Region Replication
- Define a pool of jobs for your webapp, which are in a pending state while MetaDefender analyzes the files
  - Best option would be to use the SQS service and your application to send the files from the SQS to MetaDefender
    - Every time the files are being uploaded to a temporary S3 bucket, there is a new job added to the SQS
  - If MetaDefender fails, the job is stilled defined in SQS
  - If AZ or even the region fails, if the S3 bucket is synced in multiple regions, the SQS queue should be easily be able to be reconstructed for the remaining files in the bucket
    - On initializing stage, check the files in the S3 bucket and if there is any file left, add them to the SQS
- Based on the result from MetaDefender, the file should be moved to the final location or it's sanitized copy should. Either way, the original file should be deleted from the temporary S3 bucket
- When the job is being removed from SQS, remove also the file from the S3 bucket

In case of failure of MetaDefender service, follow the instructions defined in the Troubleshooting section: Troubleshooting MetaDefender Core

Support

To receive support, please visit our Support Portal. From there after you log in click on "Cases" on menu at the top of the page and then click "New" on the right hand side to create a new case.

Support policies, costs, levels and SLA's are described on our website, at the Support section: https://www.opswat.com/support. Please view our support tiers in the table below:

*Note: Please visit*https://www.opswat.com/support*for the most updated information on support.*

	Standard	Gold	Platinum
Prices	Included with your license	20% ($500.00 minimum) *	25% ($2000.00 minimum) *
Support Hours			
Support via Portal Ticket System	8AM to 5PM business days (local to you)	7AM to 7PM business days (local to you)	24 x 7 x 365

	Standard	Gold	Platinum
Support via Chat	8AM to 5PM business days (local to you)	7AM to 7PM business days (local to you)	24 x 7 x 365
Support via Phone			24 x 7 x 365
Incident Response Times			
Blocker Issues First Response Times	Within 1 business day	Within 2 hours	Within 1 hour
Non-blocker Issues First Response Times	Within 3 business days	Within 2 business days	Within 1 business day
Support Account Management			Yes
Support Account Manager			Yes
Quarterly Conference Call Reviews			Yes
Yearly Roadmap Review with the Product Team			Yes
Quarterly Summary Reports		Yes	Yes

Accessibility

Reference Materials

All MetaDefender documentation is available on docs.opswat.com/mdcore

Localization

MetaDefender products and documentation are available exclusively in English for now.

EKS Cluster

This guide explains how to use the provisioning script provided by OPSWAT to create an AWS EKS and generate all the Kubernetes components needed to run MetaDefender Core.

Deploy using Terraform

- Locally clone the metadefender-k8s repository and go to terraform folder

Code · bash

```
git clone git@github.com:OPSWAT/metadefender-k8s.git
cd terraform/aws/
```

- Modify variables/variables.tfvars file
  - PERSISTENT_DEPLOYMENT=true -> To create K8S cluster with EC2 Worker Nodes
  - DEPLOY_FARGATE_NODES=false -> To true if wanted to create a Fargate profile
  - DEPLOY_RDS_POSTGRES_DB=true -> To create RDS instance for PostgreSQL server
  - POSTGRES_USERNAME -> if DEPLOY_RDS_POSTGRES_DB=true, set up admin database username
  - POSTGRES_PASSWORD -> if DEPLOY_RDS_POSTGRES_DB=true, set up admin database password

variables/variables.tfvars · bash

```
MD_CLUSTER_NAME        = "md-k8s"
MD_CLUSTER_REGION      = "eu-central-1"
ACCESS_KEY_ID          = "<ACCESS_KEY_ID>"
SECRET_ACCESS_KEY      = "<SECRET_ACCESS_KEY>"
PERSISTENT_DEPLOYMENT  = true
DEPLOY_FARGATE_NODES   = false
DEPLOY_RDS_POSTGRES_DB = true
POSTGRES_USERNAME      = "<POSTGRES_USERNAME>"
POSTGRES_PASSWORD      = "<POSTGRES_PASSWORD>"
```

- Run terraform init and apply. Check the resource to be created, after that enter "y"

Code · bash

```
terraform init
terraform apply
```

MetaDefenderK8S script details

- Script path: https://github.com/OPSWAT/metadefender-k8s/blob/main/metadefenderk8s.sh

- **Programming Language:** Bash
- Installation Pre-requisites for provisioning:
  - Terraform
  - Helm
  - AWS CLI
  - Kubectl
  - eksctl (When using the lb controller script)
- AWS Credentials
  - Set in your local environment credentials the IAM user credentials under ***AWS_ACCESS_KEY_ID***& ***AWS_SECRET_ACCESS_KEY***
- MetaDefender Core License Key (Required with --mdcore parameter)
  - Set it in your local environment credentials under ***MDCORE_LICENSE_KEY***

> **Warning**
>
> As security measure do not use the root user for any deployment or operation. Follow the principle of least privilege for all access granted. Keep in mind the resources listed in this page for your option selected: EKS Cluster Architecture

How to run the script

Only MD Core · bash

```
./metadefenderk8s.sh provision -l AWS --mdcore
```

MD Core + MDSS · bash

```
./metadefenderk8s.sh provision -l AWS --mdcore --mdss
```

MD Core + ICAP · bash

```
./metadefenderk8s.sh provision -l AWS --mdcore --icap
```

Script Parameters

Parameter	Flags	Options	Default	Description	Required/ Optional

Parameter	Flags	Options	Default	Description	Required/ Optional
Action		• provision • install		Action to indicate the script if we want to provision (Create resources + install Core) or install (Install Core)	**Required**
Location	-l or --location	• AWS • Azure • GCP		Where is going to be the K8S cluster	**Required**
MetaDefender Flag Installation	Combination of • --mdcore • --mdss • --icap		•	Install MetaDefender Core in the cluster provisioned	**Required**
Image Version	--image	• latest • <docker-image-tag>	latest	MetaDefender Core image version to install	Optional
Region	--region	AWS Regions	eu-central-1	AWS region where all the resources will be provisioned	Optional
Cluster Name	--name	Not uppercase allowed	md-k8s	Name of the cluster that will be used for naming all the resources	Optional
Number of Replicas	--replicas	[0-9]*	1	Number of replicas for	Optional

Parameter	Flags	Options	Default	Description	Required/ Optional
				MetaDefender Core service	
Namespace	--namespace	[A-Za-z]{1,10}		Namespace where MetaDefender products will be installed in the K8S Cluster	Optional Max Characters: 10

Limitations using parameters

- When using Fargate
  - the pods should be installed in the default namespace.
  - if using AWS Load Balancer controllers need a worker node to be installed in.

> **Info**
>
> The script will deploy a single Worker Node for the cluster with enough space for 1 replica of MD Core. AWS EC2 Instance size is **c5.2xlarge (8 vCPU & 16 GiB Memory).** Each pod would request 7 vCPU and 8 GiB Memory. To change the request to adapt each pod to the specific case go to values.yml To change the node group go to the terraform file terraform/aws/eks_cluster_module

MetaDefender Core Flowchart Provisioning in AWS

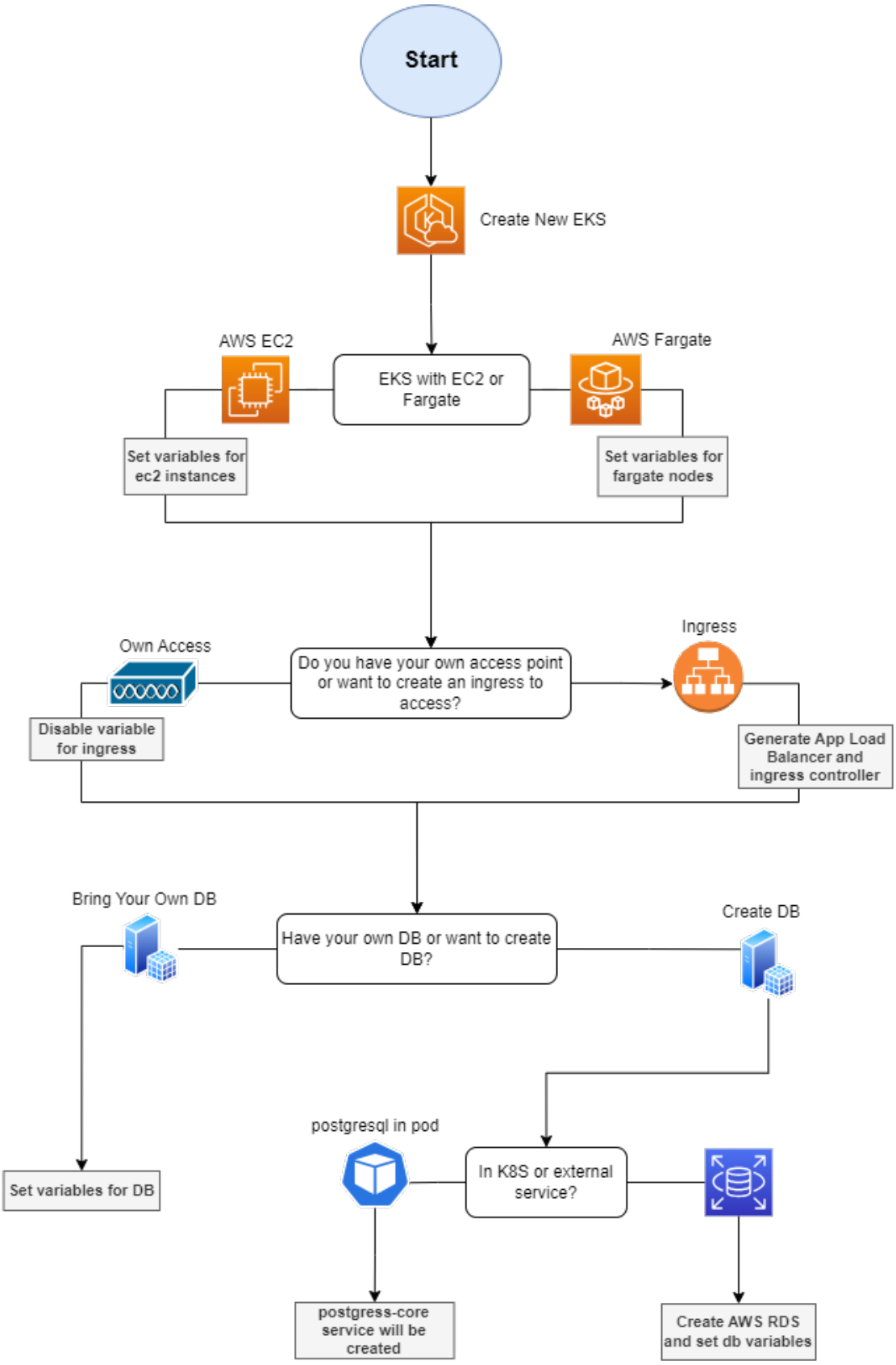
The following flowchart represents how the provisioning script will configure the environment based on the options selected for provisioning AWS EKS.

Summary options to be selected

1. EKS with **EC2** or **Fargate**
2. Access to the K8S cluster. Generate **Ingress** or provide **own** access.
   1. **An Ingress** will be create per each product flag added as parameter to the script
   2. **Own Access**, you decide how to access to the cluster so it won't generate any load balancer for accessing
3. Have your **own database** or create **new database**
   1. **Own database,** will be asked if you want the script to set up the credentials and database host url for you or the script will just indicate the secrets to edit, later on by you, for connecting the MetaDefender Core with your

database.

2. Create new **DB in K8S** or **external DB** that for AWS we will provision a ***RDS instance***



Single EC2

For a single deployment of MetaDefender in a public subnet. In this example, besides the MetaDefender EC2 instance, additional resources are being generated and set up.

Prerequisites and Requirements

AWS knowledge: This guide assumes familiarity with AWS Services

AWS account: needs permission to create AWS Services listed in Recommended Architecture Single EC2

Tools installation: Terraform

System/OS requirements

For OS, database type, 3rd party dependencies and storage requirements on Windows installations, please refer to the *Microsoft Windows Deployments* section of the Recommended System Configuration page.

For OS, database type, 3rd party dependencies and storage requirements on Linux installations, please refer to the *Linux Deployments* section of the Recommended System Configuration page.

Deploy using Terraform

- Locally clone the metadefender-csp repository and go to AWS/single-ec2

Code · bash

```
git clone git@github.com:OPSWAT/metadefender-csp.git
cd AWS/single-ec2
```

- Modify terraform.tfvars with the desired options
  - LICENSE_KEY required if wanted to have the instance activated automatically
  - APIKEY_GENERATION=true to have the apikey generated by terraform
  - There is one general section and one section for each product that is supported using this Terraform project

terraform.tfvars · bash

```
# General variables

MD_ENV_NAME             = "metadefender" # Prefix to add to all the resources
MD_REGION               = "eu-central-1" # Region for all the resources
EC2_KEY_NAME            = "" # Key pair to attach to EC2 instances (Optional)
#ACCESS_KEY_ID          = "<ACCESS_KEY_ID>"
#SECRET_ACCESS_KEY      = "<SECRET_ACCESS_KEY>"  # To give access to terraform (Optional, can
use other ways to authenticate)
PUBLIC_ENVIRONMENT      = true
APIKEY_GENERATION       = true
```

```
# MetaDefender Core variables

DEPLOY_CORE             = true
MD_VPC_CIDR             = "192.168.0.0/16"  # VPC CIDR where to create the MetaDefender products
CORE_PRODUCT_ID         = "ani6v4vb5z4t87cymrfg3m451" #MetaDefender Core ID in AWS Marketplace
|| For Windows it is "9s8powksm1cj7fuafdnv0sfj9"
CORE_INSTANCE_TYPE      = "c5.2xlarge"   # Instance type for MetaDefender Core
LICENSE_KEY_CORE        = ""
```

- In case of wanting to deploy MetaDefender ICAP Server or MetaDefender Storage Security, modify these sections

terraform.tfvars · bash

```
# MetaDefender ICAP variables

DEPLOY_ICAP             = false                    # true to deploy ICAP together with Core
ICAP_PRODUCT_ID         = "b1w10ei2pw2vgpdsjw44pbffr" #MetaDefender ICAP LINUX ID in AWS
Marketplace
ICAP_INSTANCE_TYPE      = "c5.2xlarge"   # Instance type for MetaDefender Storage Security
LICENSE_KEY_ICAP        = ""

# MetaDefender Storage Security variables

DEPLOY_MDSS             = false        # true to deploy MetaDefender Storage Security together
with Core
MDSS_PRODUCT_ID         = "3mup1qubt6hwmup405eljau0k" # MetaDefender Storage Security LINUX ID
in AWS Marketplace
MDSS_INSTANCE_TYPE      = "c5.2xlarge"   # Instance type for MetaDefender Storage Security

DEPLOY_MDSS_DOCUMENTDB  = false  # true to deploy  MetaDefender Storage Security with a managed
instance of Amazon DocumentDB
MDSS_DOCUMENTDB_INSTANCE_CLASS = "db.r5.large"
DEPLOY_MDSS_ELASTICACHE = false  # true to deploy  MetaDefender Storage Security with a managed
instance of Elasticache Redis
MDSS_ELASTICACHE_NODE_TYPE = "cache.m5.large"
```

- Run terraform init and apply. Check the resource to be created, after that enter "y"

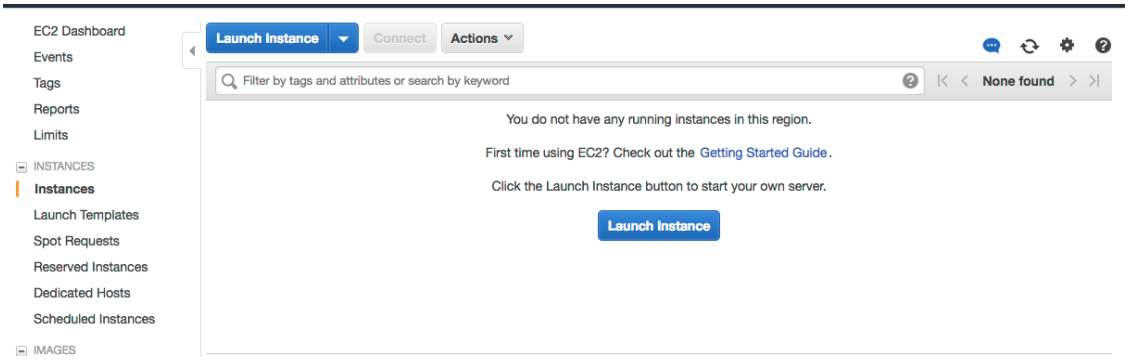
Code · bash

```
terraform init
terraform apply
```

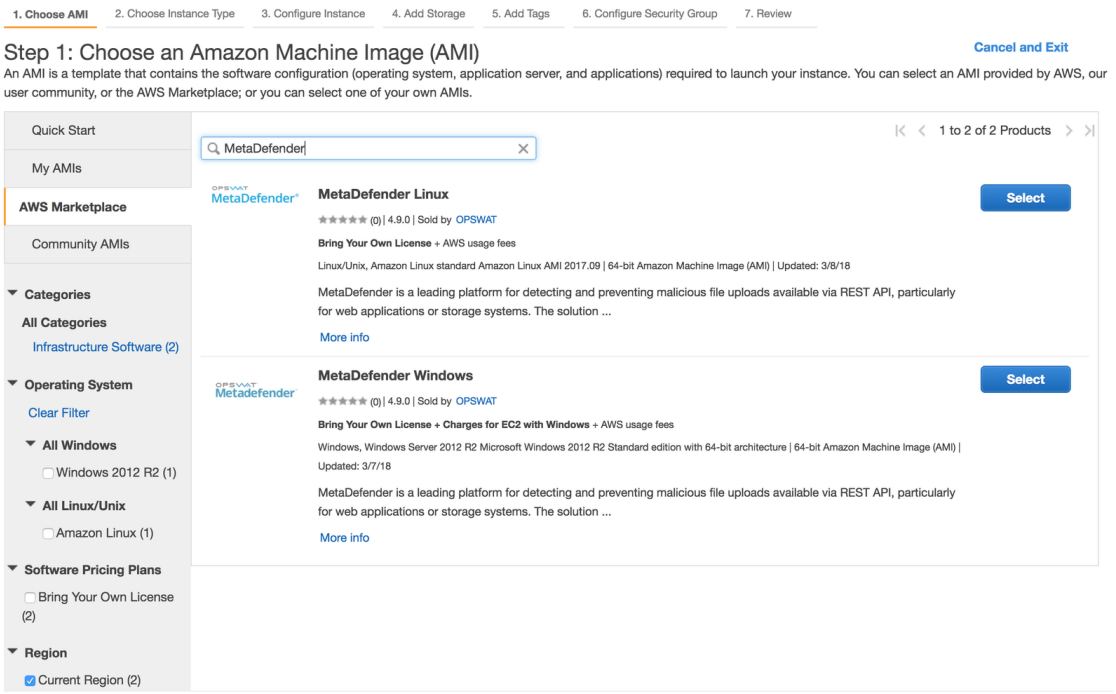
Deploy using AWS Console

Select MetaDefender Core AMI

- Go to the EC2 Management Console in AWS and select Launch Instance

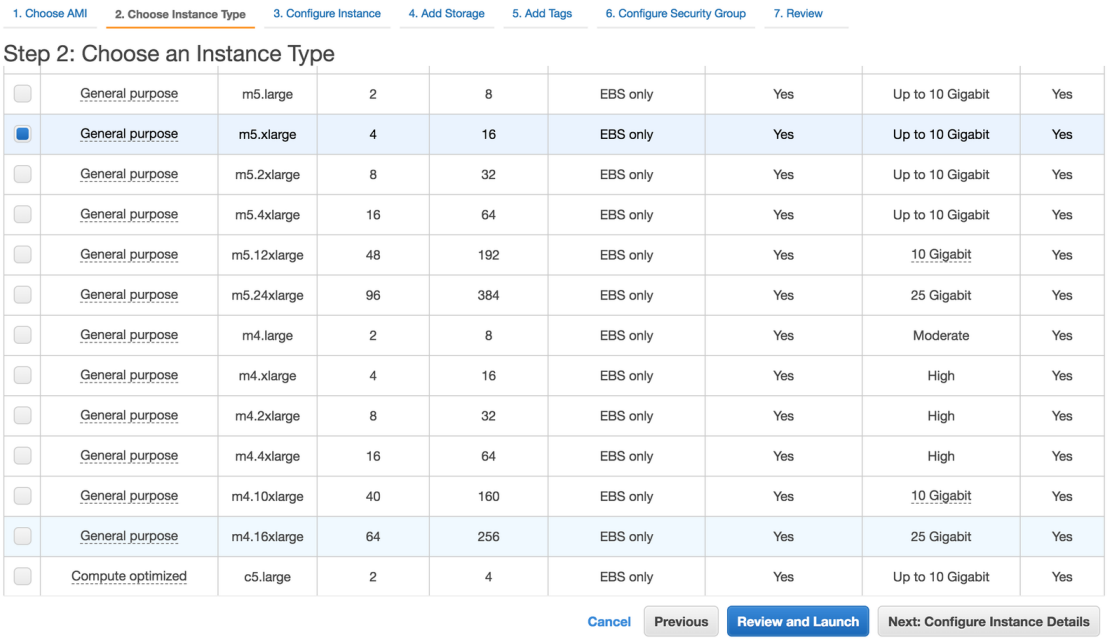


- Select the desired MetaDefender, based on the OS support:



- Choose the Instance type

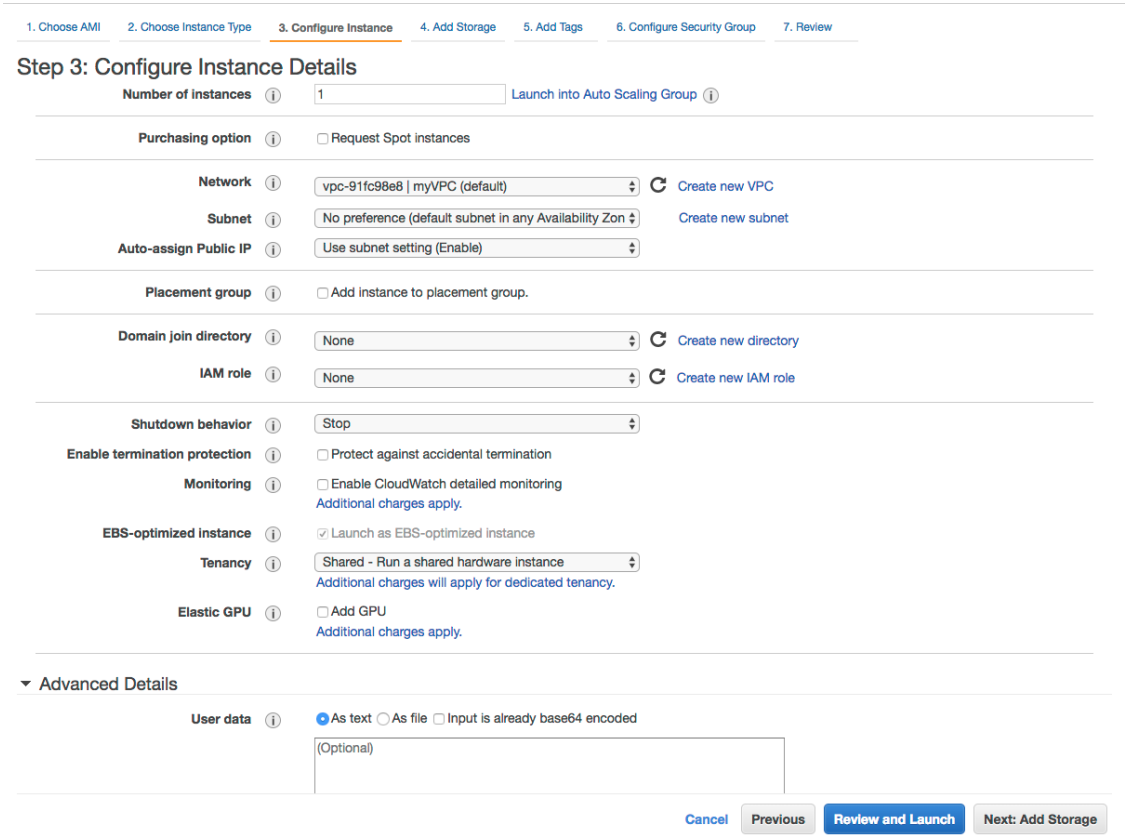
Please review MetaDefender's system requirements (OS and hardware requirements) before choosing the desired AMI and instance type.



☐	General purpose	m5.large	2	8	EBS only	Yes	Up to 10 Gigabit	Yes
☑	General purpose	m5.xlarge	4	16	EBS only	Yes	Up to 10 Gigabit	Yes
☐	General purpose	m5.2xlarge	8	32	EBS only	Yes	Up to 10 Gigabit	Yes
☐	General purpose	m5.4xlarge	16	64	EBS only	Yes	Up to 10 Gigabit	Yes
☐	General purpose	m5.12xlarge	48	192	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m5.24xlarge	96	384	EBS only	Yes	25 Gigabit	Yes
☐	General purpose	m4.large	2	8	EBS only	Yes	Moderate	Yes
☐	General purpose	m4.xlarge	4	16	EBS only	Yes	High	Yes
☐	General purpose	m4.2xlarge	8	32	EBS only	Yes	High	Yes
☐	General purpose	m4.4xlarge	16	64	EBS only	Yes	High	Yes
☐	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m4.16xlarge	64	256	EBS only	Yes	25 Gigabit	Yes
☐	Compute optimized	c5.large	2	4	EBS only	Yes	Up to 10 Gigabit	Yes

Networking for EC2

- Select the desired VPC and subnet you would like to have MetaDefender deployed.



Depending on the deployment model, the recommendation would be to deploy MetaDefender in a private subnet, with no Internet Connection.

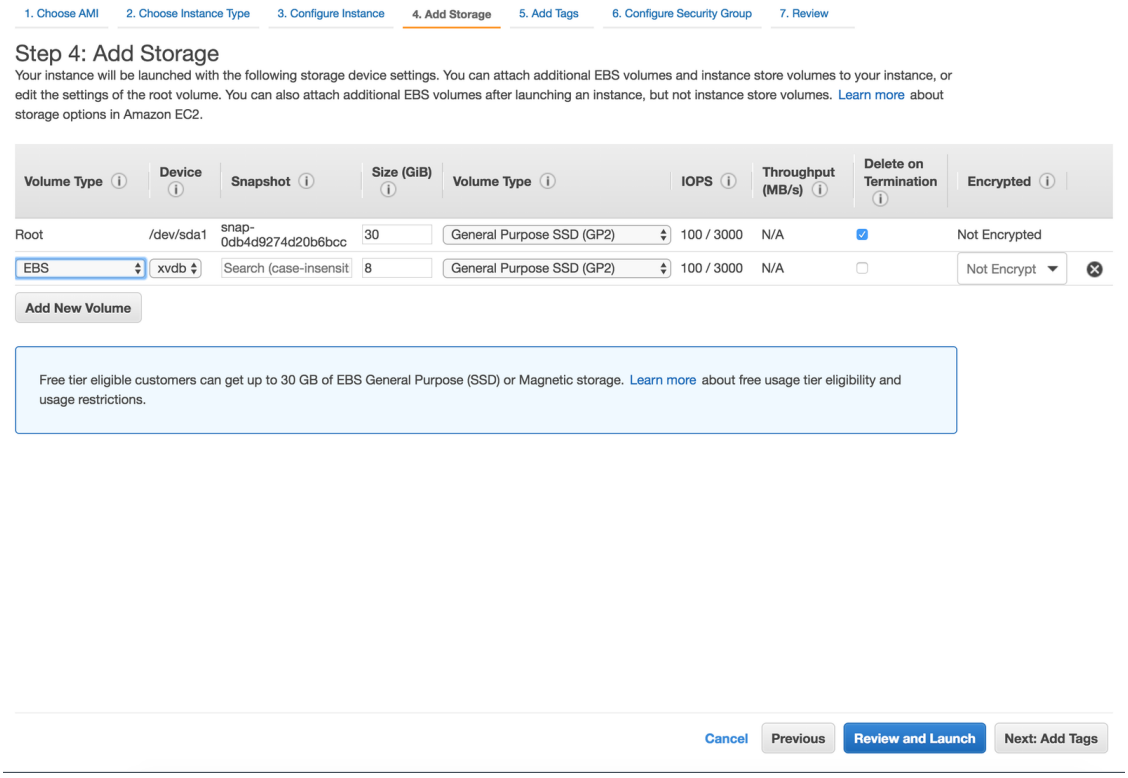
Storage configuration

Storage step can be skipped

In general there's no need for additional storage by MetaDefender. However, there are 2 situations where additional local storage might be required:

- MetaDefender will process large files or a high volume of files which submitted in MetaDefender's queue will need over 10GB files storage
- MetaDefender is configured to store files in the Quarantine section which will eventually fill the entire root volume.

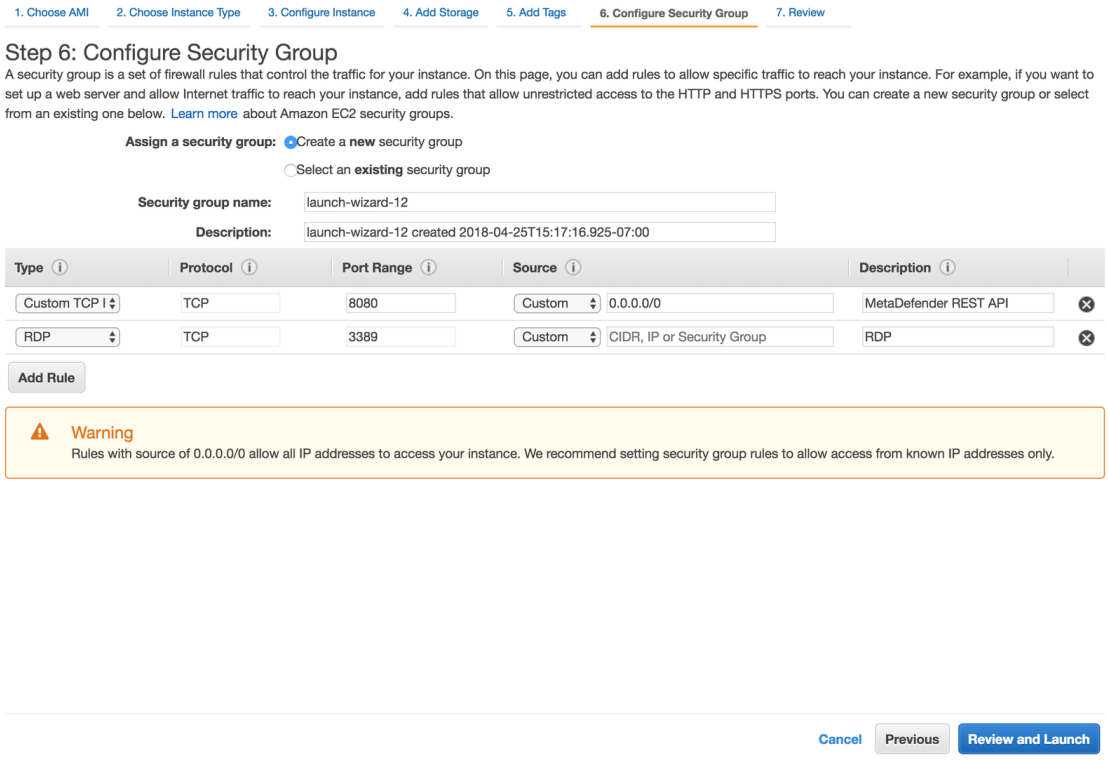
In case quarantining the files in the MetaDefender instance or analyzing high volumes of files is a requirement, please consider adding an EBS volume.



Security Groups configuration

There are 2 rules that are needed to be opened during the installation phase, after which only one rule should remain:

- Custom TCP: 8008
  - MetaDefender exposes it's REST API by default to port 8008. However this port can be changed during the installation phase or updated during it's lifetime.
- RDP / SSH
  - In order to install MetaDefender on Windows add access for RDP and for Linux add support for SSH
    - Highly recommended would be not to allow traffic to RDP or SSH from anywhere, but to limit to your IP address

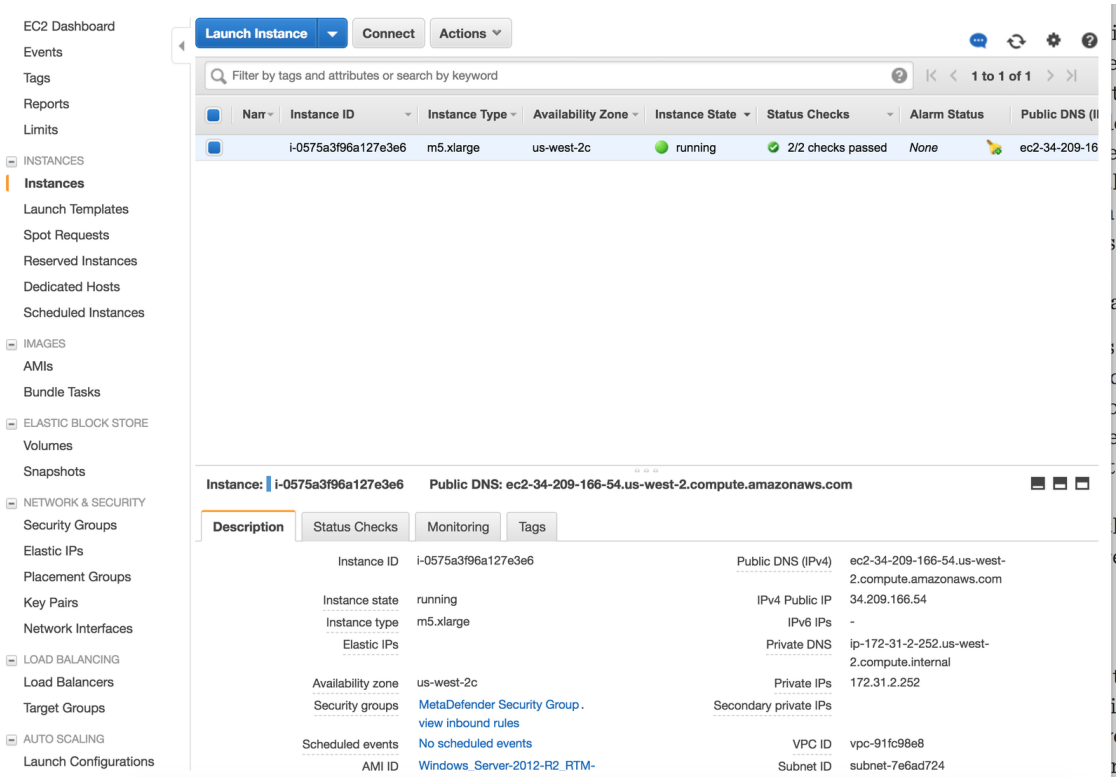


Warning

After the installation remove RDP or SSH from this Security Group!

Launch the instance

After you review the settings, hit Launch. In a few minutes the instance should become available



Licensing Automation

To automate the activation and deactivation of MetaDefender Core the marketplace image contains an automation that activate the product on start and deactivate the product on graceful stop/termination.

Customize OPSWAT MetaDefender AMI from marketplace

> **Warning**
>
> Please follow these instructions before generating a custom AMI from the OPSWAT AMI published in the AWS marketplace.

- Default username and password is `admin`/`<instance-ID>`
  - "admin" is an unacceptable password. Whenever the end-user changes the password to "admin", it will automatically change the password to `<instance-ID>` after the Instance is rebooted.
- To create custom AMI from OPSWAT AMI, some mandatory things need to be noted before converting:
  1. ***(Required)*** Deactivate the MetaDefender Core license in the EC2 instance from where the new AMI will be created.
  2. ***(Optional)*** Change MD Core password.
     - If the end-user changes the password to "admin", their default password after creating a custom AMI will be `<instance-ID>`
     - If the end-user does NOT change the Core UI password, it will be the default password in the original instance (instance-id) for all the new instances that uses the custom AMI.

Autoscaling EC2

This is the guideline an autoscaling deployment of MetaDefender. In this example, besides the MetaDefender EC2 instance, additional resources are being generated and set up.

Prerequisites and Requirements

AWS knowledge: This guide assumes familiarity with AWS services.

AWS account: needs permission to create AWS resources listed in Autoscaling EC2

Tools to be used: Terraform

System/OS requirements

For OS, database type, 3rd party dependencies and storage requirements on Windows installations, please refer to the *Microsoft Windows Deployments* section of the Recommended System Configuration page.

For OS, database type, 3rd party dependencies and storage requirements on Linux installations, please refer to the *Linux Deployments* section of the Recommended System Configuration page.

Deploy using Terraform

- Locally clone the metadefender-csp repository and go to AWS/autoscaling-group

Code · bash

```
git clone git@github.com:OPSWAT/metadefender-csp.git
cd AWS/autoscaling-group
```

- Modify terraform.tfvars with the desired options

terraform.tfvars · bash

```
# General variables

MD_ENV_NAME             = "metadefender"
MD_REGION               = "eu-central-1"
#ACCESS_KEY_ID          = "<ACCESS_KEY_ID>"
#SECRET_ACCESS_KEY      = "<SECRET_ACCESS_KEY>"
MD_VPC_CIDR             = "192.168.0.0/16"
PUBLIC_ENVIRONMENT      = true
WARM_POOL_ENABLED       = true

# MetaDefender Core variables

DEPLOY_CORE             = "false"
CORE_PRODUCT_ID         = "ani6v4vb5z4t87cymrfg3m451" # For Windows it is
```

```
"9s8powksm1cj7fuafdnv0sfj9"
CORE_INSTANCE_TYPE      = "c5.2xlarge"
LICENSE_KEY_CORE        = ""
CORE_PWD                = "admin"
CORE_PORT               = 8008
```

- Run terraform apply and check the resource to be create, after that enter "y"

Code · bash

```
terraform apply
```

Monitoring in AWS

Health Check

There are multiple factors that can result in failure of the system, some of them are AWS related (AZ fault, hardware fault) or application fault. Depending on the use case, each customer defines differently what application failed is.

Success

Starting with version 5.2.0, Get health check status helps fetch the current health check status of the MetaDefender Core server.

Health Check for EC2 instance deployment

For generic hardware or AZ fault, we recommend deploying MetaDefender Core in a distributed environment as defined in AMI - Distributed MetaDefender Deployment with Autoscaling. By deploying MetaDefender in different Availability Zones with a load balancer in front of them, you will always be sure that no hardware fault will result in service interruption.

Health Check for EKS deployment

Deploying MetaDefender Core in a Kubernetes cluster offer a high availability within the cluster thanks to the configured health check at the container level that use the "/readyz" endpoint and the auto-healing feature provide by Kubernetes

How to use the health check

1. If you have an ELB in front of MetaDefender Core, configure it to do the health checks using the engines status.

- How to setup ELB to do health checks: https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-healthchecks.html
- For health check you should use the GET Health Check API
  - If you've used GET Engine Status, we'd recommend to move to the new dedicated Health Check API, since its

validating the entire system's health, not just the engine's health.

2. In case you are not using an ELB or you would like to have a more advanced health check, our recommendation would be to setup a Lambda function to check on each MetaDefender instance. In case of the health check monitored via Lambda, you will have the flexibility to actually adapt based on the REST API response provided by MetaDefender. One of the following should apply:
   - If all engines are healthy, the instance is considered healthy
   - If some of the engines are not up to date or failing, based on the internal policy you might invalidate this MetaDefender instance.
   - If the REST API response is an error, the instance should be considered down
3. An even more advanced check can be considered actually submitting a file to be analyzed using a Lambda function. Note that depending on the file size and complexity, workflow configurations and the number of files in the queue, it might result in timing out the Lambda execution.
   - Submit always the same file to MetaDefender through the REST API
   - Compare the response with a baseline

Licensing

To check the status of your MetaDefender Core license, please review the *Check Your License__Details* section on License Activation page. In case of monitoring the license information there is available an API endpoint to get the license information

Troubleshooting and Maintenance

For troubleshooting MetaDefender Core with Kubernetes deployment , please refer to the following guides:

1. Troubleshooting K8S (installation, logging, cluster management tools)

For troubleshooting MetaDefender Core with EC2 deployment , please refer to the following guides:

1. Troubleshooting MetaDefender Core (installation, logging, node management)
2. Scripted license management (Licensing status and automation)
3. Performance Scaling (for large deployments)
4. AWS Database Troubleshooting (AWS Docs for RDS)

Regular Instance Maintenance

Please see this page for more information on Regular Maintenance of the MetaDefender Core services and components. For database maintenance, please refer to Database Defragmentation and Optimization.

Routine Maintenance

Always update the MetaDefender deployment with the new versions published either on:

- portal.opswat.com > Products section
- AWS Marketplace

Release Notes are available here.

MetaDefender provides a direct update mechanism for the licensed analysis engines (all the engines listed in Inventory > Technology). The application logic will require a product update, however the licensed engines are automatically updated (in online deployments). Both the engine and the signature updates for Anti-malware engines will be automatically downloaded and deployed on a daily basis. We recommend to configure the product to check for updates at least every 4h (for more details see Engines Update Configuration).

In general MetaDefender instance has Internet connection. If an offline deployment is considered, make sure you are using either the Central Management or the Update Downloader and that you are uploading the engines' signatures updates minimum once a day.

If you are deploying MetaDefender in your own AWS environment, the AWS service limits will not impact the product. A potential issue that can be caused by AWS service is using a lower-tier EC2 type than is recommended. Please see the Costs section above for details on which EC2 instance types we recommend for MetaDefender. Network bandwidth and EBS/General SSD IOPS can become a limitation in high-throughput environments. The solution to this is to simply use higher tier machines such as ones with 20+Gbps network bandwidth and moving to a provisioned IOPS (io2) disk(s).

MetaDefender supports the ability to rotate system credential and cryptographic keys to ensure the security of your deployment. Information for performing this rotation can be found on the Database Management page. MetaDefender users can also change user credentials via our POST - Change Password API.

Emergency Maintenance

In case of any availability failures of an AWS Service, recommended actions would be to pass the load to another MetaDefender instances, preferably deployed in a different AZ than the one affected. Any analysis in progress submitted to MetaDefender will be considered lost. However, is highly recommended to build a resilient system that won't rely on MetaDefender to recover the pending jobs, but to be managed by a queueing mechanism (e.g. AWS SQS). See Backup and Recovery suggested deployment scenarios.

Customize OPSWAT MetaDefender AMI from Marketplace

> **Warning**
>
> Please follow these instructions before generating a custom AMI from the OPSWAT AMI published in the AWS marketplace.

- Default username and password is `admin`/`<instance-ID>`
  - “admin” is an unacceptable password. Whenever the end-user changes the password to “admin”, it will automatically change the password to `<instance-ID>` after the Instance is rebooted.
- To create custom AMI from OPSWAT AMI, some mandatory things need to be noted before converting:
  1. ***(Required)*** Deactivate the MetaDefender Core license in the EC2 instance from where the new AMI will be

created.

2. ***(Required)*** Not to add any parameters during the creation process.
3. ***(Optional)*** Change MD Core password.
   - If the end-user changes the password to “admin”, their default password after creating a custom AMI will be `<instance-ID>`
   - If the end-user does NOT change the Core UI password, it will be the default password in the original instance (instance-id) for all the new instances that uses the custom AMI.

Recommended Architectures

These pages explain what are the architecture recommended for running MetaDefender Core in an Azure VM or Azure AKS (Kubernetes)

Infrastructure Options for MetaDefender Core in Azure VM

The possible infrastructure vary depending on what are the number of VMs where MetaDefender Core will be installed

Options:

1. Single Azure VM

Infrastructure Options for MetaDefender Core in Azure AKS

The possible infrastructure options in Azure AKS vary depending on how the PostgreSQL database is provisioned.

Options:

1. Azure Vm + Azure PostgreSQL
2. Azure Vm + PostgreSQL Pod

Above options are explained at AKS Cluster Architecture

We offer a provisioning script that depending on the selection, we automatically provision the cluster for you and install the K8S components needed for MD Core service. See AKS Cluster Deployment

AKS Cluster

The possible infrastructure vary depending on how the PostgreSQL database is provisioned

Options:

1. Azure AKS with VMs + Azure PostgreSQL
2. Azure AKS with VMs + PostgreSQL Pod

We offer a provisioning script that depending on the selection, we automatically provision the cluster for you and

install the K8S components needed for MD Core service. See AKS Cluster Deployment

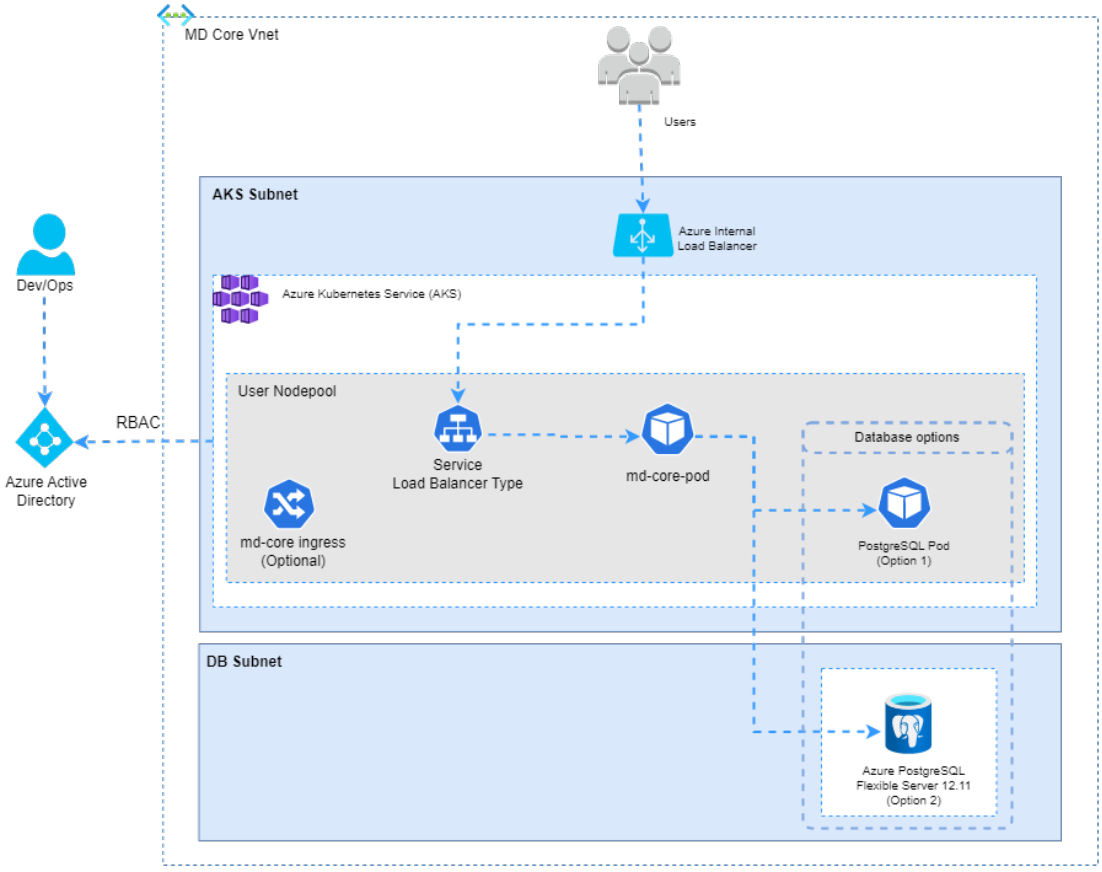
MetaDefender PostgreSQL database details

Recommendations

For running MD Core in a Kubernetes cluster we recommend to use, for production environments, an external database service for PostgreSQL. In case of **Azure** it is **Azure PostgreSQL** which the script will deploy and configure to be accessible from the cluster.

For non-production environments or for stateless use cases, we offer the option of installing the database within the cluster using a standard PostgreSQL image.

Azure VMs + Azure PostgreSQL / PostgreSQL Pod Diagram



Resources Inventory

Networking

Service Type	Resource Name
Azure Virtual Network	<resource-group-name>-vnet
Azure Subnet	<resource-group-name>-aks-subnet

Service Type	Resource Name
Azure Subnet	<resource-group-name>-db-subnet
Azure Private DNS Zone	<postgres_db_account_name>.postgres.database.azure.c
Azure Private DNS Zone Virtual Network Link	<postgres__db__account_name>.com

Monitoring

Service Type	Resource Name	Description
Azure Log Analytics WorkSpace	<log_analytics_workspace_name>	Workspace for analyzing the AKS Service
Azure Log Analytics Solution	ContainerInsights	Solution for gathering metrics of containers

Compute

Service Type	Resource Name	Description	Option
Azure AKS	<cluster_name>	Azure Kubernetes Service Cluster	All
Azure ProstgreSQL Flexible Server	postgresql-<postgres_db_account_name>	PostgreSQL Server for MetaDefender Core	Option with Azur
Azure ProstgreSQL Flexible Server Configuration	azure.extensions	Configuration for installing PG_TRGM,DBLINK,BTREE_GIN_	Option with Azur

Info

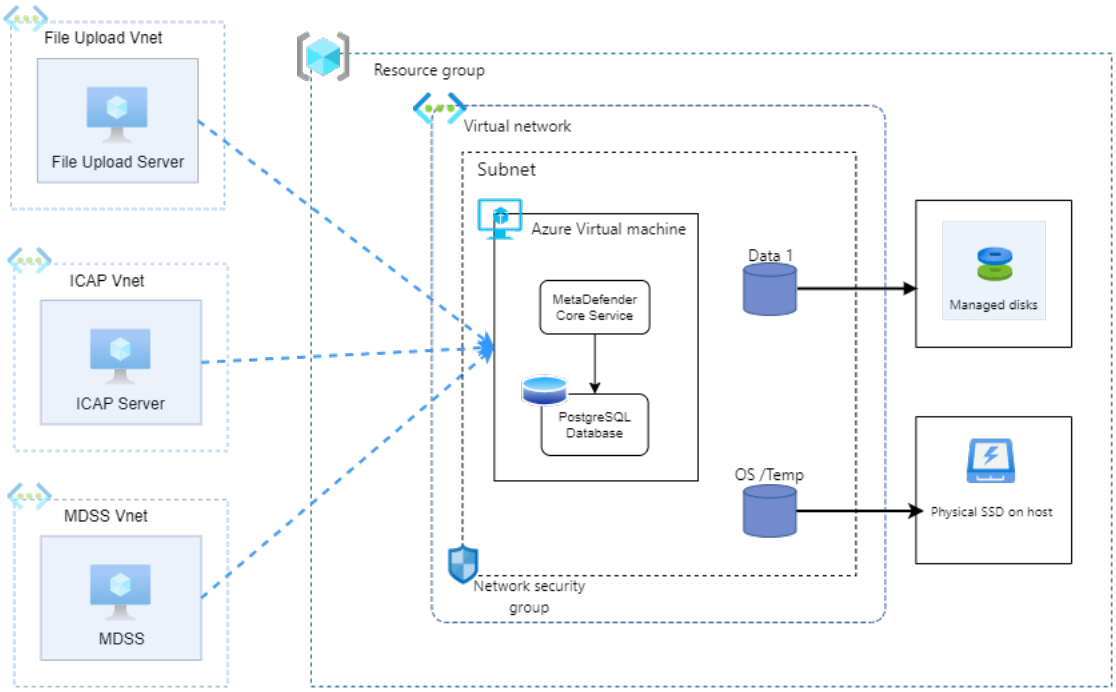
The script will deploy a single Worker Node for the cluster with enough space for 1 replica of MD Core. Azure VM size is **Standard_F8s_v2 (8 vCPU & 16 GiB Memory).** Each pod would need a minimum of 4 vCPU and 8 GiB Memory. To change the request to adapt each pod to the specific case go to values.yml To change the size of the node pool go to terraform file terraform/azure/main.tf

Single Azure VM

The single instance deployment is only recommended for small or test environments that do not need high

availability. This deployment consist of a single VM instance using the MetaDefender Core image published in the Azure marketplace that includes a Standalone MetaDefender Core where the PostgreSQL database is deployed in the same instance than MetaDefender Core.

Diagram



Resources Inventory

Resource Management

Service Type	Resource Name	Description
Resource Group	metadefender-rg	Azure Resource group for grouping all the resources needed for MetaDefender Core

Networking

Service Type	Resource Name	Description
Security Group	MetaDefenderSecurityGroup	Generated security group to allow traffic to MetaDefender REST API.
Vnet	metadefender-core-vnet	Desired Virtual Private Cloud

Service Type	Resource Name	Description
SubNet	metadefender-core-subnet	

Compute

Service Type	Resource Name	Description
Azure VM	metadefender-core-<os>	Azure VM Instance to run Windows MetaDefender Core AMI

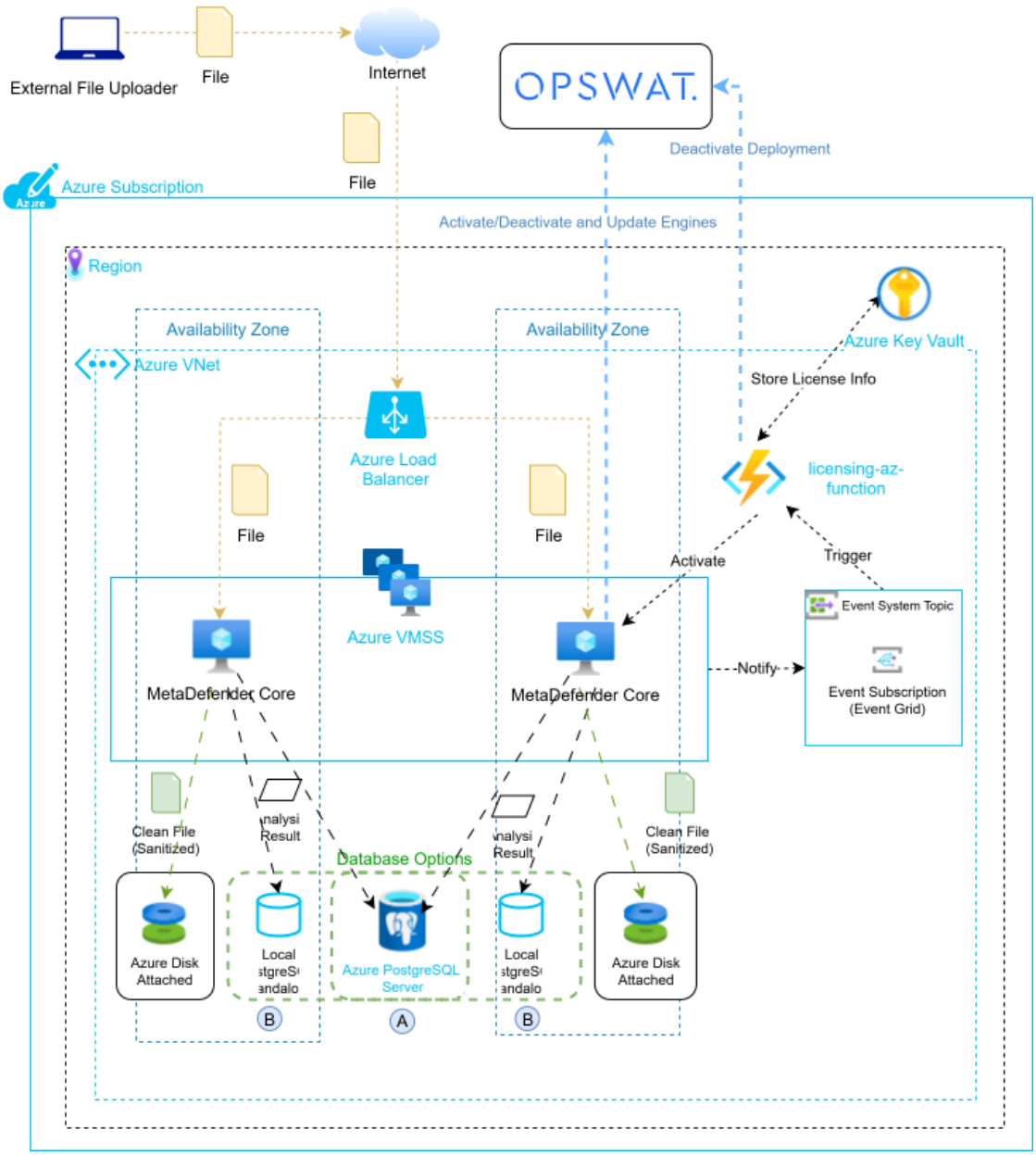
Storage

Service Type	Resource Name	Description
Azure Managed Disk	metadefender-core-data1-disk	Azure Disk for use cases where is needed additional storage

VM Scale Set

The multi instance deployment is recommended for medium or large environments that need high availability. This deployment consist of a VMSS with as many virtual machines as needed depending on the number of files to scan.

Diagram



Resources Inventory

Resource Management

Service Type	Resource Name	Description
Resource Group	metadefender-rg	Azure Resource group for grouping all the resources needed for MetaDefender Core

Networking

Service Type	Resource Name	Description
Security Group	MetaDefenderSecurityGroup	Generated security group to allow traffic to MetaDefender REST API.
Vnet	metadefender-core-vnet	Desired Virtual Private Cloud
SubNet	metadefender-core-subnet	

Compute

Service Type	Resource Name	Description
Azure VMSS	metadefender-core-<vmss>	Azure VMSS Instance to run multiple instances

Storage

Service Type	Resource Name	Description
Azure Managed Disk	metadefender-core-data1-disk	Azure Disk for use cases where is needed additional storage

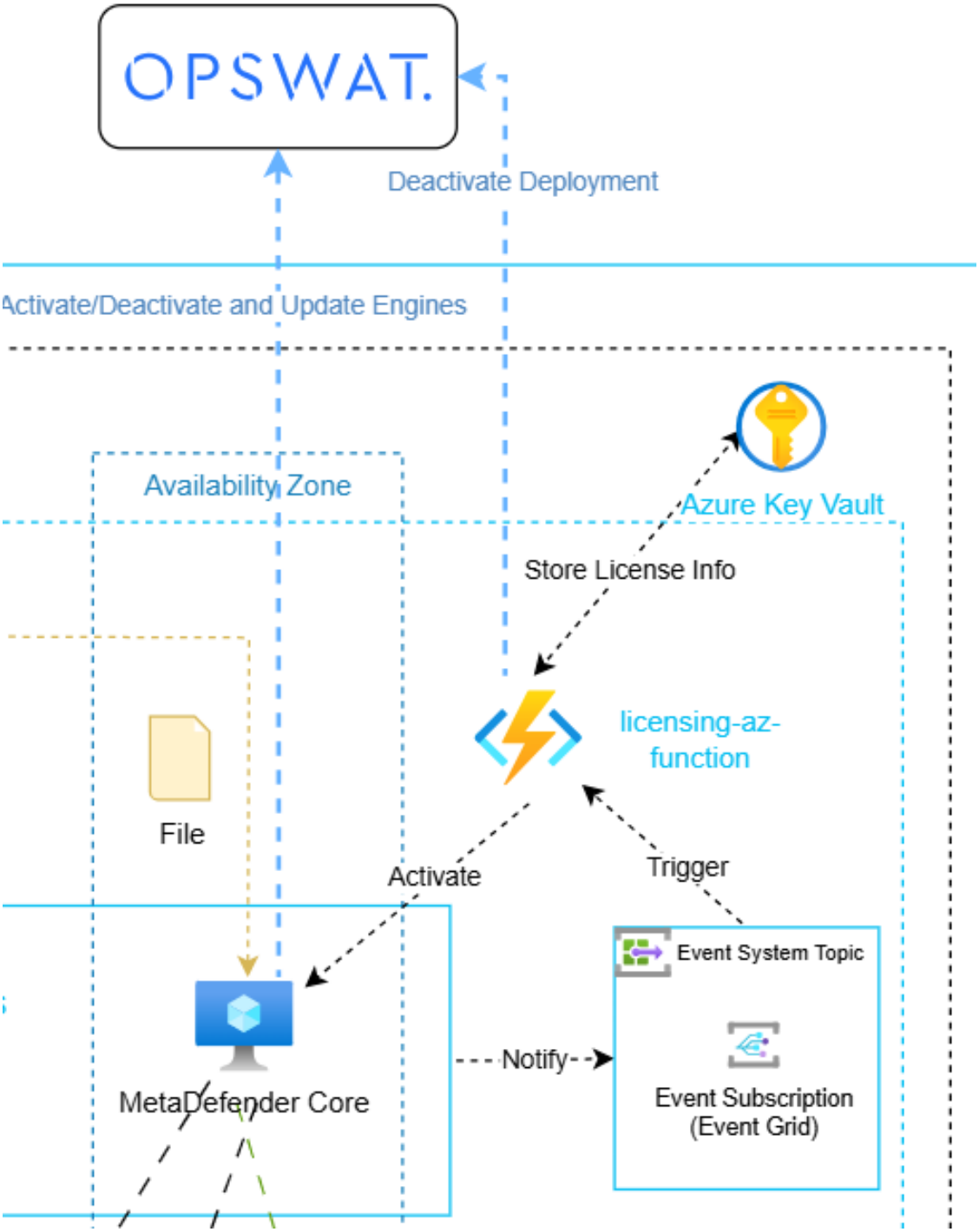
Licensing Automation on Azure

To activate the instances where the MetaDefender products are running, it is needed to activate these products providing the LICENSE_KEY.

There are 2 options to manage the licensing that will automate the activation and deactivation of the products.

Azure Function to handle the VMSS events

Diagram



Info

Solution recommended for production as the activation and deactivation does not depend if the VM is gracefully terminated or not

Pre-requisites

This sections presumed that a VMSS has been created with the Azure VM image generated based on the database

mode desired. If not please go to VM Scale Set

Resources Needed

- Azure Function App
- Azure Key Vault
- Azure Event System Topic
- Azure Event Subscription
- Azure Function to run within the Azure Function App

> **Info**
> This guidelines are for creating the resources from Azure Portal. To use Terraform please check

Azure Function

Check the Deployment Options for Azure Functions

This function has 2 different tasks.

- **Activation**: Retrieve the LICENSE_KEY and APIKEY from Azure Secret and activate the VM. Store the deployment ID to Azure Key Vault using VM as key
- **Deactivation**: Retrieve Deployment ID from Azure Secret and deactivate the deployment.

Steps to create the Azure Function

1. Create the Azure Function App (Recommended Flex Consumption Plan)
2. Create Azure Key Vault with the following required keys
    1. License Key (licenseKey)
    2. API Key (apiKey)
    3. Core UI User (coreUser)
    4. Core UI Password (corePwd)
3. Create an Access Policy in the Key Vault for the Azure Function to "Get", "Set", "List", "Delete", "Purge" and "Recover"
4. Set Up as environment variables for the Azure Function App
    1. SUBSCRIPTION_ID
    2. KEY_VAULT_NAME
5. Deploy the Event Licensing Handler Function

Trigger the Azure Function

This Azure Function has to be triggered by an Azure Event Grid System Topic and a Event Grid Subscription to link the topic and trigger the Azure Function based on events that will be specified in the subscription.

Create Event Grid System Topic

This topic will generate events for the resource group linked to it.

How to create it from Azure portal

1. Go to Event Grid > Azure Service Event > System Topics
2. Create Topic > Select Azure Subscription or Resource Group
3. Enable system assigned identity and input all the tags

Create Event Grid Subscription to System Topic and Azure Function

The subscription will link the Azure Event Grid Topic to the Azure Function already created.

How to create the Event Grid Subscription

1. Go to Event Topic and create Subscription from within it
2. Indicate name and Event Grid Schema
3. Under Event Types select only Resource Action Success
4. Configure the Endpoint to be the Azure function created in previous section
5. Under filters Enable subject filtering
    1. Subject Begins With: "/subscriptions/{subscription_id}/resourceGroups/{RG_NAME}/providers/Microsoft.Compute/virtualMachines"
6. Under Advanced Filters
    1. Key: data.operationName
    2. Values:
        1. "Microsoft.Compute/virtualMachines/start/action"
        2. "Microsoft.Compute/virtualMachines/deallocate/action"
        3. "Microsoft.Compute/virtualMachines/powerOff/action"

Setting Up Environment Variables using user-data (Only POC)

> **Info**
>
> Only works with OPSWAT Marketplace Images that are prepared to handle user-data

This option is the default and recommended mechanism to automate the licensing management when using the Azure Marketplace Image, as this is the easiest way to activate and deactivate the product when running a single VM

of MetaDefender Core

When passing as environment variables to the user-data the LICENSE_KEY it will set up the automation that will activate the product when the instance is started and will deactivate the product when the instance is terminated or stopped.

It is also recommended to indicate the APIKEY environment variables as this will set up the APIKEY to the admin user to be able to make requests to the MetaDefender Core API. In vm scale set deployment types it will help to manage all the instances with the same APIKEY when hitting the Load Balancer.

Licensing when running in Kubernetes

The licensing automation options when the MetaDefender products are installed in a Kubernetes cluster, are explained in Licensing in K8S

Deploy using Azure services

Introduction

This Deployment Guide provides step-by-step instructions for deploying MetaDefender Core on Azure Cloud Services infrastructure.

Organizations interested in protecting their solutions deployed in Azure can leverage MetaDefender to analyze and sanitize files residing, or transitioning, their Azure deployment. MetaDefender can scan and either sanitize or check for known vulnerabilities, depending on the type of traffic it's seeing. The ideal use case would be an organization that allows files to be uploaded to Azure through an external facing web portal. Analyzing files before they are made accessible to the end-users is critical to ensure that no malicious content is allowed and distributed through the web application.

Advanced attacks are concealing the malicious payload and are relying on productivity files (documents, pdfs, images) as a distribution mechanism. Productivity files allow active content to be leveraged, but these features are frequently exploited to execute the malicious behavior.

This guide is for IT infrastructure architects, administrators and DevOps professionals who are seeking to prevent potential malicious traffic being allowed in their Axure Cloud deployment. Threat Prevention is ensured for both productivity files that might be uploaded and for known vulnerabilities that can be identified for all running services/ applications deployed in Azure. The vulnerability scanning is checking known vulnerabilities for unpatched OS and running applications.

MetaDefender is provided as:

- AMI through the Azure Marketplace (MetaDefender products)
- Packaged installer available for download through the OPSWAT Portal.
- Container Image available in the OPSWAT Docker Hub Repository

There are currently no restrictions on Azure regions in which you can deploy MetaDefender. The only restriction is that MetaDefender is not available Azure for Government at this time.

OPSWAT Contact Information

Sales:sales-inquiry@opswat.com

Support:https://www.opswat.com/support

Other:https://www.opswat.com/contact

For deploying our solution in Azure, please review the guideline listed below:

- AKS Cluster

Database Encryption

MetaDefender Core supports both native OS encryption or Azure disk encrypted volumes. The database is secured via credentials and can be configured to encrypt everything at rest. In general, OPSWAT recommends the use of a 3rd party database layer, e.g. Azure PostgreSQL, which should be encrypted. It is best practice to not share the database layer on the same hosts/Azure VM

Sensitive Data Storage

Network traffic processed by MetaDefender Core is SSL encrypted during transmission. The product does not collect or store any customer data outside of analyzing the contents of files submitted for scanning.

During the scanning process, files are stored unencrypted on disk to allow the scan process to complete. Once the scan completes, the file is deleted unless configured to place the file in quarantine. If you wish to encrypt these quarantined files, please use native OS encryption or Azure Disk encrypted volumes.

Private Key Management

The Private key pair is only used to SSH or RDP to the Azure VM. This is a very uncommon scenario, used primarily for debugging. You can create a new key pair, or use an existing pair.

Costs

MetaDefender is made available as an annual subscription. The only mandatory service that needs to be purchase is MetaDefender Core. MetaDefender Core is a paid service and is currently licensed on the BYOL (Bring Your Own License) model. Contact our sales team via our Contact form, available here: https://www.opswat.com/contact

Considering that are over 60 different options to license MetaDefender, it is highly coupled to the use case and the advanced features that you are considering deploying. Note that the more functionality you are adding to MetaDefender, the more need will be of CPU from the EC2 instance.

Our recommendation would be a minimum 8 vCPU for our lower tiers.

We recommend to go with Reserved Instances, considering that you are committing to an annual subscription for MetaDefender.

For pricing per vm, please refer to the official Azure pricelist

Sizing

MetaDefender needs an Azure vm with minimum 8 vCPU, in order to have an optimal response rate to submitted files for analysis. However, depending on the use case and expected throughput (analysis SLA), higher tier instances are recommended.

The system requirements (hardware and supported operating system) are defined here: Recommended System Configuration

Operation Guidance for Azure VM deployments

Troubleshooting

For troubleshooting MetaDefender Core with Azure deployment , please refer to the following guides:

1. Troubleshooting MetaDefender Core (installation, logging, node management)
2. Scripted license management (Licensing status and automation)
3. Performance Scaling (for large deployments)
4. Engine and Database health (view Health Check section under different deployment option)

Routine Maintenance

Always update the MetaDefender deployment with the new versions published either on:

- portal.opswat.com > Products section
- Azure Marketplace

Release Notes are available here.

MetaDefender provides a direct update mechanism for the licensed analysis engines (all the engines listed in Inventory > Technology). The application logic will require a product update, however the licensed engines are automatically updated (in online deployments). Both the engine and the signature updates for Anti-malware engines will be automatically downloaded and deployed on a daily basis. We recommend to configure the product to check for updates at least every 4h (for more details see Engines Update Configuration).

In general MetaDefender instance has Internet connection. If an offline deployment is considered, make sure you are using either the Central Management or the Update Downloader and that you are uploading the engines' signatures updates minimum once a day.

If you are deploying MetaDefender in your own AWS environment, the AWS service limits will not impact the product. A potential issue that can be caused by AWS service is using a lower-tier EC2 type than is recommended. Please see the Costs section above for details on which EC2 instance types we recommend for MetaDefender. Network bandwidth and EBS/General SSD IOPS can become a limitation in high-throughput environments. The solution to this is to simply use higher tier machines such as ones with 20+Gbps network bandwidth and moving to a provisioned IOPS (io2) disk(s).

MetaDefender supports the ability to rotate system credential and cryptographic keys to ensure the security of your deployment. Information for performing this rotation can be found on the Database Management page. MetaDefender users can also change user credentials via our POST - Change Password API.

Emergency Maintenance

In case of any availability failures of an AWS Service, recommended actions would be to pass the load to another MetaDefender instances, preferably deployed in a different AZ than the one affected. Any analysis in progress submitted to MetaDefender will be considered lost. However, is highly recommended to build a resilient system that won't rely on MetaDefender to recover the pending jobs, but to be managed by a queueing mechanism. See Backup and Recovery suggested deployment scenarios.

Support

To receive support, please visit our Support Portal. From there after you log in click on "Cases" on menu at the top of the page and then click "New" on the right hand side to create a new case.

Support policies, costs, levels and SLA's are described on our website, at the Support section: https://www.opswat.com/support. Please view our support tiers in the table below:

Note: Please visithttps://www.opswat.com/supportfor the most updated information on support.

	Standard	Gold	Platinum
Prices	Included with your license	20% ($500.00 minimum) *	25% ($2000.00 minimum) *
*_Support Hours *_			
Support via Portal Ticket System	8AM to 5PM business days (local to you)	7AM to 7PM business days (local to you)	24 x 7 x 365
Support via Chat	8AM to 5PM business days (local to you)	7AM to 7PM business days (local to you)	24 x 7 x 365
Support via Phone			24 x 7 x 365
Incident Response Times			
Blocker Issues First Response Times	Within 1 business day	Within 2 hours	Within 1 hour
Non-blocker Issues First Response Times	Within 3 business days	Within 2 business days	Within 1 business day
Support Account Management			Yes
Support Account Manager			Yes

	Standard	Gold	Platinum
Quarterly Conference Call Reviews			Yes
Yearly Roadmap Review with the Product Team			Yes
Quarterly Summary Reports		Yes	Yes

Info

The following sections are for Azure deployments. When installing MetaDefender Core in Kubernetes check Installing MetaDefender Core in K8S

Reference Materials

All MetaDefender documentation is available on docs.opswat.com/mdcore

Localization

MetaDefender products and documentation are available exclusively in English for now.

AKS Cluster

This guide explains how to use terraform or the provisioning script provided by OPSWAT, to create an Azure AKS and generate all the Kubernetes components needed to run MetaDefender Core.

Deploy using Terraform

- Locally clone the metadefender-k8s repository and go to terraform folder

Code · bash

```
git clone git@github.com:OPSWAT/metadefender-k8s.git
cd terraform/azure/
```

- Modify variables/variables.tfvars file
  - PERSISTENT_DEPLOYMENT=true -> To create K8S cluster with EC2 Worker Nodes
  - DEPLOY_FARGATE_NODES=false -> To true if wanted to create a Fargate profile
  - DEPLOY_RDS_POSTGRES_DB=true -> To create RDS instance for PostgreSQL server

- POSTGRES_USERNAME -> if DEPLOY_RDS_POSTGRES_DB=true, set up admin database username
- POSTGRES_PASSWORD -> if DEPLOY_RDS_POSTGRES_DB=true, set up admin database password`

Code · bash

```
aks_service_principal_app_id = "" #OPTIONAL, IT CAN BE SET UP WITH ENV VARIABLES
aks_service_principal_client_secret = ""
aks_service_principal_object_id = ""

postgres_admin = "coreadmin"
postgres_password = "CoreAWS1$"
deploy_cosmos_db = "false"
deploy_postgres_db = "false"
cosmos_db_account_name = "111"
```

- Run terraform init and apply. Check the resource to be created, after that enter "y"

Code · bash

```
terraform init
terraform apply

# IF Variables are not setup you can use this command
terraform apply \
  -var="aks_service_principal_app_id=$ARM_CLIENT_ID" \
  -var="aks_service_principal_client_secret=$ARM_CLIENT_SECRET"
```

MetaDefenderK8S Script Details

- **GitHub Project** metadefender-k8s --> **Script path**: ./metadefenderk8s.sh
- **Programming Language:** Bash
- Installation **Pre-requisites** for provisioning:
  - Terraform
  - Helm
  - AZ CLI
  - Kubectl
- **Azure Credentials**
  - Set in your local environment variables the credentials of the IAM user under
    - **ARM_CLIENT_ID**
    - **ARM_CLIENT_SECRET**

  - ARM_SUBSCRIPTION_ID
  - ARM_TENANT_ID
- ssh public key to used to access the cluster with kubectl. Default (~/.ssh/id_rsa.pub). Change from here
- az login before running the script for the step of downloading the kubeconfig file for you

- MetaDefender **Core License Key** (Required with --mdcore parameter)
  - Set it in your local environment variables the credentials under ***MDCORE_LICENSE_KEY***

How to run the script

Only MD Core · bash

```
./metadefenderk8s.sh provision -l Azure --mdcore
```

MD Core + MDSS · bash

```
./metadefenderk8s.sh provision -l Azure --mdcore --mdss
```

MD Core + ICAP · bash

```
./metadefenderk8s.sh provision -l Azure --mdcore --icap
```

Info

The script will deploy a single Worker Node for the cluster with enough space for 1 replica of MD Core. Azure VM size is **Standard_F8s_v2 (8 vCPU & 16 GiB Memory).** Each pod would need a minimum of 4 vCPU and 8 GiB Memory. To change the request to adapt each pod to the specific case go to values.yml To change the size of the node pool for having more MD Core replicas or install additional MetaDefender products go to terraform file terraform/azure/main.tf

Script Parameters

Parameter	Flags	Options	Default	Description	Required/ Optional
Action		• provision • install		Action to indicate the script if we want to provision	**Required**

Parameter	Flags	Options	Default	Description	Required/ Optional
				(Create resources + install Core) or install (Install Core)	
Location	-l or --location	• AWS • Azure • GCP		Where is going to be the K8S cluster	**Required**
MetaDefender Flag Installation	Combination of • --mdcore • --mdss • --icap		•	Install MetaDefender Core in the cluster provisioned	**Required**
Image Version	--image	latest 5.0.1	latest	MetaDefender Core image version to install	Optional
Region	--region	Azure Regions	centralus	AWS region where all the resources will be provisioned	Optional
Cluster Name	--name		md-k8s	Name of the cluster that will be used for naming all the resources	Optional
Number of Replicas	--replicas	[0-9]*	1	Number of replicas for MetaDefender Core service	Optional
Namespace	--namespace	[A-Za-z]{1,10}		Namespace where	Optional Max Characters: 10

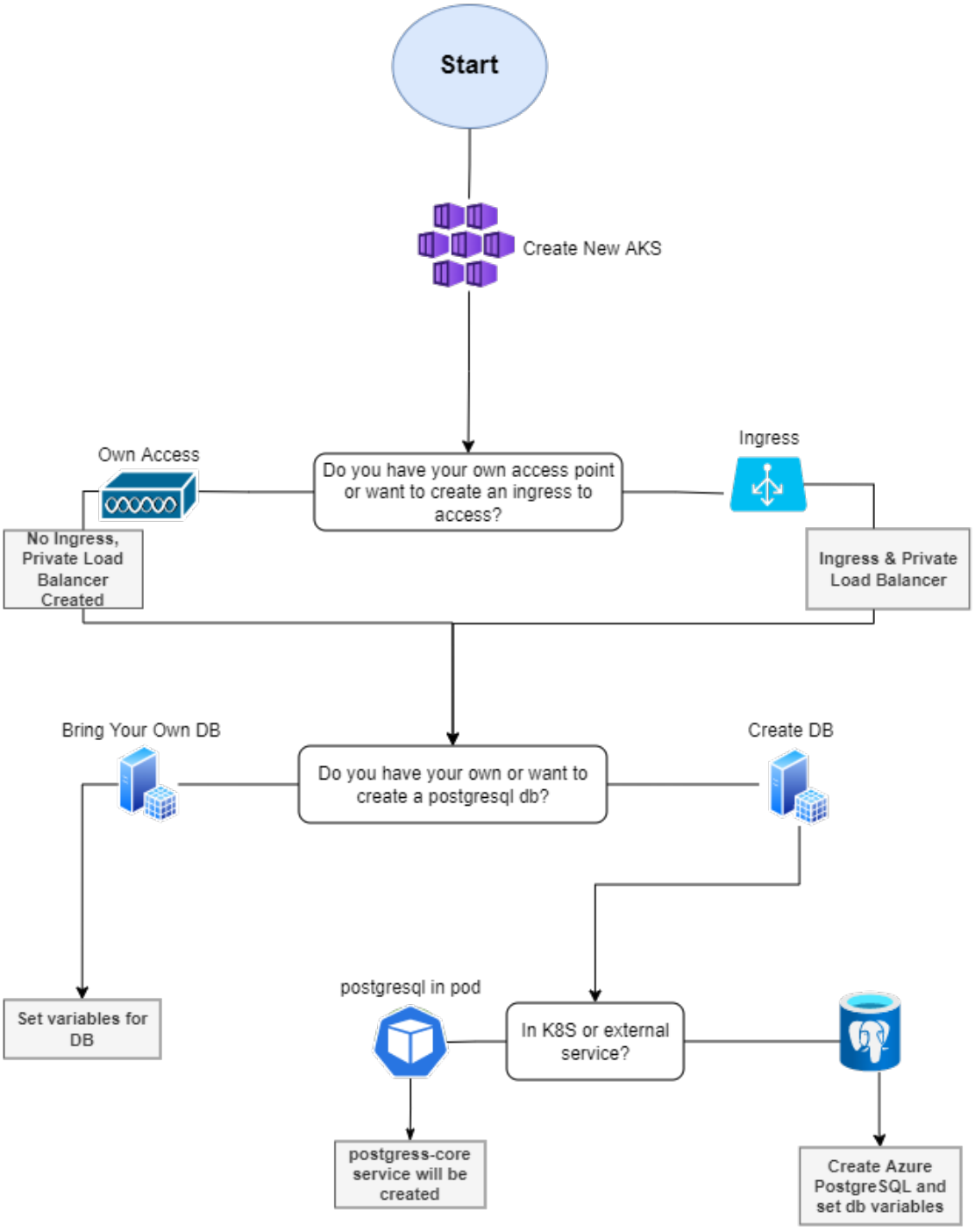
Parameter	Flags	Options	Default	Description	Required/ Optional
				MetaDefender products will be installed in the K8S Cluster	

MetaDefender Core Flowchart Provisioning in Azure

The following flowchart represents how the provisioning script will configure the environment based on the options selected for provisioning Azure AKS.

Summary options to be selected

1. Access to the K8S cluster. Generate **Ingress** or provide **own** access.
    1. **An Ingress** and an internal load balancer will be created per each product flag added as parameter to the script
    2. **Own Access**, you decide how to access to the cluster so it won't generate any ingress but will still create the internal load balancer for the product service deployed
2. Have your **own database** or create **new database**
    1. **Own database,** will be asked if you want the script to set up the credentials and database host url for you or the script will just indicate the secrets to edit, later on by you, for connecting the MetaDefender Core with your database.
    2. Create new **DB in K8S** or **external DB** that for Azure we will provision a **Azure *PostgreSQL Flexible Server***.



Single Azure VM

For a single deployment of MetaDefender Core in Azure you can use this example to test this deployment type, besides the MetaDefender EC2 instance, additional resources are being generated and set up.

Prerequisites and Requirements

Azure knowledge: This guide assumes familiarity with Azure Services

Azure account: needs permission to create Azure Services listed in Recommended Architecture for Azure

Tools installation: Terraform

System/OS requirements

For OS, database type, 3rd party dependencies and storage requirements on Windows installations, please refer to the *Microsoft Windows Deployments* section of the Recommended System Configuration page.

For OS, database type, 3rd party dependencies and storage requirements on Linux installations, please refer to the *Linux Deployments* section of the Recommended System Configuration page.

Deploy using Terraform

- Locally clone the metadefender-csp repository and go to Azure/single-vm-deployment

Code · bash

```
git clone git@github.com:OPSWAT/metadefender-csp.git
cd Azure/single-vm-deployment
```

- Modify terraform.tfvars with the desired options
  - LICENSE_KEY_CORE required if wanted to have the instance activated automatically
  - APIKEY_GENERATION=true to have the apikey generated by terraform
  - There is one general section and one section for each product that is supported using this Terraform project in case of needing to install additional OPSWAT products

terraform.tfvars · bash

```
# General variables

RG_NAME                 = "metadefender" # Prefix to add to all the resources
MD_REGION               = "eastus" # Region for all the resources
MD_VNET_CIDR            = "192.168.0.0/16"  # VPC CIDR where to create the MetaDefender products
PUBLIC_ENVIRONMENT      = true
APIKEY_GENERATION       = true
IMPORT_RG               = false
VM_PWD                  = "<SET_UP_VM_PWD>"

# MetaDefender Core variables

DEPLOY_CORE             = true
CORE_INSTANCE_TYPE      = "Standard_D8s_v5"   # Instance type for MetaDefender Core
LICENSE_KEY_CORE        = ""
OFFER_PRODUCT_CORE      = "opswat-mdcore-linux"  # Windows opswat-mdcore-windows
SKU_CORE                = "opswat-mdcore-linux"  # Windows opswat-mdcore-windows
```

- Run terraform init and apply. Check the resource to be created, after that enter "y"

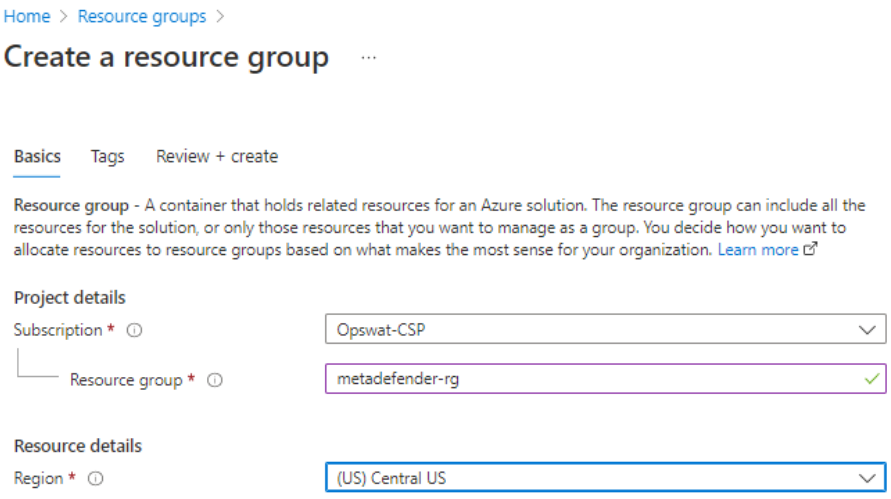
```
terraform init
terraform apply
```

Deploy using Azure portal

For having MetaDefender Core in a Single Azure VM as showed in Single Azure VM Architecture it is needed to follow the next steps

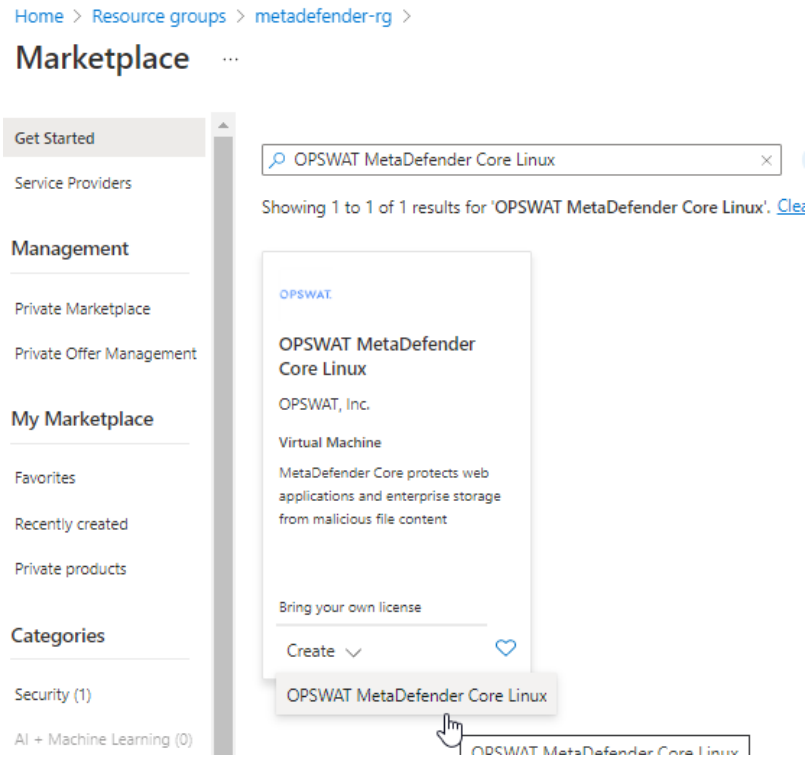
1. Create Resource Group

- Go to you Azure account and create a new resource group (we will refer to it as "metadefender-rg")

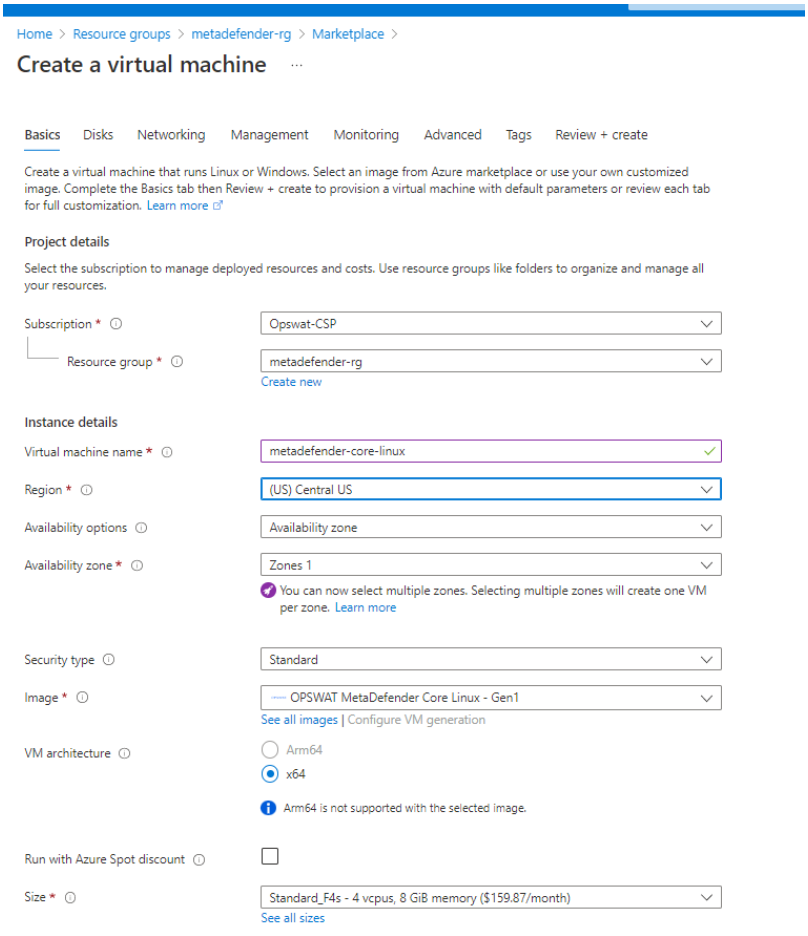


2. Create Azure VM

- Inside the resource group click on create resource and search for ***"OPSWAT MetaDefender Core Linux"***



- Input the vm name and select the size based on Recommended System Configuration.
- Under Administrator Account section select the desired way to access to the VM. For this guideline we will generate a public key to access to it



3. Storage Configuration

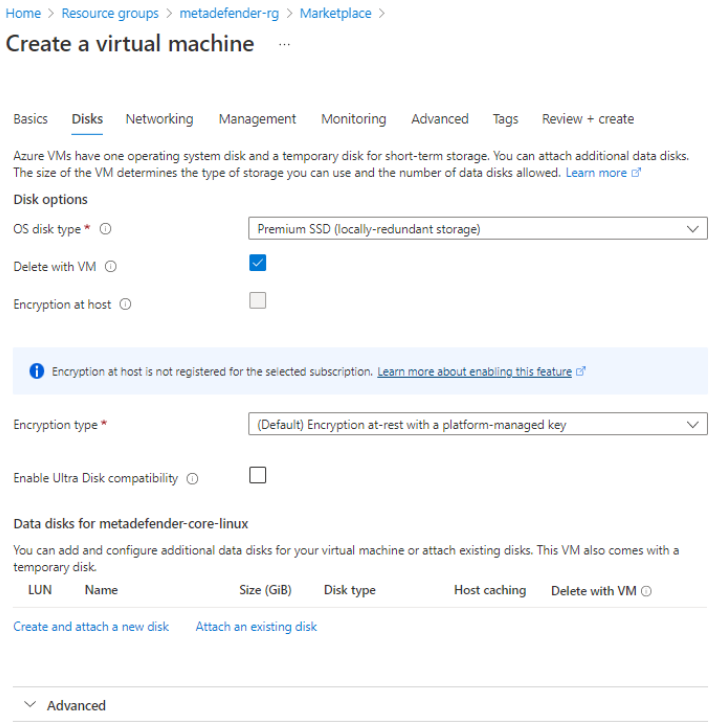
Storage step can be skipped.

In general there's no need for additional storage by MetaDefender. However, there are 2 situations where additional local storage might be required:

- MetaDefender will process large files or a high volume of files which submitted in MetaDefender's queue will need over 10GB files storage
- MetaDefender is configured to store files in the Quarantine section which will eventually fill the entire root volume.

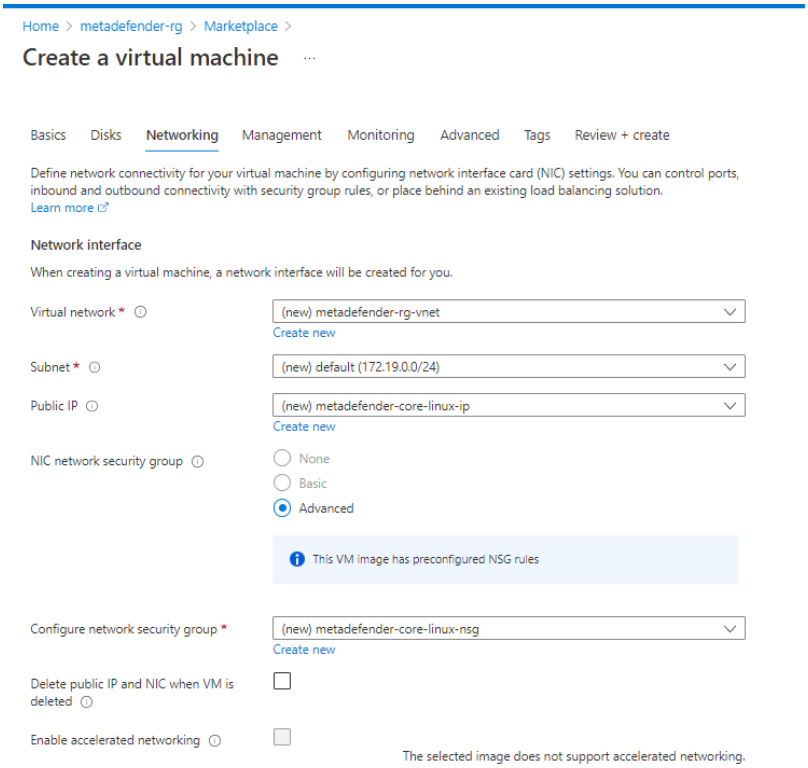
In case quarantining the files in the MetaDefender instance or analyzing high volumes of files is a requirement, please consider adding an additional Azure Managed Disk

- Premium SSD is recommended for having the best performance

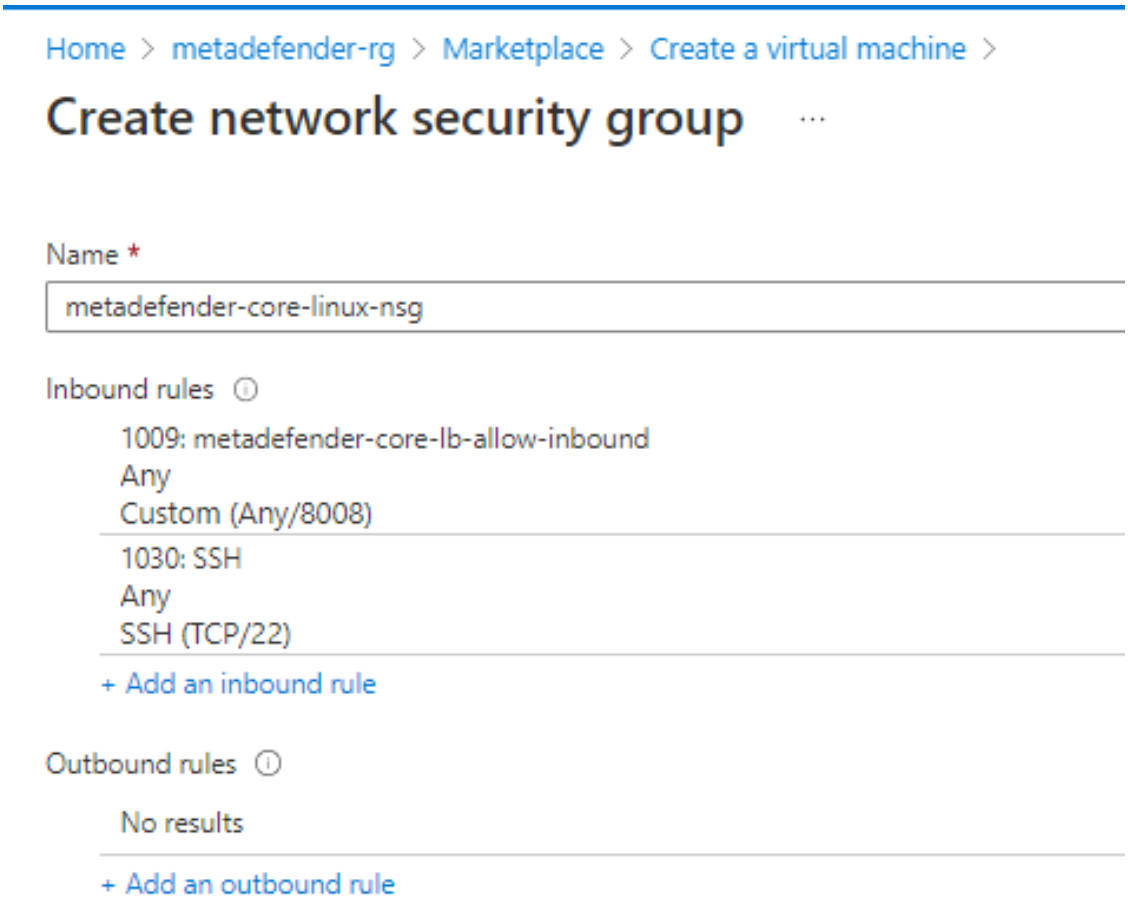


3. Network Configuration

- Change public IP to None in case you do not want to access from internet.
- Select the Virtual network and Subnet in case you already have them created, if not leave it to create them



- Create Security Group adding rules for 8008 and SSH



- Under Load Balancing section as it is a single vm it is not really needed but it can be used to set up a health check

using the /readyz endpoint

4. Review & Create the VM

- Under Management, Monitoring, Advanced and Tags there are not any configuration different from the default ones that are needed to be selected. So customize it as it is needed for each use case.
- After creating it will show you a window to download the private key to access to the VM in case that option was selected for accessing

Additional Configurations

For controlling the engine and database health OPSWAT offer an endpoint to use as health check, there are different ways of using it

VM Scale Set

Steps to follow for a Production Environment

When running MetaDefender Core is needed to generate a Azure VM image with the desired configuration as the image published in the Azure Marketplace is only covering one use case and is not recommended to be used for production.

1. Create Networking and Resource Group for MetaDefender
2. Create Azure PostgreSQL in case of wanting remote database (Optional but recommended)
3. Create Azure VM (Linux or Windows) and follow the installation process for the desired OS. See Installation
    1. Generate an APIKEY from the UI and save it for later.
    2. Generate a Azure VM image based on the current VM
4. Create a Azure VMSS
5. Create the Licensing Automation. Following Licensing Automation on Azure page

Running a POC Environment using Terraform example

For a multi instance deployment of MetaDefender Core in Azure you can use the terraform code provided by OPSWAT to test this deployment type.

> **Info**
> This terraform code is just an example to test the deployment type. Customers can reuse the code as needed.

Prerequisites and Requirements

Azure knowledge: This guide assumes familiarity with Azure Services

Azure account: needs permission to create Azure Services listed in Recommended Architecture for Azure

Tools installation: Terraform

System/OS requirements

For OS, database type, 3rd party dependencies and storage requirements on Windows installations, please refer to the *Microsoft Windows Deployments* section of the Recommended System Configuration page.

For OS, database type, 3rd party dependencies and storage requirements on Linux installations, please refer to the *Linux Deployments* section of the Recommended System Configuration page.

Deploy using Terraform

- Locally clone the metadefender-csp repository and go to Azure/vm-scale-set-deployment

Code · bash

```
git clone git@github.com:OPSWAT/metadefender-csp.git
cd Azure/vm-scale-set-deployment
```

Deploy activating the instances using Azure Function

- Modify terraform.tfvars with the desired options
  - (Required) VM_PWD required to access to the instances (ssh or RDP)
  - (Required) LICENSE_AUTOMATION_FUNCTION= true required if wanted to have the instance activated automatically
  - (Required) APIKEY_GENERATION=false only supported for User-data. The instance is activated with temporary session id
  - (Required) LICENSE_KEY_CORE required if wanted to have the instance activated automatically
  - There is one general section and one section for each product that is supported using this Terraform project in case of needing to install additional OPSWAT products

terraform.tfvars · bash

```
# General variables

RG_NAME                 = "opswatmd" # Prefix to add to all the resources
MD_REGION               = "eastus" # Region for all the resources
MD_VNET_CIDR            = "192.168.0.0/16"  # VPC CIDR where to create the MetaDefender products
PUBLIC_ENVIRONMENT      = true
APIKEY_GENERATION       = false
IMPORT_RG               = false
VM_PWD                  = ""
LICENSE_AUTOMATION_FUNCTION   = true
```

```
# MetaDefender Core variables

DEPLOY_CORE             = true
CORE_INSTANCE_TYPE      = "Standard_D8s_v5"   # Instance type for MetaDefender Core
LICENSE_KEY_CORE        = ""
APIKEY                  = ""
CORE_USER               = ""
CORE_PWD                = ""
OFFER_PRODUCT_CORE      = "opswat-mdcore-linux"  # Windows opswat-mdcore-windows
SKU_CORE                = "opswat-mdcore-linux"  # Windows opswat-mdcore-windows
NUMBER_INSTANCES_CORE   = 2
```

- Run terraform init and apply. Check the resource to be created, after that enter "y"

Code · bash

```
terraform init
terraform apply
```

Deploy activating the instances using the User-Data (POC Only)

- Modify terraform.tfvars with the desired options
  - (Required) LICENSE_KEY_CORE required if wanted to have the instance activated automatically
  - (Required) APIKEY_GENERATION=true to have the apikey generated by terraform. Need to activate the instance automatically
  - There is one general section and one section for each product that is supported using this Terraform project in case of needing to install additional OPSWAT products

terraform.tfvars · bash

```
# General variables

RG_NAME                 = "metadefender" # Prefix to add to all the resources
MD_REGION               = "eastus" # Region for all the resources
MD_VNET_CIDR            = "192.168.0.0/16"  # VPC CIDR where to create the MetaDefender products
PUBLIC_ENVIRONMENT      = true
APIKEY_GENERATION       = true
IMPORT_RG               = false
VM_PWD                  = "<SET_UP_VM_PWD>"

# MetaDefender Core variables

DEPLOY_CORE             = true
CORE_INSTANCE_TYPE      = "Standard_D8s_v5"   # Instance type for MetaDefender Core
LICENSE_KEY_CORE        = ""
OFFER_PRODUCT_CORE      = "opswat-mdcore-linux"  # Windows opswat-mdcore-windows
```

```
SKU_CORE                = "opswat-mdcore-linux"  # Windows opswat-mdcore-windows
NUMBER_INSTANCES_CORE   = 2
```

- Run terraform init and apply. Check the resource to be created, after that enter "y"

Code · bash

```
terraform init
terraform apply
```

Next Steps

- Create Azure function to handle the license automation to Azure function App using Visual Studio Code, or Command Line. See Licensing Automation on Azure
- After the function is created withing the Azure Function App. We need to trigger the function using Azure Alerts based on the

Recommended Architectures

These pages explain what are the architecture recommended for running MetaDefender Core in GCP GKE (Kubernetes)

Infrastructure Options for MetaDefender Core in GCP GKE

The possible infrastructure options in Azure AKS vary depending on how the PostgreSQL database is provisioned.

Options:

1. GCP GKE + GCP Cloud SQL

2. GCP GKE + PostgreSQL Pod

Above options are explained at GKE Architecture

We offer a provisioning script that depending on the selection, we automatically provision the cluster for you and install the K8S components needed for MD Core service. See GKE Deployment

GKE Cluster

The possible infrastructure vary depending on how the PostgreSQL database is provisioned

Options:

1. GCP GKE + GCP Cloud SQL

2. GCP GKE + PostgreSQL Pod

3. GCP GKE Autopilot + GCP Cloud SQL

We offer a provisioning script that depending on the selection, we automatically provision the cluster for you and install the K8S components needed for MD Core service. See GKE Cluster Deployment

MetaDefender PostgreSQL database details

Recommendations

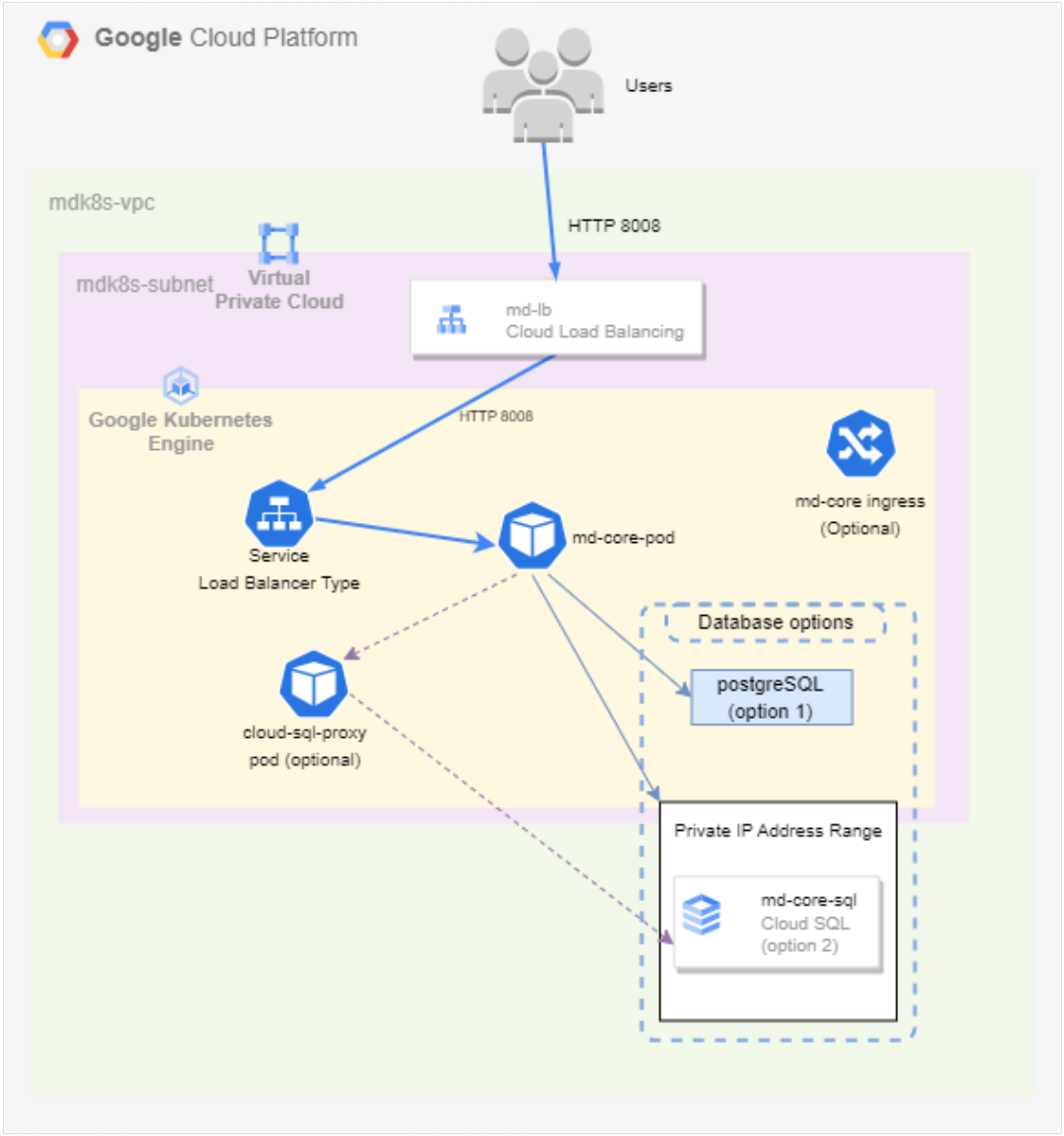
For running MD Core in a Kubernetes cluster we recommend to use, for production environments, an external database service for PostgreSQL. In case of **GCP** it is **GCP Cloud SQL** which the script will deploy and configure to be accessible from the cluster.

We offer different ways to access to the Cloud SQL Server from the GKE Cluster, these are the following ordered by recommended solutions

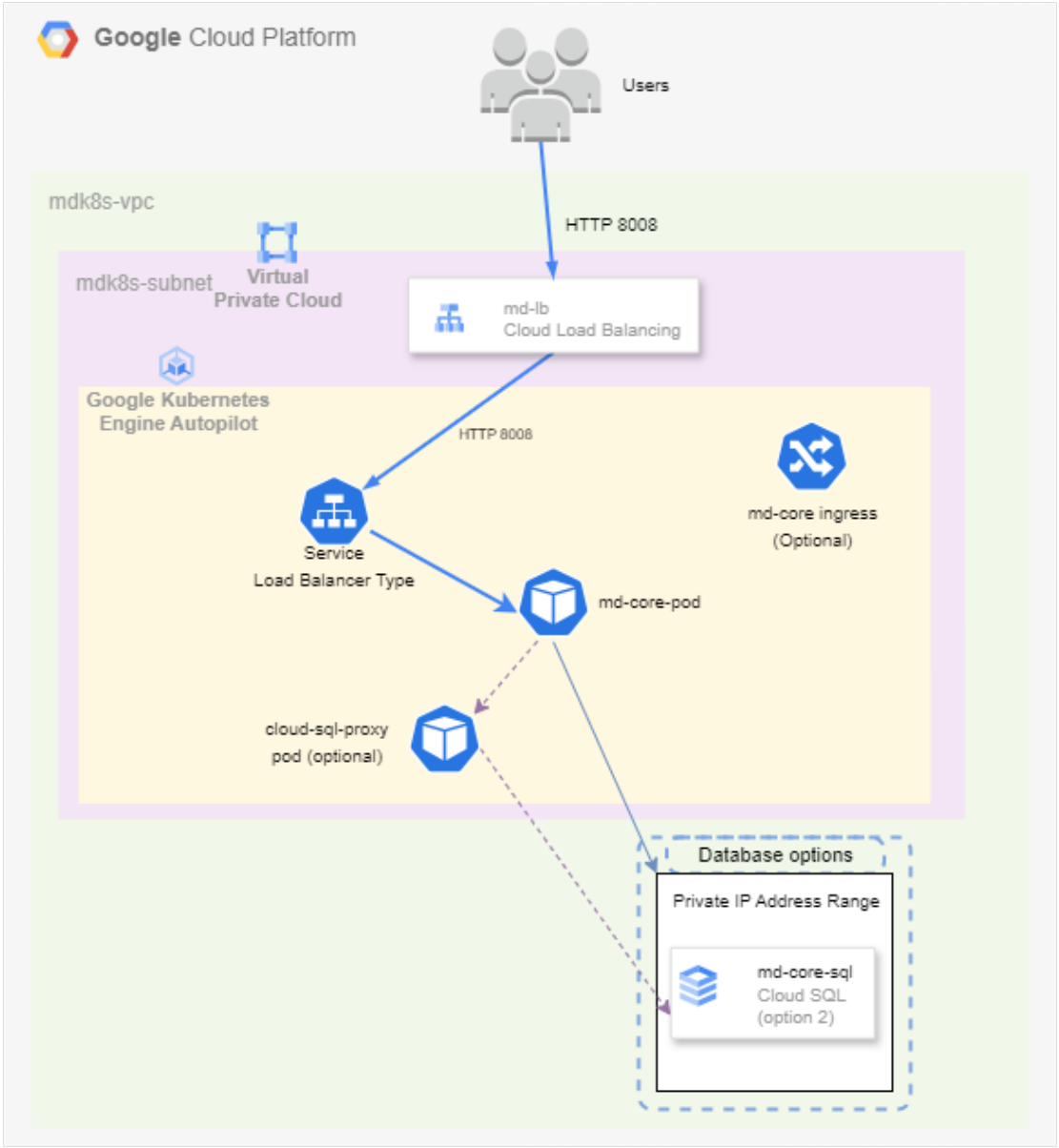
1. Private Connection
2. GCP Provided Connector in a Pod
3. GCP Provided Connector in a Sidecar container

For non-production environments or for stateless use cases, we offer the option of installing the database within the cluster using a standard PostgreSQL image.

GCP GKE + Cloud SQL Server/PostgreSQL Pod Diagram



GCP GKE Autopilot + Cloud SQL Server



Resources Inventory

Networking

Service Type	Resource Name	Description
GCP Compute Network	mdk8s-<project_id>-vpc	VPC for all the resources needed for MetaDefender products
GCP Compute Subnet	mdk8s-<project_id>-subnet	
GCP Compute Global Address	private-ip-address	
GCP Service Networking		

Service Type	Resource Name	Description
Connection		

Compute

Service Type	Resource Name	Description	Option
GCP GKE	<cluster_name>	Google Kubernetes Engine	All
Google Cloud SQL	postgresql-<postgres_db_account_name>	PostgreSQL Server for MetaDefender Core	Option 2 with Cloud

Info

The script will deploy a single Worker Node for the cluster with enough space for 1 replica of MD Core. Azure VM size is **Standard_F8s_v2 (8 vCPU & 16 GiB Memory).** Each pod would need a minimum of 4 vCPU and 8 GiB Memory. To change the request to adapt each pod to the specific case go to values.yml To change the size of the node pool go to terraform file terraform/azure/main.tf

GCP Deployments

Introduction

This Deployment Guide provides step-by-step instructions for deploying MetaDefender Core on Google Cloud Services infrastructure.

Organizations interested in protecting their solutions deployed in GCP can leverage MetaDefender to analyze and sanitize files residing, or transitioning, their GCP deployment. MetaDefender can scan and either sanitize or check for known vulnerabilities, depending on the type of traffic it's seeing. The ideal use case would be an organization that allows files to be uploaded to GCP through an external facing web portal. Analyzing files before they are made accessible to the end-users is critical to ensure that no malicious content is allowed and distributed through the web application.

Advanced attacks are concealing the malicious payload and are relying on productivity files (documents, pdfs, images) as a distribution mechanism. Productivity files allow active content to be leveraged, but these features are frequently exploited to execute the malicious behavior.

This guide is for IT infrastructure architects, administrators and DevOps professionals who are seeking to prevent potential malicious traffic being allowed in their Axure Cloud deployment. Threat Prevention is ensured for both productivity files that might be uploaded and for known vulnerabilities that can be identified for all running services/ applications deployed in GCP. The vulnerability scanning is checking known vulnerabilities for unpatched OS and running applications.

MetaDefender is provided as:

- Packaged installer available for download through the OPSWAT Portal.
- Container Image available in the OPSWAT Docker Hub Repository

There are currently no restrictions on GCP regions in which you can deploy MetaDefender.

OPSWAT Contact Information

Sales:sales-inquiry@opswat.com

Support:https://www.opswat.com/support

Other:https://www.opswat.com/contact

For deploying our solution in GCP, please review the guideline listed below:

- GKE Cluster

Database Encryption

MetaDefender Core supports both native OS encryption or GCP disk encrypted volumes. The database is secured via credentials and can be configured to encrypt everything at rest. In general, OPSWAT recommends the use of a 3rd party database layer, e.g. Google SQL Cloud, which should be encrypted. It is best practice to not share the database layer on the same hosts/GCP VM

Sensitive Data Storage

Network traffic processed by MetaDefender Core is SSL encrypted during transmission. The product does not collect or store any customer data outside of analyzing the contents of files submitted for scanning.

During the scanning process, files are stored unencrypted on disk to allow the scan process to complete. Once the scan completes, the file is deleted unless configured to place the file in quarantine. If you wish to encrypt these quarantined files, please use native OS encryption or GCP Disk encrypted volumes.

Private Key Management

The Private key pair is only used to SSH or RDP to the GCP VM. This is a very uncommon scenario, used primarily for debugging. You can create a new key pair, or use an existing pair.

Costs

MetaDefender is made available as an annual subscription. The only mandatory service that needs to be purchase is MetaDefender Core. MetaDefender Core is a paid service and is currently licensed on the BYOL (Bring Your Own License) model. Contact our sales team via our Contact form, available here: https://www.opswat.com/contact

Considering that are over 60 different options to license MetaDefender, it is highly coupled to the use case and the advanced features that you are considering deploying. Note that the more functionality you are adding to MetaDefender, the more need will be of CPU from the EC2 instance.

Our recommendation would be a minimum 8 vCPU for our lower tiers.

We recommend to go with Reserved Instances, considering that you are committing to an annual subscription for MetaDefender.

For pricing per vm, please refer to the official GCP pricelist

Sizing

MetaDefender needs minimum 8 vCPU, in order to have an optimal response rate to submitted files for analysis. However, depending on the use case and expected throughput (analysis SLA), higher tier instances are recommended.

The system requirements (hardware and supported operating system) are defined here: Recommended System Configuration

Operation Guidance for GCP VM deployments

Troubleshooting

For troubleshooting MetaDefender Core with GCP deployment , please refer to the following guides:

1. Troubleshooting MetaDefender Core (installation, logging, node management)
2. Scripted license management (Licensing status and automation)
3. Performance Scaling (for large deployments)
4. Engine and Database health (view Health Check section under different deployment option)

Routine Maintenance

Always update the MetaDefender deployment with the new versions published either on:

- my.opswat.com > Products section

Release Notes are available here.

MetaDefender provides a direct update mechanism for the licensed analysis engines (all the engines listed in Inventory > Technology). The application logic will require a product update, however the licensed engines are automatically updated (in online deployments). Both the engine and the signature updates for Anti-malware engines will be automatically downloaded and deployed on a daily basis. We recommend to configure the product to check for updates at least every 4h (for more details see Engines Update Configuration).

In general MetaDefender instance has Internet connection. If an offline deployment is considered, make sure you are using either the Central Management or the Update Downloader and that you are uploading the engines' signatures updates minimum once a day.

If you are deploying MetaDefender in your own AWS environment, the AWS service limits will not impact the product. A potential issue that can be caused by AWS service is using a lower-tier EC2 type than is recommended. Please see the Costs section above for details on which EC2 instance types we recommend for MetaDefender. Network bandwidth and EBS/General SSD IOPS can become a limitation in high-throughput environments. The solution to this is to simply use higher tier machines such as ones with 20+Gbps network bandwidth and moving to a provisioned IOPS (io2) disk(s).

MetaDefender supports the ability to rotate system credential and cryptographic keys to ensure the security of your deployment. Information for performing this rotation can be found on the Database Management page. MetaDefender users can also change user credentials via our POST - Change Password API.

Emergency Maintenance

In case of any availability failures of an AWS Service, recommended actions would be to pass the load to another MetaDefender instances, preferably deployed in a different AZ than the one affected. Any analysis in progress submitted to MetaDefender will be considered lost. However, is highly recommended to build a resilient system that won't rely on MetaDefender to recover the pending jobs, but to be managed by a queueing mechanism. See Backup and Recovery suggested deployment scenarios.

Support

To receive support, please visit our Support Portal. From there after you log in click on "Cases" on menu at the top of the page and then click "New" on the right hand side to create a new case.

Support policies, costs, levels and SLA's are described on our website, at the Support section: https://www.opswat.com/support. Please view our support tiers in the table below:

*Note: Please visit*https://www.opswat.com/support*for the most updated information on support.*

Reference Materials

All MetaDefender documentation is available on docs.opswat.com/mdcore

Localization

Info

The following sections are for GCP deployments. When installing MetaDefender Core in Kubernetes check Installing MetaDefender Core in K8S

	Standard	Gold	Platinum
Prices	Included with your license	20% ($500.00 minimum) *	25% ($2000.00 minimum) *
*_Support Hours *_			
Support via Portal Ticket System	8AM to 5PM business days (local to you)	7AM to 7PM business days (local to you)	24 x 7 x 365
Support via Chat	8AM to 5PM business days (local to you)	7AM to 7PM business days (local to you)	24 x 7 x 365

	Standard	Gold	Platinum
Support via Phone			24 x 7 x 365
Incident Response Times			
Blocker Issues First Response Times	Within 1 business day	Within 2 hours	Within 1 hour
Non-blocker Issues First Response Times	Within 3 business days	Within 2 business days	Within 1 business day
Support Account Management			Yes
Support Account Manager			Yes
Quarterly Conference Call Reviews			Yes
Yearly Roadmap Review with the Product Team			Yes
Quarterly Summary Reports		Yes	Yes

MetaDefender products and documentation are available exclusively in English for now.

GKE Cluster

This guide explains how to use the provisioning script provided by OPSWAT to create an GCP GKE and generate all the Kubernetes components needed to run MetaDefender Core.

MetaDefenderK8S script details

- **GitHub Project** OPSWAT/metadefender-k8s --> **Script path:** ./metadefenderk8s.sh
- **Programming Language:** Bash
- Installation **Pre-requisites** for provisioning:
  - Terraform
  - Helm
  - gcloud cli

- Kubectl
  - kubectl plugin "gke-gcloud-auth-plugin" (Since GKE 1.6)

- **GCP Credentials**
  - The Cloud SQL Admin API is enabled. (Optional, enabled when Cloud SQL database is going to be used)
  - Download the credentials for the service account, it should have the following permissions
    - Editor
    - Compute Network Admin
  - Set in your environment variable the path to the JSON key with the credentials for the service account to use
    - **GCP_JSON_CREDENTIALS_PATH**

- MetaDefender **Core License Key** (Required with --mdcore parameter)
  - Set it in your local environment credentials under ***MDCORE_LICENSE_KEY***

How to run script

Only MD Core · bash

```
./metadefenderk8s.sh provision -l GCP --mdcore --project_id <PROJECT_ID>
```

MD Core + MDSS · bash

```
./metadefenderk8s.sh provision -l GCP --mdcore --mdss
```

MD Core + ICAP · bash

```
./metadefenderk8s.sh provision -l GCP --mdcore --icap
```

Info

The script will deploy a single Worker Node for the cluster with enough space for 1 replica of MD Core. GCP compute size is **e2-standard-8 (8 vCPU & 32 GB Memory).** Each pod would need a minimum of 4 vCPU and 8 GB Memory. To change the request to adapt each pod to the specific case go to values.yml To change the size of the node pool for having more MD Core replicas or install additional MetaDefender products go to terraform file terraform/gcloud/main.tf

Script Parameters

Parameter	Flags	Options	Default	Description	Required/ Optional
Action		• provision • install		Action to indicate the script if we want to provision (Create resources + install Core) or install (Install Core)	**Required**
Location	-l or --location	• AWS • Azure • GCP		Where is going to be the K8S cluster	**Required**
MetaDefender Flag Installation	Combination of • --mdcore • --mdss • --icap		•	Install MetaDefender Core in the cluster provisioned	**Required**
Project ID	--project_id			GCP project where the resources will be created	**Required**
Image Version	--image	latest 5.0.1	latest	MetaDefender Core image version to install	Optional
Region	--region	GCP Regions	us-central1	AWS region where all the resources will be provisioned	Optional
Cluster Name	--name	Not uppercase allowed	md-k8s	Name of the cluster that will be used for	Optional

Parameter	Flags	Options	Default	Description	Required/ Optional
				naming all the resources	
Number of Replicas	--replicas	[0-9]*	1	Number of replicas for MetaDefender Core service	Optional
Namespace	--namespace	[A-Za-z]{1,10}		Namespace where MetaDefender products will be installed in the K8S Cluster	Optional Max Characters: 10

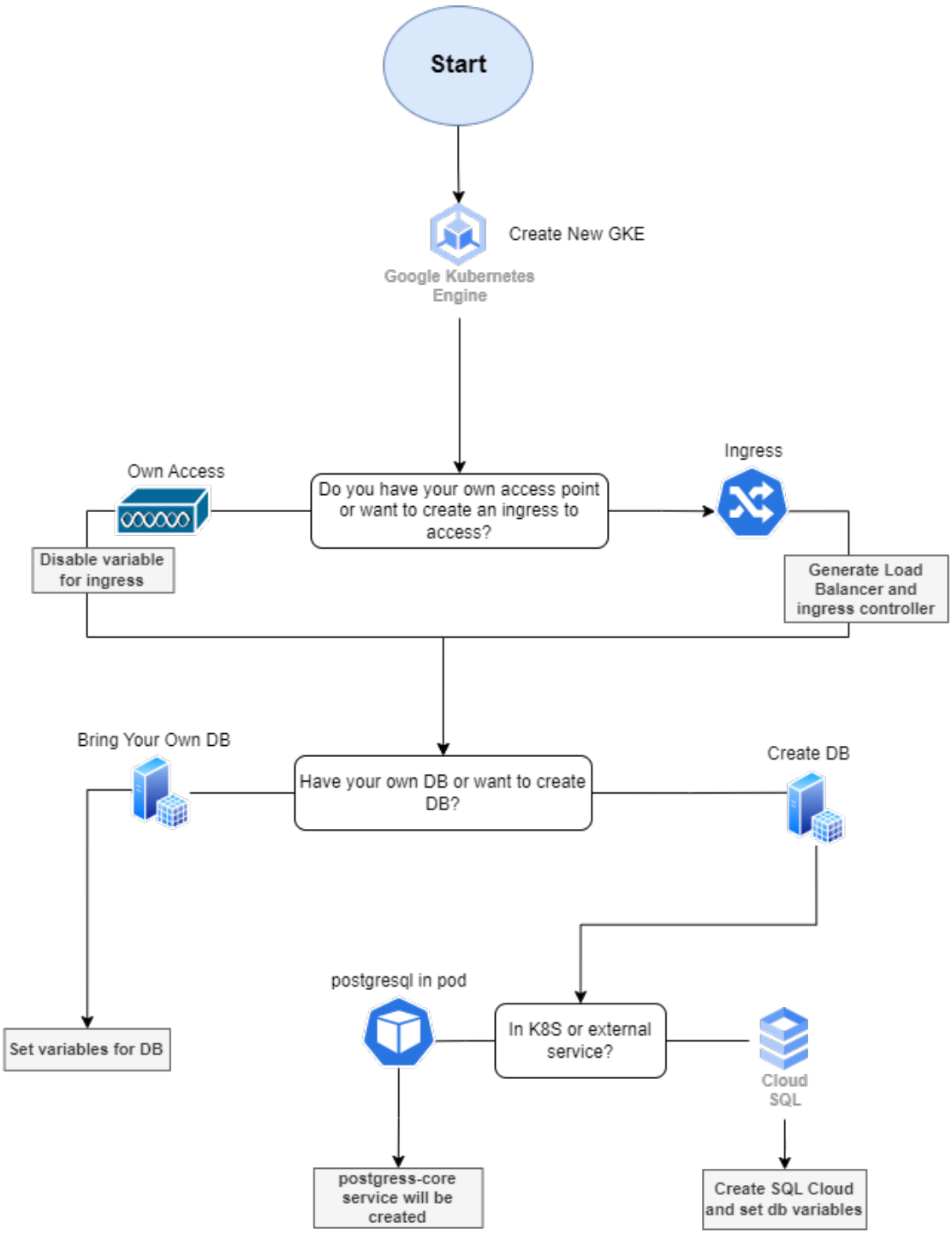
Connection to Cloud SQL Database from GKE

MetaDefender Core Flowchart Provisioning in GCP

The following flowchart represents how the provisioning script will configure the environment based on the options selected for provisioning GCP GKE.

Summary options to be selected

1. Access to the K8S cluster. Generate **Ingress** or provide **own** access.
    1. **An Ingress** and an internal load balancer will be created per each product flag added as parameter to the script
    2. **Own Access**, you decide how to access to the cluster so it won't generate any ingress but will still create the internal load balancer for the product service deployed
2. Have your **own database** or create **new database**
    1. **Own database,** will be asked if you want the script to set up the credentials and database host url for you or the script will just indicate the secrets to edit, later on by you, for connecting the MetaDefender Core with your database.
    2. Create new **DB in K8S** or **external DB** that for GCP we will provision a **Cloud SQL Server**
        1. **If external, indicate how to connect to it, by private connection or SQL Proxy pod**



GKE Autopilot Deployment

For having GKE of type Autopilot it is needed to change AUTOPILOT_GKE variable to true from terraform.tfvars. Database has to be external (Cloud SQL)

Provisioning OpenShift on your CSP

Requirements

Most cloud service providers (CSP) have options for managed deployments of RedHat OpenShift without the need to manually provision and configure VMs. The available resources for the nodes of such a deployment that are intended for MD Core should still follow the recommended system requirements from here depending on the expected load and desired performance per node.

The MD Core helm chart used for deployment already has a default request for 7 CPUs and 7 GB of memory and a limit of 8 CPUs and 8 GB of memory, these values can be changed at installation depending on the provisioned resources available.

The PostgreSQL database can be provisioned separately as a managed service (recommended) or as part of the same MD Core deployment on the cluster, in which case the Cluster needs additional resources depending on the expected demand (PostgreSQL requirements). All major CSPs have managed database solutions compatible with PostgreSQL clients like MD Core.

Database Options

Option 1. Provisioning a database managed by the cloud provider:

- Microsoft Azure Database for PostgreSQL: https://learn.microsoft.com/en-us/azure/postgresql/single-server/quickstart-create-server-database-portal
- Amazon RDS: https://aws.amazon.com/getting-started/hands-on/create-connect-postgresql-db/
- Google Cloud SQL: https://cloud.google.com/sql/docs/postgres/connect-instance-cloud-shell

Option 2. Using an OpenShift operator for PostgreSQL. There are several publicly available operators for HA deployments of PostgreSQL, this type of deployment allows running the database inside the same cluster as MD Core. A persistent volume using managed storage from the cloud provider is preferred for reliability.

Example deployment: https://portworx.com/blog/run-ha-postgresql-red-hat-openshift/

Option 3. Using the MD Core helm chart. By default, the helm chart also deploys a single PostgreSQL instance alongside MD Core and configures MD Core to use this database. This option is intended mostly for testing and demonstration.

Provisioning OpenShift

- AWS: https://docs.openshift.com/rosa/rosa_getting_ started/rosa-quickstart-guide-ui.html
- Azure: https://learn.microsoft.com/en-us/azure/openshift/tutorial-create-cluster
- Google Cloud: https://docs.openshift.com/container-platform/4.10/installing/installing_gcp/installing-gcp-default.html

After the cluster is provisioned, the MD Core helm chart can be installed: https://github.com/OPSWAT/metadefender-k8s/blob/main/helm_charts/mdcore-README.md

Deploy your Own OpenShift Cluster

RedHat OpenShift is platform agnostic and can be deployed on any compatible hardware. The installation process is

covered in detail on their documentation page:

https://docs.openshift.com/container-platform/4.12/installing/installing_platform_ agnostic/installing-platform-agnostic.html

Cluster requirements for MD Core

- Cluster nodes should still follow the recommended MD Core system requirements for the expected load
- Persistent storage is required for the MD Core database (PostgreSQL) while the MD Core containers are stateless and do not require any persistency
- In case of an on-premise cluster, internet access to hub.docker.com is required to pull all MD Core related images

MD Core installation

Resources for Kubernetes, OpenShift and cloud deployments in general are available on our GitHub repo:

https://github.com/OPSWAT/metadefender-k8s

Installation is performed using the provided helm chart for MD Core and an example configuration for OpenShift is provided in the `helm_charts/mdcore-openshift.yml` file.

Adding the helm repository

The GitHub repository can be used directly as a helm repo:

Code · bash

```
helm repo add mdk8s https://opswat.github.io/metadefender-k8s/
helm repo update mdk8s

#Example installation command
helm install my_mdcore mdk8s/metadefender_core -f mdcore-openshift.yml
```

Or the repository can be cloned locally:

Code · bash

```
git clone https://github.com/OPSWAT/metadefender-k8s.git metadefender
cd metadefender/helm_carts

#Example installation command
helm install my_mdcore ./mdcore -f mdcore-openshift.yml
```

Cluster requirements

1. A configured image pull secret for the current OpenShift user for the RedHat docker repo: `registry.redhat.io`. The helm values for OpenShift use the following image from RedHat: `registry.redhat.io/rhel8/postgresql-`

`12` . This is only required if using the database deployment from the Helm chart, a managed external database service can be configured instead if available. The repo credentials ca be configured with the following `oc` commands:

Code · bash

```
oc create secret docker-registry imagepullsecret --docker-server=registry.redhat.io --docker-
username=<REDHAT_USER> --docker-password=<REDHAT_PASSWORD> --docker-email=<REDHAT_EMAIL>

oc secrets link <OPENSHIFT_USER> imagepullsecret --for=pull
```

2. An existing persistent volume or storage class to be used for database persistency. The `helm_charts/mdcore-openshift.yml` values file is configured with an example persistent volume claim using a certain storage class.

Storage

MD Core containers are stateless and don't require any persistent storage. If the PostgreSQL database is deployed in the cluster, then it's recommended to use persistent storage managed by a cloud provider.

The helm chart can be configured to use a custom storage class or persistent volume by setting the `storage_provisioner` value to custom, adding the Kubernetes yaml for the persistent volume claim in the `storage_configs` value and then using the pvc name in the `storage_name` value to tell the Postgres pod where to save it's files. Here's an example using managed storage in Azure:

Helm Values · yaml

```
storage_provisioner: custom       # Type of storage to use in the pod definition
storage_name: ocstorage           # Name of the PVC to use in the pod definition

storage_configs:          # Example using a PVC with dynamic provisioning from an existing
storage class
  pvc-example:
    apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      name: ocstorage
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 1Gi
      storageClassName: managed-premium
```

The `managed-premium` storage class is already available in the cluster if provisioned by Azure:



Helm chart

To deploy the helm chart directly in a RedHat OpenShift cluster we have the `mdcore-openshift.yml` values file. This file can be used as an example of the changes required for OpenShift:

- **PostgreSQL image**: the docker image has been changed to use the RedHat repo: `registry.redhat.io/rhel8/postgresql-12`
- **Storage**: a persistent volume claim has been configured to use an existing storage class since `hostPath` is not supported on an unprivileged container

Example installation when using local helm files and setting the custom values manually:

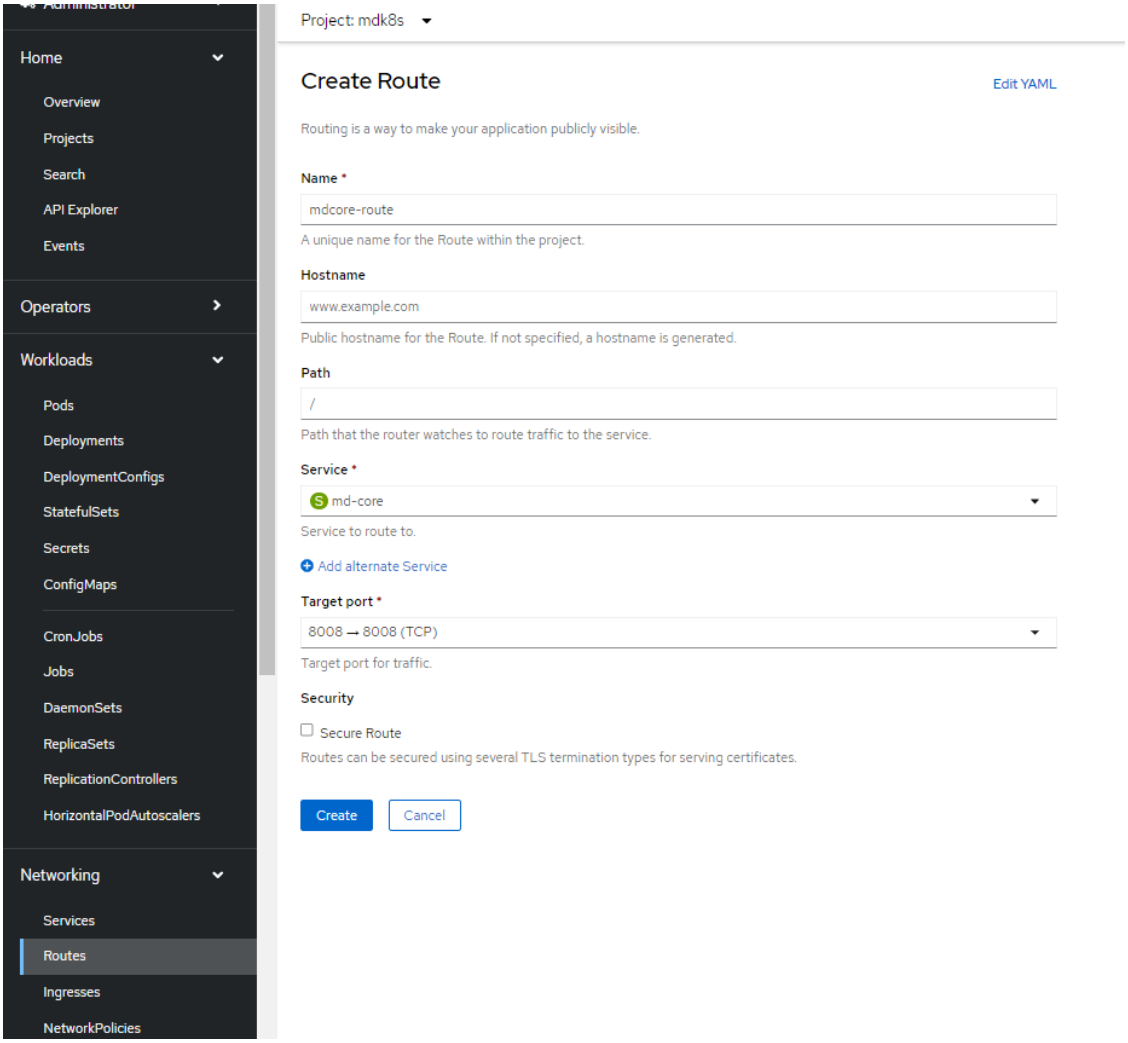
Code · bash

```
helm install my_mdcore ./helm_charts/mdcore -f mdcore-openshift.yml \
--set 'db_password=<SET_POSTGRES_PASSWORD>' \
--set 'env.POSTGRESQL_ADMIN_PASSWORD=<SET_POSTGRES_ADMIN_PASSWORD>' \
--set 'storage_configs.pvc-example.spec.storageClassName=<SET_STORAGE_CLASS_NAME>' \
--set 'mdcore_license_key=<SET_LICENSE_KEY>'
```

Exposing MD Core

After installation MD Core can be exposed in OpenShift by creating a new route in the `Networking -> Routes` section with the following settings:

- Path: `/`
- Service: `md-core`
- Target port: `8008 -> 8008`



An ingress is also created by default and can be disabled by setting the `core_ingress.enabled` value to `false`.

Overview

Introduction

This Installation Guide provides step-by-step instructions for installing MetaDefender Core in your own Kubernetes cluster in whatever environment you have it deployed.

Organizations interested in protecting their solutions deployed in Kubernetes technology can leverage MetaDefender to analyze and sanitize files. MetaDefender can scan and either sanitize or check for known vulnerabilities, depending on the type of traffic it's seeing. Analyzing files before they are made accessible to the end-users is critical to ensure that no malicious content is allowed and distributed through the web application.

This guide is for IT infrastructure architects, administrators and DevOps professionals who are seeking to prevent potential malicious traffic being allowed in their environments. Threat Prevention is ensured for both productivity files that might be uploaded and for known vulnerabilities that can be identified for all running services/applications. The vulnerability scanning is checking known vulnerabilities for unpatched OS and running applications.

Guide Structure

1. Kubernetes Components
2. Deploying on K8S
3. Upgrade/Deployment Strategy on K8S
4. High Availability on K8S
5. Scaling K8S Cluster
6. Troubleshooting

Kubernetes Supported Versions

Kubernetes Version	Known Limitations
1.22.17 and later	None
Before 1.22.17	When pod eviction, automation of core deactivation fails.

Kubernetes Components

Introduction

MD Core can be installed in Kubernetes thanks to the components that are defined in the MetaDefender core helm chart stored in https://github.com/OPSWAT/metadefender-k8s/tree/main/helm_charts/mdcore . This guide explain what are the different components that we propose for MetaDefender Core to properly run in Kubernetes.

Info

For deploying this components we offer an installation script see Installing MetaDefender Core in K8S for installing MetaDefender Core in an already created cluster or EKS Cluster for provisioning a Kubernetes cluster in AWS and install MetaDefender Core on it

K8S Component Architecture Options

There will be deployed different services depending on how the cluster is accessed and what type of database will be used to work with MetaDefender Core.

These are the possible combinations:

- MetaDefender Core Service + Ingress + PostgreSQL service
  - md-core service
  - postgres-core service
  - core-ingress component (Optional)
- MetaDefender Core Service + Ingress + External Database
  - md-core service
  - core-ingress service (Optional)
  - PostgreSQL external database

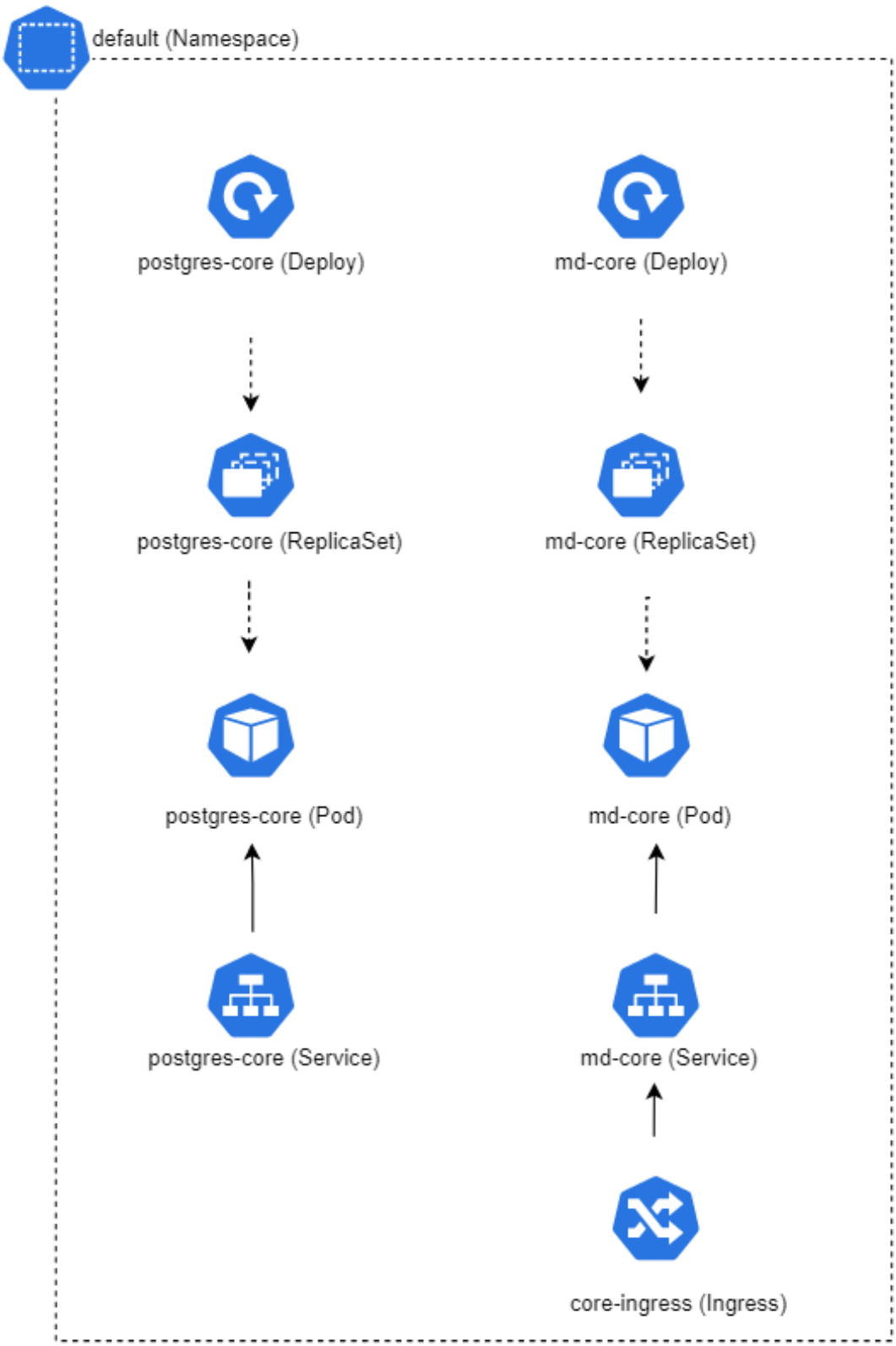
Overview Architecture of K8S Component

Components Table

Component Type	Component Name	Description
Ingress	core-ingress	Ingress component for accessing to the md-core service from outside of the cluster. In case you have an external load balancer, it is optional, as you can setup an ingress controller for any of the CSPs load balancers
Deployment	md-core	Define what are the replicas and the containers that are needed for md-core
Deployment	postgres-core	Define what are the replicas and

Component Type	Component Name	Description
		the containers that are needed for the postgreSQL database. Optional in case you don't have an external PostgreSQL database to connect with md-core service
Service	md-core	Represent a single point to access to all the pods for md-core
Service	postgres-core	Represent a single point to access to all the pods for postgres-core
Pod	md-core	Contains all the containers needed for running MetaDefender Core app.
Pod	postgres-core	Contains the container needed for running the PostgreSQL for MetaDefender Core.

Diagram of Components



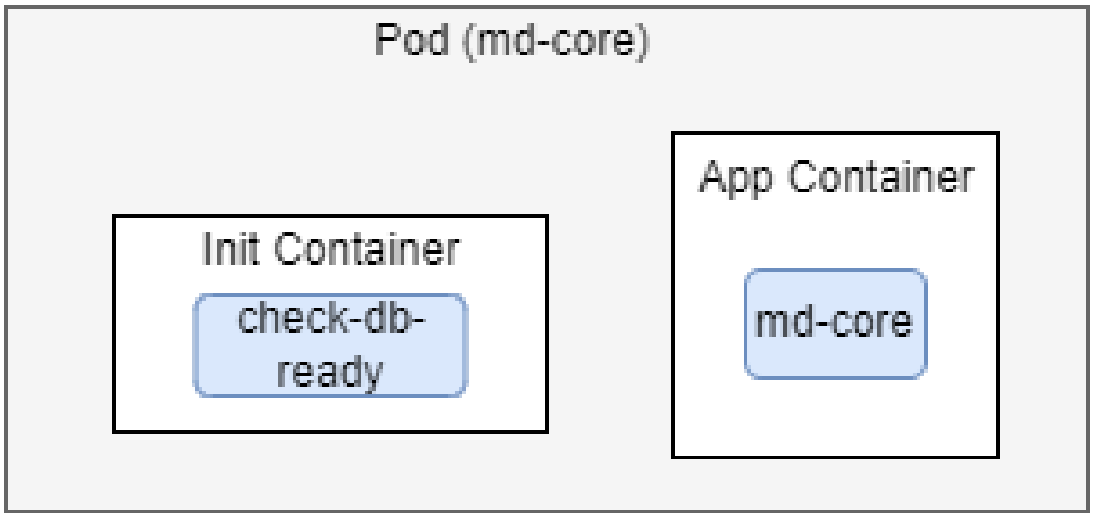
Info

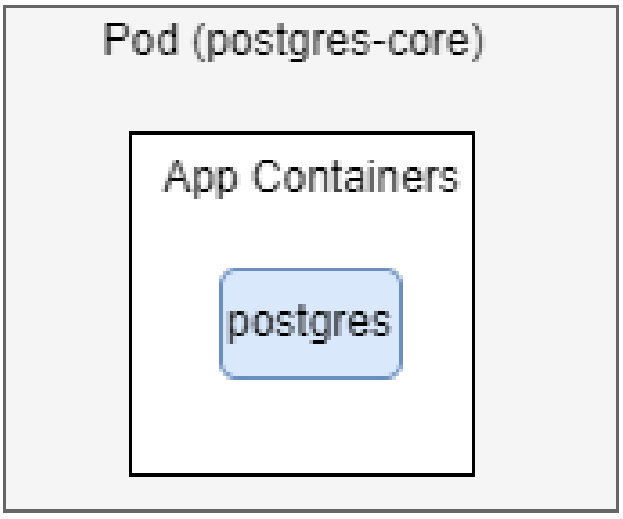
postgres-core components will be needed to be generated just in case an external database is not connect to the md-core service

Containers Design

Containers Description

- **Init Containers:** Run at starting to check if the database is ready and reachable from the pod.
- **App Container:** Main container of the application that host al the logic to activate/deactivate the service with the license





Secrets Configuration

- **mdcore-api-key:** API key autogenerated in case it is not indicated in the helm chart
- **mdcore-cred:** Admin user and password autogenerated in case it is not indicated in the helm chart
- **mdcore-license-key:** MetaDefender Core activation key needed to activate the pods.
- **mdcore-postgres-cred:** Database Admin user and password for md-core service to have access to the database

Features

Load balancers in AWS

For accessing to the md-core service you can set up a load balancer in AWS that will need to install a controller for AWS. For creating the controller we have included in this script al the commands indicated by AWS in this docs

Deploying on Kubernetes

Using Kubernetes technology provides us with some features that distinct it from other type of deployments. These are the main features:

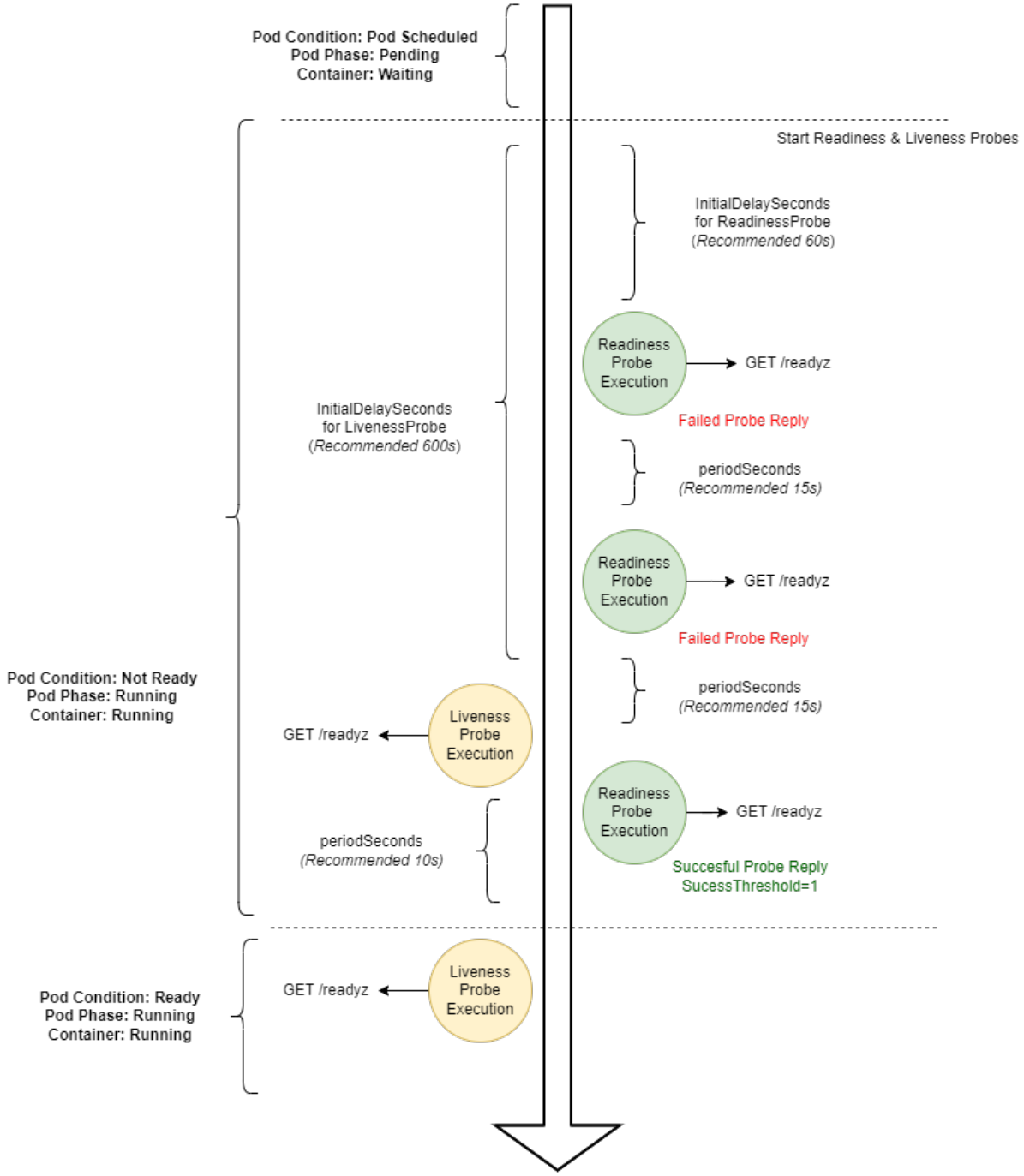
- **Service discovery and load balancing.** See Installing MetaDefender Core in K8S
- **Storage orchestration.** Allows you to automatically mount a storage system. See Storage Configuration on K8S
- **Automated rollouts and rollbacks.** See Upgrade/Deployment Strategy on K8S
- **Automatic bin packing.** You tell Kubernetes how much CPU and memory (RAM) each container needs.
- **Self-healing.** Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve. Se **Kubernetes Probes** explained below
- **Secret and configuration management.** Kubernetes lets you store and manage sensitive information. See Kubernetes Components
- **Horizontal scaling.** Scale your application up and down. See Scaling K8S Cluster

There are recommended configuration for each of the features to use by default when deploying MetaDefender Core in a Kubernetes cluster.

Kubernetes Probes

The Kubernetes probes are needed to control the correct behavior of the application on this environment. Each probe need to use the health check endpoint GET - Get health check status (/readyz)

- **Readiness Probe** (Check /readyz endpoint to know if the Health Check is OK)
  - `initialDelaySeconds`= 60
  - `periodSeconds` = 10
  - `successThreshold` = 1
  - `timeoutSeconds` = 10
- **Liveness Probe** (Check /readyz endpoint to know if the Health Check is OK after Pod is ready)
  - `initialDelaySeconds`= 90
  - `timeoutSeconds` = 10
  - `periodSeconds` = 10
  - `failureThreshold`= 3



Installing MetaDefender Core in K8S

This guide explains how to use the provisioning script for generating all the Kubernetes components needed to run MetaDefender Core in you already created K8S cluster that depending on some configuration options chosen it will adapt the helm chart values for configuring it properly.

Also includes the details needed to understand how to install MetaDefender Core using directly the Helm chart.

Install using the Helm chart

The MD Core k8s deployment can be performed directly using the provided helm chart in our public GitHub repo here

and example configuration files for different environments are provided in the `helm_charts` directory.

Using the helm repository

The GitHub repository can be used directly as a helm repo:

Code · bash

```
helm repo add mdk8s https://opswat.github.io/metadefender-k8s/
helm repo update mdk8s

#Example installation command
helm install mdcore mdk8s/metadefender_core -f <CUSTOM_VALUES_FILE.yml>
```

Or the repository can be cloned locally:

Code · bash

```
git clone https://github.com/OPSWAT/metadefender-k8s.git metadefender
cd metadefender/helm_charts

#Example installation command
helm install mdcore ./mdcore -f <CUSTOM_VALUES_FILE.yml>
```

Exposing MetaDefender Core Service

By default, the helm chart deploys a `ClusterIP` service for MD Core and this can be changed to any service type supported by the Kubernetes cluster. For example, a `LoadBalancer`service type can be created by overwriting the `service_type` value in the `md-core` component:

Code · yaml

```
core_components:
  md-core:
    service_type: LoadBalancer
```

MD Core can also be exposed using an ingress:

Code · yaml

```
core_ingress:
  host: <APP_NAMESPACE>-mdcore.k8s  # Hostname for the publicly accessible ingress, the
`<APP_NAMESPACE>` string will be replaced with the namespace where the chart is deployed
  enabled: true                     # Enable or disable the ingress creation
  class: nginx                      # Sets the ingress class depending on the installed ingress
controller
```

Scaling MD Core

Multiple MD Core pods can be deployed by setting the `replicas` value in the `md-core` component:

Code · yaml

```yaml
core_components:
  md-core:
    replicas: 3
```

TLS Configuration

In case we want to use a self-signed certificate, we need to create the secrets with the crt and key

Code · bash

```bash
mkdir ~/cert
openssl req -new -newkey rsa:4096 -days 36500 -nodes -x509 -keyout ~/cert/mdcore.key -out ~/
cert/mdcore.crt

# Needed to replace <user>. kubectl do not accept "~"
kubectl create secret generic mdcore-tls-cert --from-file=/home/<user>/cert/mdcore.crt -n
default
kubectl create secret generic mdcore-tls-cert-key --from-file=/home/<user>/cert/mdcore.key -n
default
```

We can set up the configuration parameters in the values.yaml file

Code · yaml

```yaml
# From Values.yaml
tls:
  enabled: false
  certSecret: mdcore-tls-cert
  certSecretSubPath: mdcore.crt
  certKeySecret: mdcore-tls-cert-key
  certKeySecretSubPath: mdcore.key

ReadinessProbe:
  httpGet:
    scheme: HTTPS
livenessProbe:
  httpGet:
    scheme: HTTPS
```

Ingress Configuration By CSP

1. In case we want to use a self-signed certificate, we need to create the tls secret with the crt and key

Code · bash

```
mkdir ~/cert
openssl req -new -newkey rsa:4096 -days 36500 -nodes -x509 -keyout ~/cert/mdcore.key -out ~/
cert/mdcore.crt

# Needed to replace <user>. kubectl do not accept "~". Also change <namespace>
kubectl create secret tls mdcore-tls --key="/home/<user>/cert/mdcore.key" --cert="/home/<user>/
cert/mdcore.crt" -n <namespace>
```

2. Set up the following configuration parameters in the values.yaml file

Code · yaml

```yaml
# Ingress setting for md core (md core setting are ignored if deploy_with_core is false)
core_ingress:
  host: <APP_NAMESPACE>-mdcore.k8s       # Hostname for the publicly accessible ingress, the
`<APP_NAMESPACE>` string will be replaced with the namespace where the chart is deployed
  service: md-core                  # Service name where the ingress should route to, this
should be left unchanged
  port: 8008                        # Port where the ingress should route to
  enabled: true                    # Enable or disable the ingress creation
  spec_className: true              # true -> for adding class as spec || false -> for adding
class as annotation. false for GKE as it needs to be as annotation.
  class: nginx                      # Sets the ingress class
  tls: true                        # Flag for set up tls section in ingress
  secret: mdcore-tls                # SecretName of the tls secret created to be used for
ingress
  ingress_annotations:
    nginx.ingress.kubernetes.io/affinity: cookie     # To set affinity when having more than one
pod running behind the ingress
```

3. Install MetaDefender Core using Helm
4. Install Nginx Ingress Controller
    1. For Azure see example below
    2. For AWS you need to install AWS LB Controller and set up the ingress with the annotations indicated
    3. GCP install it automatically, see below configuration needed

Azure

Code · bash

```bash
helm upgrade --install mdcore-ingress ingress-nginx/ingress-nginx  --debug \
--create-namespace \
--namespace nginx-ingress \
--version 4.6.1 \
```

```
--set controller.replicaCount=2 \
--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-
internal"=false \ #Change to true if you want to have an internal load balancer
--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-health-
probe-request-path"=/healthz
```

AWS

yaml · yaml

```yaml
core_ingress:
  host: <APP_NAMESPACE>-mdcore.k8s
  enabled: true
  spec_className: true
  ingress_annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
```

GCP

values.yml · yaml

```yaml
core_ingress:
    spec_className: false

core_components:
  md-core:
        service_type: NodePort
```

Proxy Configuration

To enable Proxy, the following criteria must be met

- The proxy server should be already deployed and reachable from the MetaDefender Core pod.
- Add env vars needed

values.yaml · bash

```bash
env:
  # MD Core proxy settings
  PROXY: false              # Enable the proxy settings in MD Core
  PROXY_HOST: ""            # Proxy server host address
```

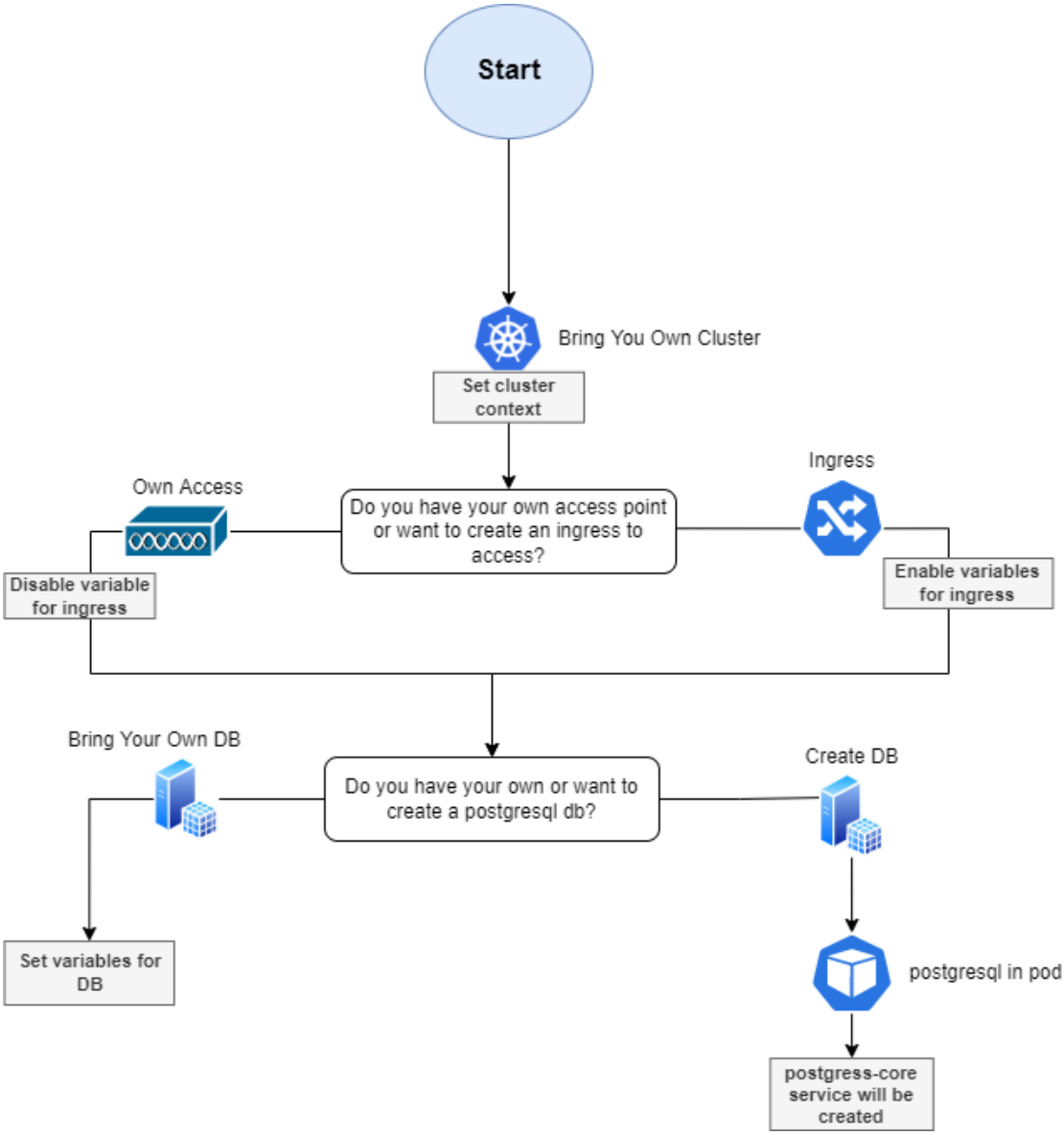
```
  PROXY_PORT: 8080          # Proxy server port
  PROXY_USER: ""            # (optional) Username for proxy authentication
  PROXY_PWD: ""             # (optional) Password for proxy authentication
  PROXY_EXCLUSION: ""       # Not use the proxy server for the addresses starting with the
following entries, separated by comma
```

Flowchart for MetaDefender K8S Script

The following flow chart represents how the MetaDefenderK8S script will configure the environment based on the options selected for installing MetaDefender products in an already created cluster.

Summary options to be selected

1. Select your cluster context where you want to install the MetaDefender products
2. Access to the K8S cluster. Generate **Ingress** or provide **own** access.
    1. **An Ingress** will be create per each product flag added as parameter to the script
    2. **Own Access**, you decide how to access to the cluster so it won't generate any ingress for accessing
3. Have your **own database** or create **new database**
    1. **Own database,** will be asked if you want either
        1. the script to set up the credentials and database host url for you
        2. the script will just indicate the secrets to edit, later on by you, for connecting the MetaDefender Core with your database.
    2. Create new **DB in K8S.** It will generate a postgreSQL pod inside the cluster



How to run script

Only MD Core · bash

```
./metadefenderk8s.sh install --mdcore --name <k8s-cluster-name>
```

MD Core + MDSS · bash

```
./metadefenderk8s.sh install --mdcore --name <k8s-cluster-name> --mdss
```

Script Parameters

Parameter	Flags	Options	Default	Description	Required/ Optional

Parameter	Flags	Options	Default	Description	Required/ Optional
Cluster Name	--name		md-k8s	Name of the cluster that will be used for naming all the resources	**Required**
MetaDefender Flag Installation	Combination of • --mdcore • --mdss • --icap		•	Install MD Core in the cluster provisioned	**Required**
Image Version	--image	latest 5.0.1	latest	MD Core image version to install	Optional
Region	--region	[AWS Regions] (AWS Regions)	eu-central-1	AWS region where all the resources will be provisioned	Optional
Number of Replicas	--replicas	[0-9]*	1	Number of replicas for MD Core service	Optional
Namespace	--namespace	[A-Za-z]{1,10}		Namespace where MetaDefender products will be installed in the K8S Cluster	Optional Max Characters: 10

Resources Requirements

CPU & RAM Allocation

See system requirement for CPU and RAM for Linux system defined in [Linux Recommended System Configuration,linux-recommended-system-configuration] based on License type purchased, multiscanning with 5, 10

or MAX engines. In case of having Deep CDR and Proactive DLP see values recommended for CPU and RAM.

To achieved a desired performance when scanning the files it is recommended to do a load test with the system required resource assigned. There are 3 factors that are important to take into consideration that would mainly affect to the CPU usage.

1. Number of files to scan per minute

2. Average file size

3. File types

Knowing well these 3 factors will help to allocate the proper resources to have best scan time.

Sharing Kubernetes Cluster with other applications

Helm chart - request only, include notes for limits in case of sharing apps

Disk Space Allocation

See Storage Usage Information for minimum GB disk space allocation. It will differ depending on the space that is needed to allocate for the temporary files during processing, quarantine and sanitized files.

By default the pod is stateless so it is using the disk space allocated by the worker node where the pod is running on, to request a minimum of storage it is needed to use "ephemeral-storage:" key in the values.yaml file.

In case of wanting to mount a disk for specific files, check the location for those file in [Linux Installation Details,linux-installation-details].

Database Requirements

How to assign CPU & Memory to pod

To assign resources from the helm chart it is needed to modify the resources component in the values.yaml file

values.yaml · yaml

```yaml
core_components:
  md-core:
    resources:
      requests:
        memory: "7Gi"                                   # Minimum reserved memory
        cpu: "4.0"                                      # Minimum reserved cpu
        ephemeral-storage: "60Gi"                       # Minimum reserved storage
      limits:
        memory: "8Gi"                                   # Maximum memory limit
        cpu: "8.0"
```

Licensing in K8S

Environment variable for containers

The container image will use the `LICENSE_KEY` environment variable to run the automation to activate and deactivate the container license.

Main Licensing Automation in K8S

Every time a pod of md-core is created it will create a deployment id attached to that pod so it is automatically activated. In case the pod goes down it will be automatically deactivated thanks to the logic included in the app container that will make a call to the OPSWAT Activation Server.

Following the best practices for Kubernetes, MetaDefender Core handles the SIGTERM signal, sent by a Kubernetes cluster component to the app container, and run the logic to deactivate the deployment id linked to the pod.

> **Warning**
>
> The md-core pods use the secrets **mdcore-license-key** to generate a deployment id and activate that pod. In case of failure on activation the app container will restart until that secret is properly set up with a license key with enough activations.

Licensing Clean Up on Involuntary Disruption (Optional)

Aiming to cover as much as possible situations of disruption when running MetaDefender Core application in Kubernetes, OPSWAT provides with a mechanism to cleanup the zombie activations in case an old app container terminated did not get the SIGTERM signal from the Kubernetes cluster.

How to enable licensing Clean Up

When providing the environment variable `LICENSING_CLEANUP` in the MetaDefender Core pod, it will store licensing information in a the secret **mdcore-licensing** that will be created by the pod in case it is not already created. Therefore it is needed to create a role that will be bind to the default service account that the pod is using by namespace.

Step by step using helm and kubectl

- Set Up environment variables
  - variables already in the latest helm chart version. Only needed to change the flag to "true"

Values.yml · bash

```
env:
  LICENSING_CLEANUP: true

## ADD to the rest of the environment variables
md-core:
    env:
       - name: MY_POD_NAME
      valueFrom:
        fieldRef:
```

```
          fieldPath: metadata.name
    - name: MY_POD_NAMESPACE
      valueFrom:
        fieldRef:
          fieldPath: metadata.namespace
    extra_labels:
        name: md-core
```

- Modify `core-worker-role.yml` and `core-worker-role-binding.yml` changing the namespace if needed
- Create **Role** and **RoleBinding** with the following commands

Code · bash

```
# from root path of github metadefender-k8s project
kubectl apply -f example_scripts/core-worker-role.yml
kubectl apply -f example_scripts/core-worker-role-binding.yml
```

Storage Configuration on K8S

MD Core containers are stateless and don't require any persistent storage by default. However, there are some cases where it is needed to attach a volume to the app container or database container.

Configure Database Storage

If the PostgreSQL database is deployed in the cluster, then it's recommended to use persistent storage managed by a cloud provider.

When using an external database that is not deployed from the MD Core chart, the `deploy_with_core_db` value has to be set to `false` in order to not deploy an additional database from the chart.

The helm chart can be configured to use a custom storage class or persistent volume by setting the `storage_provisioner` value to custom.

Available storage providers:

- **hostPath**: doesn't use any PVCs in the pod definitions and just mounts a path on the host starting with the hostPathPrefix
- **custom**: mounts from the PVC named as the storage_name value and uses component name as the subpath

Adding the Kubernetes yaml for the persistent volume claim in the `storage_configs` value and then using the pvc name in the `storage_name` value to tell the container where to mount to volume. Here's an example using managed storage in Azure:

Code · yaml

```
storage_configs:          # Example using a PVC with dynamic provisioning from an existing
storage class
  pvc-example:
```

```
    apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      name: md-core-store-data-pvc
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 1Gi
      storageClassName: managed-premium

core_components:
  md-core:
    extraVolumeMounts:
      - name: md-core-store-data
        mountPath: /var/lib/ometascan
    extraVolumes:
      configStore:
        name: md-core-store-data
        persistentVolumeClaim:
          claimName: md-core-store-data-pvc
```

Upgrade/Deployment Strategy on K8S

For upgrading MetaDefender Core the container of the application contains the automation to upgrade the database schema and migrate the configuration data. There are 3 strategies that we would like to point out to use when upgrading it.

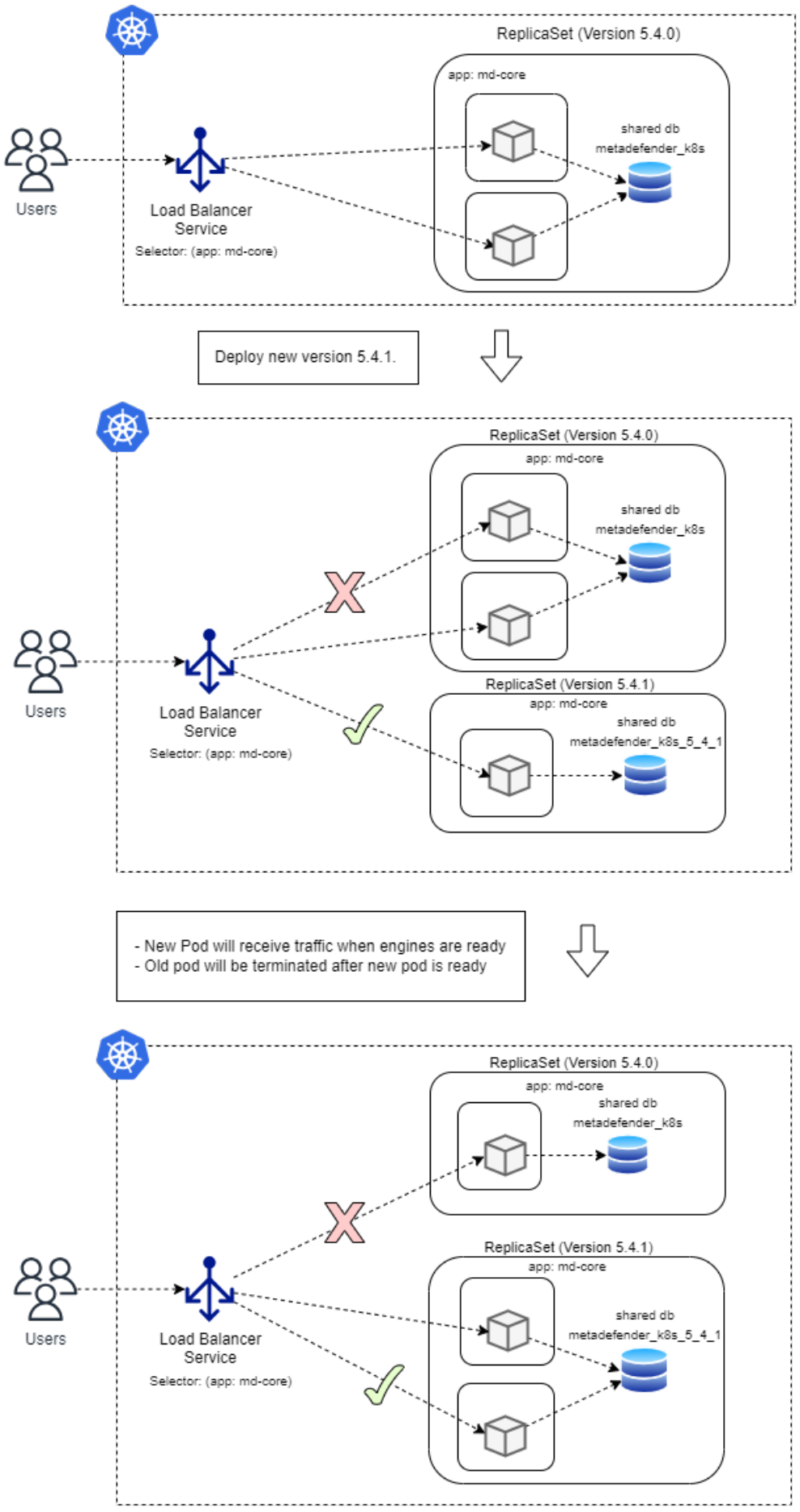
1. Rolling Update (Easy deployment where functionality test with new version is based on Kubernetes Probes)
2. *Blue-Green Deployment (2 or more activations licensed where is possible to test the new image before) - On RoadMap*
3. *Blue-Green Deployment + Hot Backup (When needed to go above the activations licensed) - On RoadMap*

As indicated in Deploying on K8S , the Kubernetes probes like Readiness and Liveness probes help the application to control the expected behavior, not to have a downtime, when doing an upgrade.

Rolling Update Strategy

When using the Rolling Update strategy provided by Kubernetes technology, these are the main points to consider

- Traffic wont be sent to new pods until readiness probe is successful
- Old version pods won't be terminated until new version pods are ready to serve traffic
- Rolling Update is the default strategy for deployment


ReplicaSet (Version 5.4.0)
app: md-core
shared db
metadefender_k8s
Users
Load Balancer
Service
Selector: (app: md-core)
Deploy new version 5.4.1.
ReplicaSet (Version 5.4.0)
app: md-core
shared db
metadefender_k8s
ReplicaSet (Version 5.4.1)
app: md-core
shared db
metadefender_k8s_5_4_1
Users
Load Balancer
Service
Selector: (app: md-core)
- New Pod will receive traffic when engines are ready
- Old pod will be terminated after new pod is ready
ReplicaSet (Version 5.4.0)
app: md-core
shared db
metadefender_k8s
ReplicaSet (Version 5.4.1)
app: md-core
shared db
metadefender_k8s_5_4_1
Users
Load Balancer
Service
Selector: (app: md-core)

How to upgrade using Rolling Update

Decide what data to migrate from version to version

When upgrading from one version to another, MetaDefender Core will create a new database migrating different data depending on what is indicated during the process. It can be migrated only configuration data or configuration data + processing history data.

- Configuration data:
  - Workflow
  - Health check
  - User Management
- Processing History Data
  - Scan results of previous scanned files

Pre-steps when migrating Processing History Data

Feature available starting from version 5.10.1

As the processing history data is the heaviest part of the data stored and is not always needed to be migrated, OPSWAT offers a way to only migrate the configuration data and save the time that it takes to migrate that sometimes can be high depending on the size of the processing history data.

There is a limitation when migrating the processing history, it is that is needed to stop the current pods running as we need to terminate all the connections to the database to migrate.

Following pre-steps are needed to follow when migrating the processing history

Pre-steps

- Must disable health check liveness probe in the helm chart before run with **MIGRATE_HISTORY** (Depending on the size of the database it can take a while to migrate all the data)
- Check that there is enough space in the database server before starting the migration
- Needed to stop MetaDefender Core pods

Configure Rolling Update Strategy

When using the Rolling Update strategy, there are two more options that let you fine-tune the update process:

- **maxSurge:** The number of pods that can be created above the desired amount of pods during an update
- **maxUnavailable:** The number of pods that can be unavailable during the update process

Both maxSurge and maxUnavailable can be specified as either an integer (e.g. 2) or a percentage (e.g. 50%), and they cannot both be zero.

To change this you need to edit the deployment strategy.

Code · bash

```
core_components:
    md-core:
    strategy:
      type: RollingUpdate
      rollingUpdate:
        maxSurge: 0
        maxUnavailable: 1
```

Variables to use during an upgrade

1. **MDCORE_UPGRADE _ FROM __*DB* NAME**: old database name to migrate from (1st installation metadefender_core, concurrent installations metadefender__core_<version>, _metadefender__core_5__17_0)
2. **UPGRADE_DB**: To indicate if we want to enable the DB upgrade to the new version. It will create a new database with database name metadefender_core<new___version>
3. **MIGRATE_HISTORY**: To migrate processing history. Follow Pre-steps

Step by step using kubectl

- Set Up the environment variables for the pod to know what data will be needed to migrate
  - For MIGRATE_HISTORY please see pre-steps above and decide if it is really needed or not

Code · bash

```
kubectl patch configmap mdcore-env -n <namespace> \
  --type merge \
  -p '{
    "data": {
      "MDCORE_UPGRADE_FROM_DB_NAME": "<CURRENT_DB_NAME>",
      "UPGRADE_DB": "true",
      "MIGRATE_HISTORY": "false"
    }
  }'
```

- Set new image version on the MetaDefender Core deployment

Code · bash

```
kubectl set image -n <namespace> deployments/md-core md-core=opswat/metadefendercore-
debian:<NEW_IMAGE_TAG>
```

Step by step using helm

- Set new image version in values.yaml helm file
  - For MIGRATE_HISTORY please see pre-steps above and decide if it is really needed or not

values.yaml · yaml

```
env:
    MDCORE_UPGRADE_FROM_DB_NAME: <CURRENT_DB_NAME>
  UPGRADE_DB: "true"
  MIGRATE_HISTORY: <true/false> ## Change to true to migrate processing history. It will delay
the upgrade depending on the database size. Needed to follow Pre-steps

core_components:
    md-core:
      image: opswat/metadefendercore-debian:<NEW_IMAGE_TAG>
```

- Run helm upgrade command with the environment variables desired if not indicated in values.yaml file when first deployment

Code · bash

```
helm upgrade mdcore mdcore/ --namespace <Namespace> -f <Additional_yaml_file> \
        --set core_ingress.enabled=$ingress_enabled \
        --set mdcore_license_key=<MDCORE_LICENSE_KEY> \
        --set deploy_with_core_db=<boolean_k8s_db_running_pod> \
        --set core_components.md-core.replicas=<replicas> \
        --set mdcore_password=<mdcore_password> \
        --set mdcore_user=<mdcore_user> \
        --set db_user=<db_user> \
        --set db_password=<db_password> \
        --set MDCORE_DB_HOST=<db_host>
```

Rollout Failed Update

Code · bash

```
kubectl rollout undo deployments/md-core -n <namespace>
```

How to upgrade with API Export/Import Configuration (Old-Fashioned way)

The steps needed to follow are:

1. Export the current configuration, like user management, worflows or settings configured. For it is needed to use the API endpoint Export Configuration (New version)

Code · bash

```
curl --request GET \
 --url 'http://localhost:8008/admin/export/v2' \
 --header 'apikey: {apikey}' \
 --header 'password: {password}' --output <filename>.zip
```

2. Adapt helm configuration to upgrade to the new image version

- In case of **controlling all the data from your values.yaml** file, be sure that the ***maxUnavailable*** and ***maxSurge*** attributes of the deployment strategy is adapted to your deployment use case and the ***image version*** is changed to the new one

Code · yaml

```
md-core:
    name: md-core
    image: opswat/metadefendercore-debian:<new_core_version> #Change <new_core_version> with any
tag in Docke Hub
    #....
    #....
    #....
    strategy:                           #Example for use case when having more than 1 pod
running
      type: RollingUpdate
      rollingUpdate:
        maxSurge: 0
        maxUnavailable: 1
```

Code · bash

```
helm upgrade mdcore mdcore/ -f <external_file_for_csp.yaml> \
      --set env.UPGRADE_DB=true \
      --set env.RETRY_UPGRADE=5
```

- In case of **controlling attributes from the helm install command,** be sure that the maxUnavailable and maxSurge attributes of the deployment strategy is adapted to your deployment use case and the image version is changed to the new one

Code · bash

```
helm upgrade mdcore mdcore/ -f <external_file_for_csp.yaml> \
      --set core_ingress.enabled=<ingress_boolean> \
      --set mdcore_license_key=<MD_Core_Lincese_Key> \
      --set deploy_with_core_db=<Postgres_Pod_Boolean> \
      --set core_components.md-core.replicas=<Number_of_replicas_Core> \
      --set db_user=<DB_USER> \
      --set db_password=<DB_PWD> \
```

```
    --set MDCORE_DB_HOST=<DB_HOST> \
    --set core_components.md-core.image="opswat/metadefendercore-debian:<Image_Version_TAG>" \
    --set core_components.md-core.strategy.rollingUpdate.maxUnavailable=1 \
    --set core_components.md-core.strategy.rollingUpdate.maxSurge=0 \
    --set env.UPGRADE_DB=true \
    --set env.RETRY_UPGRADE=5
```

3. **Import the configuration exported in the step 1** For it is needed to use the API endpoint Import Configuration (New version) . See how to use it in the example below.

Code · bash

```
curl --request POST \
 --url 'http://localhost:8008/admin/import/v2' \
 --header 'Content-Type: application/octet-stream' \
 --header 'apikey: {apikey}' \
 --header 'password: {password}' \
 --data-binary '@<file_name>.zip'
```

High Availability on Kubernetes

HA solutions for Postgres

In production environments it's recommended to use a managed database service from a cloud provider, Azure, AWS and Google cloud all provide Postgres compatible SQL services. These services have built-in HA and can scale dynamically depending on the required load.

If a highly available database is required inside the k8s cluster, then there are publicly available solutions that can deploy locally. One of the most popular is the Bitnami chart for HA PostgreSQL: https://bitnami.com/stack/postgresql-ha/helm and can be deployed using helm:

Code · bash

```
# Adding the Bitnami Helm repo
helm repo add bitnami https://charts.bitnami.com/bitnami

# Installing the chart with example values
helm upgrade --install postgresql-ha postgresql-ha \
    --wait \
    --set pgpool.authenticationMethod=scram-sha-256 \
    --set postgresql.image.tag=12.12.0 \
    --set postgresql.pgHbaTrustAll=true \
    --set global.pgpool.adminUsername=<SET_PGPOOL_USERNAME> \
    --set global.pgpool.adminPassword=<SET_PGPOOL_PASSWORD> \
    --set postgresql.password=<SET_POSTGRES_PASSWORD> \
    --set postgresql.postgresPassword=<SET_POSTGRES_PASSWORD> \
    --set postgresql.numInitChildren=50 \
    --set postgresql.maxConnections=100
```

In order to avoid performance bottlenecks, the following values have to be set in the PostgreSQL chart depending on the number of running MD Core pods using the same database. Example when running 3 MD Core pods:

Code · yaml

```yaml
postgresql:
    numInitChildren: 50     # set to about the number of MD Core Pods x 16
  maxConnections: 100     # set to about the value of numInitChildren plus 50
```

The following PGPOOL command has to be run manually the first time to add credentials to enhance authentication via PGPOOL. In case the PGPOOL k8s deployment restarts you must run the command again.

To make this setting persistent, a PV has to be mounted in `/opt/bitnami/pgpool/conf` in each Postgres pod.

Code · bash

```bash
# Add user and password for internal PG user with auth method scram-sha-256
pg_enc  -m -k /opt/bitnami/pgpool/conf/.pgpoolkey -f /opt/bitnami/pgpool/conf/pgpool.conf  -u
"<user>" "<passwd>"
```

HA deployment for MD Core

Multiple MD Core pods can be deployed by setting the `replicas`value in the`md-core` component:

Code · yaml

```yaml
core_components:
  md-core:
    replicas: 3
```

MD Core can also be set up with separate credentials for read/write operations. For example, the following command can be used to deploy MD core using a HA Postgres deployment as above:

Code · bash

```bash
helm install mdcore-deployment ./mdcore \
  --wait \
  --set db_password=<SET_POSTGRES_PASSWORD> \
  --set deploy_with_core_db=false   \
  --set core_components.md-core.replicas=3 \
  --set MDCORE_DB_HOST=<SET_POSTGRES_SERVICE_HOSTNAME> \
  --set env.MDCORE_DB_PRIVATE_USERNAME=<SET_PG_USERNAME> \
  --set env.MDCORE_DB_PRIVATE_PASSWORD=<SET_PG_PASSWORD>
```

Each MD Core pod used one activation on the given license and is automatically deactivated by the activation-manager sidecar when the pod is destroyed. It's advisable to have more activations available than the number of active pods in order to avoid activation issues when the number of pods might surge (like on a rolling update).

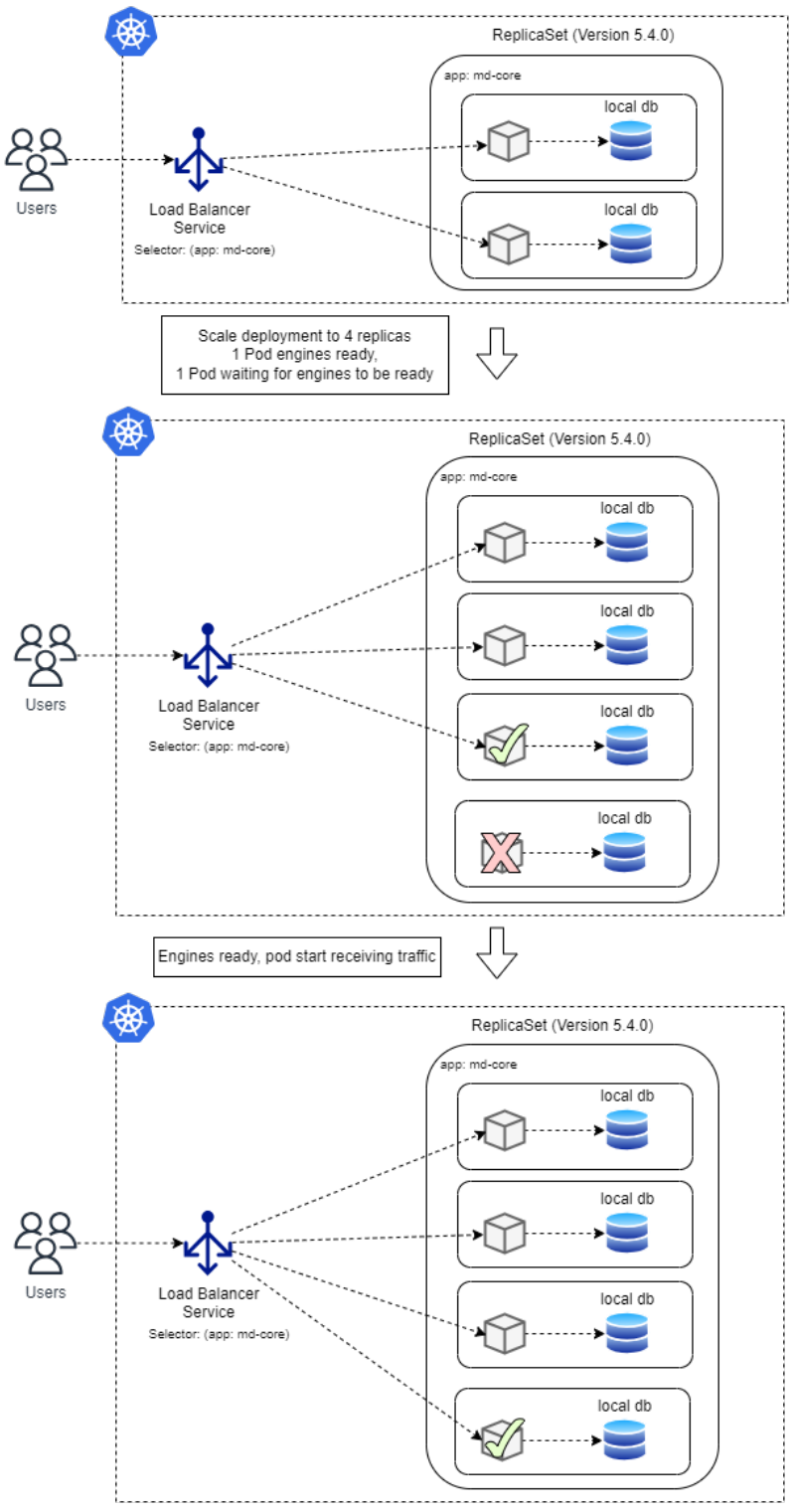
For redundancy, the pods can also be set to run on different nodes by using the `nodeSelector` or `nodeAffinity` Kubernetes features according to the cluster architecture.

Scaling K8S Cluster

Scaling up Cores

Flow Diagram

- Traffic won't be sent to pods until readiness probe is successful
  - Readiness probe is using health check API to check that engines are ready.
- If we scale it down, older pods will be terminated



Command to Scale up

Code · bash

```
kubectl scale deployment md-core --replicas=<num_replicas>
```

Autoscaling MetaDefender Core

There are different levels and ways to auto-scale an application when running in a Kubernetes cluster.

- When autoscaling the pods running in the Kubernetes cluster it is needed to run the Horizontal Pod AutoScaler
- To ensure that the Kubernetes cluster has enough nodes to schedule the pods to scale, it is recommended to use the Kubernetes Cluster AutoScaler.

Pre-requisites

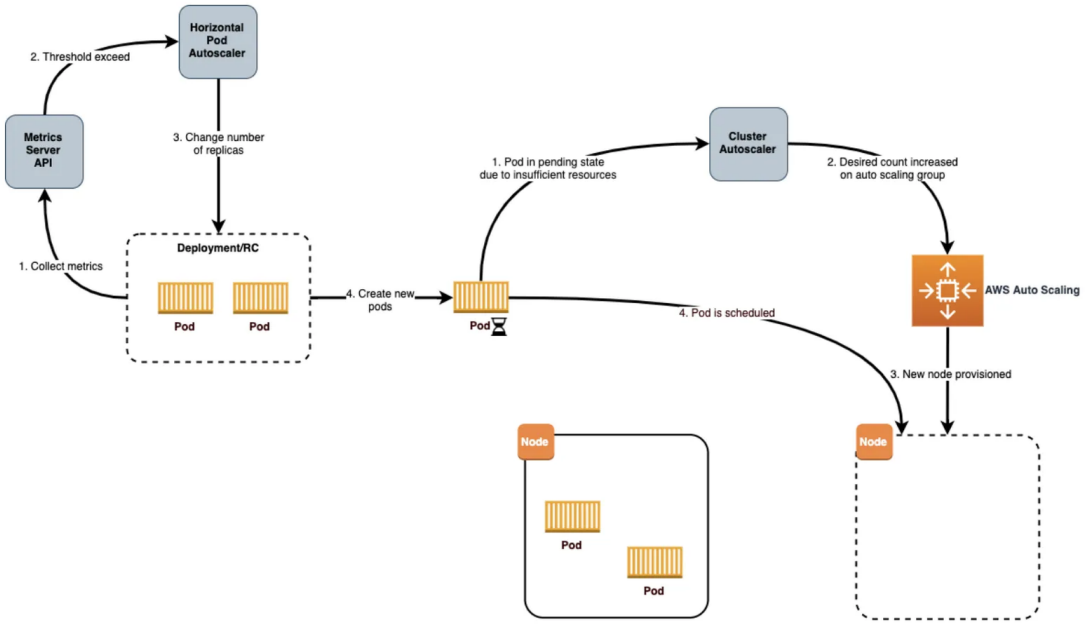
- Kubernetes Cluster already deployed
- Kubernetes Metrics Server installed - Installing the Kubernetes Metrics Server - Amazon EKS
- Kubectl client installed

Limitations

- Manual scale get disabled by the HPA

Autoscaling Architecture on K8S Cluster

In grey the components needed to be added to have a full autoscaling architecture with AWS as example.



Steps to install Horizontal Pod Autoscaler for Core

- Deploy HPA with kubectl

Code · bash

```
kubectl autoscale deployment md-core --cpu-percent=95 --min=1 --max=5
```

Steps to install Cluster Autoscaler (Manual Configuration)

Permissions

Cluster Autoscaler requires the ability to examine and modify EC2 Auto Scaling Groups. We recommend using IAM roles for Service Accounts to associate the Service Account that the Cluster Autoscaler Deployment runs as with an IAM role that is able to perform these functions

For that it is needed to attach the following policy __example_scripts/iam_policy_cluster_autoscaler.json to the service account with name `cluster-autoscaler`

One ASG Setup

Cluster Autoscaler can be configured manually if you wish by passing the `--nodes` argument at startup. The format of the argument is `--nodes=<min>:<max>:<asg-name>`, where `<min>` is the minimum number of nodes, `<max>` is the maximum number of nodes, and `<asg-name>` is the Auto Scaling Group name.

- Both `<min>` and `<max>` must be within the range of the minimum and maximum instance counts already specified by the Auto Scaling group.
- When manual configuration is used, the Auto Scaling groups must use EC2 instance types that provide equal CPU and memory capacity.

It is needed to replace the following variables in example_scripts/cluster-autoscaler-one-asg-aws.yaml before applying the changes <MIN_NODES>, <MAX_NODES> and <K8S_ASG_NAME>

Code · bash

```
kubectl apply -f example_scripts/cluster-autoscaler-one-asg-aws.yaml
```

Script provided by OPSWAT with all Steps

OPSWAT provides an script that automatically replace all the variables from the files needed to create HPA and Cluster Autoscaler

Code · bash

```
cd example_scripts
install_hpa_and_cluster_autoscaler_aws.sh --min <min_nodes> --max <max_nodes> --region
<asg_region> --cluster_name <cluster_name> --asg_name <k8s_asg_name> --account_id
<aws_account_id>"
```

Troubleshooting

To perform load testing, we recommend using the Gatling tool which you can find on https://github.com/OPSWAT

How to check the logs with Kubectl

- We can check the logs of the different containers running for the MetaDefender Core pod.
  - For init container "check-db" run `kubectl logs [-n <namespace> ] <pod-name> -c check-db-ready`
  - For app container "md-core" run `kubectl logs [-n <namespace> ] <pod-name>` (No container needed to be indicated)
- To describe the pod and see all the configurations of it.
  - `kubectl describe [ -n <namespace> ] <pod-name>`

How to collect logs generating a support package

For generating and exporting the logs we need to run the following commands

kubectl · bash

```
# Generate support package, it will log the support package name at the last line
(<sp_file_name>)
kubectl exec <mdcore_pod_name> -n <namespace> -- bash -c "cd /opt/ometascan/core_data/var/lib/
ometascan && /opt/ometascan/usr/bin/ometascan-collect-support-data.sh"

#export support package, replace <namespace>, <mdcore_pod_name> & <sp_file_name>
kubectl cp -n <namespace> <mdcore_pod_name>:opt/ometascan/core_data/var/lib/ometascan/
<sp_file_name>.tar.gz <sp_file_name>.tar.gz
```

Installation Script is failing

- Follow pre-install requirements in Installing MetaDefender Core in K8S
- Message Error "INSTALLATION FAILED: found in Chart.yaml, but missing in charts/ directory: aws-load-balancer-controller".
  - Run `helm dependency update`

Nginx Ingress Issues

Problems found:

1. Tls is not enabled for ingress
2. Controller is not picking up the changes of the services

TLS not enabled

How to debug

- Check ingress rule deployed and name of the tls secret that contains the certificate and key

Solution Proposed

- Create tls secret if not created and add the secret name matching the tls secret created. See TLS configuration

Ingress Controller not picking up the changes

How to debug

- Check the logs of the ingress controller pod. Search for any error or warning that could point to the root cause

Solution Proposed

- If error is not clear try to find in the ingress controller user guide to review the configuration.

Errors Connecting to Database

Problems found

1. MetaDefender Core cannot connect to database
2. PostgreSQL extenstion is not installed

Cannot connect to database

How to debug

- MetaDefender Core init container is not finishing. Check logs of check-db init container to see if the database if reachable.
- If MetaDefender Core init container is terminated means that the database could be reached with the configuration provided. In this case it is needed to check the MetaDefender Core app container and search for 'connection error'

Solutions Proposed

- Check **DB_HOST** in ***mdcore-env*** configmap
- Check user and password values in ***mdcore-postgres-cred*** secret

PostgreSQL extension not installed

How to debug

- Check logs of MetaDefender Core pod and search for éxtension missing' message.

Solutions Proposed

- There are some extensions that are needed to be installed, sometimes they are installed by the admin user that MetaDefender Core use for connecting to the database but other times there are some database services in the CSPs that does not allow this kind of actions and the extensions have to be installed from the CSP console.

Error Activating MetaDefender Core license

Problems found

1. MetaDefender Core pod does not get activated

MetaDefender Core does not get activated

How to debug

- MetaDefender Core pod is restarting as it does not get activated because of the livenessprobe is failing. By default the health check endpoint /readyz is checking if the pod is activated, so if this fails the pod will get restarted.
- It is needed to review the MetaDefender Core pod logs to see what is the specific error, it could be things like
  - Not enough activations left
  - Activation Key expired

Solutions Proposed

1. Each activation key has a number of slots/Core that can be activated and generate a deployment id, if your reach the limit could be because either any of the deployments was not deactivated properly or because you need to get more Core licensed in your activation key
   1. Check the sidecar container *activation-manager* logs to see if there is any error on deactivation
   2. Contact support if there is any error in the logs, or you need to extend the number of slots/Core
2. In case it is saying that the activation key is expired, please contact Sales to extend your license

Activate license key failed even though the import zip config file was successful

Problems found

1. Discrepancy between the **APIKEY** environment variable on K8S and the **API key** stored in the zip config file

Discrepancy between APIKEYs

How to Debug

- Check if the API Key is protected (Feature available from version 5.9.0, For more details see link)

Solution Proposed

- Case 1: If user does not enable **API key protection**
  - Step 1: Remove the current helm chart and re-deploy it again with the correct APIKEY env var (**plain text API key** stored in the zip config file)
- Case 2: If user enabled **API key protection** and user does not remember **plain text API key**

- Step 1: Ignore Health Checks (livenessProbe, readinessProbe) for md-core deployment via kubectl command

Code · bash

```
kubectl patch deployment md-core --type json -p='[{"op": "remove", "path": "/spec/template/spec/containers/0/livenessProbe"}]'
kubectl patch deployment md-core --type json -p='[{"op": "remove", "path": "/spec/template/spec/containers/0/readinessProbe"}]'
```

- Step 2: Login to the Core UI, adjust settings that users want (workflow, user, other settings, etc), then generate a **new apikey** and export **new zip config file** - Step 3: Copy the **new zip config file** from the client to the NFS share folder - Step 4: Re-deploy helm chart again with correct APIKEY environment variable (newly generated API key in the Core UI) - Step 5: Restart K8S md-core deployment via kubectl command

Code · bash

```
kubectl rollout restart deploy md-core
```

MetaDefender Core Engines Errors

Problems found

1. Engines does not get activated

Engines does not get activated

How to debug

- Check, in the MetaDefender Core console, the configuration under Inventory > Modules > Metascan. There would be an error indicating that the engines are failing to be activated or installed.
- Check the logs of the MetaDefender Core pod and search for errors when installing the engines. "Waiting for init message timeout, no response from engine"

Solutions Proposed

- Review the resources allocated to the MetaDefender Core pod. The limit resources settings in K8S must meet the recommended system in the documentation (for more details: link)

IKARUS engine

For IKARUS, `--shm-size=1gb` should be added when starting the docker.

If you're using K8S, the section below should be also specified in the yaml file:

Code · yaml

```
spec:
```

```
  volumes:
  - name: dshm
    emptyDir:
      medium: Memory
  containers:
  - volumeMounts:
      - mountPath: /dev/shm
        name: dshm
```

Xcitium engine

For Xcitium, `--shm-size=300m` should be added when starting the docker.

If you're using K8S, the section below should be also specified in the yaml file:

Code · yaml

```
spec:
  volumes:
  - name: dshm
    emptyDir:
      medium: Memory
  containers:
  - volumeMounts:
      - mountPath: /dev/shm
        name: dshm
```

Notice

If you are using both IKARUS and Xcitium, please use `--shm-size=1300m` to satisfy the requirements of both engines.

Unable to deploy MetaDefender Core in Kubernetes with containerd engine 2.2.x

Summary

We have identified an issue affecting our application deployments in Kubernetes environments using **containerd 2.2.0-2.2.2**. This behavior is caused by a known bug in the container runtime and is not related to changes in our application.

Issue Description

A known bug in **containerd** version **2.2.0-2.2.2** prevents containers from being successfully created in certain Kubernetes environments. As a result, Pods may fail during startup and remain in a non-running state.

The following is an example when deploying to EKS 1.35 (the log is redacted)

Code · bash

```
Events:
  Type     Reason            Age               From               Message
  ----     ------            ----              ----               -------
  Normal   Scheduled         82s               default-scheduler  Successfully assigned default/
md-core-7fb8c84585-vk2gt to ip-172-31-5-122.ec2.internal
  Normal   Created           76s               kubelet            Container created
  Normal   Started           75s               kubelet            Container started
  Warning  Failed            70s               kubelet            Error: failed to create
containerd container: mount callback failed on /var/lib/containerd/tmpmounts/containerd-
mount1122317056: openat xxx: path escapes from parent
  Warning  Failed            60s               kubelet            Error: failed to create
containerd container: mount callback failed on /var/lib/containerd/tmpmounts/containerd-
mount3050653287: openat xxx: path escapes from parent
  Warning  Failed            48s               kubelet            Error: failed to create
containerd container: mount callback failed on /var/lib/containerd/tmpmounts/containerd-
mount2061916826: openat xxx: path escapes from parent
  Warning  Failed            33s               kubelet            Error: failed to create
containerd container: mount callback failed on /var/lib/containerd/tmpmounts/containerd-
mount2702648643: openat xxx: path escapes from parent
  Warning  Failed            18s               kubelet            Error: failed to create
containerd container: mount callback failed on /var/lib/containerd/tmpmounts/containerd-
mount1893797964: openat xxx: path escapes from parent
  Normal   Pulled            6s (x6 over 70s)  kubelet            Container image "opswat/
metadefendercore-debian:5.17.1" already present on machine and can be accessed by the pod
  Warning  Failed            6s                kubelet            Error: failed to create
containerd container: mount callback failed on /var/lib/containerd/tmpmounts/containerd-
mount3629785830: openat xxx: path escapes from parent
```

Reference: https://github.com/containerd/containerd/issues/12683

Affected Environments:

- ** Kubernetes 1.35 or **containerd 2.2.0-2.2.2**

Workaround

Until the vendor provides a fix, use one of the following mitigations:

- Use a Kubernetes node image that does not include **containerd 2.2.3 and later**
- Pin node runtime version in cluster provisioning

Overview

Starting with MetaDefender Core 4.21.0, we will support you to deploy and run the product on the docker containerization environment, comes with various options:

1. OPSWAT docker images hosted on our docker hub repository. The docker images are bundled with MetaDefender Core latest official release builds Docker image published on OPSWAT Docker Hub.
2. Provide a build toolkit (with docker file and licensing auto deactivation scripts), use them as a deployment support with your own docker image Using build tool kit with your own docker image.

Info

Supported platforms

- Rocky / RedHat
- Debian / Ubuntu

Docker image published on OPSWAT Docker Hub

Info

OPSWAT publish all official **public** docker images on Docker Hub:

`opswat/metadefendercore-<os-type>:<version>`

The docker images are all bundled with the official release MetaDefender Core.

Note:

To ensure better tracking of the running image version, it is advisable for users to not use the `:latest` tag when deploying containers in production.

Instead, it is recommended to specify a meaningful tag such as `5.13.2`.

More information:

https://hub.docker.com/r/opswat/metadefendercore-debian

https://hub.docker.com/r/opswat/metadefendercore-rocky-nonroot

Pull from the OPSWAT Docker Hub repository

pull image · bash

```
docker pull <repository>/metadefendercore-<platform>[:<version>]
```

- `<repository>` - OPSWAT repository address
- `<platform>` - can be `debian` , `rocky`
- `<version>` - desired Core version (optional, default is `latest`)

Example:

pull debian · bash

```
docker pull opswat/metadefendercore-debian:latest
```

Run MetaDefender Core Docker image

docker run cmd · bash

```
docker run -d [--name <container_name>] \
  [-e "<env_var>=<value>"] \
  [-v <ignition_folder>:<container_ignition_folder>] \
  [-v <host_folder>:<container_folder>] \
  [-u <user ID>] \
  -p <rest_port>:8008 <image_name>
```

IKARUS engine

For IKARUS, `--shm-size=1gb` should be added when starting the docker.

If you're using Kubernetes (K8S), the section below should be specified in the yaml file:



[Parameter] Container Name

Argument: `--name <container_name>`

Description: Your container's name

Example: `--name mdcore01`

[Parameter] Init Details (Environmental Variables & Ignition File)

Argument: `-v <ignition_folder>:<container_ignition_folder> -e "<env_var>=<value>"`

Description:

You must configure MetaDefender Core (default local admin account, database connection etc.) before running MetaDefender Core docker image. It could be done via either one of following options **(do not use all options**, otherwise the ignition file settings will be ignore and overwritten):

1. Using ignition file (`-v`)

2. Using environmental variables (`-e`)

3. Using `IGNITION_JSON`

Option 1: Using environmental variables

`-e "<env_var>=<value>"` : set an environmental variable to configure, each environmental variable need one -e argument.

Available environmental variables:

Environmental variable	Necessity	
DB_MODE	required	database mode
DB_TYPE	required	database type
DB_HOST	required	database host
DB_PORT	required	database port
DB_USER	required	database user
DB_PWD	required	database password Support some specific ch `;>:{}[]()*&^%$#@~` Not support `"` (double qu (quotation).
MD_INSTANCE_NAME	optional	instance name Note: For deployments us database type and multip assigning the same instar
MD_USER	optional	username to create the fi
MD_PWD	optional	password to create the fi
MD_EMAIL	optional	email to create the first ac
APIKEY	optional	the API key will be assign user for license auto deac HTTPS enabling, proxy en registration, health check Splunk registration. Requirements: • Must be exactly 36 cha

Environmental variable	Necessity	
		• Can contain only lower characters [0-9],[a-f]. • Must include at least 1 [a-f]. • Must include at least 1 (0-9]. • Must not contain more lowercase letters. • Must not contain more numeric digits. Note: At the moment, the accepts the simplified AP However, in future release API key does not comply security requirements, the to start during the initializ
LICENSE_KEY	optional	an license key for license
STORAGE_PATH	optional	a full path of storage for s processed, quarantined fi Forensic
DATA_DIR	optional	a full path of MetaDefend data directory
REST_ADDRESS	optional	REST binding address for Core's Nginx to be allowe
REST_PORT	optional	REST binding port for Met Nginx to be allowed
IGNITION_JSON	optional	The ignition file settings, is accepted. For example: `IGNITION_J e": "admin", "user/pa ", "user/email": "adm` For database password, t specific characters: `*/-+ *&^%$#@~`;.,`

Environmental variable	Necessity	
		Not support " (double quo (quotation).
CORE_CONF_JSON	optional	MetaDefender Core confi settings, only JSON forma For example: CORE_CONF_ restport": "8009", "l : "info"}'
CERT_PATH	optional	A full path to folder conta and private key file. There files must have the meanwhile its extensions and **.key** After being added, the file extension will be the nam in MetaDefender Core
MDCORE_CERT_PASSPHRASE	optional	A passphrase for encrypt when adding new certifica environment variable. Passphrase must be less characters Passphrase does not sup character '
MDCORE_SSL_PROTOCOLS	optional	Specify the TLS version. A (case-insensitive): TLSv1
MDCORE_SSL_ADVANCED_CONFIGS	optional	Configure advanced SSL s cipher suites, session tim Accepted values (case-ins ssl_verify_depth 10; he none; *In case of usin compose, they should not ' and double quote "
MDCORE_DB_NAME	optional	Used for customizing the database name of MetaD By default, in shared DB n

Environmental variable	Necessity	
		`ender_core` Name must l (`a-z`) or underscore (`_`). Su characters in a name can (`0-9`), or underscores. Max
MDCORE_DB_PRIVATE_USERNAME	optional	PostgreSQL server's inter MetaDefender Core own c purpose. If not specified, Core will auto generate th Customize Internal Postg
MDCORE_DB_PRIVATE_PASSWORD	optional	PostgreSQL server's inter created for MetaDefender operational purpose. If no MetaDefender Core will a user. Details: Customize I User
PROXY	optional	**Default**: false If the user wants to use p
PROXY_HOST	optional	Default: empty string This PROXY is set to **true**
PROXY_PORT	optional	Default: empty string This PROXY is set to **true**
PROXY_USER	optional	Username for proxy authe
PROXY_PWD	optional	Password for proxy authe
PROXY_EXCLUSION	optional	List of IPs to be added to
MDCORE_HEALTH_CHECK	optional	**Note:** This environment va accepts boolean values (t Any other value or format **true** • Enables the health chec • Applies up to three cus check criteria, if config

Environmental variable	Necessity	

		false
		• Disables the health che
		• Ignores all customized criteria.
		`not defined`
		• Enables the health chec default settings:
		  • Queue threshold**:**
		  • Minimum number of
		  • Required engine list:
		  • Disable the threshold required engines
		  • Disables disk space validation.

MDCORE_QUEUE_THRESHOLD	optional	Health check criteria. Threshold for scan queue 100 (%). When allocated slots perc the threshold, MetaDefen itself unhealthy.
MDCORE_MINIMUM_NUMBER_AV	optional	Health check criteria. Minimum number of AV e When number of active A than or equal to the prede MetaDefender Core will s
MDCORE_REQUIRED_ENGINES	optional	Health check criteria. A list of mandatory engin engines of a license. Whe engines are not active, Me will set itself unhealthy.
MDCORE_REQUIRED_ENGINES_THRESHOLD	optional	Use this setting to define for required engines.

Environmental variable	Necessity	

		Format
		Provide the value in the fo format:
		`[{ "id": "<engine ble": <true|false>, <number_of_days> }]`
		Example
		`[{ "id": "clamav" true, "threshold" "id": "avira", "e "threshold": 5 }]`
		If this setting is **undefined** Health Check feature trea validation as **disabled** for engines.
		Important: If the engines `RE_REQUIRED_ENGINES` v set up the health check

MDCORE_DISK_SPACE_THRESHOLD	optional	Use this setting to define policy for disk space If **unset** or **empty**, Health value as `0`, which disables threshold validation. To configure a threshold, value representing the pe
UPGRADE_DB	optional	**true** • Enable database upgra migrates Workflow Rul settings. • Require `MDCORE_UPGRA E` variable. • Require super-user per `R`. • Only support shared-da `ODE=4` and remote Post `emote` .

Environmental variable	Necessity	
		false / not defined • Database upgrade proc
MDCORE_UPGRADE_FROM_DB_NAME	optional	**Required** when UPGRADE If this variable does not e process will exit with cod
MIGRATE_HISTORY	optional	**Required** when UPGRADE **true** • Database upgrade proc processing history, qua module update and cor **false / not defined** • Database upgrade proc migrate these kinds of
MDCORE_LICENSE_COMMENT	optional	**Default**: empty string Add the comment for the activating.
OLMS_HOST_URL	optional	The OLMS server endpoir use OLMS, this variable is MetaDefender Core will tr priority than the normal w activation.
OLMS_TOKEN	optional	Token created from OLMS want to use OLMS, this va
OLMS_RULE	optional	Rule containing licenses f If users want to use OLMS required.
OLMS_REST_PORT	optional	**Default:** 8040 OLMS port for REST API.
OLMS_SOCKET_PORT	optional	**Default**: 3316

Environmental variable	Necessity	
		OLMS port for socket inte
OLMS_CONNECTION_TYPE	optional	**Default:** `production` Can be `production` or `ho`
OLMS_USE_PROXY	optional	**Default**: false If the user want to use pro activation
OLMS_PROXY_SERVER	optional	**Default**: empty string Proxy server address This field is required if OL is set to **true**
OLMS_PROXY_PORT	optional	**Default**: empty string Proxy port This field is required if OL is set to **true**
OLMS_PROXY_PROXY_TYPE	optional	**Default**: empty string Proxy type. Accepted valu • socks4 • socks5 • http This field is required if OL is set to **true**
OLMS_PROXY_USERNAME	optional	**Default**: empty string Proxy username
OLMS_PROXY_PASSWORD	optional	**Default**: empty string Proxy password
MDCORE_CONFIG___IMPORT_FILE	optional	Absolute path inside the c configuration `.zip` to imp The user should mount a

Environmental variable	Necessity	
		exported file to a containe Leave unset/empty to dis
MDCORE_CONFIG_IMPORT_PASSWORD	optional	Password used when the exported. Leave unset/em had no password.
MDCORE_IMPORT_FORCE_UPDATE_YARA	optional	When `true`, force-update import. It is required to be config file contained Yara Any value other than `true` `alse`.
OCM_URL	optional	**Default**: empty string The OCM endpoint URL th enroll into. Notes: • OCM traffic via a proxy • Unenrollment only trigg shutdown • If OCM is provisioned t bootstrap config (**IGNIT OCM_URL** / **OCM_REG_** config enrolls the insta boot, then the env-drive Core rejects the second **(Warning: OCM enrollm Response (500)**. This is Recommendation: prov one channel only
OCM_REG_CODE	optional	**Default**: empty string The OCM registration cod
SPLUNK_SERVER	optional	Specifies the base URL of that receives log or event must include the protocol `/`) and the port number. Example: `tcp://splunk.`

Environmental variable	Necessity	
		`4` **Default**: empty string
SPLUNK_LOG_LEVEL	optional	Defines the minimum sev messages forwarded to S Supported values are: • `INFO` • `EXTINFO` • `DEBUG` • `WARNING` • `ERROR` **Default**: `INFO`
SPLUNK_ALLOW_SELF_SIGNED_CERT	optional	Controls whether the appl signed SSL/TLS certificat the Splunk server. Supported values are: • `true` • `false` **Default**: false

Configurations overriding

The priority for overriding configs is:

JSON environmental variable (IGNITION_JSON, CORE_CONF_JSON) > single environmetal variable

For example, the following command will start a container with `global/restport=8009`, and ignore `REST_PORT=8010`:

```
docker run -it --name core -p 8009:8009 \
-e DB_MODE=4 \
-e DB_TYPE=remote \
-e DB_HOST=192.168.0.1 \
-e DB_PORT=5432 \
-e DB_USER=postgres \
-e DB_PWD=admin \
```

```
-e REST_PORT=8010 \
-e IGNITION_JSON='{"user/name": "admin", "user/password": "admin", "user/email": "admin@local"}' \
-e CORE_CONF_JSON='{"global/restport": "8009", "logger/loglevel": "info"}' \
opswat/metadefendercore-debian:latest
```

Warning

If missing any of the required environmental variables, docker run **will return failed** with error message `<ENV> is missing. Please set <ENV> before running container.`

Warning

When deploying in a Kubernetes environment with multiple replicas and a remote database, avoid setting the same instance name across all Pods. Specifically, be cautious when using the `MD_INSTANCE_NAME` environment variable or the `global/instance_name` field in `CORE_CONF_JSON`. Using a shared instance name will cause all Pods to use the same database private user. If any Pod restarts, this private user's password will be regenerated, resulting in the other Pods becoming non-functional.

Option 2: Using ignition file

`-v <ignition_folder>:<container_ignition_folder>` - (optional) mounting the folder containing the ignition file to the container's folder

- `<ignition_folder>` - ignition folder path containing the ignition file `<ignition_folder>/ometascan.conf`
  - Deployment automation support
- `<container_ignition_folder>` - container's folder to be mounted to `/opt/ometascan/core_data/opswat` (by default)

Info

The `internal/data_directory` parameter is **not** supported in the `ometascan.conf` ignition file when using containerized deployments. Any configuration of this parameter will be ignored.

Example

Setup a PostgreSQL version 14 server. After that, we assume that remote PostgreSQL server's information is

- host = **192.168.0.1**
- port = **5432**
- user = **postgres**

- password = **admin**
- type = **remote**

In case of local PostgreSQL:

- host = **localhost**
- type = **local**

Option 1 - using environmental variables

env vars · bash

```
mkdir /ometascan-config
docker run -d --name core01 \
  -e APIKEY=9f54e1b62d19a889d718cb3d9db145a4b8bb \
  -e LICENSE_KEY=... \
  -e DB_MODE=4 \
  -e DB_TYPE=remote \
  -e DB_HOST=192.168.0.1 \
  -e DB_PORT=5432 \
  -e DB_USER=postgres \
  -e DB_PWD=123 \
  -e MD_USER=localadmin \
  -e MD_PWD=password \
  -e MD_EMAIL=admin@local \
  -e STORAGE_PATH=/metadefendercore \
  -e IGNITION_JSON='{"user/name": "admin", "user/password": "admin", "user/email":
"admin@local"}' \
  -e CORE_CONF_JSON='{"global/restport": "8008", "logger/loglevel": "info"}' \
  -e CORE_DATA_PATH=/home/core_data_dir \
  -e CERT_PATH=/home/cert \
  -p 8008:8008 opswat/metadefendercore-debian:latest
```

Option 2 - using ignition file

ignition file · bash

```
mkdir /ignition_folder
touch /ignition_folder/ometascan.conf

# Create /ignition_folder/ometascan.conf based on
# https://onlinehelp.opswat.com/corev4/10.2._Deployment_automation_support.html
echo "[global]" >> /ignition_folder/ometascan.conf
echo "dbmode=1" >> /ignition_folder/ometascan.conf

echo "[dbserver]" >> /ignition_folder/ometascan.conf
echo "type = remote" >> /ignition_folder/ometascan.conf
echo "host = 10.40.50.194" >> /ignition_folder/ometascan.conf
echo "port = 5432" >> /ignition_folder/ometascan.conf
```

```
echo "user = postgres" >> /ignition_folder/ometascan.conf
echo "password = 123" >> /ignition_folder/ometascan.conf

echo "[user]" >> /ignition_folder/ometascan.conf
echo "name = localadmin" >> /ignition_folder/ometascan.conf
echo "password = password" >> /ignition_folder/ometascan.conf
echo "email = admin@local" >> /ignition_folder/ometascan.conf

docker run -d --name core01 \
  -e "APIKEY=9f54e1b62d19a889d718cb3d9db145a4b8bb" \
  -e "LICENSE_KEY=..." \
  -v /ignition_folder:/opt/ometascan/core_data/opswat \
  -p 8008:8008 opswat/metadefendercore-debian:latest
```

Option 3 - using ignition file with a full configuration package (.zip)

Note:

a. When using Local Update Folder, let's make sure *pickupfolder* (*Pick up update from*) in the settings_export.json is correct.

- Format: `<DATA_DIR>/path/to/update-folder`
- For example:
  - `"pickupfolder": "/opt/ometascan/core_`data/var/lib/ometascan/update_`autoadd"`

b. If you want to activate license or enable HTTPS during container starting, please make sure apikey of an administrator in .zip package matches with

- apikey under [user] section (if present) in ignition file
- or `APIKEY` environment variable (if present)

ignition file · bash

```
mkdir /ignition_folder
touch /ignition_folder/ometascan.conf

# Create /ignition_folder/ometascan.conf based on
# https://onlinehelp.opswat.com/corev4/10.2._Deployment_automation_support.html
echo "[global]" >> /ignition_folder/ometascan.conf
echo "dbmode=1" >> /ignition_folder/ometascan.conf

echo "[dbserver]" >> /ignition_folder/ometascan.conf
echo "type = remote" >> /ignition_folder/ometascan.conf
echo "host = 10.40.50.194" >> /ignition_folder/ometascan.conf
echo "port = 5432" >> /ignition_folder/ometascan.conf
echo "user = postgres" >> /ignition_folder/ometascan.conf
echo "password = 123" >> /ignition_folder/ometascan.conf

echo "[user]" >> /ignition_folder/ometascan.conf
```

```
echo "name = localadmin" >> /ignition_folder/ometascan.conf
echo "password = password" >> /ignition_folder/ometascan.conf
echo "email = admin@local" >> /ignition_folder/ometascan.conf

echo "[config]" >> /ignition_folder/ometascan.conf
# Path to a file in json/zip format that contains a previously exported configuration to be
imported
# Make sure the file path of zip to be mounted. For example, -v /path/to/:/path/to/
echo "import = /path/to/settings_export_package.zip" >> /ignition_folder/ometascan.conf
# Password to decrypt .zip configuration package.
# This can be omitted if using json file
echo "import_password = password" >> /ignition_folder/ometascan.conf

docker run -d --name core01 \
  -e "APIKEY=9f54e1b62d19a889d718cb3d9db145a4b8bb" \
  -e "LICENSE_KEY=..." \
  -v /ignition_folder:/opt/ometascan/core_data/opswat \
  -p 8008:8008 opswat/metadefendercore-debian:latest
```

[Parameter] User

Argument: `-u <user ID>`

Description: [Optional] run container with any user ID (UID) or group ID (GID). Range: [1000, 65533]

If not specified, the container will run as `root` user. Otherwise, a user will be used for `non-root` option.

Example: `-u 1011`

> **When running container with non-root user**
>
> When running container with non-root user (`-u <user ID>` param), you are expected to grant write permission to the **_ignition folder** _and **_ignition file**_for a user with the respectively uid on the host machine, before running the container. There are 2 ways to do this:
>
> 1. Grant write permission for others (e.g. chmod 777)
> 2. Change owner to uid 1011 or gid 1011 (e.g. chown 1011:1011), then grant write permission for owner or group owner (e.g.: chmod 755)

[Parameter] Host Folder

Argument: `-v <host_folder>:<container_folder>`

Description: [Optional] mounting any additional folder from your host to the container (for managing config file, persistent data, storing log)

Example: `-v /var/mdcore:/temp`

[Parameter] REST Port

Argument: `-p <rest_port>:<internal_port>`

Description: Port on the host machine to map with Core REST and web console management port

Example: `-p 30808:8008`

[Parameter] Image Name

Argument: `<image_name>`

Description: MetaDefender Core docker image was built/pulled earlier

Example: `mdcore:debian`

Start / Stop container

start container · bash

```
docker start <container_name>
```

docker stop · bash

```
docker stop -t <timeout> <container_name>
```

- `<container_name>` - container's name for using later
- `<timeout>` - timeout in second before Docker forces stop the container, this can result in lost data and restart the container abnormally, so this need to be set to long timeout, minimum 30 is recommended.

Example:

docker stop · bash

```
docker stop -t 30 core01
docker start core01
```

Access to view MetaDefender Core logs

get logs · bash

```
docker logs [-f] [-n <number_of_lines>] <container_name>
```

- `<container_name>` - container's name for using later
- `[-f]` - keep following the log
- `[-n <number_of_lines>]` - number of last lines existed in log file will be shown instantly when running this command.

Example:

tail logs · bash

```
docker logs -f -n 100 core01
```

Collecting Logs by Generating a Support Package

Bash · bash

```
#Step 1: Run the following command to generate support package file
docker exec <container_name> /bin/bash -c "cd /opt/ometascan/core_data && /opt/ometascan/usr/
bin/ometascan-collect-support-data.sh"

#Step 2: Copy the file from container to host machine
docker cp <container_name>:/opt/ometascan/core_data/<support-package-file>.tar.gz
<target_folder>
```

- `<container_name>`: The name of the container.
- `<support-package-file>.tar.gz`: The generated support package file. You can find the file name in the console output after executing the command in Step 1.
- `<target_folder>`: The destination folder on the host machine where the support package file will be saved.

Example:

Bash · bash

```
docker exec mdcore /bin/bash -c "cd /opt/ometascan/core_data && /opt/ometascan/usr/bin/
ometascan-collect-support-data.sh"

#output: Support file created: ometascan-5.14.1-1-support-1666784477.tar.gz

docker cp mdcore:/opt/ometascan/core_data/ometascan-5.14.1-1-support-1666784477.tar.gz ./
```

Configure MetaDefender settings

You can modify the `/etc/ometascan/ometascan.conf` file. It can be modified in 2 ways:

- Directly inside the container by using docker exec to go inside the container and modify
- Mount the `/etc/ometascan` folder to host when starting the container, then modify the file on host folder

Database Maintenance

Bash · bash

```
#Run the following command
```

```
docker exec <container_name> /bin/bash -c "/opt/ometascan/usr/sbin/ometascan-vacuum-db"

#Example output:
Running vacuum with config file: /opt/ometascan/core_data/ometascan.conf
Postgres info: Postgres private user is not defined,
Postgres info: Database name, metadefender_core_brxnpu
Postgres info: Postgres private password is not defined,
Postgres info: Generating private password for the first time, metadefender_core_brxnpu
Postgres info: PG_DATA path in dat file = /opt/ometascan/core_data/var/lib/ometascan/pg_data,
Postgres info: PG_DATA path in conf file = /opt/ometascan/core_data/var/lib/ometascan/pg_data,
Postgres info: Same DIR,
Postgres info: The service and the config are mismatched, re-config the service..., /opt/
ometascan/core_data/var/lib/ometascan/pg_data
Postgres info: PG_DATA path in dat file = /opt/ometascan/core_data/var/lib/ometascan/pg_data,
Postgres info: PG_DATA path in conf file = /opt/ometascan/core_data/var/lib/ometascan/pg_data,
Postgres info: Saving /var/tmp/ometascan/ometascan-pg.init,
Postgres info: Replacing /etc/init.d/ometascan-pg,
Postgres info: Checking for database upgrade,
Postgres info: DBMode is standalone, checking for deploymentId,
Postgres info: Upgrade database status: 0,
Postgres service init done
Creating connection with: localhost
Creating connection done
Created out dir: /opt/ometascan/core_data/analyze_and_cleanup_database
Working dir: /opt/ometascan/usr/lib/ometascan/postgres
Running vacuum DONE
```

- `<container_name>`: The name of the container.

Reference: Database Maintenance | MetaDefender Core 4.19.0 or above

Using build tool kit with your own docker image

> **Info**
>
> The docker build tool kit could be found on OPSWAT Portal - Products - Utilities
>
> https://my.opswat.com/home/metadefender-core

The build tool kit includes following files:

- `Dockerfile` - main script for building docker image.
- `entrypoint.sh` - startup script that runs when the container starts.
- `patch.sh` - utility script for string replacing in file, needed for building docker image.
- `systemctl.py` the replacement for systemctl, needed for non-privileged container.
- `mdcore.sh` - functionalities that `entrypoint.sh` need to start MetaDefender Core service.

To build the image, just run the following command:

build cmd · bash

```
docker build -t <image_name[:tag]> [--build-arg VERSION=<version>] <build-kit_path>
```

- `<image_name[:tag]>` - name of your built image with optional tag (e.g. `latest`).
- `--build-arg VERSION=<version>`- the desired MetaDefender Core version to be downloaded from OPSWAT. If obmitted then a local installer will be used, installer should be placed in working directory with name is formatted /mdcore/ometascan-*.rpm or .deb (configurable in the Dockerfile).
- `<build-kit_path>` - relative path to your build-kit which contains Dockerfile.

For example:

build cmd · bash

```
docker build -t core:debian --build-arg VERSION=5.13.2 .
```

In Dockerfile, there are some environment variables that can be modified for various use cases. **You should modify the environment variables before building the MetaDefender Core docker image**.

config · bash

```
# Product version for downloading installer package, ignored if use local installer
ARG VERSION=local

# Some convenient keys for building, do not change these
ENV VERSION         ${VERSION}
ENV PACKAGE_NAME    ometascan-${VERSION}-1.x86_64.rpm
ENV WORK_DIR        /mdcore
ENV PACKAGE_PATTERN ometascan-*.rpm
ENV PACKAGE_FILE    ${WORK_DIR}/ometascan.rpm


# All designated data files and folders to be used by MetaDefender Core upon installed will
reside inside
ENV INSTALL_ROOT    /opt/ometascan

# MetaDefender Core internal configurations
ENV REST_PORT       8008
ENV IGNITION_FILE   ${INSTALL_ROOT}/etc/opswat/ometascan.conf
ENV DATA_DIR        ${INSTALL_ROOT}/var/lib/ometascan
ENV CORE_CONF       ${INSTALL_ROOT}/etc/ometascan/ometascan.conf


# User for installing Core and running
ENV RUN_USER        metascan
ENV RUN_GROUP       metascan
```

```
ENV RUN_UID         1000
ENV RUN_GID         1000


# Define apikey path to store apikey for license activation and deactivation
ENV APIKEY_PATH     ${INSTALL_ROOT}/etc/opswat/apikey
```

- `VERSION` variable needs to be set to the desired version of MetaDefender Core to be downloaded and used.
- `RUN_USER` variable is the user that is automatically created in the container and then used to run the MetaDefender Core. It could be non-root user.
- `INSTALL_ROOT` variable to define root folder for all MetaDefender Core data files and folder to reside inside.
- `APIKEY_PATH` variable to define file path containing API key of your desired local user admin on MetaDefender Core, that will be used for license activation and deactivation.

MetaDefender Cluster Deployment

Integrating MetaDefender Core with MetaDefender Cluster empowers users with seamless scalability, allowing them to effortlessly handle surging volumes of file uploads and scans across distributed MetaDefender Core instances without compromising performance.

This combination delivers exceptional fault tolerance, ensuring continuous operation even if individual MetaDefender Core instances fail, while providing centralized management for streamlined oversight and policy enforcement.

More details at MetaDefender Cluster.

Anti-malware capabilities

Engine name	Vendor name	Country	Detection	Heuristic Detection
ClamAV	ClamAV	USA	Signature based	Yes
AhnLab	Harbin AhnLab Technology	South Korea	Signature based	No
Avira	Avira	Germany	Signature and AI based	Yes
ESET	ESET	Slovakia	Signature based	Yes
Bitdefender	Bitdefender	Romania	Signature based	Yes**
K7	K7 Computing Private	India	Signature based	Yes*
Quick Heal	Quick Heal Technologies	India	Signature based	Yes
Vir.IT	TG Soft	Italy	Signature based	No
Varist	OK	Iceland	Signature based	Yes
Emsisoft	Emsisoft	New Zealand	Signature based	Yes
IKARUS	IKARUS Security Software	Austria	Signature based	No
TACHYON	INCA Internet	South Korea	Signature based	No
Zillya!	ALL IT Services	Ukraine	Signature based	Yes*
Antiy	Antiy Labs	China	Signature base	Yes*
Xcitium	Xcitium	USA	Signature based	Yes
Trellix	Trellix	USA	Signature based	Yes

Engine name	Vendor name	Country	Detection	Heuristic Detection
Sophos	Sophos	UK	Signature based	No
CrowdStrike Falcon ML	CrowdStrike	USA	AI based	No
Webroot SMD	OpenText	USA	Signature and AI based	No
Lionic	Lionic Corp	Taiwan	Signature based	No
Filseclab	Filseclab Corp	China	Signature based	Yes
Huorong	Huorong Security	China	Signature based	No
Microsoft Defender	Microsoft	USA	Signature based	Yes**
NETGATE	Netgate Technologies	Slovakia	Signature based	Yes*
Scrutiny	Cyberstanc	USA	AI based	No
Xvirus Anti-Malware	Xvirus	Portugal	Signature based	No
Systweak	Systweak	India	Signature based	No
Aurora	Arctic Wolf	USA	AI based	No
CMC	CMC Cyber Security	Vietnam	Signature and AI based	No
BKAV Pro	BKAV Corporation	Vietnam	Signature and AI based	No
Gridinsoft	Gridinsoft	Ukraine	Signature based	Yes**
SentinelOne	SentinelOne	USA	AI based	No

*The engines have a "Heuristic Scan" configuration but is not enabled by default

** The engines have Heuristic detection but don't have configurations

For the other capabilities such as Archive extraction, Cloud scan, etc... please check the engine details in the MetaDefender product.

Supported file type

Engine Name	***Supported File Type***
CrowdStrike Falcon ML	Portable Executable (PE) Mach object file (Mach-O) Executable and Linkable Format (ELF) CDF & OOXML based Microsoft Office files Adobe File (PDF)
Webroot SMD	PE and JS embedded on PDF
RocketCyber	Any windows file that contains a PE header. That would include the file format for Windows executables, object code, and DLLs in both 32-bit and 64-bit versions.
Scrutiny	exe, dll, sys.
Cylance	Windows executables (.acm, .ax, .cpl, .drv, .efi, .mui, .ocx, .src, .sys, .tsp, .exe, .dll) macOS executables (.o, .dylib, .bundle) Linux executables (.o, .ko, .mod, .so) OLE files (.doc, .xls, .ppt) OOXML files (.docx, .xlsx, .pptx) PDF files (.pdf) Archive files (.zip, .7z, .rar, .tar, .gz, .bz2, .xz) **Note:** Files with the .zip extension are supported whether they are password-protected or not. Files with the .rar extension are supported only if they are not password-protected.
SentinelOne	Windows PE format, Linux ELF format, MACH format (Future), PDF format, DOS COM file, OLE compound file format, OpenXML Document File Format (.docx, xlsx, pptx...), ZIP Archive file format, RAR Archive file format, LZMA Archive file format, BZIP2 Archive file format, TAR

Engine Name	*Supported File Type*
	Archive file format, CABINET Archive file format, SFX Archive file format, Windows .NET PE File, Microsoft LNK file format, Email Storage Format, Powershell Script files, Python script executable file, Linux shell script executable file, gzip compressed file, PHP script file, ASP script file, ASP.NET Core script file, Batch file, JavaScript file, VBScript file

Info

Apart from the engines mentioned above, the other antivirus engines do not have specific file type limitations.

MetaDefender MetaScan Engine Packages

Windows Packages:

Packages	List of Engines
Metadefender 8 Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7 and Vir.IT
Metadefender 12 Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft and Tachyon
Metadefender 16 Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft, Tachyon, CMC, Xcitium , BKAV and Zillya!
Metadefender 20 Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft, Tachyon, CMC, Xcitium , BKAV, Zillya!, Sophos, Lionic, Trellix and Xvirus
Metadefender MAX Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft, Tachyon, Xcitium, Zillya!, Sophos, Lionic, Trellix, Xvirus, Antiy, CrowdStrike Falcon ML, Microsoft Defender, NETGATE, Systweak, Filseclab, Huorong, Scrutiny, Gridinsoft, Webroot SMD, Aurora, CMC, BKAV and SentinelOne

Packages	List of Engines
Metadefender Goverment & Defense	AhnLab, Avira, ClamAv, Eset, Vir.IT, Bitdefender, QuickHeal, Emsisoft, Tachyon, Varist, K7, Ikarus, CMC, CrowdStrike, Trellix, Sophos, Microsoft Defender and Xcitium

Linux Packages:

Packages	List of Engines
Metadefender 5 Engines	AhnLab, Bitdefender, ClamAV, Eset and K7
Metadefender 10 Engines	AhnLab, Bitdefender, ClamAV, Eset, K7, Avira, QuickHeal, Tachyon, Varist and Ikarus
Metadefender MAX Engines	AhnLab, Bitdefender, ClamAV, Eset, K7, Avira, QuickHeal, Tachyon, Varist, Ikarus, Sophos, Trellix, CrowdStrike Falcon ML, Lionic, Xvirus, Webroot SMD, Aurora, CMC and Xcitium

Government & Defense | Anti-malware capabilities

Engine name	Vendor name	Country	Detection
ClamAV	ClamAV	USA	Signature based
AhnLab	Harbin AhnLab Technology	South Korea	Signature based
Avira	Avira	Germany	Signature and AI based
ESET	ESET	Slovakia	Signature based
Bitdefender	Bitdefender	Romania	Signature based
K7	K7 Computing Private	India	Signature based
Quick Heal	Quick Heal Technologies	India	Signature based
Vir.IT	TG Soft	Italy	Signature based

Engine name	Vendor name	Country	Detection
Varist	OK	Iceland	Signature based
Emsisoft	Emsisoft	New Zealand	Signature based
IKARUS	IKARUS Security Software	Austria	Signature based
TACHYON	INCA Internet	South Korea	Signature based
Xcitium	Xcitium	USA	Signature based
Trellix	Trellix	USA	Signature based
Sophos	Sophos	UK	Signature based
CrowdStrike Falcon ML	CrowdStrike	USA	AI based
Microsoft Defender	Microsoft	USA	Signature based
CMC	CMC Cyber Security	Vietnam	Signature and AI based

Please check the configuration page in the product for the detail Detection methods (Heuristic, Archive extraction, ...) of each engine

AV engine return code

The following values are available for ***scan_results.scan_details.<engine name>.scan_result_i***

Returned Code	Description
0	No threat detected
1	Infected
2	Suspicious
3	Failed to scan

Returned Code	Description
10	Not scanned
19	Cancelled
22	PUA/PUP detected
23	File format not supported

Configure ClamAV to use a custom database

ClamAV allows you to create your own database to improve the detection, please see this page to lean more about how to create a custom database.

To use the custom database in MetaDefender Core:

- go to Inventory > Modules > Metascan > ClamAV
- Enable Custom Database, set Custom Database Path to the custom database folder



After saving the configuration, you can verify if the Metascan Core uses the new file or not by checking the ClamAV folder in **<MetaDefender Core installation folder>\data\engines\clamav_**_\database**, if there is an file started with **EXT_** which means the configuration was set successfully.

A simple test to validate this feature

- Create a text file with this format: **<md5>**:**<file size>**:**<malware name>**, for example**:** 385141700748a2cfa2d746fb8899efe6:35:My_Malware_Name**,** save it with **.hdb** extension to a folder
- Set the ClamAV custom database path to that folder
- Scan the file which has the same MD5 in text file, ClamAV should detect the file as infected with "<malware name>.UNOFFICIAL" as a threat name

Custom Database Example

Disclaimer

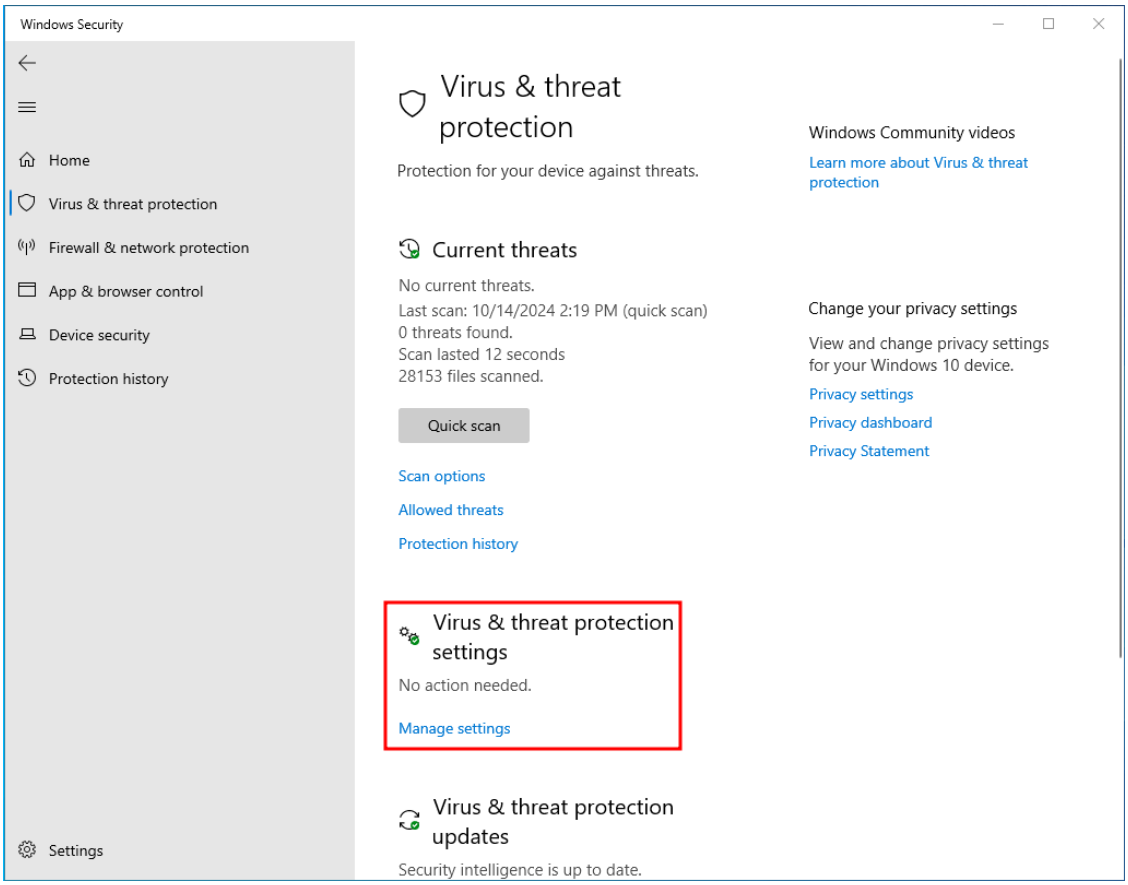
This information is provided for general informational purposes only. OPSWAT is not responsible for the accuracy or reliability of any information provided. The examples mentioned are for illustration only and do not constitute an endorsement by OPSWAT.

- SecuriteInfo [https://www.securiteinfo.com]
- Sanesecurity [https://sanesecurity.com/]
- URLhaus [https://urlhaus.abuse.ch/api/#clamav]
- PhishTank [https://phishtank.com/]
- Interserver [https://www.interserver.net/]
- Linux Malware Detect [https://www.rfxn.com/projects/linux-malware-detect/]
- OpenPhish [https://openphish.com/]

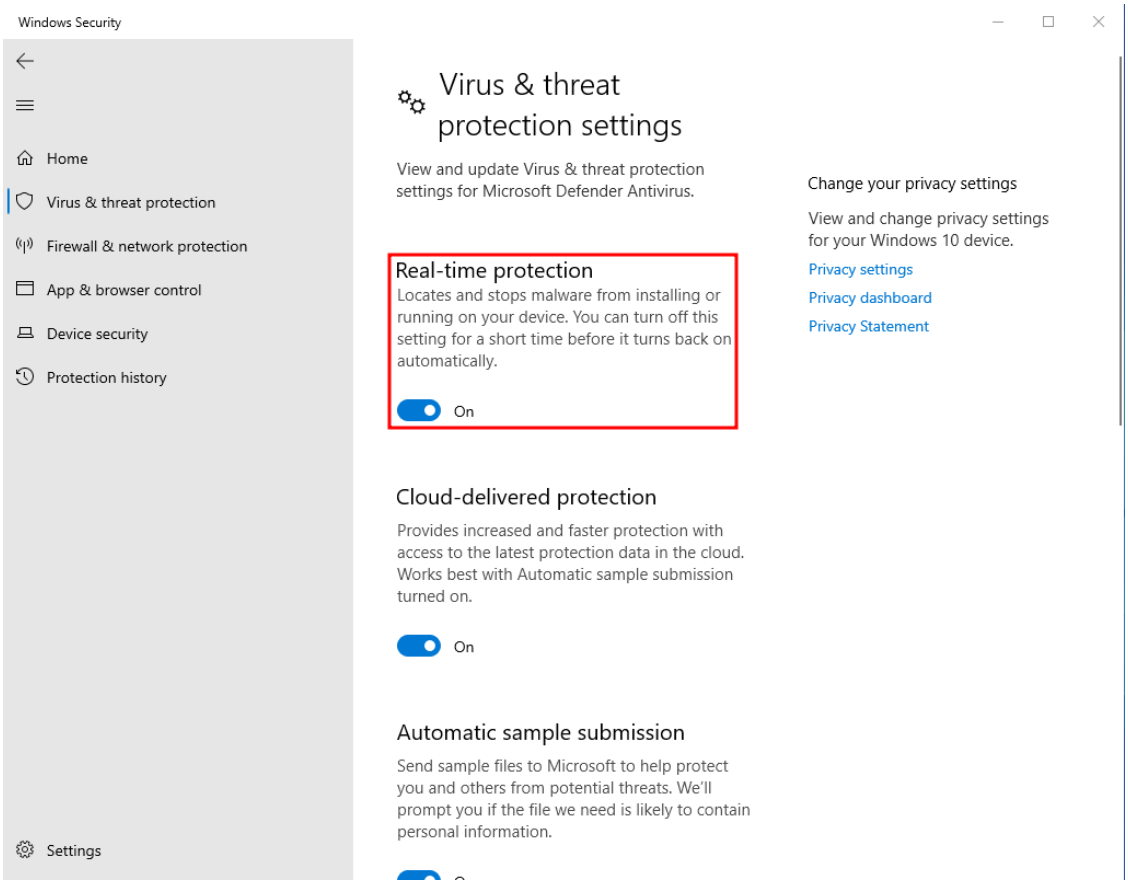
Set up Windows Defender as a custom engine

In order to set up and use Windows Defender as a custom engine in MetaDefender Core, the following conditions must be met:

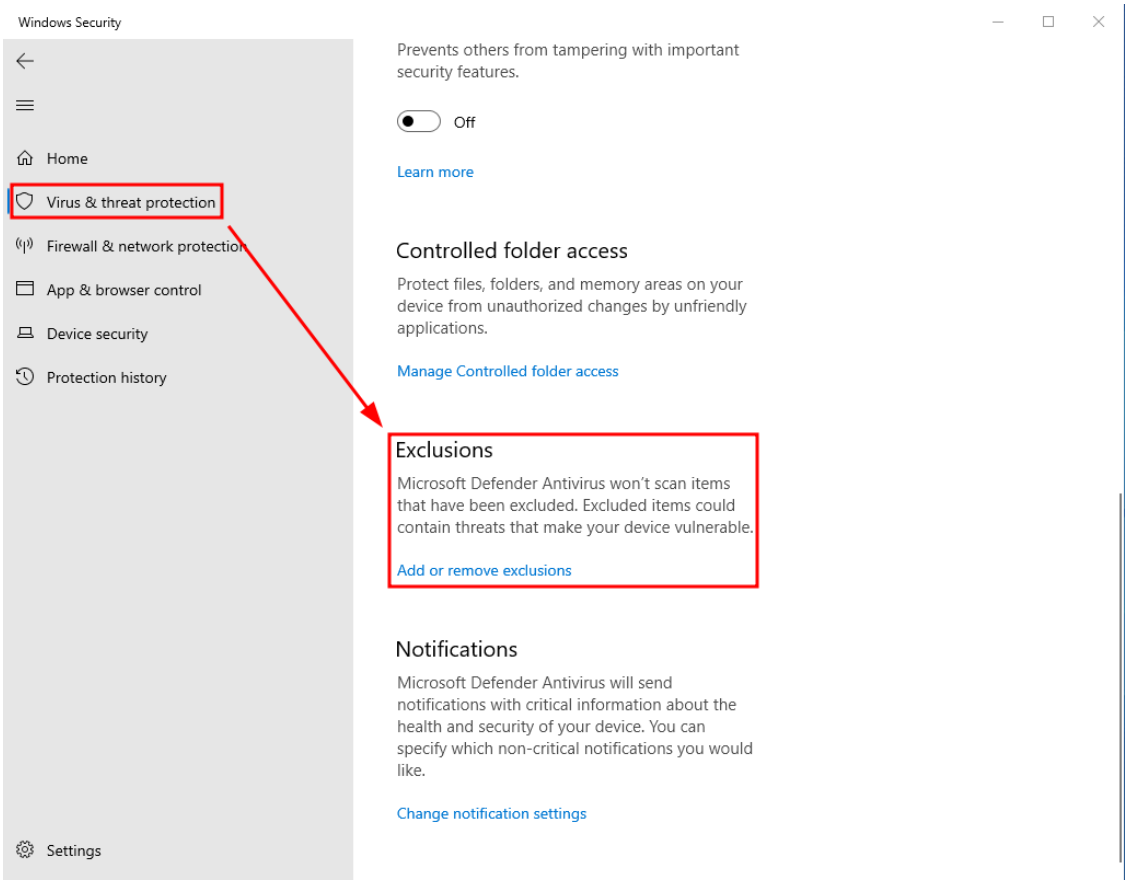
1. The native Windows Defender / Windows Security must exist on the Windows OS.
2. **Real-time protection** must be turned on.



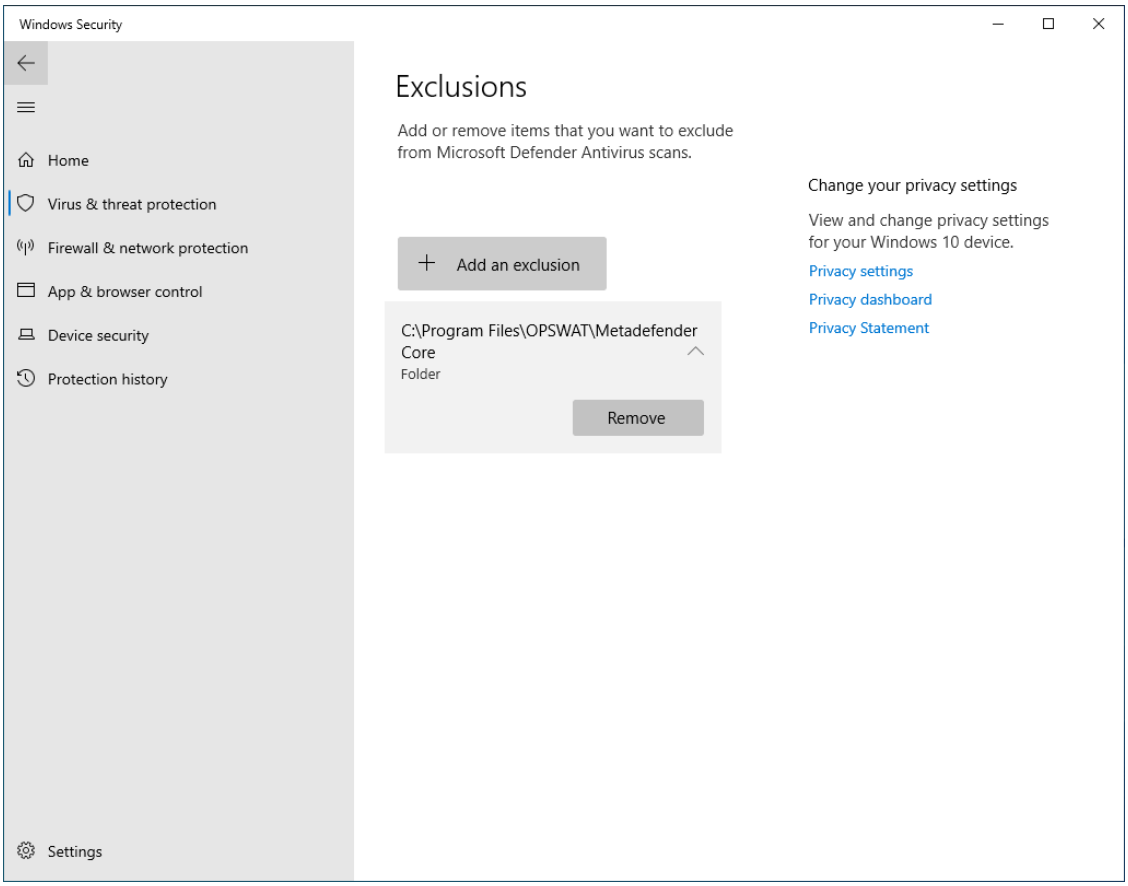
In case of the **Real-time protection** feature is disabled, it can be enabled on the **Manage settings** screen.



3. The MetaDefender Core **Installation** folder must be **Allowlisted** (Excluded):



The **Add or remove exclusions** feature can add a folder as an exclusion.



4. **Passive Mode** must be enabled for Windows Defender:
    1. download the Windows_Defender_Passive_Mode archive and extract it.
    2. execute the **enable_windows_defender_passive_mode.reg** to automatically add the following two keys to the registry.
    3. If you want to change Windows Defender back to active mode, execute the **disable_windows_defender_passive_mode.reg** or modify the two above registries to **00000000**.

Pro Tip:

Always check if RTP is still on after enabling passive mode! If not, try disabling passive mode!

Code · bash

```
[HKEYLOCAL MACHINE\SOFTWARE\Policies\Microsoft\Windows Advanced Threat Protection]
"DisableRoutinelyTakingAction"=dword:00000001
"ForceDefenderPassiveMode"=dword:00000001
```

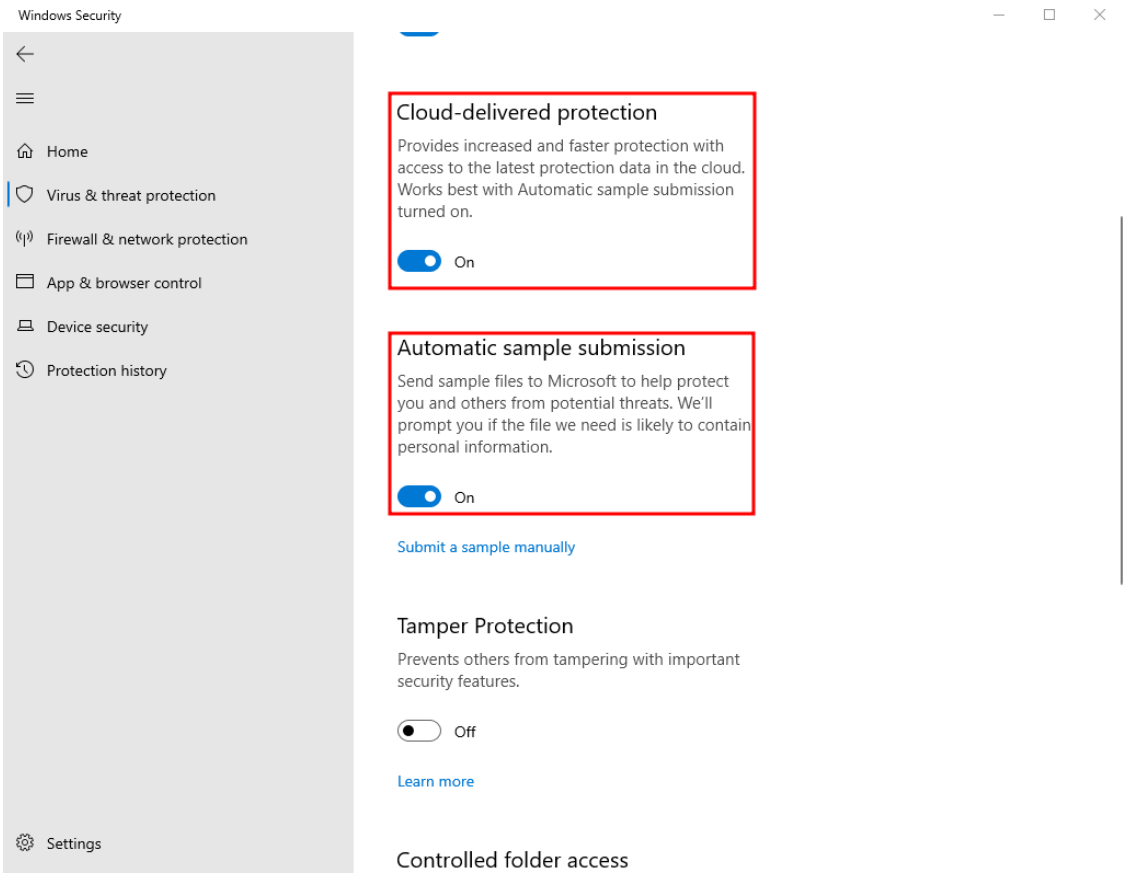
Pro Tip:

Considering that this custom engine uses the native Windows Defender available on your system, the behavior of the engine relies on your Windows Defender local settings.

So, for example, if you do not want to submit files to Microsoft servers using the cloud feature, you should turn these settings off in the Windows Defender configuration.

Turning off Cloud protection

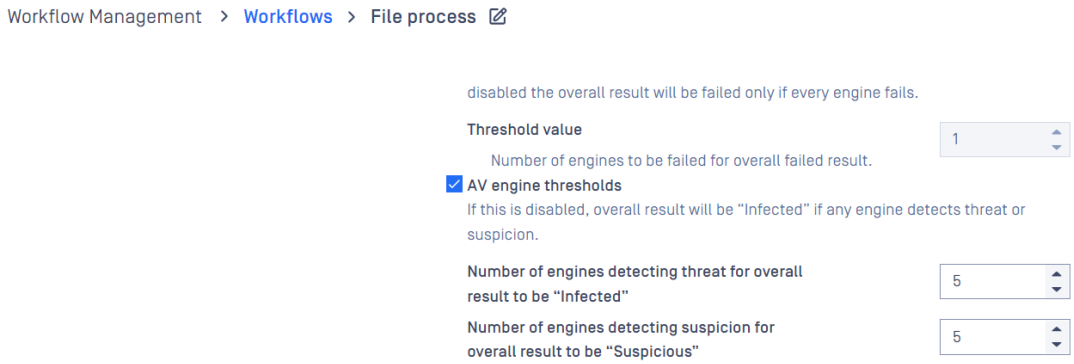
In order to turn off cloud protection or automatic sample submission, turn each feature off in Windows Defender's settings



AV scoring with post action

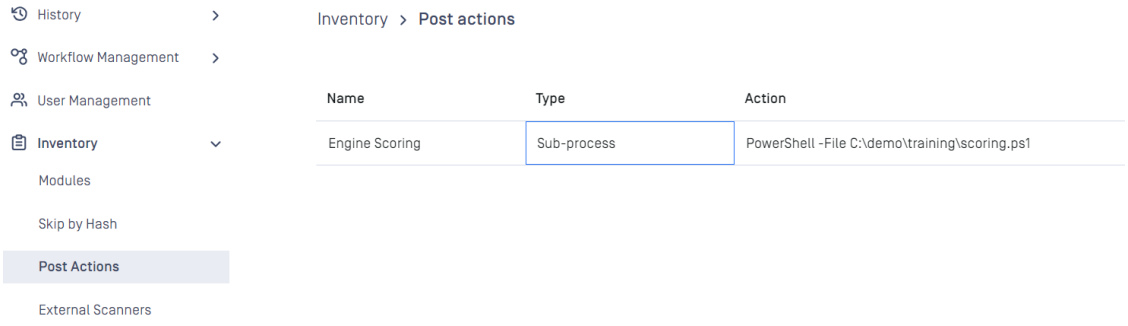
If certain engines frequently produce false positive detections, you can use the "Post Action" feature (available in MetaDefender Core 5.8.0 and later) to execute a script that modifies the final verdict. Within the script, you can implement logic to assign high scores to specific engines and low scores to others.

Step 1: The "Post Action" script can only set the final verdict to infected and NOT clean. This means if you want to fully rely on scoring for the AV engine results, you have to set the infected AND suspicious AV thresholds to at least the number of your AV engines in your workflow under the Metascan tab.

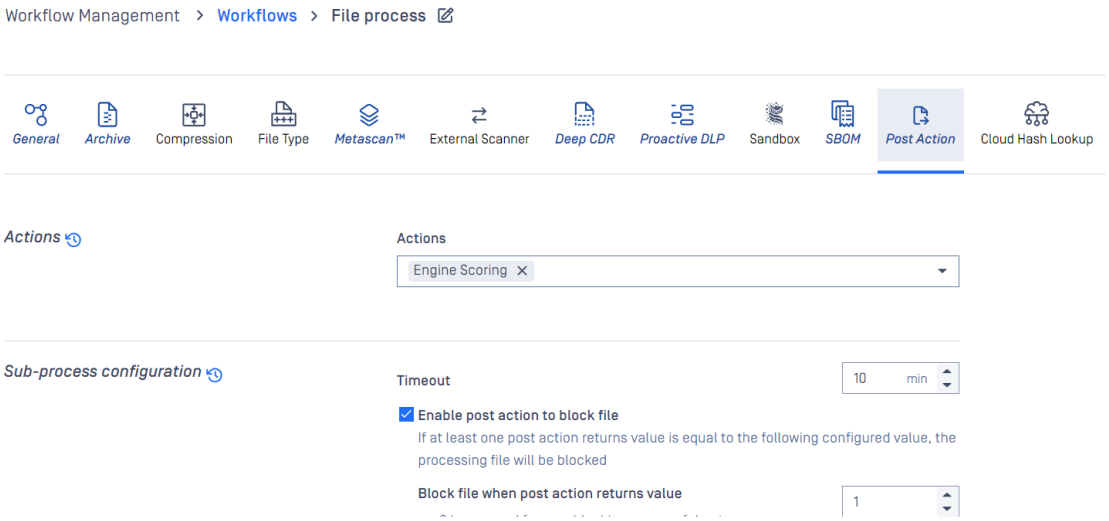


Step 2: Download the scripts at OPSWAT Github and follow the readme (This project is just an example, use and modify it at your own risk)

Technically, you create a post action, for example, on Windows:



Step 3: Configure a workflow to use this post action.



Supported File Types

What is CDR?

An increasingly popular and effective method of compromising computer security, especially as part of a targeted attack, involves sharing common document types or image files with victims. Even though the original versions of these files do not contain executable data, attackers have found ways to trigger these files to execute embedded malicious code. Popular techniques used to accomplish this include VBA macros, exploit payloads, and embedded Flash or JavaScript code. This type of attack has a high success rate because most users don't expect common file types to contain infections. For high-risk files or scenarios, Content Disarm & Reconstruction (CDR) prevents any possibility of malicious content (including zero-day threats) from executing. High-risk files can be sanitized through several different methods:

- Removing hidden exploitable objects (e.g., scripts, macros, etc.)
- Converting the file format

Supported File Types (total 232)

Source File Type	Description	Target Sanitized Types
doc	Microsoft Word 97-2003 Document	**doc**, docx, pdf, rtf
dot	Microsoft Word 97-2003 Template	**dot**, dotx
xls	Microsoft Excel 97-2003 Workbook	**xls**, pdf, csv, xlsx
xlt	Microsoft Excel 97-2003 Template	**xlt**, pdf, png
ppt	Microsoft PowerPoint 97-2003 Presentation	**ppt**, pdf, pptx
pot	Microsoft PowerPoint 97-2003 Template	**pot**, pdf, png
rtf	Microsoft Rich Text Format	**rtf**, pdf*
docx	Microsoft Word Document	**docx**, doc, txt, html, pdf, ps, jpg, bmp, png, tiff, svg, rtf
docm	Microsoft Word Macro-Enabled	**docm**, docx, txt, html, pdf, ps, jpg,

Source File Type	Description	Target Sanitized Types
	Document	bmp, png, tiff, svg, rtf
dotx	Microsoft Word Template	**dotx**
dotm	Microsoft Word Macro-Enabled Template	**dotm**, dotx
xlsx	Microsoft Excel Workbook	**xlsx**, xls, csv, html, tiff*, pdf, ps, jpg, bmp, png, svg
xlsm	Microsoft Excel Macro-Enabled Workbook	**xlsm**, xlsx, csv, html, tiff*, pdf, ps, jpg, bmp, png, svg
xlsb	Microsoft Excel Binary Workbook	**xlsb**
xltx	Microsoft Excel Template	**xltx**, pdf, png, csv
xltm	Microsoft Excel Macro-Enabled Template	**xltm**, pdf, png, csv*
xlam	Microsoft Excel Add-in	**xlam**
csv	Comma-separated values	**csv,** xlsx
tsv	Tab-separated values	**tsv**
pptx	Microsoft PowerPoint Presentation	**pptx**, ppt, html, pdf, ps, jpg, bmp, png, tiff*, svg
potx	Microsoft PowerPoint Template	**potx**, pdf, png
pptm	Microsoft PowerPoint Macro-Enabled Presentation	**pptm**, pptx, html, pdf, ps, jpg, bmp, png, tiff*, svg
potm	Microsoft PowerPoint Macro-Enabled Template	**potm**, pdf, png
pps	Microsoft PowerPoint 97-2003 Show	**pps**, pdf, png
ppsm	Microsoft PowerPoint Macro-	**ppsm**, pdf, png

Source File Type	Description	Target Sanitized Types
	Enabled Show	
ppsx	Microsoft PowerPoint Show	**ppsx,** bmp
ppam	Microsoft Powerpoint add-on	**ppam**
sldx	Microsoft Powerpoint Slide 2007+	**sldx**
sldm	Microsoft Office PowerPoint 2007 Slide - Macro Enabled	**sldm**
vsd	Microsoft Visio Drawing	**pdf**
vsdx	Microsoft Visio Drawing	**vsdx**, pdf, xps, jpg, png, bmp, tiff, svg, emf*, html, xaml, swf
vssx	Microsoft Visio Stencil	**vssx**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
vstx	Microsoft Visio Template	**vstx**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
vsdm	Microsoft Visio Macro-Enabled Drawing	**vsdm**, pdf, xps, jpg, png, bmp, tiff, svg, emf*, html, xaml, swf
vssm	Microsoft Visio Macro-Enabled Stencil	**vssm**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
vstm	Microsoft Visio Macro-Enabled Template	**vstm**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
vsx	Microsoft Visio XML Stencil	**vsx**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
vtx	Microsoft Visio XML Template	**vtx**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
vdx	Microsoft Visio XML Drawing	**vdx**, pdf, xps, jpg, png, bmp, tiff_, svg, emf_, html, xaml, swf
one	Microsoft OneNote	**one**

Source File Type	Description	Target Sanitized Types
odt	OpenDocument Text	**odt**, pdf
ods	OpenDocument Spreadsheet	**ods**
ott	OpenDocument Document Template	**ott**, pdf
ots	OpenDocument Spreadsheet Template	**ots**
odp	OpenDocument Presentation	**odp**
otp	OpenDocument Presentation Template	**otp**
htm/html	Hypertext Markup Language	**html**, pdf, ps, jpg, bmp, png, svg, txt
mht	MIME HTML	**mht**, pdf, jpg, bmp, png, tiff
hta	HTML Application	**hta**
mar	Mozilla Archive	**mar**
pdf	Adobe Portable Document Format	**pdf**, html, svg, jpg, bmp, png, tiff, txt, pptx
fdf	Forms Data Format	**fdf**
ai	Adobe Illustrator	**ai**
ait	Adobe Illustrator Template	**ait**
idml	InDesign Markup Language	**idml**
xps	XML Paper Specification	**pdf**
hwp	Hangul Word Processor	**hwp**
hwt	Hangul Word Template	**hwt**

Source File Type	Description	Target Sanitized Types
hwpx	Hangul Word Processor	**hwpx**
cell	Hancom Cell	**cell**
show	Hancom Show	**show**
jtd	Ichitaro Document	**jtd**
jtdc	Ichitaro Compressed Document	**jtdc**
jhd	Justsystem Hanako	**jhd**
xml	Extensible Markup Language	**xml**
xml-doc	Microsoft Word 2003 XML Document	**pdf**
xml-docx	Microsoft Word XML Document	**xml-docx**, pdf
xml-xls	Microsoft XML Spreadsheet 2003	**pdf**
xml-pptx	Powerpoint XML Presentation	**xml-pptx**
jnlp	Java Network Launching Protocol	**jnlp**
bml	Bean Markup Language	**bml**
twbx	Tableau packaged workbook	**twbx**
twb	Tableau workbook	**twb**
tds	Tableau Datasource	**tds**
pbix	Microsoft Power BI	**pbix**
rdf	Resource Description Framwork	**rdf**
mpp	Microsoft Project File	**mpp**, pdf, xlsx
vcs	vCalendar	**vcs**

Source File Type	Description	Target Sanitized Types
ics	iCalendar	**ics**
vcf	Virtual Contact File	**vcf**
lnk	Windows Shortcut	**lnk**
url	URL	**url**
jpg	JPEG Image	**jpg**, bmp, png, tiff, svg, gif, ps, eps, pdf*
mj2	Motion JPEG 2000	**mp4**, avi, mov, wmv
jpx	JPEG 2000	**jpx**
bmp	Windows Bitmap Image	**bmp**, jpg, png, tiff, svg, gif, ps, eps, pdf*
png	Portable Network Graphics	**png**, jpg, bmp, tiff, svg, gif, ps, eps, pdf*
apng	Animated PNG	**apng,** png
mng	Multiple Network Graphic	**mng**
tiff	Tagged Image File Format	**tiff**, jpg, bmp, png, svg, gif, ps, eps, pdf
tiff64	Big Tagged Image File Format	**tiff64**, jpg, png, gif, bmp
nef	Nikon Raw Image	**nef**, jpg, bmp, png, svg, gif
svg	Scalable Vector Graphics	**svg**, jpg, bmp, png, tiff, gif, ps, eps
gif	Graphics Interchange Format	**gif,** jpg, bmp, png, tiff, svg, ps, eps, pdf*
tga	Truevision Advanced Raster Adapter	**tga**

Source File Type	Description	Target Sanitized Types
wmf	Windows Metafile	**wmf**, jpg, bmp, png, tiff, svg, gif, ps, eps, pdf*
emf	Windows Enhanced Metafile	**emf**
emz	Windows Compressed Enhanced Metafile	**emz**
ico	Icon	**ico**, bmp, png
cur	Cursor	**cur**
webp	Google Image File Format for Web	**webp**
wdp	HD Photo	**wdp**
dwfx	Design Web Format XPS	**dwfx**
dwg	AutoCAD	**dwg**
dwt	AutoCAD Drawing Template	**dwt**
dws	AutoCAD Drawing Standards	**dws**
sfc	SXF Feature Comment	**sfc**
p21	STEP Data Model	**p21**
stp	STEP Data Model	**stp**
ifc	Industry Foundation Classes	**ifc**
jww	JW CAD	**jww**
jwc	JW CAD	**jwc**
bfo	V-nas BFO	**bfo**
jt	Jupiter Tessellation	**jt**

Source File Type	Description	Target Sanitized Types
dxf	AutoCAD DXF	**dxf**, pdf, jpg, png, bmp, gif, tiff
dwf	Design Web Format	**dwf**, pdf, jpg, png, bmp, gif, tiff
3ds	3D Studio	**3ds**, dae, stl, fbx
3MF	3D Manufacturing Format	**3MF**
dae	Digital Asset Exchange	**dae**, 3ds, stl, fbx
u3d	Universal 3D	**u3d**, 3ds, dae, stl, pdf, drc, rvm, fbx
drc	Google Draco	**drc**, 3ds, dae, pdf, u3d, rvm, fbx
rvm	AVEVA Plant Design Management System Model	**rvm**, 3ds, dae, stl, pdf, u3d, drc, fbx
hl7	Health Level Seven	**hl7**
dcm	Digital Imaging and Communications in Medicine	**dcm**
shp	Shapefile	**shp**
shx	Shapefile	**shx**
dbf	Shapefile	**dbf**
lyrx	ArcGIS Layer Files	lyrx
heic	High Efficiency Image Container	**heic**, jpg, pdf
heif	High Efficiency Image File	heic, bmp, png, jpg
avif	AV1 Image File Format	**avif**
avc	Advanced Video Codec	**avc**
wsq	Wavelet Scalar Quantization FBI fingerprint format	**wsq**

Source File Type	Description	Target Sanitized Types
odg	OpenDocument Drawing	**odg**
wmv	Windows Media Video	**wmv**, mp4, avi
wma	Windows Media Audio	**wma**
mpeg	Moving Picture Experts Group	**mpeg,** mp4, avi
wav	Waveform Audio	**wav**, mp3, mp4
mp3	MPEG-1 Audio Layer-3	**mp3,** mp4
paf	Ensoniq's Paris Audio File	**wav**
mpc	Musepack Audio Format	**wav**
mp4	MPEG-4 Part 14	**mp4**, avi, wmv
mov	QuickTime video format	**mov,** mp4, avi
avi	Audio Video Interleave	**avi**, mp4, wmv
webm	Video file format	**webm**
flv	Flash Video	**flv**
fla	Flash Audio	**wav**
swf	Shockwave Flash	**wmv**
bwf	Broadcast wave format	**bwf**
bw64	Broadcast wave 64	**bw64**
w64	Sony wave64 autdio format	**w64**
rf64	BWF-compatible multichannel audio file format	**rf64**
m4a	MPEG-4 Audio	**m4a**, mp3, wav

Source File Type	Description	Target Sanitized Types
m4v	MPEG-4 Video	**mp4,** wmv, avi
mkv	Matroska Video	**mp4,** wmv, avi
hevc	High Efficiency Video Coding	**hevc**
3gp	Third Generation Partnership Project	**mp4,** wmv
mts	MPEG-2 Tranmission	**mts,** wmv, avi, mp4,wmv
ogg	Ogg Vorbis Compressed Audio	**ogg,** mp4
aiff	Audio Interchange File Forma	**aiff**
aac	Advanced Audio Coding	**aac**
acm	Audio Compression Manager	**acm**
caf	Core Audio Format	**wav**
opus	Ogg Opus	**opus,** mp4, mp3, wav
mxf	Material Exchange Format	**mxf**
vtxt	AmiVoice Voice To Text	**vtxt**
eml	Electronic mail	**eml,** zip
msg	Microsoft Outlook Message	**msg,** zip
tnef	Transport Neutral Encapsulation Format	**tnef**
oft	Microsoft Outlook Template	**oft**
pst	Outlook Personal Folder	**pst**
mbx	Outlook Express Mailbox	**mbx,** zip

Source File Type	Description	Target Sanitized Types
txt	Text	**txt**, pdf
json	JSON	**json**
ttml	Timed Text Markup Language	**ttml**
mpd	MPEG-Dash	**mpd**
xdw	DocuWork Image	**xdw***
xbd	DocuWork Binder	**xbd***
xct	DocuWork Container	**xct***
crl	Certificate Revocation List	**crl**
spf	StorageCraft ShadowProtect	**spf**
prn	HP Printer Job Language data	**prn**
zei	ZUGFeRD	**zei**
utib	UTIB	**utib**
har	HTML Archive	**har**
story	Storyist Document	**story**
7z	7-zip Archive	**7z**, zip, gz, xz, tar
gz/gzip	GNU Zipped Archive	**gz**, 7z, zip, xz, tar
rar	WinRAR Archive	**rar,** zip, 7z, gz, xz, tar
xz	XZ Archive	**xz**, zip, 7z, gz, tar
zip	ZIP Archive	**zip**, 7z, gz, xz, tar
alz	ALZip	**zip**, 7z, gz, xz, tar

Source File Type	Description	Target Sanitized Types
tar	Tape Archive	**tar,** zip, 7z, gz, xz
bz2	BZ2 Archive	**zip**, 7z, gz, xz, tar
lzma	LZMA Archive	**zip**, 7z, gz, xz, tar
lzh	LZH Archive	**zip**, 7z, gz, xz, tar
arj	ARJ Archive	**zip**, 7z, gz, xz, tar
cab	Cabinet Archive	**cab,** zip, 7z, gz, xz, tar
lz4	Lempel-Ziv 4 Compressed Archive	**lz4**
wsp	Windows Sharepoint	**zip**, 7z, gz, xz, tar
ace	WinAce archive format	**zip**, 7z, gz, xz, tar
tse	TIP Test Selection Engine	**tse**, zip, 7z, gz, xz, tar
tsez	TIP Test Selection Engine	**tsez**, zip, 7z, gz, xz, tar
tsec	TIP Test Selection Engine	**tsec**, zip, 7z, gz, xz, tar
egg	EGG archive format	**zip**, 7z, gz, xz, tar
gpkg	GeoPackage Encoding Standard	**gpkg,** zip
b3dm	Batched 3D Model	**b3dm**, zip
x3p	Xml 3D Surface Profile	**x3p**
asics	Associated Signature Container	**asics**
asice	Associated Signature Container	**asice**
jar	Java Application	**jar**
war	Java Web Application Archive	**war**

Source File Type	Description	Target Sanitized Types
log	Log file	**log**
pck	System Center Configuration Manager Package Files	**pck**
aem	AppleSingle/AppleDouble Encoded Macintosh	**aem**
ribc	Research Institute on Building Cost	**ribc**
ribcd	RIBC Data Table	ribcd
esz	E-szigno	**esz**, zip
base64	Base64 Encoding	**base64**
yenc	yenc Encoding	**yenc**
zstd	Zstandard Compressed	**zstd**
yz1	Yamazaki Zipper Compressed Archive	**yz1***
scdoc	SpaceClaim Document	**scdoc**
mdzip	MagicDraw	**mdzip**
lz	Lzip Compressed Archive	**lz**
asar	Atom Shell Archive	**asar**
3dml	3D Mesh Layer	**3dml**
easm	3D CAD Assembly File	**easm**
vsix	Visual Studio Extension	**vsix**
blz	BriefLZ	**blz**

Source File Type	Description	Target Sanitized Types
taz	TAR Zipped Archive	**taz**
aff4	Advanced Forensic Format 4	**aff4**

- (*) Only supported on Windows for now.
- For the archive sanitization, please enable it in the Compression tab

Info

Sanitization is in BETA for these file types:

AVC, SPF, TWBX, UTIB, STORY, ASICS, ASICE, ES3, PCK, B3ML, B3DM, MDZIP, LZ, ASAR, MPD, DWFX, AVIF, 3DML, MPP, STP, VSIX, EASM, HEIF, YENC, VCF, IFC, LYRX, BLZ, X3P, TAZ, JT, IDML, AFF4

XML sanitization is specific to XML vulnerability. It does not eliminate other threat such as Microsoft Office XML formats.

HTML/TXT sanitization is designed for Email Security purposes, should not use for sanitizing normal HTML, TXT traffic. TXT with "Remove Invisible Characters" can be used for the LLM Guard use cases.

HWP: there are two versions of HWP, v3.0 and v5.0. v3.0 document can be created from only legacy old Hangul Word Processor. For this reason, we do not support HWP v3 and result in "failed to sanitize". We recommend this old version file as suspicious. If you need support for v3.0, please contact support.

XDW/XBD/XCT: Customers need to> install and activate DocuWorks 9 or newer on the system to use this feature

HEIC/HEVC/HEIF/AVC/AVIF: it is enabled on-demand, please contact Customer Support or Sales team to enable it.

Single / Multiple Output File

If target contains only one file, it will be not zipped and treat as single output file. For example, If a PDF file has only one page, converts to JPG will be JPG. If a PDF file has more than one page, there will be multiple JPG files and will result in a ZIP file. The following sanitization result in potentially multiple files (single ZIP file).

- PDF->HTML
- PDF->IMG
- DOCX HTMLIMG
- XLSX->HTML, CSV, IMG
- PPTX HTMLIMG

Notes:

Deep CDR removes active content that can drive a malicious behavior. These objects are usually non-visual, such as

javascript, hidden malicious code in an image (steganography). However, those objects can also be visible such as hyperlinks, active code that change the data (e.g. macros). Even though is not within CDR process's scope, it might also alter the content if it's configured to do so.

For images and media files, we don't "edit" the content, which means the original file and sanitized file will have the same content. However, the quality may be altered due to several steps performed to disarm the content by decoding, processing and encoding it again.

Known Issues

1. Conversion from HTML to an image would fail if the size of the HTML file is bigger than 90KB
2. AutoCAD file (.DWG): with version 2007-2009, when removing macro from the original file (if it has), opening sanitize file will display an error message "Failed to load project from storage" appeared but the file still works as usual
3. Support TXT in ASCII, UTF-8, Shift-JIS and ISO/IEC 8859-1 encoding only
4. When converting Excel files to TXT, only the first sheet is converted
5. Support AI in PDF format
6. If the file names inside XCT contain multibyte characters after sanitization, it will be different from the original. As a workaround, users can change the system default code page setting by the following steps: Settings > Time & language > Language & region > Administrative language settings > Change system locale, and check Beta: Use Unicode UTF-8 for worldwide language support

Supported versions

File types	Supported Versions
PDF	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 2.0
MS Office	97-2003, 2007 and later. Not support version 95 or older
JPG	1.01, 1.02, 1.03, 2.1, 2.2, 2.3, jpeg-ls
PNG	1.0, 1.1, 1.2, 1.4, 1.6
BMP	1.0, 2.0, 3.0, 3.1, 4.0, 5.0
GIF	Gif87a, Gif89a
TIFF	1.0, 2.0, 3.0, 4.0, 5.0, 6.0, BigTIFF, GeoTIFF
DICOM	3.0, 3.1, 3.2, 3.3, 3.4, 3.5

File types	Supported Versions
WEBP	0.1, 0.2, 0.3, 1.0, 1.1
HEIC	1.0, 1.1, 2.0
APNG	1.0
MNG	1.0, 1.0.1, 1.0.2, 1.0.3
SVG	1.0, 1.1, 2.0
WAV	1991
OGG	1.0 Xiph (2000)
MP3	1.0, 2.0, 2.5
BW64	1.0 (2014 EBU Tech 3306)
W64	1.0 (2002)
EMF/WMF	Original WMF, Placable WMF, EMF, EMF+
AIFF	1.0 (1988)
AAC	AAC MPEG-4 Part 3 (1999), HE-AAC v1 2003, HE-AAC v2 (2006), AAC-LED (2007)
OPUS	1.0 RFC 6716 (2012), 2.0 RFC 7845 (2015)
BWF	1.0 (2001), 2.0 (2003)
MP4	ISO/IEC 14496-14 (MPEG-4 Part 14)
MOV	MPEG-4 AVC, MPEG-4 HEVC
MPEG	MPEG-2
WMV	WMV1/7, WMV2/8, WMV 9 (SMPTE 421M or VC-1)
AVI	avi 1.0, avi 2.0

File types	Supported Versions
WEBM	2010 (initial release), 2013 (support VP9 video and Opus audio)
FLV	v1
M4A	ISO/IEC 14496-3 (MPEG-4 Part 3)
MTS	MTS 2006
WMA	WMA v1, v2, v9, Pro, Voice, Lossless
HTML	HTML 2.0, HTML 3.2, HTML 4.0, HTML 4.01, HTML 5.0, HTML 5.1, HTML 5.2, XHTML 1.0, XHTML 1.1
JSON	JSON (RFC 4627, ECMA-404, RFC 8259, ISO/IEC 21778:2017)
HAR	1.1, 1.2, 1.3
MHT	MHT
TWB	TWB
XML	1.0, 1.1, 2.0
RTF	1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9.1
ODT, OTT, ODS, OTS, ODP, OTP	1.0, 1.1, 1.2, 1.3
xdw, xbd	v7.3, v9.0, v9.1
SHOW, CELL	2014, 2018, 2020, 2024
HWPX	2014
VSDX, VSSX, VSTX, VSDM, VSSM, VSTM	15.0, 16.0
VSX, VTX, VDX	11.0, 12.0, 14.0, 15.0, 16.0
DWG, DWT, DWS	DWG 2000, DWG 2004, DWG 2007, DWG 2010, DWG

File types	Supported Versions
	2013, DWG 2018
DWF	DWF 6.0, DWF 6.1, DWG 7.0, DWF 7.5
DXF	DXF 13, DXF 14, DXF 2000, DXF 2004, DXF 2007, DXF 2010, DXF 2013, DXF 2018
JWW	Version 7.0 and before
JTD	Version 9+ (1998)
JTDC	Version 11+ (2001)
SFC, P21	1994, 2002 Conformance class 1
BFO	v9.0, v12.0, v14.0, v2021, v2016, v2012, v2011
ONE	OneNote 2010 and later
CRL	version 2
DBF	III+, V
AI	Only support AI files in PDF format, doesn't support AI files in PostScript format
PST	Only support Unicode PST. ANSI PST is not supported.

Max File Size

Info

This document does not specify the maximum supported file size that Deep CDR can handle. Instead, it only presents the file sizes that OPSWAT tested and the time Deep CDR takes to process a single large file.

Processing time may vary depending on the dataset used for testing. The primary factor affecting processing time is the number of objects within a file. For example, a large PDF with a single page containing a big image is processed faster than a large PDF with thousands of pages containing smaller images.

For audio and video files, Deep CDR verifies the input files against their respective specifications, remove unapproved objects, making the process fast. The table below does not cover conversion between different

file types.

System spec:

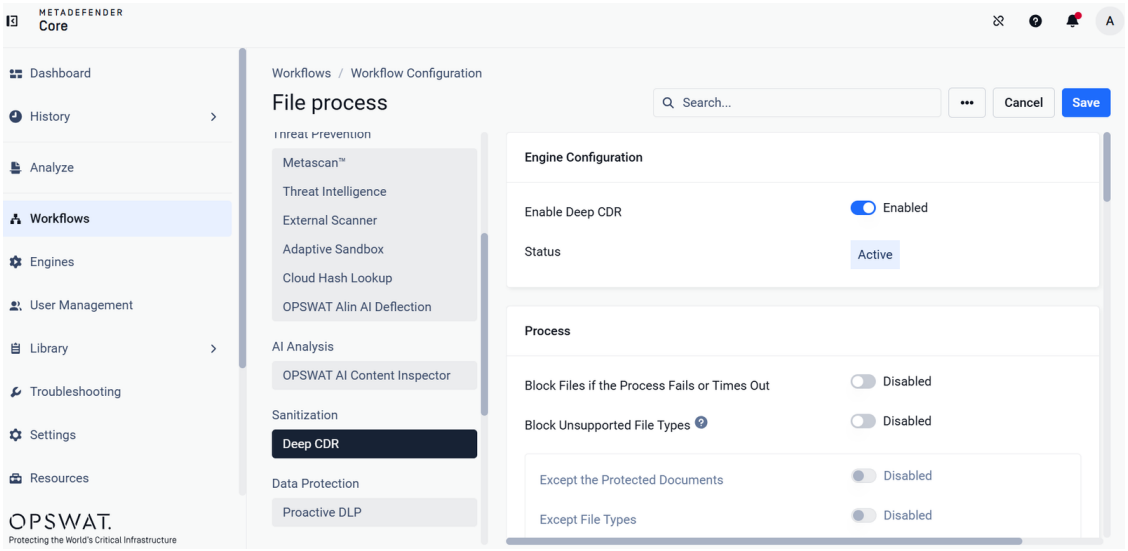
- CPU: Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz (8 CPUs)
- RAM: 32GB
- SSD: 500GB

File type	File Size (MB)	Processing time (s)	Notes
PDF	2,112	393	
MS Excel 2003	1,998	86	Microsoft 2003 allows max 2GB file size
MS Excel 2007	2,584	555	
MS Word 2003	952	118	Microsoft 2003 allows max 2GB file size
MS Word 2007	2,308	194	
MS PowerPoint 2003	1,912	229	Microsoft 2003 allows max 2GB file size
MS PowerPoint 2007	2,656	165	
RTF	507	30	In Word, the maximum file size for RTF files is typically around 512MB
Visio 2013	775	79	
Visio XML	1,221	66	
JPG	361	571	Dimensions: 40500 x 37670
PNG	530	443	Dimensions: 43200 x 21600
AVI	827	5	

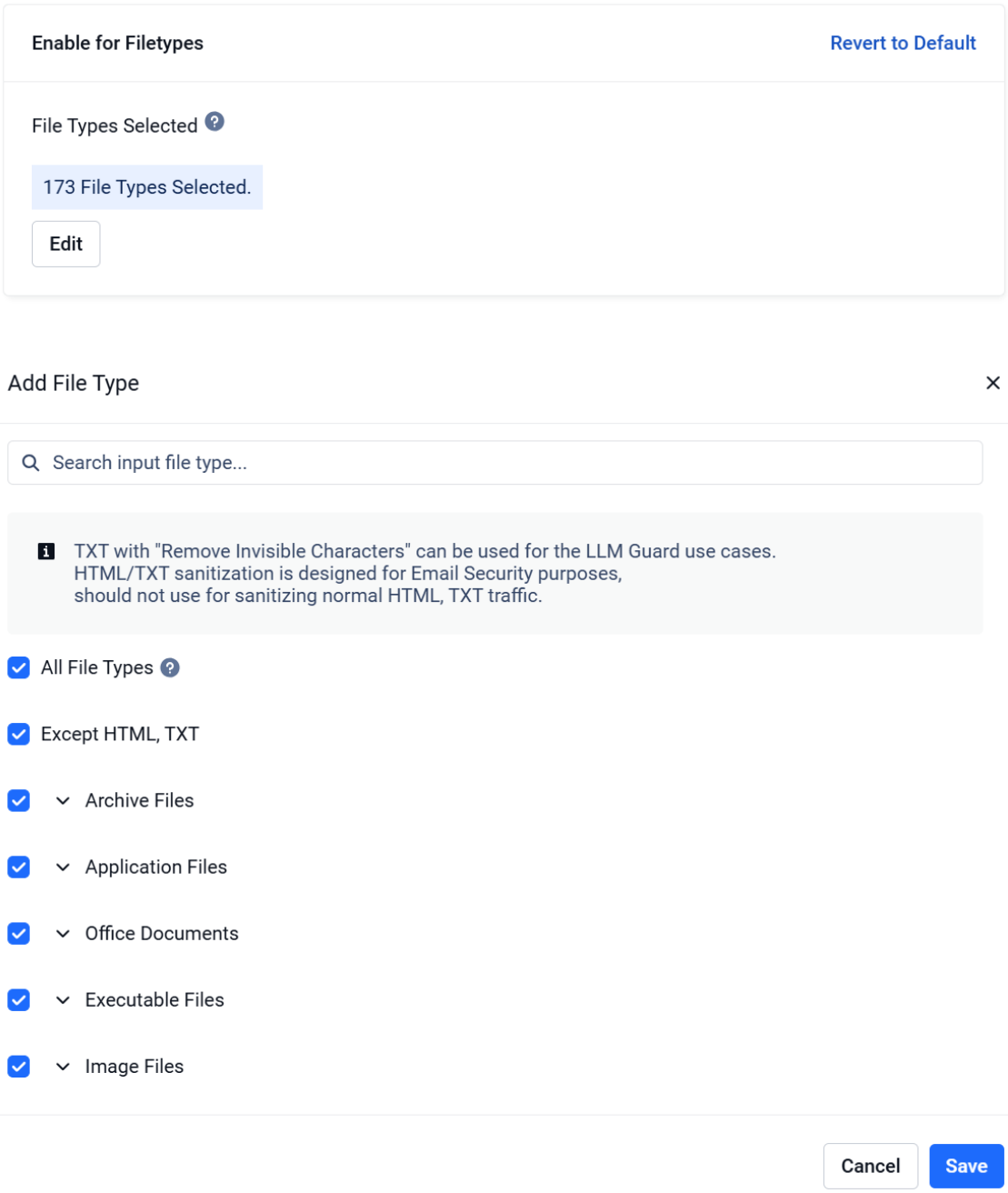
File type	File Size (MB)	Processing time (s)	Notes
MOV	1,088	9	
MP4	951	3	
MPEG	1,251	4	
MTS	1,304	3	
MWV	1,344	3	
WEBM	652	2	
MP3	1,908	13	
OGG	798	4	
WAV	688	35	

Configurations

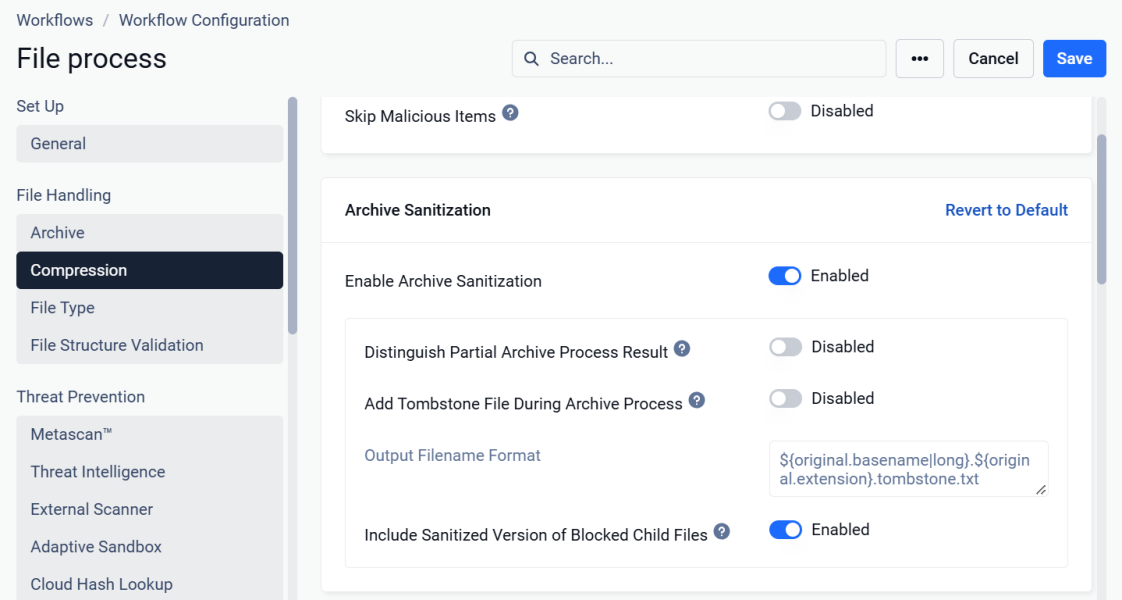
To enable Deep CDR, go to **Workflow Managements > Workflows > [Workflow name] > Enable Deep CDR**



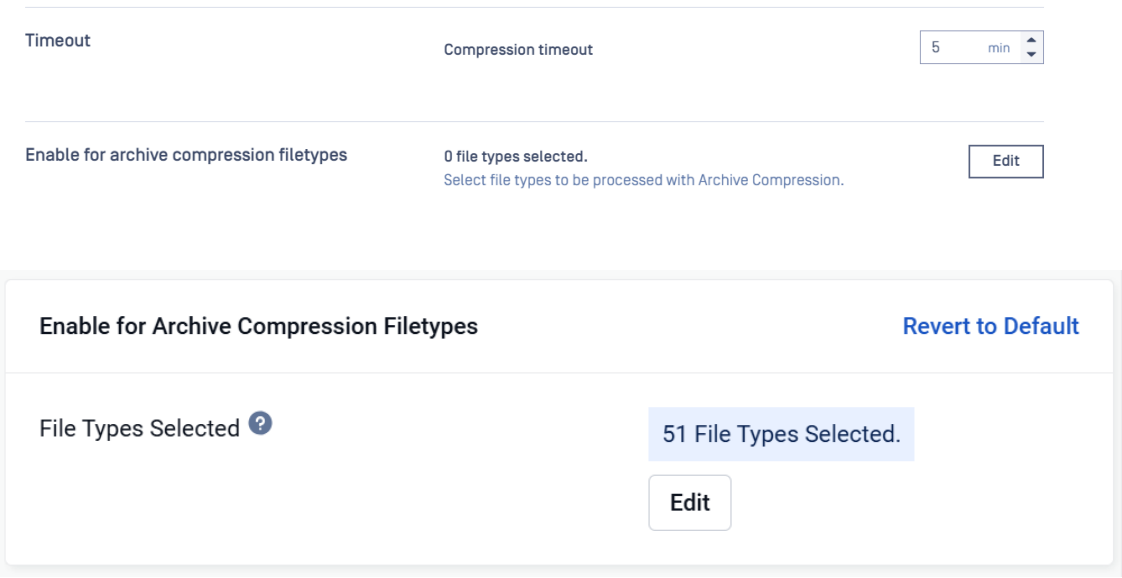
To enable file types to be processed



To enable Archive file types sanitization, go to the Compression tab



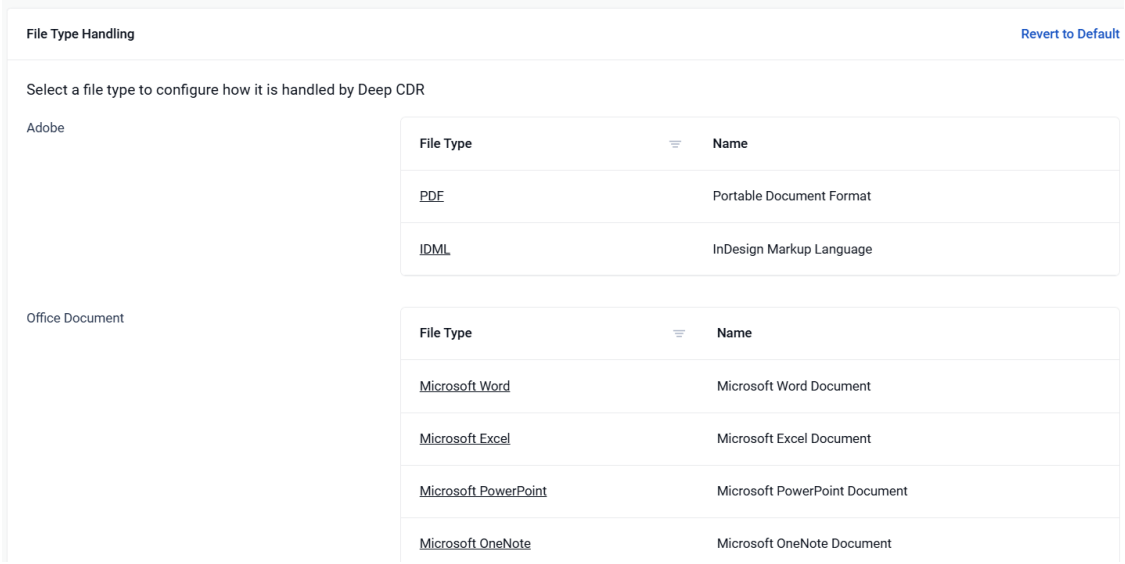
Update the "Enable for archive compression filetypes"



Advanced Configurations

Deep CDR can be configured via workflow or rules within administrator management console. For each file type, it is customizable via module configuration, which objects to remove. For example, you can configure to remove macro while keeping hyperlinks.

The engine configuration is accessible under **Workflow Management > Workflow > {Workflow name} > Deep CDR > Advanced configuration > File type handling > {File type}**



PDF

- **Remove Macro**: Remove JavaScript and document open action
- **Remove Embedded Object**: Remove embedded objects including attachments, embedded files, etc. Applicable when recursive sanitization is not performed.
  - **Excluded File Types**: List of file types, separated by commas, of which files are skipped from the removal.
- **Recursive Level**: Set to 0 to disable recursive sanitization
- **Process Hyperlink Behavior**: Remove hyperlink annotations and change hyperlinks in text (plain text strings that PDF readers recognize as hyperlinks and highlight automatically)
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink annotations only: Text links may still be highlighted by PDF readers
  - Remove hyperlink annotations and text links: Text links will not be clickable
  - Remove hyperlink annotations and prevent clickable links auto-turning: Prevent clickable links auto-turning feature of readers by changing "." to "[.]"
  - Remove hyperlink annotations and leave them in text form
  - Add hyperlink prefix
  - Return list of hyperlinks: Do nothing with hyperlink but list it in returned message
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Display Hyperlinks in Comments**: Add comment annotations to show the original hyperlinks along with other actions on the hyperlinks. Not applicable to hyperlinks in XFA form
- **Process Image**
  - **Process Raw Image**: Raw images are images that cannot be extracted to be standalone like normal images. They usually require large resources to process.
  - **Process QR code**: Hyperlink in QR code is detected and reported.

- **Image quality**: Image quality is applied for jpg image
  - Optimized
  - As original
  - Balanced
- **Remove Metadata**
- **Remove Embedded Font**: Remove all embedded fonts; may break some non-English content (like Hebrew or Arabic)
- **Process Form**: Process operational form fields
- **Remove Web Capture Content**: Remove Web Capture content database objects (/IDS, /URLS) and related data (SpiderInfo and capture command)
- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues
- **Skip Signed File**: Skip processing PDF file with digital signature
  - **Skip Only if Signature is Valid**: Only skip when PDF file is validated and valid against digital signature. Only applicable when Validate digital signature is enabled.
- **Validate digital signature**: Check file integrity to ensure the content has not been modified.
  - **Validate signer's certificate**: Confirm the signer is trusted by checking their certificate with known Certificate Authorities (CAs).
    - **Trusted CAs directory**: Path to directory containing only trusted root CAs.
- **Process 3D object**: Process 3D object
- **Process PostScript Object**: PostScript object data are set to empty and related streams are removed
- **Process XSS in text fields**: Detect and sanitize XSS injection in text fields
- **Remove User-Defined Named Action**: Remove the user-defined Named action
  - **Allowed Named Actions**: List of Named actions should be allowed
- **Process concatenated PDF**: Detect concatenated PDF and sanitize the last PDF
- **Convert to rasterized PDF**
  - **Scale with custom ratio**: Scale rasterized images with ratio to 4x page dimension rather than relying on Image quality
    - **Custom image ratio**

ISG-Compliant PDF

- **Remove Text Annotation (ISG PDF 5.39)**: Remove text annotations
- **Remove Popup Annotation (ISG PDF 5.43)**: Remove popup annotations

- **Remove Watermark Annotation (ISG PDF 5.47)**: Remove watermark annotations

PDF to Image

- **Image quality**: Image quality is applied to converted image
  - Very low: Output image dimension is approximate A8 page size at 300dpi
  - Low: Output image dimension is approximate A5 page size at 300dpi
  - Medium: Output image dimension is approximate A3 page size at 300dpi
  - High: Output image dimension is approximate A2 page size at 300dpi

Office Document

Word

- **Process Macro**: Process macro, DDE protocol and document open action in DOC/DOT/DOCM/DOTM
  - Do nothing
  - Remove all macro
  - Skip allowlisted macro: Skip macro listed in Macro Hashes setting
- **Macro Hashes**: List of macro SHA256 hashes to allowlist. Add prefix "filepath:" to allow loading newline-separated hashes from file. Applicable when choosing 'Skip allowlisted macro'
- **Skip Signed Macro**: Signed macro is detected and validated against its signature and excluded from removal if valid. Applicable to DOCM and DOTM
- **Remove Custom XML**
- **Remove Chart**: Only charts that are not recursively sanitized, due to Recursive Level set to 0 or recursive sanitization failure, are removed
- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files,...
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
  - Return list of hyperlinks: Do nothing with hyperlink but list it in returned message
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
  - **Sanitize QR code**: Hyperlink in QR code is detected and processed based on Image Configuration
  - **Allowlist External Image**: Skip external image listed in Allowlist Pattern setting

  - **Allowlist Pattern**: List of regular expressions for paths of external image to be allowlisted, e.g., "file:///D:/opswat/sample.*.jpg$". Add prefix "filepath:" to allow loading newline-separated patterns from file

- **Process ASCII-based QR code**: QR code composed by ASCII characters is detected and reported. The detection is highly resource-consumed and may degrade performance.
- **Remove Comment**: Remove Comment is applied to DOCX family (DOCX, DOTX, DOCM, etc.) only
- **Remove Revision**: Remove Revision is applied to DOCX family (DOCX, DOTX, DOCM, etc.) only
- **Remove Metadata**
- **Remove Embedded Font**
- **Remove Hidden Text**: Remove texts that were hidden in the document
- **Remove Smart Tag**: Remove Smart Tags which is an Office feature that associates specific actions with text content matching a certain pattern. Applicable to DOCX family (DOCX, DOTX, DOCM, etc.) only
- **Cleanup Unused Resources**: Remove unused styles and lists from DOC/DOT files
- **Recursive Level**: Set to 0 to disable recursive sanitization
- **Excluded File Types for Recursion**: List of file types that are not applied to recursive sanitization. Applicable when the 'Recursive Level' value is greater than 0
- **Skip Signed File**: Skip processing Word file with digital signature
- **Validate Document Properties**: Enable validation on Document Properties through syntactic checks. Applicable to DOCX family and XML-DOCX
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure

- **Validate Template Name**: Perform validation on Template Name including syntactic checks
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure

- **Validate Header and Footer**: Perform validation on Header and Footer including syntactic checks
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure

- **Validate structural XML through syntactic checks**: Enable validation on structural XML. Applicable to DOCX family and XML-DOCX
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result

  - Success
  - Failure

- **Validate Zip Structure**: Perform validation on Zip Structure including syntactic checks
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure

- **Convert to rasterized PDF**

ISG-Compliant Word

- **Remove Field Code (ISG Office 2007 4.1 and 4.18)**: Only field codes DATABASE, LASTSAVEDBY, AUTHOR, USERINITIALS, USERNAME are supported
- **Remove Document-Specific Image Metadata (ISG Office 2007 4.17)**: Remove image metadata stored in document structure rather than the image itself
- **Remove Custom Properties (ISG Office 2007 4.9)**: Remove all custom properties in the document
  - **Allowed Custom Properties**: A list of custom properties to preserve. Any custom property whose name and value pair matches an entry in this list will be preserved in the document.
    - **Property Name**: Name of the custom property to be preserved
    - **Property Value**: Value of the custom property to be preserved

- **Remove Comment-Related Data (ISG Office 2007 4.3)**: Remove comment-related data. Applicable when Remove Comment is enabled
- **Remove Image (ISG Office 2007 4.4)**
- **Remove embedded object completely (ISG Office 2007 4.4)**: Remove usage, definition, and everything related to embedded objects. Applicable when Remove Embedded Object is enabled
- **Remove Structured Document Tags (ISG Office 2007 4.21)**: Remove custom markup and content control elements (in-text comments, date pickers, drop-downs, etc.)

Excel

- **Process Macro**: Process macro and document open action in XLS/XLT/XLSM/XLTM/XLSB/XLAM
  - Do nothing
  - Remove all macro
  - Skip allowlisted macro: Skip macro listed in Macro Hashes setting

- **Macro Hashes**: List of macro SHA256 hashes to allowlist. Add prefix "filepath:" to allow loading newline-separated hashes from file. Applicable when choosing 'Skip allowlisted macro'

- **Skip Signed Macro**: Signed macro is detected and validated against its signature and excluded from removal if valid. Applicable to XLSM, XLTM and XLAM
- **Remove Custom XML**
- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files,...
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
  - Return list of hyperlinks: Do nothing with hyperlink but list it in returned message
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
  - **Sanitize QR code**: Hyperlink in QR code is detected and processed based on Image Configuration
  - **Allowlist External Image**: Skip external image listed in Allowlist Pattern setting. Applicable to Excel 2007 only
  - **Allowlist Pattern**: List of regular expressions for paths of external image to be allowlisted, e.g., "file:///D:/opswat/sample.*.jpg$". Add prefix "filepath:" to allow loading newline-separated patterns from file
- **Remove Comment**: Remove Comment is applied to XLSX family (XLSX, XLSM, XLSB, etc.) only
- **Remove Revision**
- **Remove Metadata**
- **Remove Smart Tag**: Remove Smart Tags which is an Office feature that associates specific actions with text content matching a certain pattern. Applicable to XLSX family (XLSX, XLSB, XLSM, etc.) only
- **Cleanup Unused Resources**: Remove unused styles from XLS/XLT files
- **Recursive Level**: Set to 0 to disable recursive sanitization
- **Remove External Content**: Remove DDE (Dynamic Data Exchange) protocol, external sheet and external data connection
- **Skip Signed File**: Skip processing Excel file with digital signature
- **Validate Document Properties**: Enable validation on Document Properties through syntactic checks. Applicable to XLSX family
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure
- **Validate Header and Footer**: Perform validation on Header and Footer including syntactic checks. Applicable to XLSX family
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result

    - Success
    - Failure

- **Validate structural XML through syntactic checks**: Enable validation on structural XML. Applicable to XLSX family
    - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
        - Success
        - Failure

- **Validate Zip Structure**: Perform validation on Zip Structure including syntactic checks
    - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
        - Success
        - Failure

- **Convert to rasterized PDF**

ISG-Compliant Excel

- **Remove Document-Specific Image Metadata (ISG Office 2007 4.17)**: Remove image metadata stored in document structure rather than the image itself
- **Remove Custom Properties (ISG Office 2007 4.9)**: Remove all custom properties in the document
    - **Allowed Custom Properties**: A list of custom properties to preserve. Any custom property whose name and value pair matches an entry in this list will be preserved in the document.
        - **Property Name**: Name of the custom property to be preserved
        - **Property Value**: Value of the custom property to be preserved

- **Remove Comment-Related Data (ISG Office 2007 4.3)**: Remove comment-related data. Applicable when Remove Comment is enabled
- **Remove embedded object completely (ISG Office 2007 4.4)**: Remove usage, definition, and everything related to embedded objects. Applicable when Remove Embedded Object is enabled

PowerPoint

- **Process Macro**: Process macro, DDE protocol and document open action in PPT/PPS/POT/PPTM/PPSM/SLDX/SLDM
    - Do nothing
    - Remove all macro
    - Skip allowlisted macro: Skip macro listed in Macro Hashes setting

- **Macro Hashes**: List of macro SHA256 hashes to allowlist. Add prefix "filepath:" to allow loading newline-separated hashes from file. Applicable when choosing 'Skip allowlisted macro'
- **Skip Signed Macro**: Signed macro is detected and validated against its signature and excluded from removal if valid. Applicable to PPTM, PPSM, POTM, PPAM and SLDM
- **Remove Custom XML**
- **Remove Chart**: Only charts that are not recursively sanitized, due to Recursive Level set to 0 or recursive sanitization failure, are removed
- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files,...
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
  - Return list of hyperlinks: Do nothing with hyperlink but list it in returned message
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks (not internal links). Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
  - **Sanitize QR code**: Hyperlink in QR code is detected and processed based on Image Configuration
  - **Allowlist External Image**: Skip external image listed in Allowlist Pattern setting. Applicable to PowerPoint 2007 only
  - **Allowlist Pattern**: List of regular expressions for paths of external image to be allowlisted, e.g., "file:///D:/opswat/sample.*.jpg$". Add prefix "filepath:" to allow loading newline-separated patterns from file
- **Remove Comment**: Remove Comment is applied to PPTX family (PPTX, PPTM, PPSX, etc.) only
- **Remove Metadata**
- **Remove Embedded Font**
- **Remove Smart Tag**: Remove Smart Tags which is an Office feature that associates specific actions with text content matching a certain pattern. Applicable to PPTX family (PPTX, PPTM, etc.) only
- **Recursive Level**: Set to 0 to disable recursive sanitization
- **Skip Signed File**: Skip processing PowerPoint file with digital signature
- **Validate Document Properties**: Enable validation on Document Properties through syntactic checks. Applicable to PPTX family and XML-PPTX
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure
- **Validate structural XML through syntactic checks**: Enable validation on structural XML. Applicable to PPTX family

and XML-PPTX

- **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
  - Success
  - Failure

- **Validate Zip Structure**: Perform validation on Zip Structure including syntactic checks
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure

ISG-Compliant PowerPoint

- **Remove Document-Specific Image Metadata (ISG Office 2007 4.17)**: Remove image metadata stored in document structure rather than the image itself
- **Remove Custom Properties (ISG Office 2007 4.9)**: Remove all custom properties in the document
  - **Allowed Custom Properties**: A list of custom properties to preserve. Any custom property whose name and value pair matches an entry in this list will be preserved in the document.
    - **Property Name**: Name of the custom property to be preserved
    - **Property Value**: Value of the custom property to be preserved

- **Remove Comment-Related Data (ISG Office 2007 4.3)**: Remove comment-related data. Applicable when Remove Comment is enabled

OneNote

- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files, etc.
- **Process Image**
- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink
  - Add hyperlink prefix

- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

RTF

- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files,...

- **Remove Metadata**
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Remove Embedded HTML**: Remove HTML tags containing malicious HTML nodes
  - Do nothing
  - Remove HTML tags containing malicious nodes
  - Remove all HTML tags
- **Process Image**
- **Recursive Level**: Set to 0 to disable recursive sanitization
- **Process font table**: Remove embedded font table
  - **Limit of font count in font table**: Font table exceeding this font count will be removed

Visio

- **Remove Macro**
- **Remove Embedded Object**
- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
- **Remove Metadata**: Remove Metadata is not applied to Visio XML
- **Recursive Level**: Set to 0 to disable recursive sanitization (apply for Visio XML - vdx, vsx, vtx)

Project

- **Remove Embedded Object**
- **Remove Macro**

OpenDocument

OpenDocument Text

- **Remove Macro**: Remove JavaScript and macro
- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files, flash files,...
- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink: Hyperlinks will not be clickable
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
- **Remove Metadata**
- **Remove Embedded Font**
- **Remove Comment**
- **Remove Hidden Text**
- **Remove Revision**
- **Recursive Level**: Set to 0 to disable recursive sanitization

OpenDocument Sheet

- **Remove Macro**
- **Process Image**
- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink: Hyperlinks will not be clickable
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Remove Metadata**
- **Remove Embedded Font**
- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files, flash files,...
- **Recursive Level**: Set to 0 to disable recursive sanitization

OpenDocument Presentation

- **Process Image**: Sanitize attached images
- **Remove Metadata**
- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files, flash files,...

- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink: Hyperlinks will not be clickable
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Remove Embedded Font**
- **Remove Macro**
- **Recursive Level**: Set to 0 to disable recursive sanitization

OpenDocument Drawing

- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files, flash files,...
- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink: Hyperlinks will not be clickable
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
- **Remove Metadata**
- **Remove Embedded Font**
- **Remove Comment**
- **Remove Macro**
- **Recursive Level**: Set to 0 to disable recursive sanitization

JTD

- **Remove Macro**
- **Remove Embedded Object**
- **Remove Hyperlink**: Make links unclickable
- **Process Image**
- **Remove Embedded Font**

Hangul Word

- **Remove Macro**

- **Remove Embedded Object**: Remove all embedded objects including attachments, embedded files, flash files,...
- **Process RTF Document**: Sanitize embedded RTF files, remove if fail and Remove Embedded Object is enabled. Applicable to HWP only.
- **Process Hyperlink Behavior**
  - Do nothing: Keep hyperlinks as-is
  - Remove hyperlink: Hyperlinks will not be clickable
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
- **Process Chart Behavior**
  - Do nothing
  - Remove chart
- **Remove Metadata**: Remove Metadata is applicable to HWPX only
- **Recursive Level**: Set to 0 to disable recursive sanitization. Applicable to HWPX only.
- **Process Embedded Font**: May break some non-English content (like Hebrew or Arabic) if embedded fonts are removed. Applicable to HWPX only
  - Do nothing
  - Remove embedded font

Hancom Cell

- **Process Macro**
  - Do nothing
  - Remove all macro
  - Skip allowlisted macro: Skip macro listed in Macro Hashes setting
- **Macro Hashes**: List of macro SHA256 hashes to allowlist. Add prefix "filepath:" to allow loading newline-separated hashes from file. Applicable when choosing 'Skip allowlisted macro'
- **Remove Embedded Object**
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

- **Process Image**
  - **Sanitize QR code**: Hyperlink in QR code is detected and processed based on Image Configuration
  - **Allowlist External Image**: Skip external image listed in Allowlist Pattern setting
  - **Allowlist Pattern**: List of regular expressions for paths of external image to be allowlisted, e.g., "file:///D:/opswat/sample.*.jpg$". Add prefix "filepath:" to allow loading newline-separated patterns from file
- **Remove Comment**
- **Remove Revision**
- **Remove Metadata**
- **Recursive Level**: Set to 0 to disable recursive sanitization

Hancom Show

- **Remove Embedded Object**
- **Remove Chart**: Only charts that are not recursively sanitized, due to Recursive Level set to 0 or recursive sanitization failure, are removed
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**
  - **Sanitize QR code**: Hyperlink in QR code is detected and processed based on Image Configuration
  - **Allowlist External Image**: Skip external image listed in Allowlist Pattern setting
  - **Allowlist Pattern**: List of regular expressions for paths of external image to be allowlisted, e.g., "file:///D:/opswat/sample.*.jpg$". Add prefix "filepath:" to allow loading newline-separated patterns from file
- **Remove Comment**
- **Remove Metadata**
- **Recursive Level**: Set to 0 to disable recursive sanitization

Image

Image

- **Sanitize QR code**: Hyperlink in QR code is detected and processed
  - **Applicable File Types**

  - JPG
  - PNG
  - BMP
- **Process Hyperlink Behavior**
  - Add hyperlink prefix
  - Return list of hyperlinks: Do nothing with hyperlink but list it in returned message
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

- **Remove Metadata**
- **Image Size Limit (MB)**: Maximum decompressed image size on memory (for JPG, PNG, BMP and TGA), set to 0 for unlimited size
- **Remove ICC Profile**: Remove ICC profile in JPG files
- **Image quality**: Image quality is applied for jpg image
  - As original
  - Balanced
- **GIF Layer Count Limit**: Maximum normalized layer count of GIF, applicable to GIF only. GIF files with layer count exceeding this value will be skipped from processing.
- **TIFF Size Limit (MB)**: Maximum decompressed image size on memory for TIFF, set to 0 for unlimited size
- **Restrict processing of big TIFF files in single thread**: TIFF files with decompressed image size greater than or equal to restricted size will be forced in single thread
  - **Restricted size (MB)**
- **Preserve GeoTIFF tags**: Preserve private TIFF tags storing georeferencing information
- **Deep Sanitization**: Deeply sanitize image pixel content to prevent stegomalware

Image Conversion

- **BMP Output Size**: When width or height of an image exceeds a threshold, it will be resized by the specified percentage. Applicable when converting TIFF to BMP. Input the Dimension threshold (px) in the first box and Resize ratio (%) in the second box.

SVG

- **Remove JavaScript**: Remove JavaScript code that may harm the system.
- **Remove CDATA**: Remove character data that may embed harmful program.
- **Remove Injection**: Remove SVG injection to prevent execution of harmful program.
- **Process Image**: Process raster images included inside SVG file.

- **Process Embedded Image**: Process raster images and vector graphiccs within the SVG files.
- **Process Foreign Object**
  - Do nothing
  - Foreign object data is recursively sanitized: Foreign object data is recursively sanitized. If recursive sanitization fails, the object is removed
  - Remove foreign object

DWF

- **Remove Embedded Font**: Remove all embedded fonts
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Image**

DWG

- **Remove Macro**

JWW

- **Process Image**: Process images in the JWW file

Text

TXT

- **Process Hyperlink Behavior**
  - Do nothing
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Preferred Encodings**: Preferred encodings for auto encoding detection. Supported: UTF-8, UTF-16BE, UTF-16, ISO-8859-1 and Shift-JIS
  - UTF-8
  - UTF-16BE

- UTF-16
- ISO-8859-1: Hebrew language, ISO-8859-1
- Shift-JIS: Japanese language, Shift-JIS

- **Remove Invisible Text**: Remove invisible UTF-8 characters
  - **Categories**: Categories of characters to remove
    - Format Characters
    - Control Characters
    - Private Use Characters
    - Unassigned Characters
    - Variation Selectors Character
  - **Exceptional Characters**: Unicode of characters in hexadecimal to be excluded from removal. Characters which are necessary for visual content of text, e.g., line feed (000A) should be added.
- **Process ASCII-based QR code**: QR code composed by ASCII characters is detected and reported. The detection is highly resource-consumed and may degrade performance.

CSV

- **Process Formula**
  - Do nothing
  - Sanitize formulas with function
  - Sanitize all formulas

Markup

HTML

- **Remove Script**
- **Remove Object**
- **Remove Applet**
- **Process Form**: Remove actions in forms
- **Remove Comment**
  - **Remove Conditional Comment**
- **Remove Iframe**
- **Process Hyperlink Behavior**

- Do nothing
- Display hyperlink in text form
- Remove hyperlink
- Display hyperlink with only domain
- Return list of hyperlinks
- Add hyperlink prefix
- Replace hyperlink by text
- Do nothing except remove JavaScript
- Prevent hyperlinks from being clickable: Prevent clickable links by changing "." to "[.]"

- **Process Text Link**: Process hyperlinks in text format. Only applicable when choosing 'Add hyperlink prefix' or 'Prevent hyperlinks from being clickable'
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Replaced Text**: Hyperlinks will be replaced with this text (English characters only). Applicable when choosing 'Replace hyperlink by text'
- **Display Image Source**: If enabled, copy image links ('http(s)://', 'ftp://', 'file://', 'www.', 'ms-its:') in 'src' attribute and place them outside
- **Process Embedded Image**: Process images that are embedded directly in HTML files in Base64 format
- **Accept Multiple HTML Documents**: Accept documents with multiple separate HTML documents
  - **Include Documents without HTML Tag**: Accept documents even if they do not have HTML tag
- **Process Zero Font Text**: Process node with zero font text
- **Use Numeric Entity**: HTML special characters are escaped with numeric entities
- **Prefer Pretty Print**: Create HTML output with spacing in markup code where possible. Spacing visible to users when displayed will not be affected.
- **Process ASCII-based QR code**: QR code composed by ASCII characters is detected and reported. The detection is highly resource-consumed and may degrade performance.
- **Remove Unapproved Tags**: If enabled, only HTML tags specified in 'List of Approved Tags' are kept in output HTML files. The others are removed during the processing.
  - **List of Approved Tags**: A list of HTML tags that can be kept in output HTML files.

HTA

- **Process Script**
  - Do nothing
  - Remove script with Windows shell commands only
  - Remove all script

- **Remove Object**
- **Remove Applet**
- **Process Form**: Remove actions in forms
- **Remove Comment**
- **Remove Iframe**
- **Process Hyperlink Behavior**
  - Do nothing
  - Display hyperlink in text form
  - Remove hyperlink
  - Display hyperlink with only domain
  - Return list of hyperlinks
  - Add hyperlink prefix
  - Replace hyperlink by text
  - Do nothing except remove JavaScript
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Replaced Text**: Hyperlinks will be replaced with this text (English characters only). Applicable when choosing 'Replace hyperlink by text'
- **Display Image Source**: If enabled, copy image links ('http(s)://', 'ftp://', 'file://', 'www.') in 'src' attribute and place them outside
- **Process Embedded Image**: Process images that are embedded directly in the HTML file
- **Accept Multiple Html Tags**: Accept documents with multiple separate html tags
- **Process zero font text**: Process node with zero font text

MHT

- **Remove Embedded Object**: Remove embedded objects including attachments, embedded files, etc. Applicable when recursive sanitization is not performed.

XML

- **Remove Macro**: Remove JavaScript
- **Remove CDATA**
  - Do nothing
  - Remove only if it contains script
  - Remove if it contains script or event
  - Remove all

- **Remove Injection**
- **Process Base64 encoded data**: Process XML values that are Base64 encoded data
  - **Data following "data" URL scheme**: Base64 encoded data that start with "data" URL scheme, e.g.,
  - **Data at certain XML path**: Regular expressions of XML path to fields to be processed, e.g., user..*Picture$
  - **Action on data when sanitization not applied**: Choose what action should be applied to data when sanitization is not supported or fails
    - Do nothing
    - Remove data
- **Validate against Schema**: Enable validation against XSD schema
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure
  - **XSD Schema Location**: Path to directory containing XSD schema files
  - **Schema File Name**: Relative path to XSD schema file name against which XML files are validated
- **Enable character encoding fallback**: When enabled, processing will be retried on failures related to encoding.
  - **Fallback encoding**: Encoding to use when retrying.
    - UTF-8
    - Shift-JIS

JSON

- **Preserve Format**: Preserve formats like whitespaces, newlines, etc.
- **Support Newline Delimited**: Support Newline Delimited JSON
- **Process Base64 encoded data**: Process JSON values that are Base64 encoded data. Setting Preserve Format becomes invalid if there are processed data
  - **Data following "data" URL scheme**: Base64 encoded data that start with "data" URL scheme, e.g.,
  - **Data at certain JSON path**: Regular expressions of JSON path to fields to be processed, e.g., user..*Picture$
  - **Action on data when sanitization not applied**: Choose what action should be applied to data when sanitization is not supported or fails
    - Do nothing
    - Remove data

ICS

- **Sanitize Inline Attachment**
  - Do nothing
  - Sanitize, remove if not supported
  - Sanitize, keep if not supported
- **Remove External Attachment**: Remove linked attachment
- **Process Hyperlink Behavior**
  - Do nothing
  - Disable hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

VCF

- **Process Hyperlink Behavior**
  - Do nothing
  - Disable hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
- **Process Hyperlink in Text Form**: Apply selected hyperlink behavior to hyperlink in text form
- **Process Base64 encoded data**: Process VCF values that are Base64 encoded data
  - **Action on data when sanitization not applied**: Choose what action should be applied to data when sanitization is not supported or fails
    - Do nothing
    - Remove data
  - **Apply to all data fields**: Apply Base64 processing to all data fields rather than just specified fields (KEY, LOGO, PHOTO, SOUND, and URL)

General Media

- **Preferred frame rate**: Preferred frame rate for video processing. Applies to all conversions to WMV
  - **Frame rate**: Preferred frame rate for video processing. Applicable when choosing 'Preferred frame rate'

- **Number of Video Conversion Threads**: Number of threads to be used for decoding and encoding in video conversion. Use 0 to let the number of threads selected automatically

MP3

- **Remove Metadata**
- **Validate audio frame data**: Perform anomaly detection on audio frame data to identify suspicious data like JavaScript
  - **Invalid data occurrence threshold**: Determine threshold for invalid data occurrence to avoid false positive. Invalid data are keywords related to JavaScript like <script, </script, exec(, etc.

AAC

- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues

AIFF

- **Remove Metadata**

WAV

- **Remove Metadata**
- **Remove Hidden Data**
- **Signal-to-noise ratio (SNR) in dB**: If remove hidden data is enabled, add random noise with the given SNR value to the audio stream
- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues

OGG

- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues
- **Remove Metadata**: If true, remove all metadata

MPEG4

- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues
- **Remove Metadata**: If true, remove all Metadata

WEBM

- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues

WMV

- **Remove Metadata**: If true, remove all Metadata

WMA

- **Remove Metadata**: If true, remove all Metadata

FLV

- **Remove Metadata**: If true, remove all Metadata

MXF

- **Strict Validation**: If false, ignore slight error in file structure, otherwise report standard compliance issues

LNK

- **Commands to remove (separated by |)**

URL

- **Hyperlink Processing**
  - Report hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

PowerBI

- **Process Image**
- **Cleanup Unused Resources**: Remove unused resources from PBIX files

TWB

- **Remove Macro**: Remove JavaScript
- **Remove Web Data Connection**: Remove suspicious web data connections
- **Process Hyperlink Behavior**
  - Do nothing

  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

XDW

- **Remove Image Annotation**: Remove all image annotations in xdw file
- **Remove Embedded Object**: Remove all embedded objects including OLE annotation
- **Process Hyperlink Behavior**
  - Do nothing
  - Remove hyperlink
  - Add hyperlink prefix
- **Hyperlink Prefix**: Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'

VSIX

- **Process Image**
- **Validate structural files through syntactic checks**: Enable validation on structural files like package.json, extension.vsixmanifest, [Content_Types].xml, *.imagemanifest, etc.
  - **Validation Effect on Sanitization Result**: Choose how failure result of validation affects sanitization result
    - Success
    - Failure
- **Process Tracked Files**: Performs sanitization on tracked files contained within the package. This applies to files in JSON, XML, and RTF formats.
- **Process Untracked Files**: Performs sanitization on untracked files contained within the package.
  - **Supported File Types**: List of file types that recursive sanitization is applied to.

SFC

- **Remove Comment**

P21

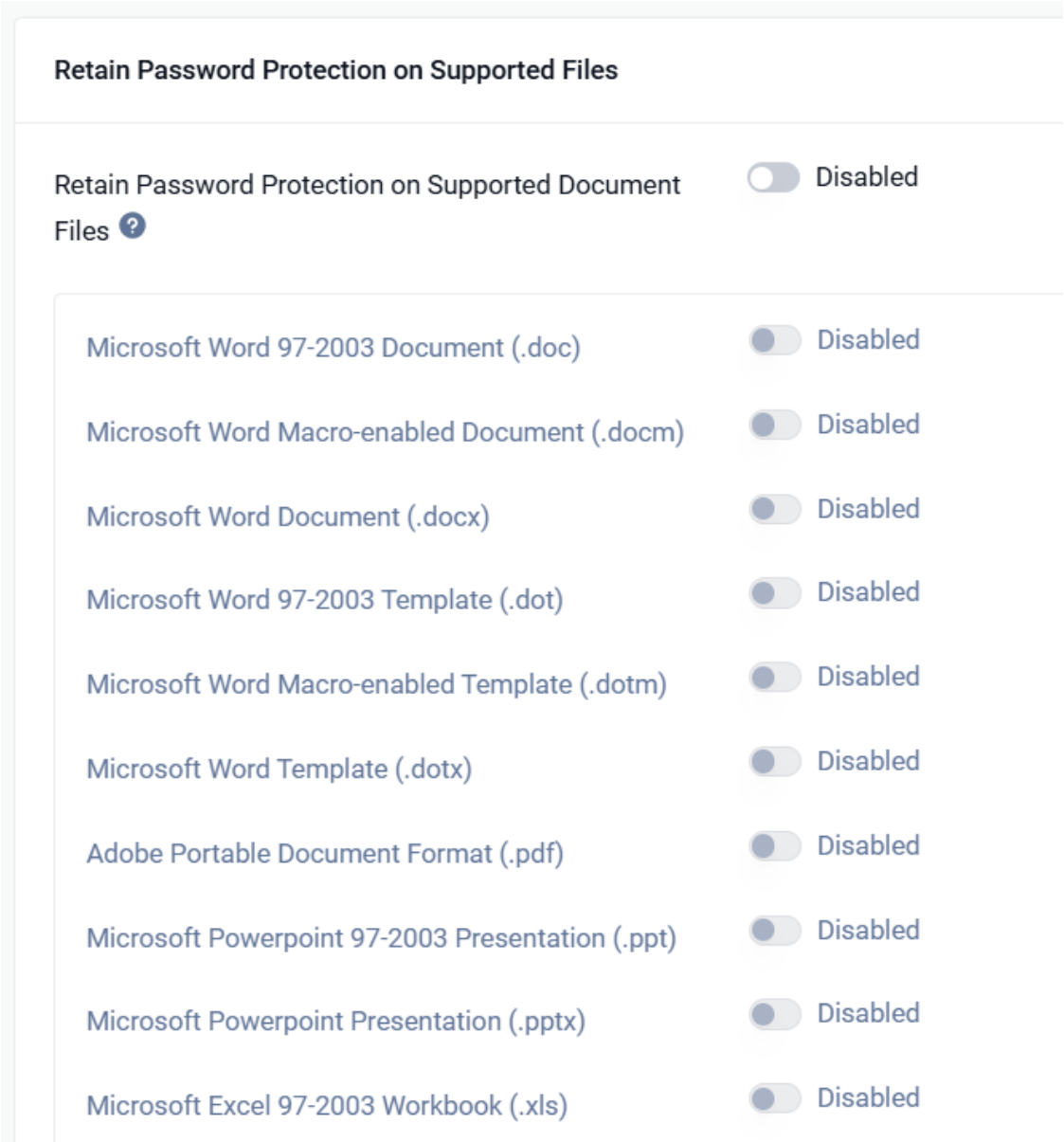
- **Remove Comment**

Retain password protection on sanitized files

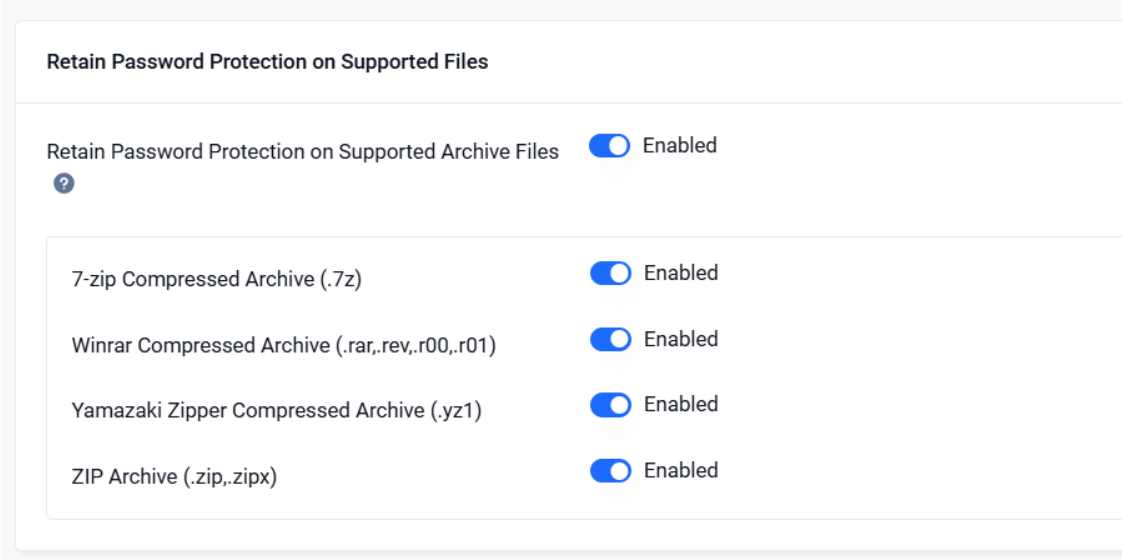
When having a password-protected encrypted file processed by Deep CDR, you can decide whether to encrypt the sanitized file with the same password or no password.

To retain password protection, modify rule under

- **Workflow Management> Workflows > Workflow Name > Deep CDR** for document files.



- **Workflow Management> Workflows > Workflow Name > Compression** for archive files.



Supported file types:

- zip, 7z, yz1 (Windows only), rar, pdf, xls, xlt, xlsx, xlsm, xltx, xltm, xlsb, ppt, pptx, doc, dot, docx, docm, dotx, dotm

Note:

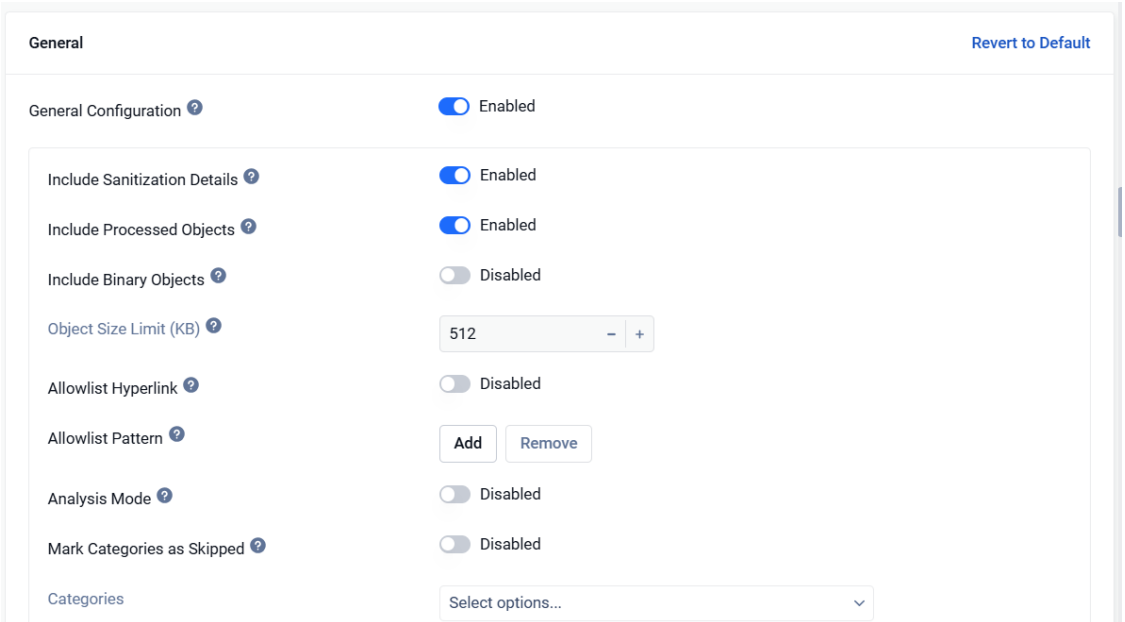
- If you enable "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" on Windows, the encryption will not work with PPTX file.

Allow Macros

When processing a file containing a macro, you can configure the Deep CDR engine to either remove the macro or keep it while sanitizing the other active objects.

Please follow the below steps to allow a specific macro.

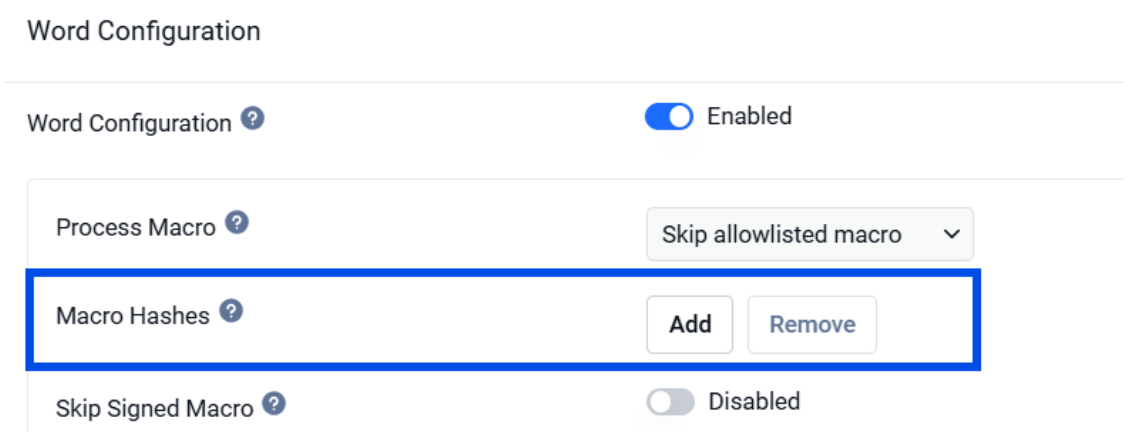
- Enable “INCLUDE PROCESSED OBJECTS” in Workflow Deep CDR Setting



- Process the file containing the Macro that you want to allowlist and check the JSON scan result. You should see the **object_details** and **object_sha256** as in the example below**:**

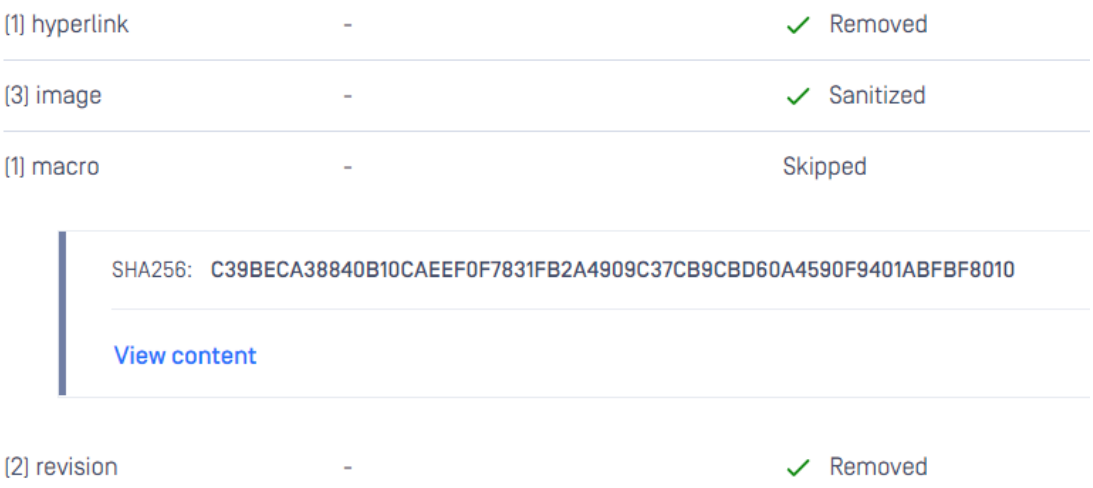

```
{
  "action": "removed",
  "count": 1,
  "object_details": [
    "Private Sub Document_Open()\r\n    TaskID = Shell(\"calc.exe\", 1)\r\nEnd Sub\r\n"
  ],
  "object_name": "macro",
  "object_sha256": [
    "c39beca38840b10caeef0f7831fb2a4909c37cb9cbd60a4590f9401abfbf8010"
  ]
},
```


- Go back to Workflow Deep CDR Setting, under Advanced, Microsoft Word, “Process Macro” select “Skip allowlisted macro” for the file type you sanitized, input the object_sha256 value into the “Macro Hashes” field.
- You can process the file again to test that your setup is working correctly. The Macro now should be skipped.



- Process the file again


```
{
  "action": "skipped",
  "count": 1,
  "object_details": [
    "Private Sub Document_Open()\r\n    TaskID = Shell(\"calc.exe\", 1)\r\nEnd Sub\r\n"
  ],
  "object_name": "macro",
  "object_sha256": [
    "c39beca38840b10caeef0f7831fb2a4909c37cb9cbd60a4590f9401abfbf8010"
  ]
},
```



Supported file type: Microsoft Office 2007

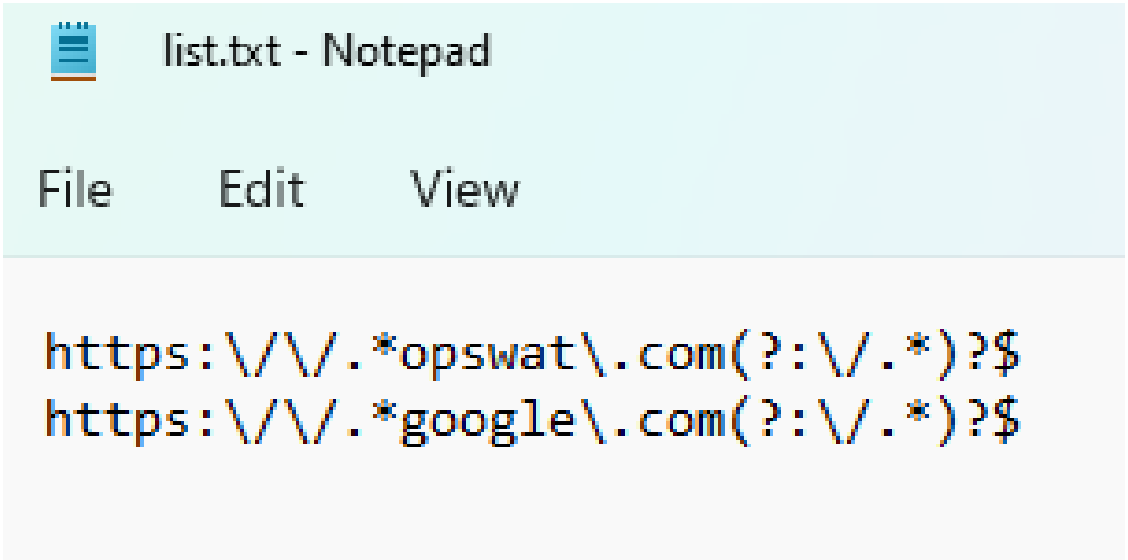
Allow Hyperlinks

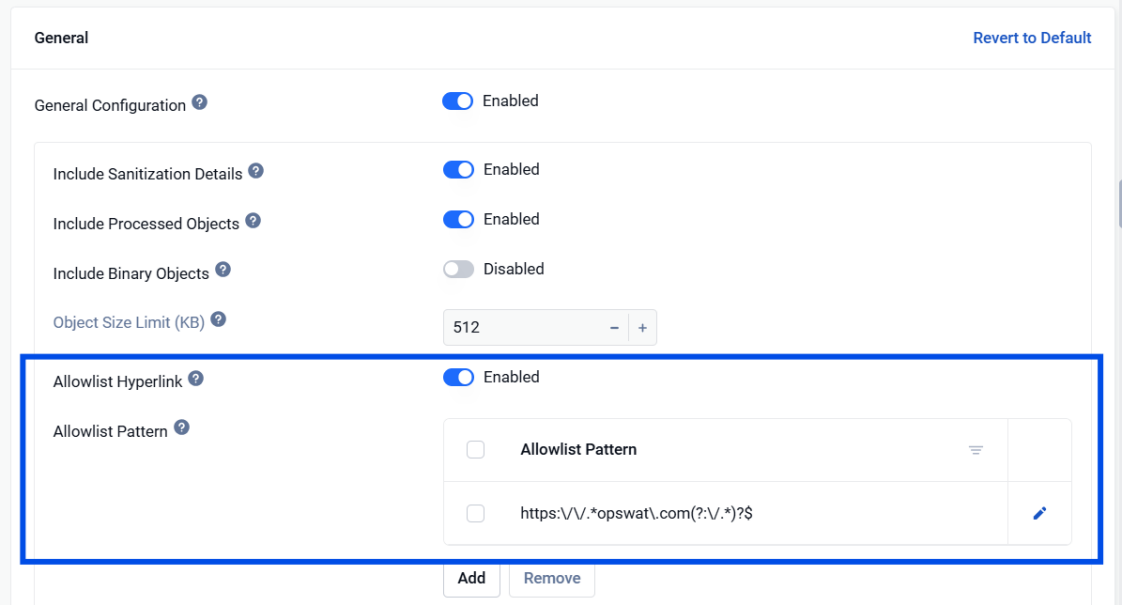
Despite hyperlinks' centrality to the user experience, connecting pages and sites with a single click, they could be a high-risk attack vector. To ensure that all potential threats in files were eliminated before reaching the users, hyperlinks should be removed during Deep CDR process by default. However, we understand that usable hyperlinks are essential and important for some businesses. Thus, we introduce a flexible configuration enabling administrators to allow hyperlinks or domains that they deem safe.

Workflow Management > Workflows > Workflow name > Enable Allowlist Hyperlink

Allowlist Pattern:

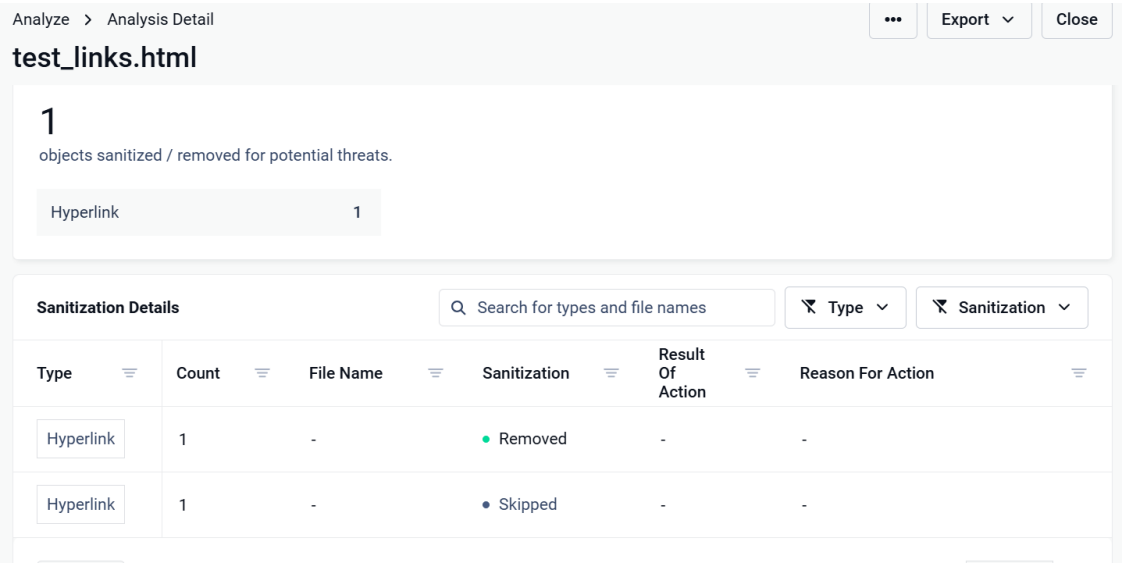
- Regular expression
- or **filepath**:<file path to a file which contains a regular expression list, line by line>, e.g.,





The above example allows all links from **opswat.com** domain/sub domain.

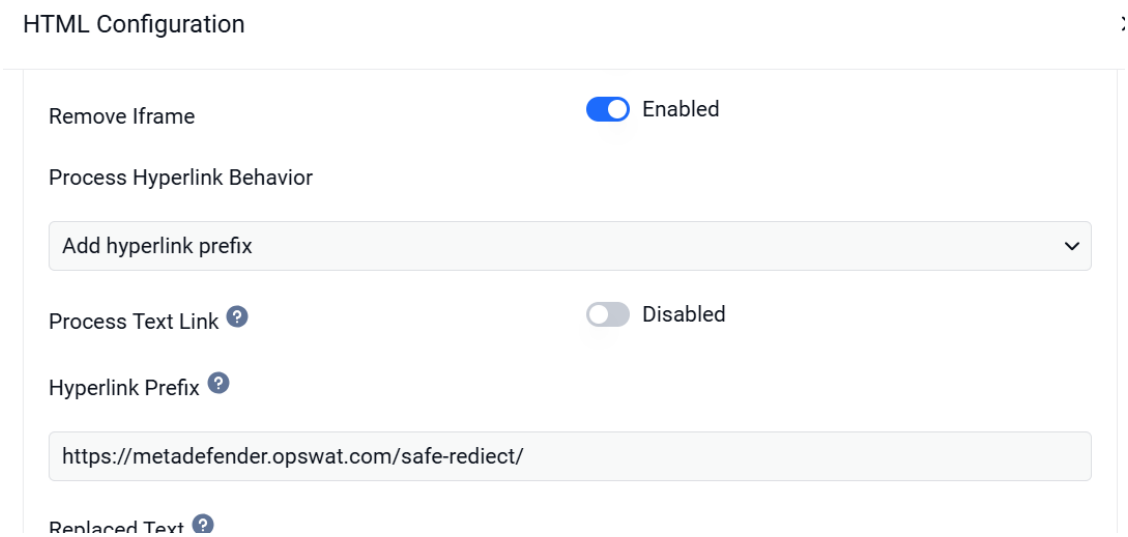
An example output when processing a file with two hyperlinks (http://malicious.com, https://docs.opswat.com/mdcore/deep-cdr) on it.



Supported file types: MS Office, HTML, Calendar

Hyperlink redirection

This option is a good choice when you have a redirection service. Deep CDR will add a prefix hyperlink to current hyperlinks. When the user clicks on a hyperlink, your inspection service will scan the link and redirect them to the original URL.



This is an example when integrating with MetaDefender Cloud Safe URL Redirect:

Original: "Please click here to contact us"

Sanitized: "Please click here to contact us"

Dynamic hyperlink redirection

The feature Dynamic hyperlink redirection allows users to pass a `container_identifier` via `engines-metadata` to generate dynamic URL output.

Container identifier serves as an identifier of container document of hyperlinks. Clients may leverage for their own purposes.

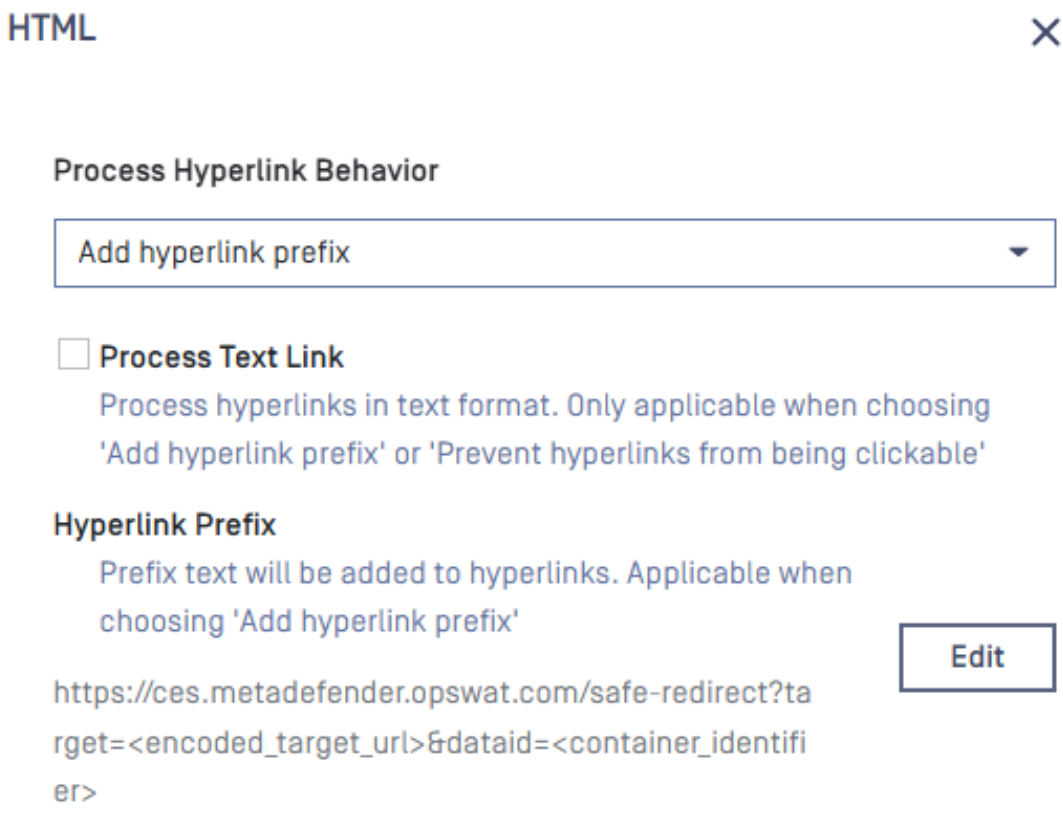
Supported file formats

This feature applies to **HTML**, **TXT**, and **RTF** file formats only.

Configuration

Configuration: **Workflow > [Workflow Name] > Deep CDR > Advanced Configuration > Others > HTML**, then:

- **Process Hyperlink Behavior:** Select how hyperlinks should be processed. Set this to **Add hyperlink prefix**.
- **Hyperlink Prefix:** Define the prefix that will be added to output hyperlinks in HTML. To use the feature, the hyperlink prefix must include two placeholders: `encoded_target_url` and `container_identifier` , e.g., `https://metadefender.opswat.com/safe-redirect?target=<encoded_target_url>&dataid=<container_identifier>`.



Now we can pass `container_identifier` via the `engines-metadata` (see Supported engines-metadata JSON) to generate dynamic output URL.

Alternatively, we can pass `hyperlink_prefix` with placeholder `<encoded_target_url>` and pre-inserted `container_identifier`. This setting takes a higher priority over the workflow setting. `hyperlink_prefix` should come along with a setting section name of a corresponding file format. Below is the list of setting sections.

File format	Setting section name
HTML	html2html
TXT	txt2txt
RTF	rtf2rtf

Code · json

```
{
      "deep_cdr": {
      "container_identifier": "69aab91a5e37baa713442af9"
    }
}
```

Code · json

```
{
      "deep_cdr": {
      "html2html.hyperlink_prefix": "https://metadefender.opswat.com/safe-redirect?
target=<encoded_target_url>&dataid=69aab91a5e37baa713442af9"
    }
}
```

Example

With an extracted hyperlink from a document, `https://www.opswat.com/docs`, the output URL will be `https://metadefender.opswat.com/safe-redirect?target=https%3A%2F%2Fwww.opswat.com%2Fdocs&dataid=69aab91a5e37baa713442af9` .

Image limit configurations

Processing image is an expensive task, it can be utilized to DDoS the system, making the system be overloaded. Having proper image limit configurations helps to protect in this case.

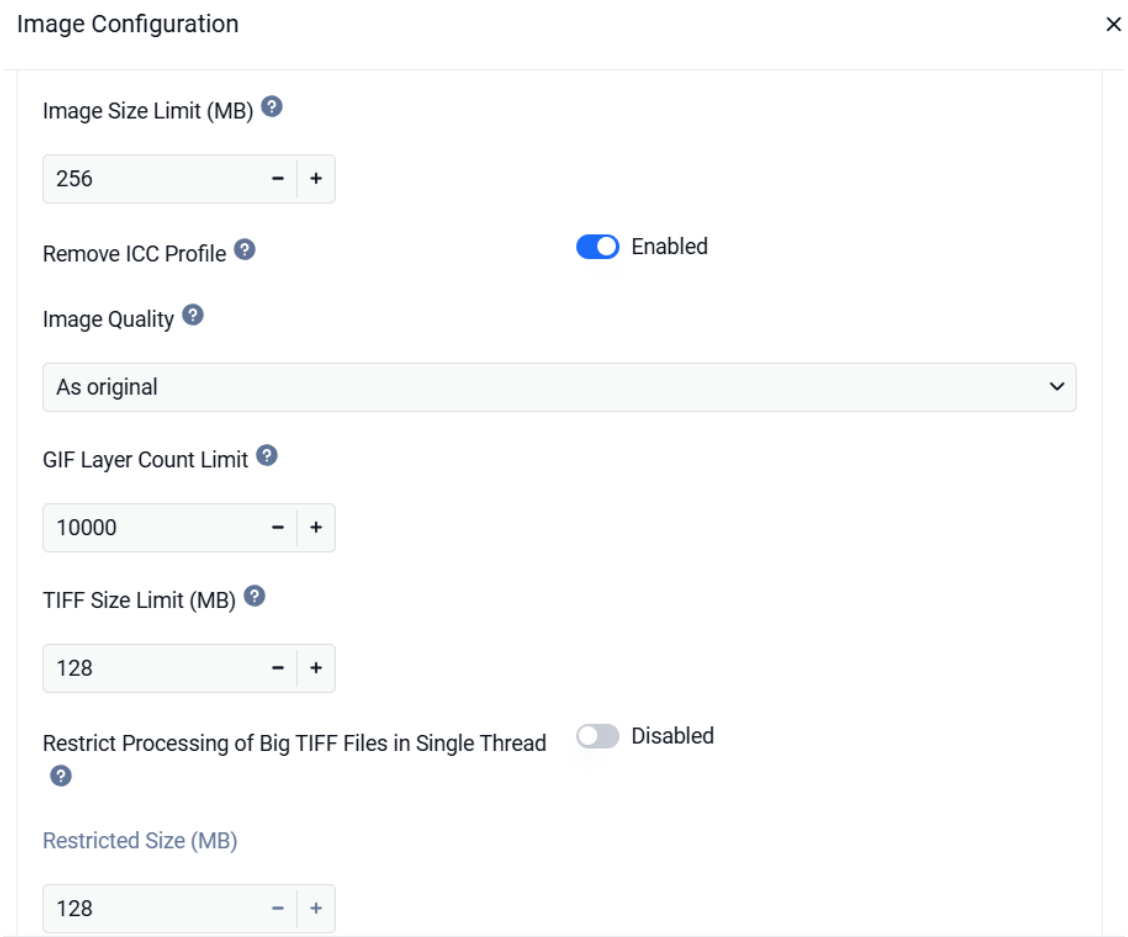


Image Size Limit (MB): A JPG/PNG file with a few byte in file size can be decompressed to become a few to a hundred MB. Therefore, this configuration is applied for the **decompressed raw data size**.

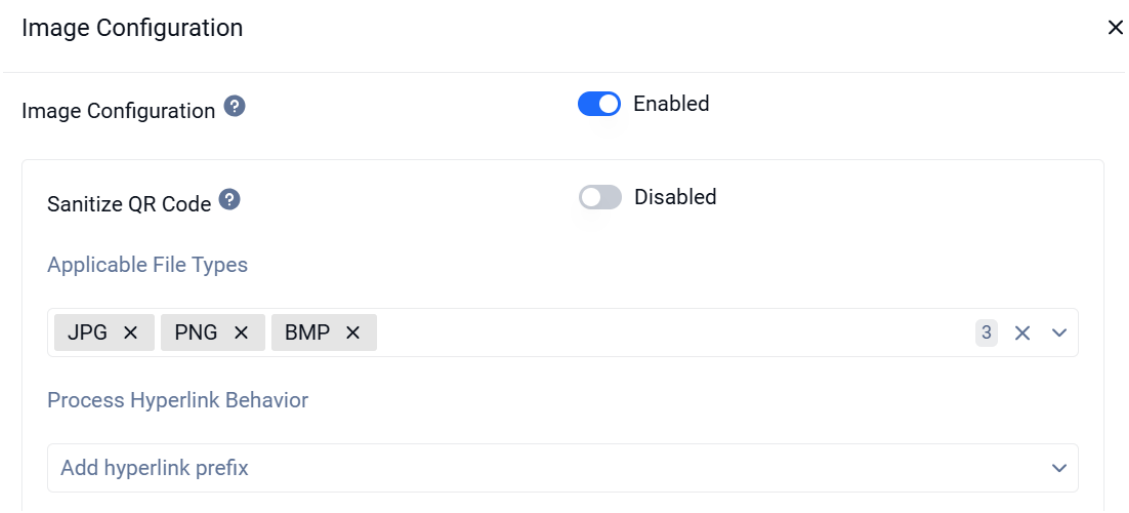
TIFF Size Limit (MB): why do we separate PNG/JPG size limit from the TIFF size limit? The TIFF file processing is expensive when it uses Group 4 compression but not in the other format. Also, the TIFF raw data size is calculated in a different way from PNG/JPG. Therefore, using the same values with JPG/PNG configuration doesn't work well. This configuration is applied for the **decompressed raw data size**.

GIF Layer Count Limit: GIF processing performance depends on not only the number of layers but also the image dimension. For example, a GIF with 40000 layers with 1x1 (Width x Height) pixels is taking more time to process than a GIF with 400 layers with 100x100 pixels. In this case, we use a "**normalized**" version that combines the number of layers and image dimension, it is not the same as the original GIF layer count.

If the values in the files are over the limitation, the sanitization result will be **Sanitization Failed**

QR code sanitization

To enable QR code sanitization, please check "Sanitize QR code" the image configuration



Supported file type: JPG, PNG, BMP, MS Office

Limitation:

- Style or rotated QR codes are changed to normal QR code.

ASCII-based QR code

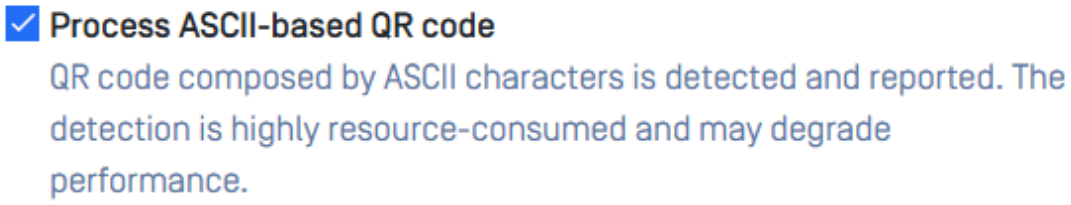
QR code can be created in a document by a combination of ASCII block characters (e.g., ▯).This tactic is designed to prevent security software from extracting malicious URLs from QR codes. Attackers can embed malicious links into the QR code and encourage users to scan the code, which would then take them to a fake page that appears to be a trusted service or application.

Deep CDR engine can detect and report the link from ASCII-based QR code to users (not sanitized).

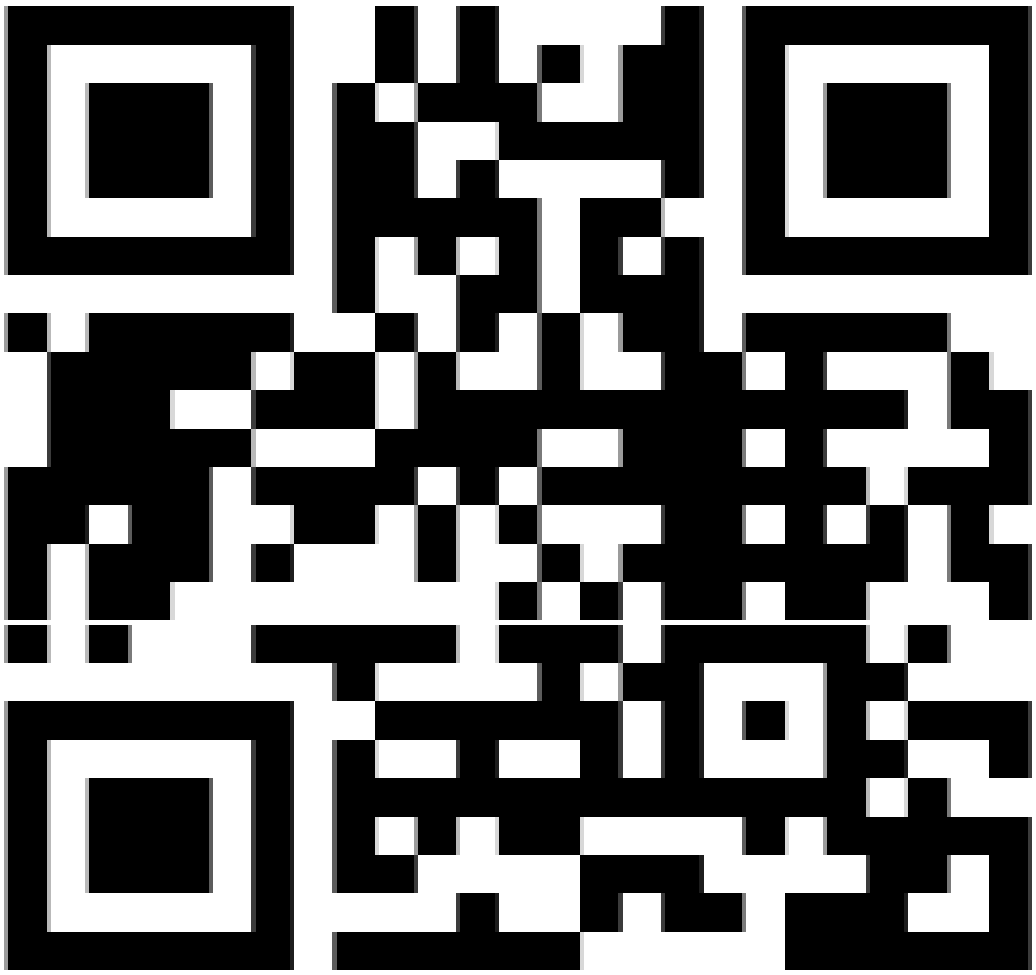
Users can configure to process QR codes that are composed by ASCII characters.

The configuration can be modified under rule:

Workflow Management> Workflows > Workflow Name > Deep CDR > Advanced configuration > File type handling



Here is an screenshot of an QR code created by ASCII block characters:



Supported file types: MS Office, RTF.

Base64-encoded data in JSON

JSON

JSON data can contain base64-encoded data within certain attributes. For example:

Code · json

```
{
  "user": {
```

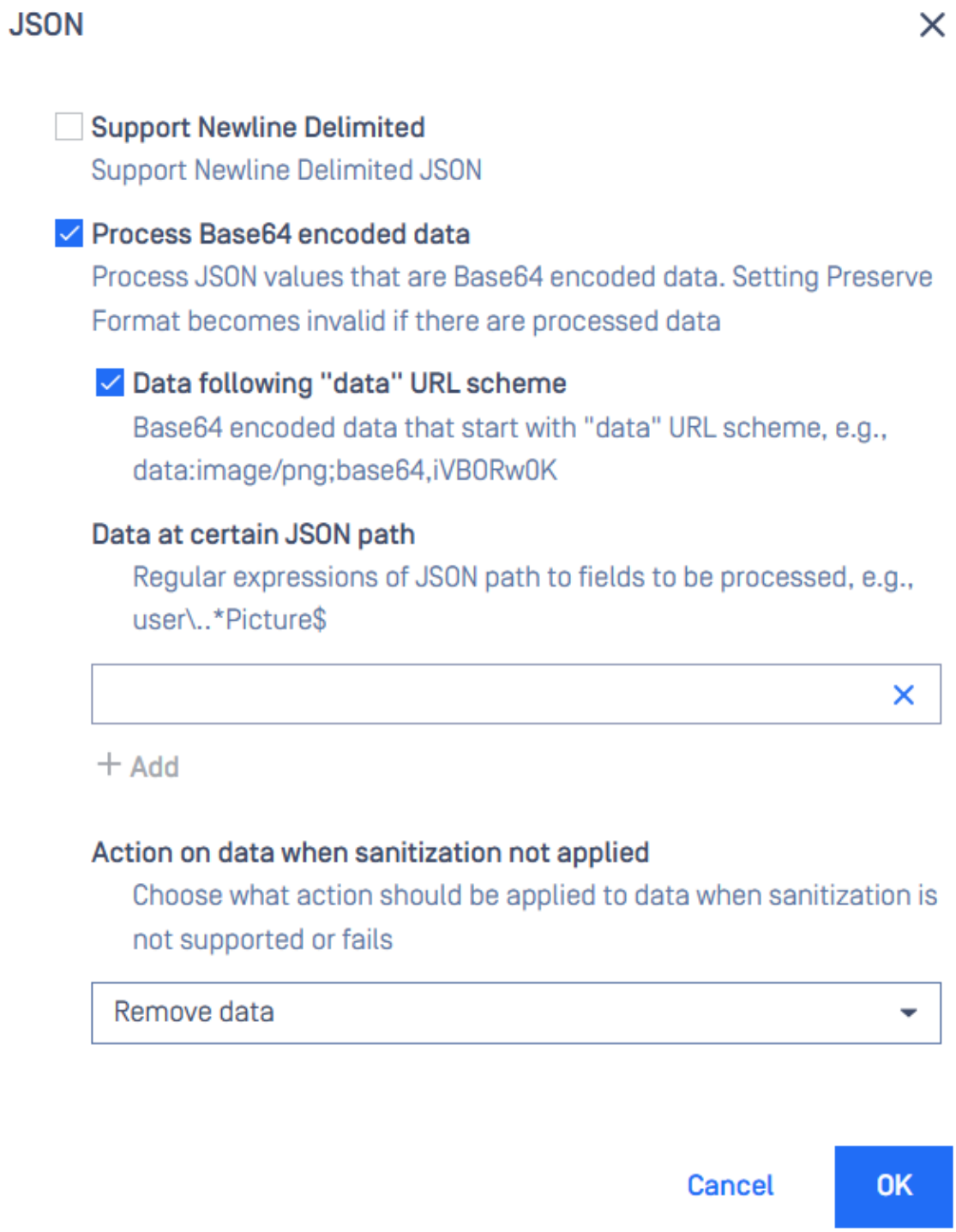
```
      "id": "12345",
      "name": "John Doe",
      "profilePicture": "...",
      "photos": [
        {
          "id": "photo1",
          "description": "A scenic view",
          "image": "asset:oc6neh5prnji..."
        },
        {
          "id": "photo2",
          "description": "A portrait",
          "image": "asset:2y8nslo98gur..."
        }
      ]
    },
    "metadata": {
      "timestamp": "2024-12-24T12:00:00Z",
      "source": "Generated Example"
    },
    "data": {
      "image1": "R0lGODlhAQABAIAAAISwxwAAACwAAAAAQABAAACAkQBADs=",
      "image2": "iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAgMAAABinRfyAA...",
      "images": [
        "/9j/4AAQSkZJRgABAQEAeAB4AAD/4QAiRXhpZgAATU0AKg...",
        "iVBORw0KGgoAAAANSUhEUgAAABAA..."
      ]
    }
  }
```


Deep CDR can be configured to perform an in-depth inspection of such data by:

- Decoding base64-encoded content.
- Recursively sanitizing the extracted data.
- Re-encoding the sanitized data in base64.
- Reinserting the processed data back into the JSON structure.

Configuration

This feature can be enabled at **Workflow Management> Workflows > [Workflow name] > Deep CDR > Advanced configuration > File type handling > Others > JSON > Process Base64 encoded data**.



- **Data following "data" URL scheme**: Base64 encoded data that start with "data" URL scheme, e.g., .
- **Data at certain JSON path**: Regular expressions of JSON path to fields to be processed, e.g., user..*Picture$.
- **Action on data when sanitization not applied:** Choose what action should be applied to data when sanitization is not supported or fails

Regular expression examples

Below are some examples on regular expressions and corresponding JSON nodes from an example JSON that

match with the expressions.

Code · json

```
{
  "data": {
    "image1": "R0lGODlhAQABAIAAAISwxwAAACwAAAAAQABAAACAkQBADs=",
    "image2": "iVBORw0KGgoAAAANSUhEUgAAABAAAAAQAgMAAABinRfyAA...",
    "images": [
      "/9j/4AAQSkZJRgABAQEAeAB4AAD/4QAiRXhpZgAATU0AKg...",
        "iVBORw0KGgoAAAANSUhEUgAAABAA..."
    ]
  }
}
```

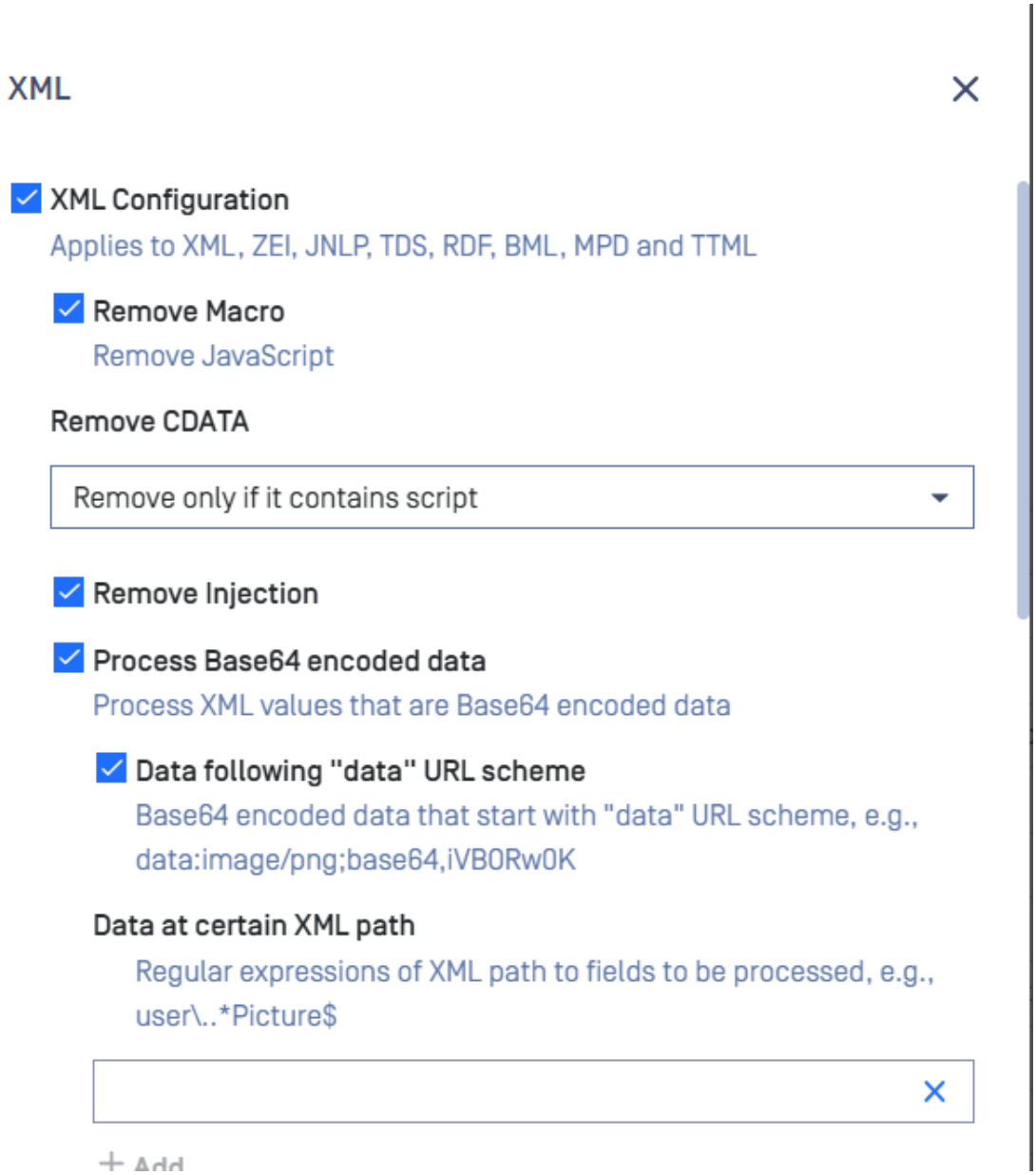
Expression	JSON node at path that matches
data.image1	data.image1
data.images.0	data.images[0]
data.images..*	data.images[0], data.images[1]
data.image.*	data.image1, data.image2, data.images[0], data.images[1]

Node: item index in a JSON array starts from 0.

XML

Configuration

This feature can be enabled at **Workflow Management> Workflows > [Workflow name] > Deep CDR > Advanced configuration > File type handling > Others > XML > Process Base64 encoded data**.



Code · xml

```
<root>
    <file1>SGVsbG8gV29ybGQhIFRoaXMgaXMgYSBzaW1wbGUgdGV4dCBmaWxlLg==</file1>
    <file2 name="invalid.jpg"
content="/9j/4AAQSkZJRgABAQEAeAB4AAD/4QAiRXhpZgAATU0AKgAAAAgAAQESAAMAAAABAAEAAAAAAAD/"/>
    <files name="icon.png"
content1="iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAADUlEQVR42mNk+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg=="
        <content2>R0lGODlhAQABAIAAAISwxwAAACwAAAAAAQABAAACAkQBADs=</content2>
    </files>
</root>
```

Regular expression examples

Expression	XML node/attribute at path that matches
root.file1	root.file1
root.file2.content	root.file2.content
root.files.content.*	root.files.content1, root.files.content2
root.files..*	root.files.name, root.files.content1, root.files.content2
root.file.*	root.file1, root.file2.name, root.file2.content, root.files.name, root.files.content1, root.files.content2

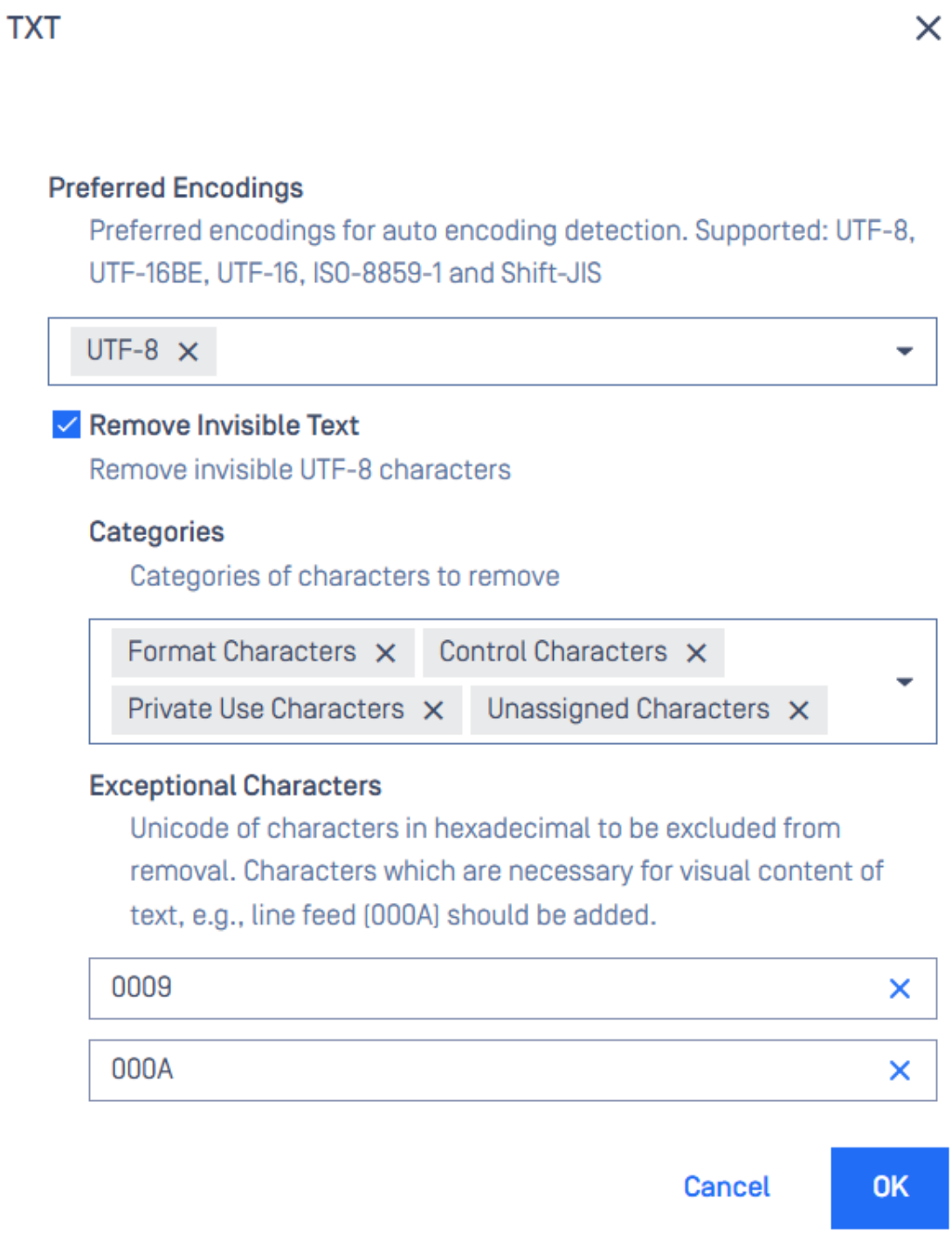
Remove invisible characters in TXT

Characters like **zero-width spaces (ZWSP)**, **soft hyphens**, or **bidirectional control characters** can be inserted into text without being visible to humans but still interpreted by machines. This method can be exploited in AI prompts to inject hidden instructions, manipulate outputs, or bypass security filters.

Deep CDR can efficiently detect and neutralize such attacks by cleansing the input and ensuring only legitimate content is processed.

Workflow > Deep CDR > Advanced Configurations > File type handling > Others > TXT.

Choose the **Preferred Encodings** is **UTF-8**. Enable **Remove Invisible Text**.

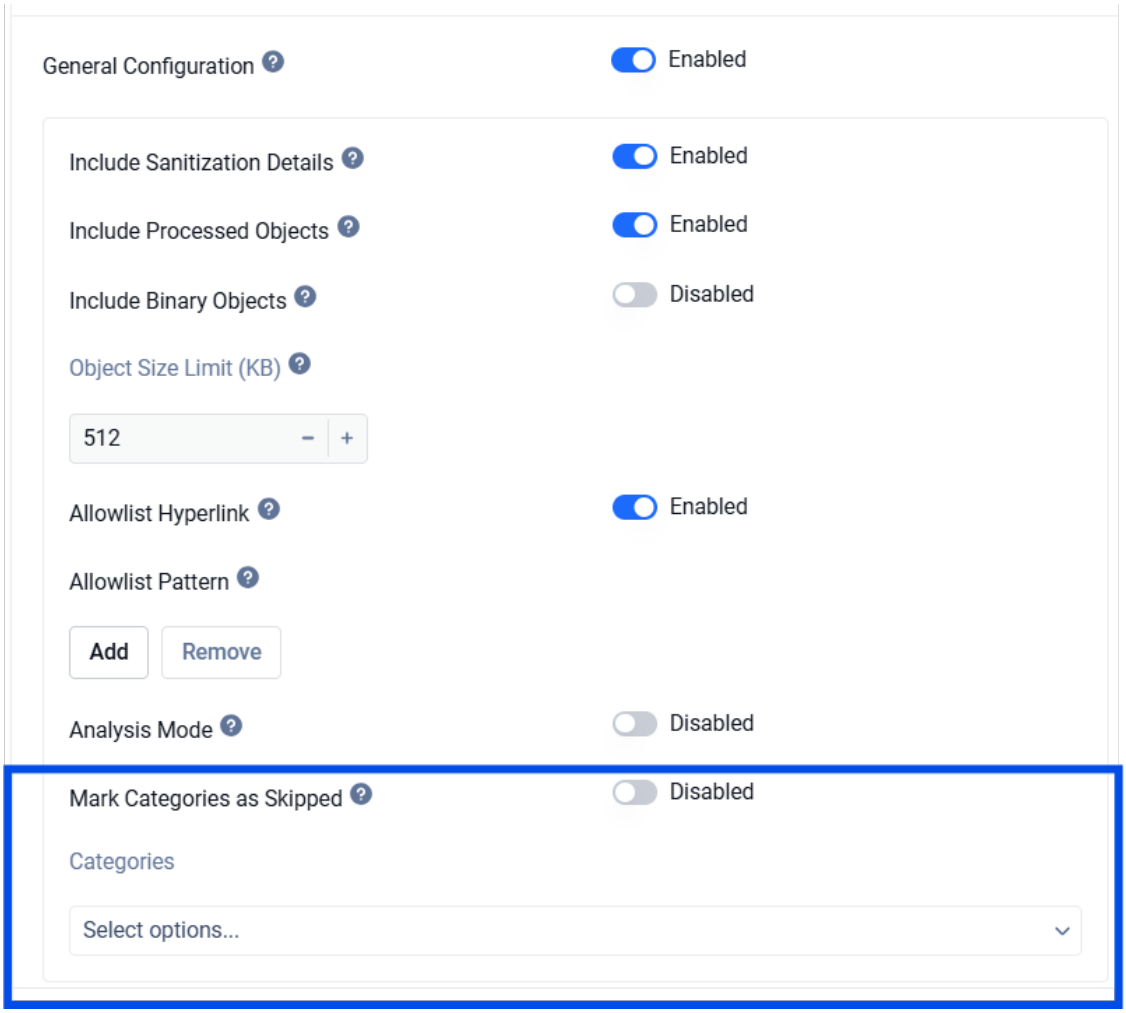


Mark Sanitization Result as Skipped

Users can configure to mark Sanitization results with certain categories as Skipped instead of Failure.

To mark sanitization result as Skipped, enable **Mark Categories as Skipped** under **Workflow Management> Workflows > Workflow Name > Deep CDR > Advanced configuration > General Configuration.**

Deep CDR can be configured to mark sanitization result as skipped, for the following categories:

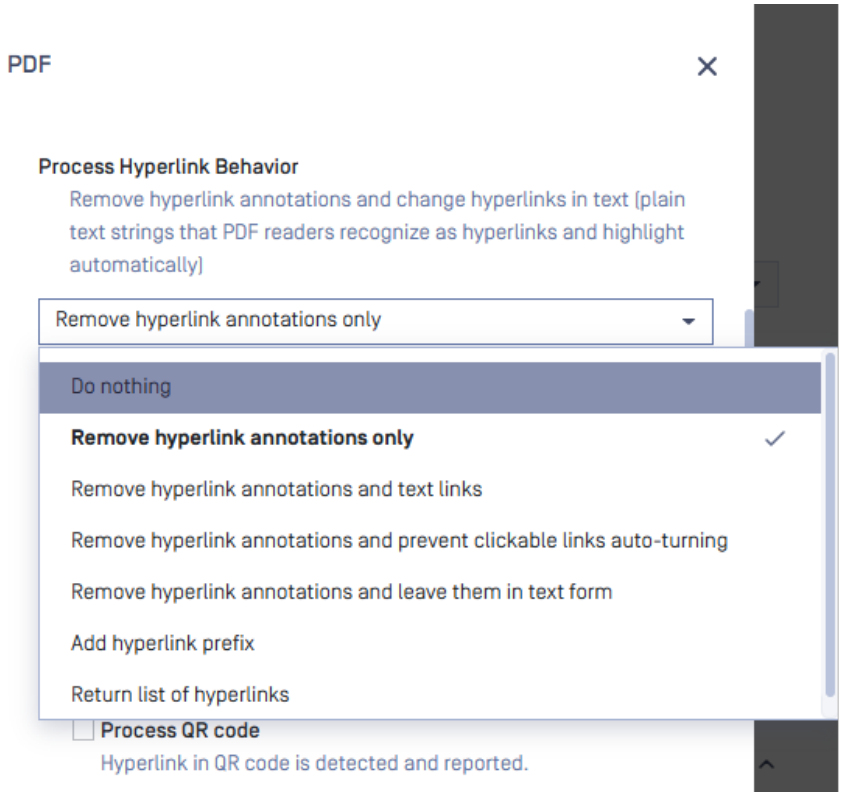


Process Hyperlink Behavior in PDF

Overview

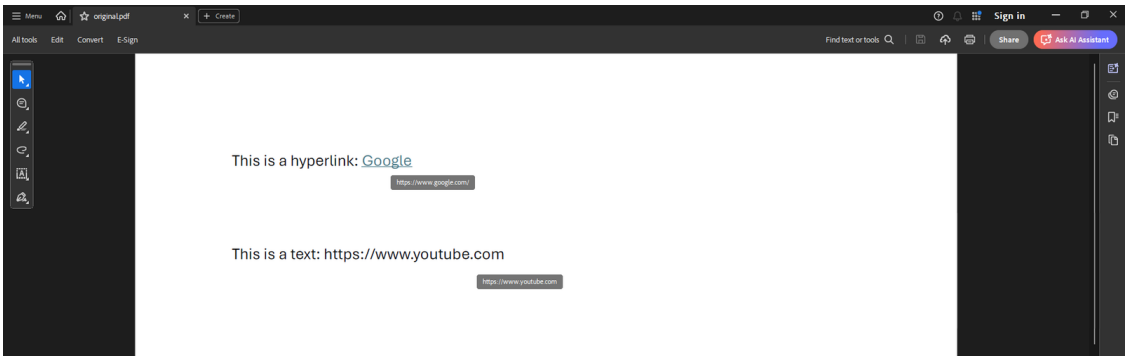
On this page, we will describe the configuration of **Process Hyperlink Behavior** for PDF files and provide examples for each option. This configuration allows users to remove hyperlink annotations and change hyperlinks in text (plain text strings that PDF readers recognize as hyperlinks and highlight automatically).

To configure the **Process Hyperlink Behavior**, please check **Process Hyperlink Behavior** in **PDF Configuration**.

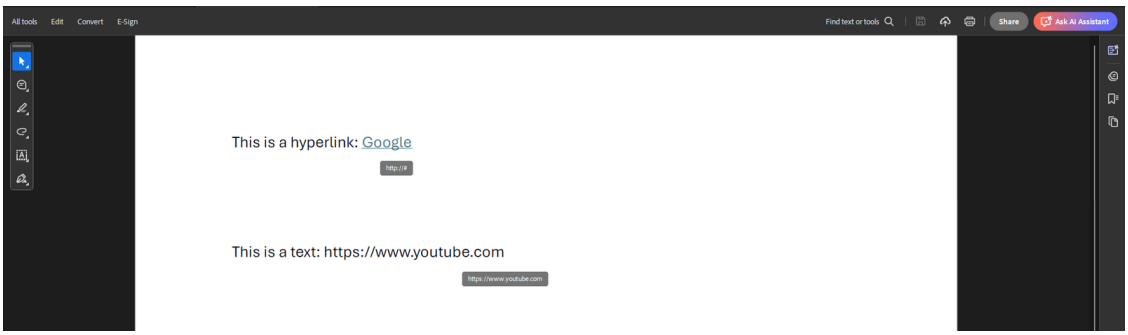


Options for Process Hyperlink Behavior

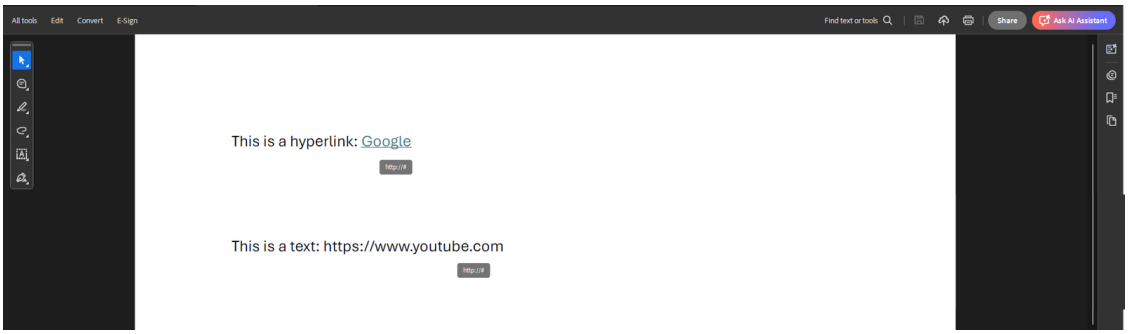
- **Do nothing:** Keep hyperlinks as is.



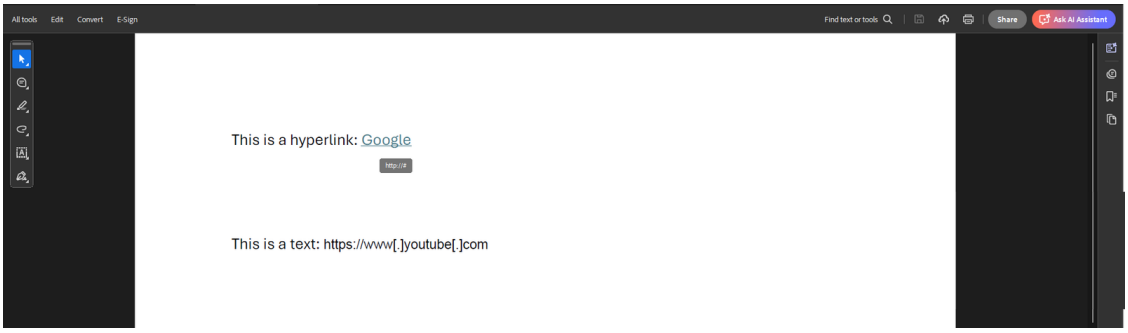
- **Remove hyperlink annotations only**: Remove hyperlink annotations. Text links (hyperlinks but written as texts in the content) may still be highlighted by PDF readers.



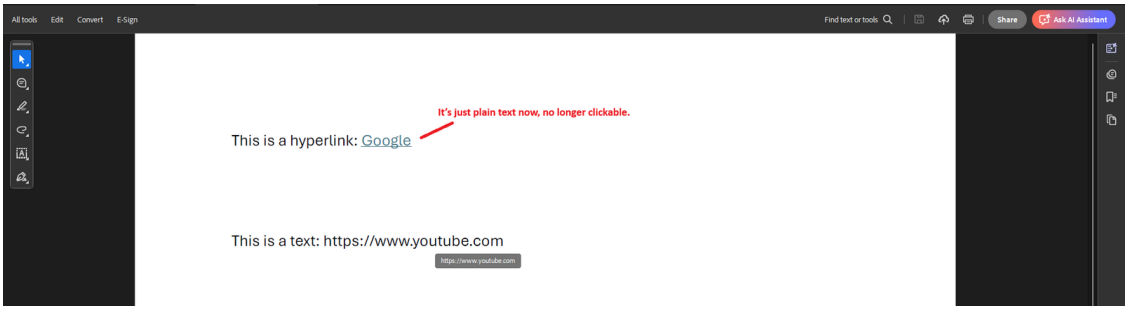
- **Remove hyperlink annotations and text links**: Remove hyperlink annotations and replace text links with null links (`http://#`). Text links will not be clickable.



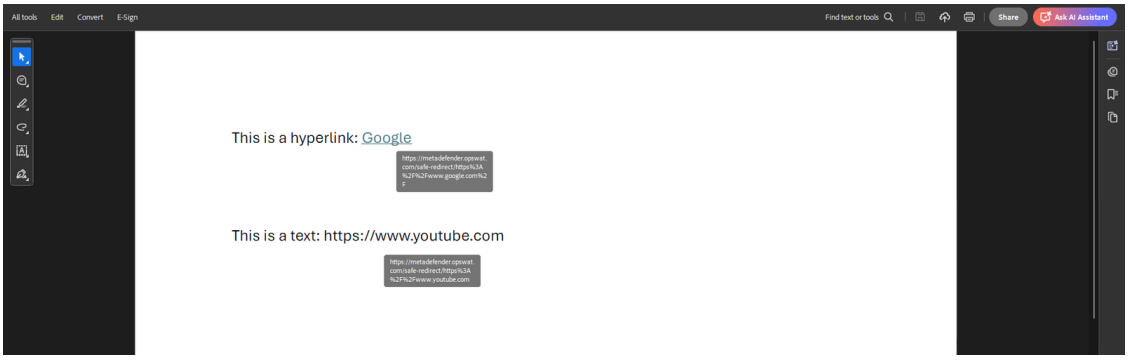
- **Remove hyperlink annotations and prevent clickable links auto-turning**: Prevent the clickable links auto-turning feature of readers by changing "." to "[.]" in text that is in hyperlink format.



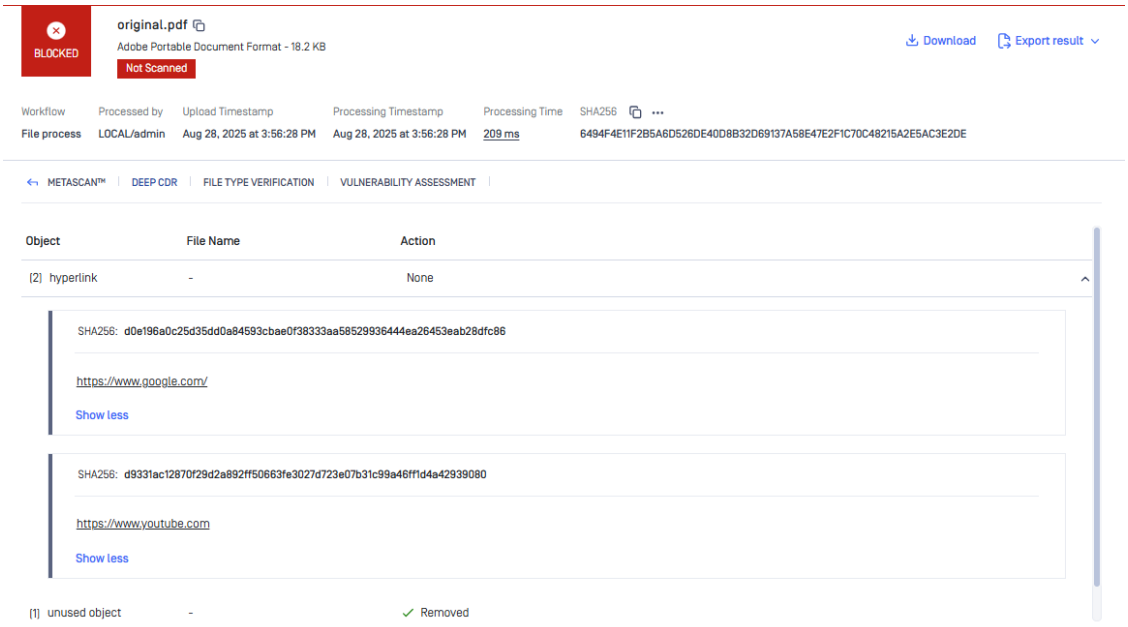
- **Remove hyperlink annotations and leave them in text form**: Remove hyperlink annotations and leave them in text form.



- **Add hyperlink prefix**: Prefix hyperlinks to allow redirecting them to a hyperlink verification or processing service.



- **Return list of hyperlinks**: Do nothing with hyperlinks, but list them in the returned message.

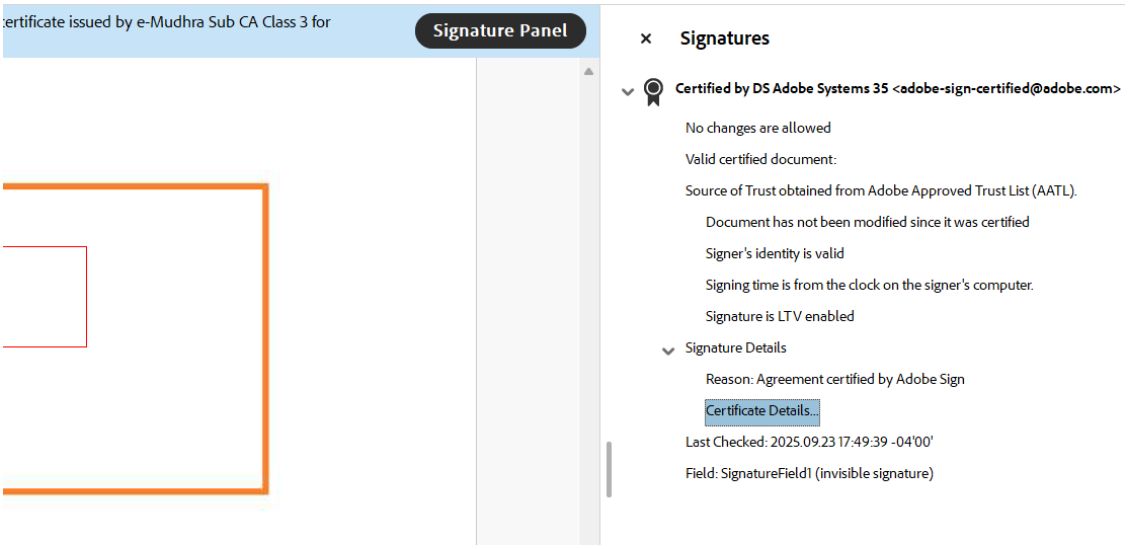


Preserve signed PDFs without sanitization

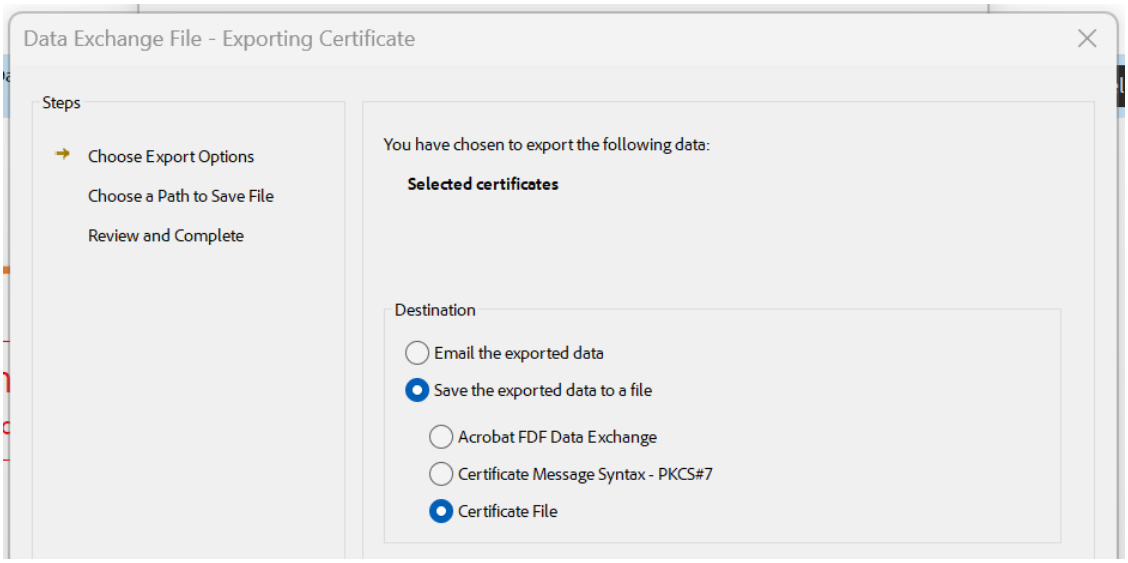
Sanitizing a signed PDF changes its content; therefore the original signature no longer validates. If the signature is important. Deep CDR provides configuration options that let users decide whether and how to skip sanitization for signed files.

1. Enable "**Skip Signed File**" and "**Validate Digital Signature**": Deep CDR validates the digital signature and file content to ensure integrity. Warning: there is a risk if an attacker uses a self-signed certificate.
2. **Recommended Configuration:** Enable the following options for maximum security:

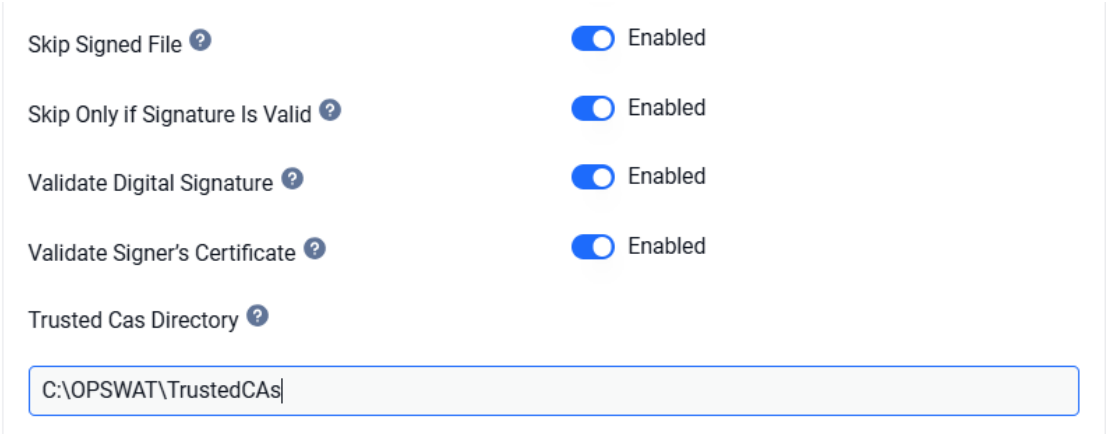
- Skip Signed File
- Skip Only if Signature is Valid
- Validate Digital Signature
- Validate Signer's Certificate
- Additionally, configure the certificate path. You can obtain the certificate by opening the PDF file in Adobe Reader, check the Signatures panel



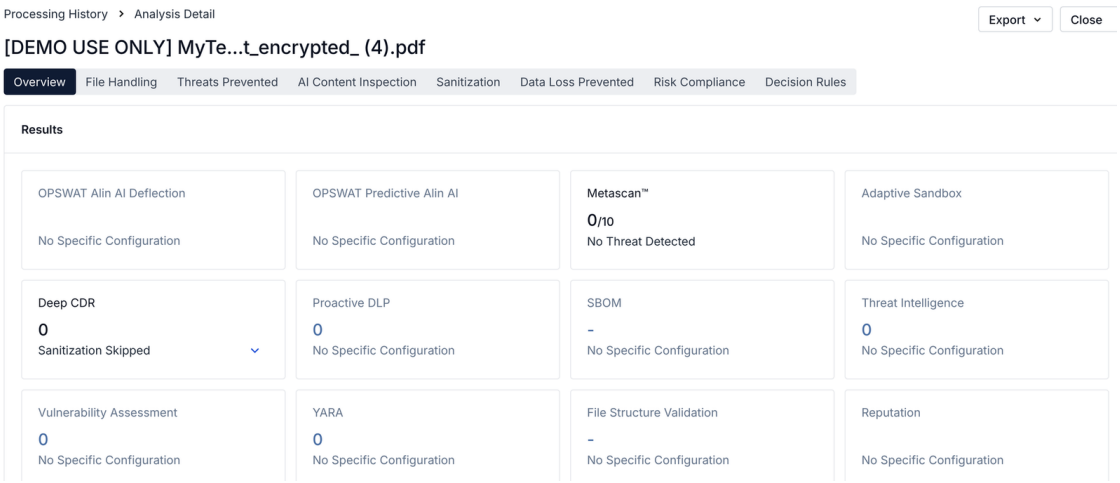
Choose "Certificate Details" and "Export", choose the "Certificate File" format and save to a folder.



Configure "Trusted CA directory" in the PDF Advanced configuration to point to that folder.



Files signed with this certificate will be **skipped during sanitization**.



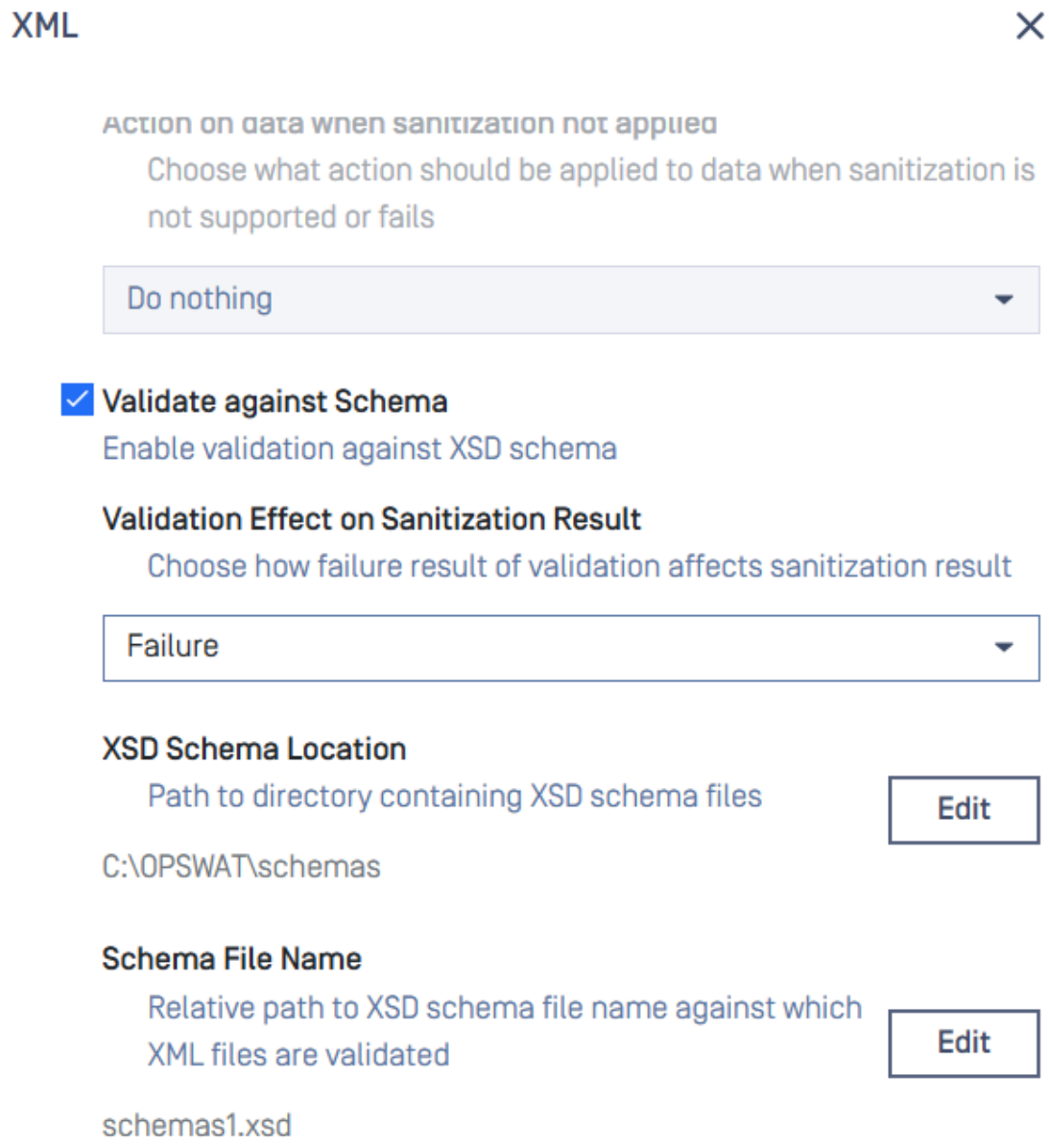
With these configurations, Deep CDR only allows if the file has exact match to certificate.

Validate XML files against an XSD schema

Deep CDR allows organizations to enforce schema validation on XML files. If a file does not conform to the specified schema, it will fail validation.

Configuration: **Workflow > [Workflow Name] > Deep CDR > Advanced Configuration > Others > XML,** then:

- **Enable "Validate against Schema":** Activates XML schema validation.
- **Validation Effect on Sanitization Result:** Choose how the system should behave when validation fails. Selecting "**Success**" will treat the failure as informational only, not as a processing error.
- **XSD Schema Location:** The folder where your schema files are stored. Multiple schema files can be placed in this location.
- **Schema File Name:** The default schema file to be applied for this workflow.



If you want to use different schema files for different input files, you can utilize the **engines-metadata** header when calling REST API

Code · json

```
{
    "deep_cdr": {
        "xml2xml.validate_against_schema.schema_name": "person.xsd"
    }
}
```

Skip Sanitizing Already-sanitized Files

If a submitted file is the output of a previous Deep CDR sanitization, MetaDefender Core skips Deep CDR for that file.

For example, Deep CDR sanitizes file A and produces A'. If A' is submitted later, MetaDefender Core recognizes it as

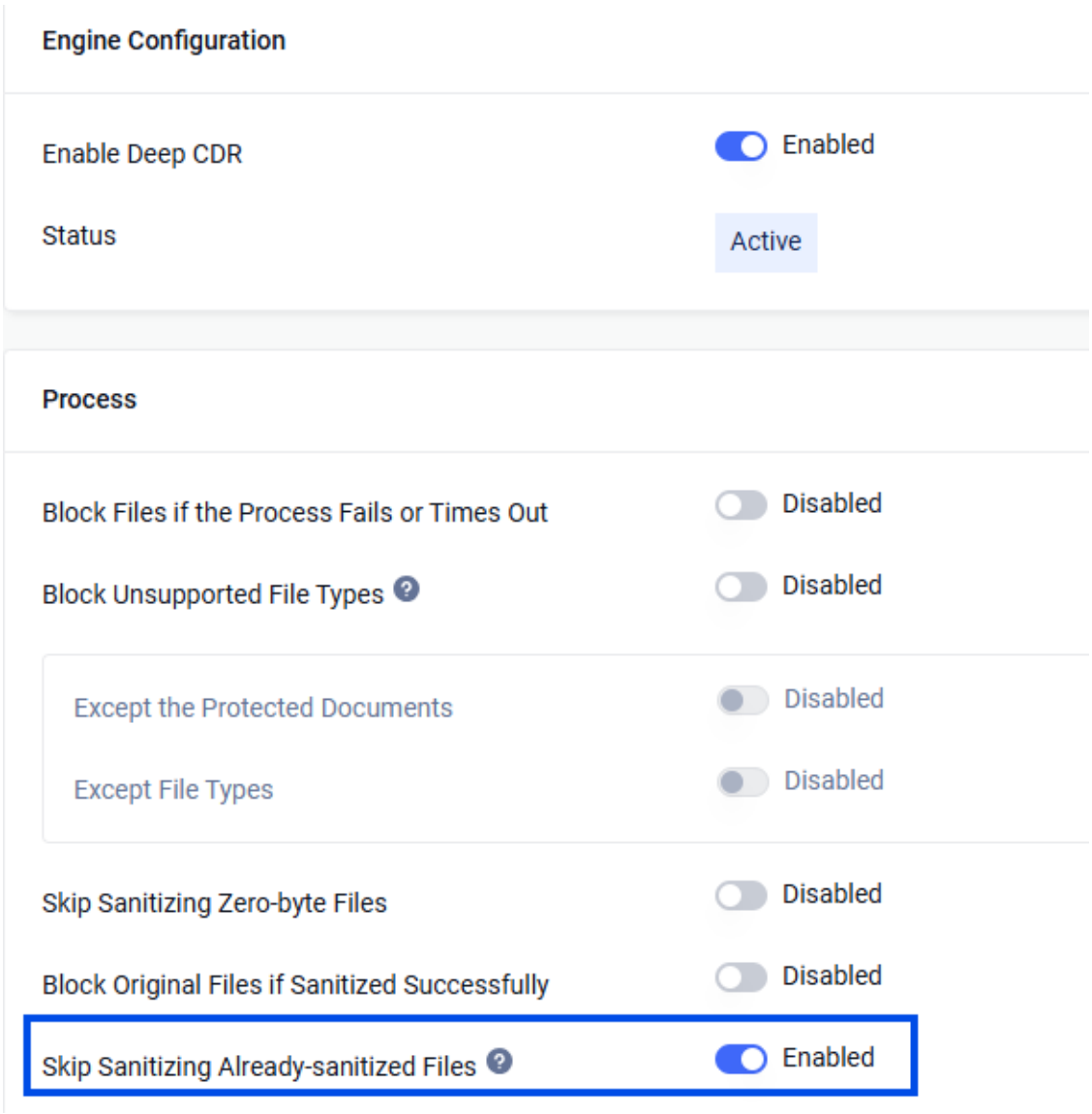
already sanitized, skips Deep CDR, and does not create a new sanitized file.

Sanitization is skipped only when both of the following match the original sanitization:

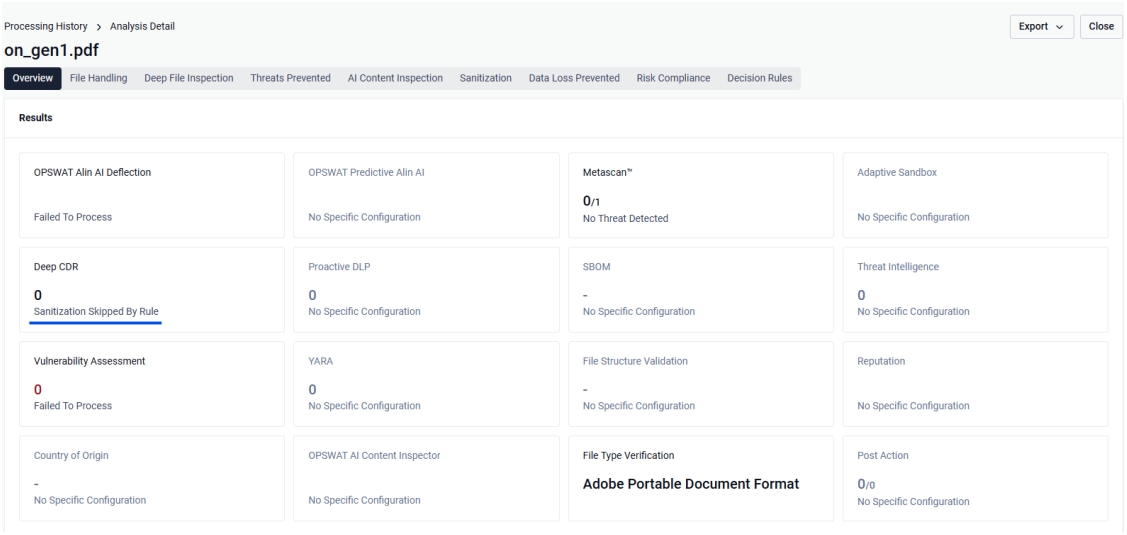
- Deep CDR settings
- Deep CDR engine version

To enable **Skip Sanitizing Already-sanitized Files** option, modify rule under

- **Workflow Management> Workflows > Workflow Name > Deep CDR > Process**.



- Sanitize result will be **Sanitization Skipped By Rule**



Supported file types:

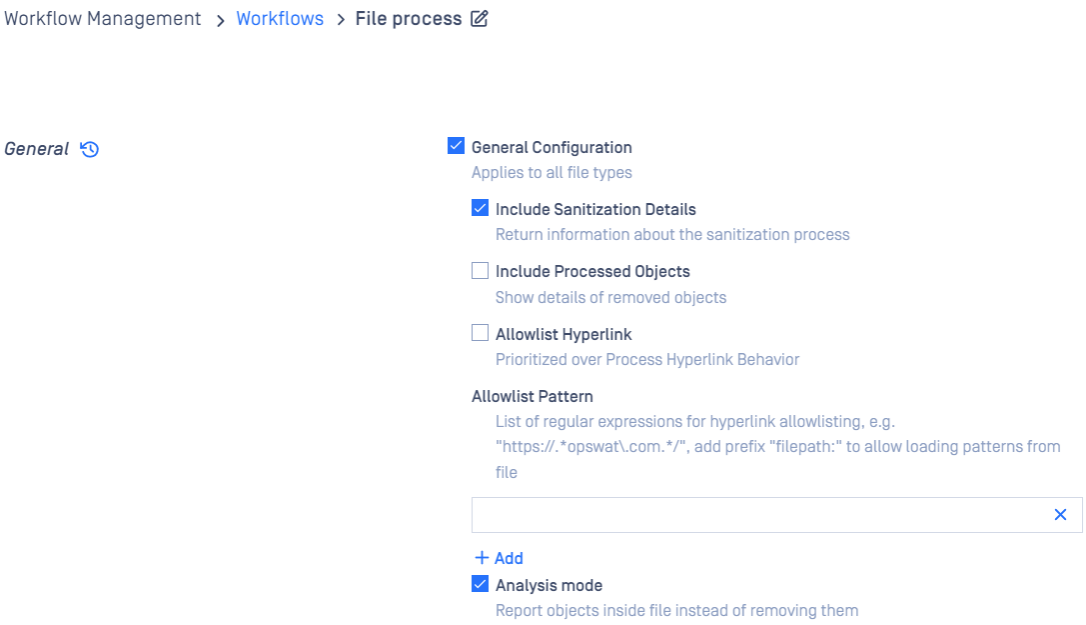
- Non archive file only

Analysis mode

> The **Analysis Mode** and **Blocking Mode** features in Deep CDR will be moved to Maintenance Mode for now. These capabilities have been migrated to the File Structure Validation modules, which provide more extensive and comprehensive functionality.

If your integrations don't need to use the sanitized file, you can enable this configuration to have a better performance.

To enable this feature: Workflow Management > [Workflow Name] > Deep CDR > Advanced > Enable Analysis Mode (the "Include Sanitization Details" must be enabled)



When this feature is enabled, **the sanitized file is not generated**

JSON output

File is processed with the analysis mode

Code · json

```
{
  "action_ran": "Sanitization Skipped",
  "sanitization_details": {
    "skipped_reason": "analysis_mode",
    "description": "Sanitization is skipped due to analysis mode being selected.",
    "details": [
      {
        "action": "none",
        "object_name": "XML content"
      },
      {
        "action": "none",
        "count": 1,
        "object_name": "image"
      },
      {
        "action": "none",
        "description": "Sanitization is skipped due to analysis mode being selected.",
        "details": [
          {
            "action": "none",
            "count": 5,
            "object_name": "attachment"
          },
```

```
          {
            "action": "none",
            "count": 1,
            "object_name": "hyperlink"
          }
        ],
        "file_name": "Portfolio.pdf",
        "object_name": "PDF file",
        "skipped_reason": "analysis_mode"
      }
    ]
  }
}
```


File type is not supported by the analysis mode

Code · json


```
{
  "actions_ran": "Sanitization Skipped",
  "sanitization_details": {
    "description": "Processing is skipped due to analysis mode being selected but not supported
for the source and target types.",
    "details": [],
    "skipped_reason": "unsupported_analysis_mode"
  }
}
```


Supported file types

- DOCX, DOCM, DOTX, DOTM, DOC, DOT
- PPTX, PPTM, POTX, POTM, PPSX, PPSM, PPT, POT, PPS
- XLSX, XLSM, XLTX, XLTM, XLSB, XLS XLT
- ODT, OTT, ODP, OTP, ODS, OTS
- VDX, VTX, VSX, VSDM, VSDX, VSSM, VSSX, VSTM, VSTX
- PDF, AI
- CELL, SHOW
- SLDM, SLDX
- XML, SVG, CSV, TXT, JSON
- ICS, VCS
- CRL
- RTF
- HTML, HTA

- HWP, HWT, HWPX
- JTD, JTDC
- EMF, WMF
- P21, SFC, JWC, JWW
- LNK
- XDW
- DWG, DWS, DWT
- Images

Performance

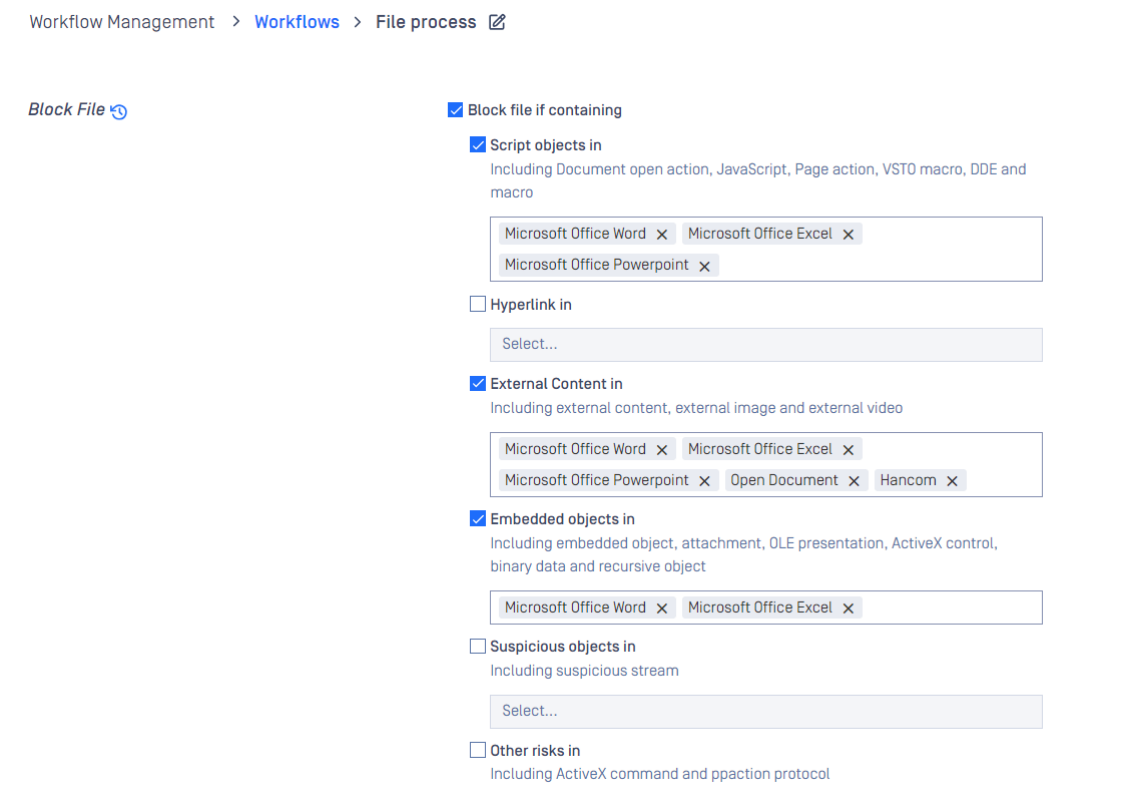
File types	Sanitization mode (s)	Analysis mode (s)	Difference (%)
MS Word 2007	0.397	0.163	143
MS Excel 2007	0.458	0.316	44
MS PowerPoint 2007	3.518	0.763	361
MS Word 2003	0.248	0.148	67
MS Excel 2003	0.224	0.064	250
MS PowerPoint 2003	9.161	4.733	93
PDF	0.457	0.131	248
XML	0.039	0.022	77
RTF	0.039	0.014	178
JWW	0.145	0.004	3525
JWC	0.025	0.003	733
Calendar	0.014	0.003	366
Visio XML	0.573	0.069	730
OpenDocument Text	0.728	0.136	435

File types	Sanitization mode (s)	Analysis mode (s)	Difference (%)
OpenDocument SpreadSheet	0.356	0.161	121
OpenDocument Presentation	4.4	0.539	716

Block file based on objects

In some use cases, the sanitized file can't be used dues to an integrity check. We recommend to block file based on objects to prevent zero-day attack. To configure this feature, please follow these steps

- *Workflow Manageement > Workflows > Workflow name > Advanced >* Enable **Block file if containing**
- Enable the Object types you want to block and which file families should be blocked



If a file contains the blocked objects, MetaDefender Core will block the file with a new verdict - Blocked Verdict by Deep CDR - an example JSON result

Code · bash

```
"process_info": {
    "archive_handling_details": {},
    "blocked_reason": "Blocked Verdict by Deep CDR",
    "blocked_reasons": [
```

```
      "Blocked Verdict by Deep CDR"
    ],
    "file_type_skipped_scan": false,
    "hash_time": 0,
    "post_processing": {
      "actions_failed": "",
      "actions_ran": "Sanitized",
      "converted_destination":
"Test_sanitized_by_OPSWAT_MetaDefender_85fd41ae04704eab83895dd2e6a46685.docm",
      "converted_to": "docm",
      "copy_move_destination": "",
      "sanitization_details": {
        "description": "Sanitized successfully.",
        "details": [
          {
            "action": "removed",
            "count": 1,
            "object_name": "macro"
          }
        ],
        "result_template_hash":
"101e1f24371ac0133dcb46d3e801833230c9bbfec3db03b2bb62abc26e2a72c3",
        "sanitized_file_info": {
          "file_size": 10824,
          "sha256": "ef09bf23546890751f888f2dc9f1cdcb21682597aea3d73a36d4b0fd4a5c8964"
        },
        "verdict": "blocked",
        "verdict_explanations": [
          "contain macro"
        ]
      }
    }
```


Supported Category and File Family

Object category	Included Object	Applicable file families
Script Object	Macro	Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint Open Document Hancom Ichitaro Markup Image

Object category	Included Object	Applicable file families
		Visio CAD PDF
	JavaScript	PDF Markup Image Video
	Script (Shell script, Script command)	Markup Video
	Document Open Action	PDF
	Page Action	PDF
	DDE	Markup Microsoft Office Word Microsoft Office Excel RTF
	VSTO macro	Microsoft Office Word Microsoft Office Excel
Embedded Object	Embedded object	PDF Markup Visio RTF Ichitaro Hancom Open Document Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint

Object category	Included Object	Applicable file families
	Embedded font	PDF Microsoft Office Word Microsoft Office Powerpoint Open Document Hancom Ichitaro RTF
	Embedded image	Markup Image
	Embedded html	RTF
	Attachment	PDF
	OLE presentation	Hancom Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint
	OLE	Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint Hancom Open Document Ichitaro RTF Visio ONE
	ActiveX control	Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint
	Binary data	Hancom

Object category	Included Object	Applicable file families
		Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint
	Recursive object	
External Content	External image	Image Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint Hancom Open Document
	External video	Microsoft Office Word Microsoft Office Powerpoint Hancom
	External content	PDF Microsoft Office Word Microsoft Office Excel
	External attachment	Markup
	External sheet	Microsoft Office Excel Hancom
Hyperlink	Hyperlink	PDF Microsoft Office Word Microsoft Office Excel Microsoft Office Powerpoint Open Document Hancom Ichitaro Markup Visio

Object category	Included Object	Applicable file families
		RTF Text ONE
Suspicious object	Suspicious stream (Suspicious streams in OLE structure, e.g., \u0005SebiesnrMkudrfcoIaamtykdDa in XLS or DOC_INFO in HWP.)	Microsoft Office Excel Hancom
	Suspicious node (Suspicious nodes like datastore or nodes starting with bin in RTF structure.)	RTF
	Suspicious CDATA (`CDATA` nodes in XML structure.)	Markup Image
Other risks	ActiveX command	Microsoft Office Powerpoint
	ppaction protocol	Microsoft Office Powerpoint

Supported file type in file family

File family	Supported File type
Microsoft Office Word	doc, dot, docx, dotx, docm, dotm
Microsoft Office Powerpoint	ppt, pps, pot, pptx, pptm, ppsx, ppsm, potx, potm, sldx, sldm, ppam
Microsoft Office Excel	xls, xlt, xlsx, xlsm, xlsb, xltx, xltm
Markup	xml, twb, html, hta, ics, csv, tsv
Image	svg
RTF	rtf
Video	wmv
PDF	pdf, ai

File family	Supported File type
Text	txt
CAD	dwg, dwt, dws
Open Document	odp, otp, ods, ots, odt, ott
Visio	vdx, vtx, vsx, vsdx, vstx, vssx, vsdm, vstm, vssm
Hancom	show, cell, hwp, hwt, hwpx
Ichitaro	jtd, jtdc, jhd
ONE	one

Deep CDR Performance (Throughput)

Performance Tuning Guide

Processing time and resources consumed by the engine vary relatively greatly depending on several aspects, e.g., nature and complexity of file structure, number of objects or components inside input files, etc. File size can be an aspect but is not necessary to contribute proportionally to processing time and resources consumption.

Depending on load and its complexity, users can tune some parameters to achieve a suitable balance on processing speed and throughput.

- **Timeouts**: a duration given to the engine for processing a file. Higher value gives the engine more chances to complete processing with complex files.
  - To set a value for this parameter, go to **Workflow Management > Workflows > [Workflow name] > Deep CDR**
- **Parallel count**: number of files being processed by the engine in parallel. Lower value puts low stress on the engine, giving it chances to complete processing with complex files.
  - Too large value can cause a domino effect that results in series of failures due to processing timed out.
  - Ones may keep it at 20 by default and, when necessary, should not increase it over 40.
  - To set a value for this parameter, see **parallelcount_ds** on MetaDefender Configuration.
- **Number of Image Processing Threads**: number of CPU cores being used for processing images. Higher value may drain CPU resources. Recommended value: <number of CPU cores>/8.
  - To set a value for this parameter, go to **Inventory > Modules > Deep CDR**.

Load with complex files

- Timeouts should be set to a high value, e.g., 30 minutes or 60 minutes.
- Parallel count should be set to a low value, e.g., 15 or 10.
- Number of Image Processing Threads should be set as recommended, <number of CPU cores>/8.

Load with less complex files

- Timeouts can be set to a low value, e.g., 5 minutes or 10 minutes.
- Parallel can be set to a high value, e.g., 20 but should not exceed 40.
- Number of Image Processing Threads should be set as recommended, <number of CPU cores>/8.

Throughput Test

> **Note**
> We do not guarantee the same performance in your environment. Performance can vary significantly depending on data sets and systems used when running the tests. The sole purpose of this section of the User Guide is to provide a high-level indicator of performance impact when enabling sanitization in your business logic.

Windows System Info

- RAM: 32.0 GB
- CPU: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz
- OS: Window 10 x64
- Disk Drive: SSD 100.0 GB

Linux System Info

- RAM: 32 GB
- CPU: 16
- OS: CentOS Linux release 7.6.1810
- Disk Drive: 100 GB

Resources

- MetaDefender Core version
  - v5.x Windows: **MetaDefender Core v5.0.0 with 8 engines**
  - v5.x Linux: **MetaDefender Core v5.0.0 with 10 engines**

- Default Deep CDR configuration
  - Window version: 6.0.0.10522

Test Results

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
pdf2pdf	700	840.608	0.612	0.371
ai2ai	500	1218.385	0.319	0.249
docx2docx	700	423.394	0.352	0.299
dotx2dotx	600	595.435	0.371	0.312
docm2docm	600	159.145	0.451	0.246
dotm2dotm	450	891.869	0.422	0.384
xlsx2xlsx	750	188.021	0.327	0.208
xlsm2xlsm	300	255.125	0.459	0.359
xlsb2xlsb	400	91.181	0.262	0.224
xltx2xltx	500	175.446	0.281	0.196
xltm2xltm	600	242.917	0.373	0.206
pptx2pptx	600	920.062	0.548	0.563
pptm2pptm	450	1316.683	0.605	0.714
potx2potx	300	532.595	0.441	0.466
potm2potm	300	534.049	0.472	0.484
ppsm2ppsm	300	533.087	0.455	0.483
ppsx2ppsx	300	661.106	0.417	0.331

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
doc2doc	800	490.112	0.491	0.477
dot2dot	500	222.872	0.366	0.349
xls2xls	1000	711.596	0.656	0.408
xlt2xlt	900	856.013	0.740	0.379
ppt2ppt	500	1340.349	0.909	0.948
pot2pot	200	1455.610	0.752	0.673
pps2pps	200	1324.390	0.697	0.621
jpg2jpg	700	712.840	0.498	0.369
png2png	700	469.696	0.688	0.509
bmp2bmp	600	788.215	0.293	0.276
gif2gif	400	2055.788	0.833	0.406
tiff2tiff	300	125.937	2.336	1.589
html2html	2000	333.108	0.232	0.206
rtf2rtf	2000	119.624	0.195	0.192
csv2csv	1350	129.233	0.184	0.180
hwp2hwp	1111	538.338	0.283	0.253
hwt2hwt	1111	538.338	0.244	0.212
hwpx2hwpx	500	550.725	0.277	0.208
jtd2jtd	1112	799.942	0.226	0.212
jtdc2jtdc	900	930.778	0.203	0.187

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
svg2svg	1800	19.566	0.179	0.180
emf2emf	1900	343.260	0.187	0.183
wmf2wmf	2000	165.267	0.207	0.183
vsdm2vsdm	600	204.497	0.302	0.248
vsdx2vsdx	800	101.889	0.230	0.210
vssm2vssm	600	244.680	0.383	0.358
vssx2vssx	600	55.265	0.190	0.226
vstm2vstm	600	207.689	0.384	0.378
vstx2vstx	500	54.581	0.198	0.225
vdx2vdx	200	903.018	0.524	0.316
vsx2vsx	200	435.260	0.348	0.307
vtx2vtx	200	582.117	0.465	0.317
txt2txt	1000	101.732	0.174	0.197
xml2xml	2700	130.711	0.178	0.187
xml-doc2pdf	800	246.213	0.199	0.248
xml-docx2pdf	500	263.272	0.215	0.269
xml-xls2pdf	550	366.042	0.310	0.867
odt2odt	1000	271.258	0.331	0.462
ott2ott	1000	280.982	0.340	0.492
ods2ods	799	66.608	0.195	0.186

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
ots2ots	800	129.761	0.253	0.241
odp2odp	700	1128.388	0.645	1.008
otp2otp	700	699.048	0.449	0.701
show2show	450	825.788	0.469	0.442
cell2cell	700	136.628	0.228	0.204
ico2ico	500	191.910	0.184	0.191
cur2cur	550	43.626	0.176	0.171
dwg2dwg	900	567.597	0.194	0.234
dwt2dwt	900	567.597	0.193	0.22
dws2dws	900	567.597	0.195	0.225
dwf2pdf	400	85.633	0.768	1.029
dxf2pdf	800	77.971	0.177	0.565
hta2hta	600	366.105	0.374	0.357
3ds23ds	550	1093.854	0.198	0.220
dae2dae	650	868.029	0.222	0.214
drc2drc	800	67.599	0.249	0.264
u3d2u3d	750	515.637	0.242	0.284
rvm2rvm	600	1251.032	0.204	0.234
ics2ics	600	31.381	0.182	0.231
vcs2vcs	750	0.436	0.174	0.287

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
wdp2wdp	400	2343.969	0.701	0.630
dcm2dcm	700	3320.839	0.324	0.357
sldx2sldx	600	920.062	0.536	0.554
sldm2sldm	450	1316.683	0.614	0.637
webp2webp	300	290.296	0.827	1.426
lnk2lnk	400	1.623	0.190	0.267
avi2avi	400	3005.967	0.312	0.157
mp32mp3	300	4293.660	0.329	0.260
mpeg2mpeg	100	13352.240	0.695	0.343
wav2wav	100	13425.500	0.740	0.365
mht2pdf	700	308.138	0.452	0.214

Archive Sanitization

MetaDefender Core is able to sanitize whole uploaded archives and give a modified version of the original archive.

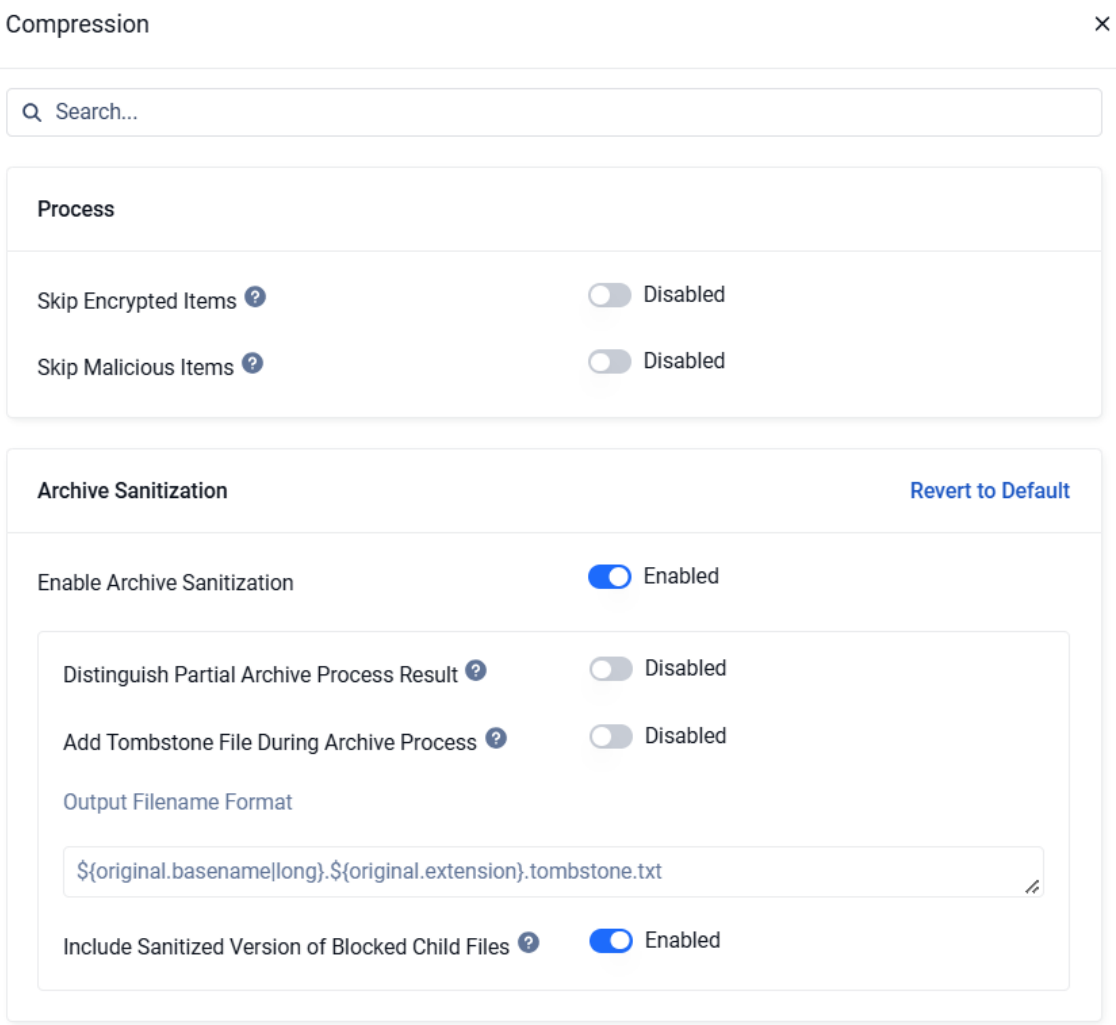
Usually this means that a sanitized archive contains:

- All allowed child files.
  - Or their sanitized versions (if available).
- Sanitized versions of blocked child files (if blocked child files are sanitized successfully).

Operation

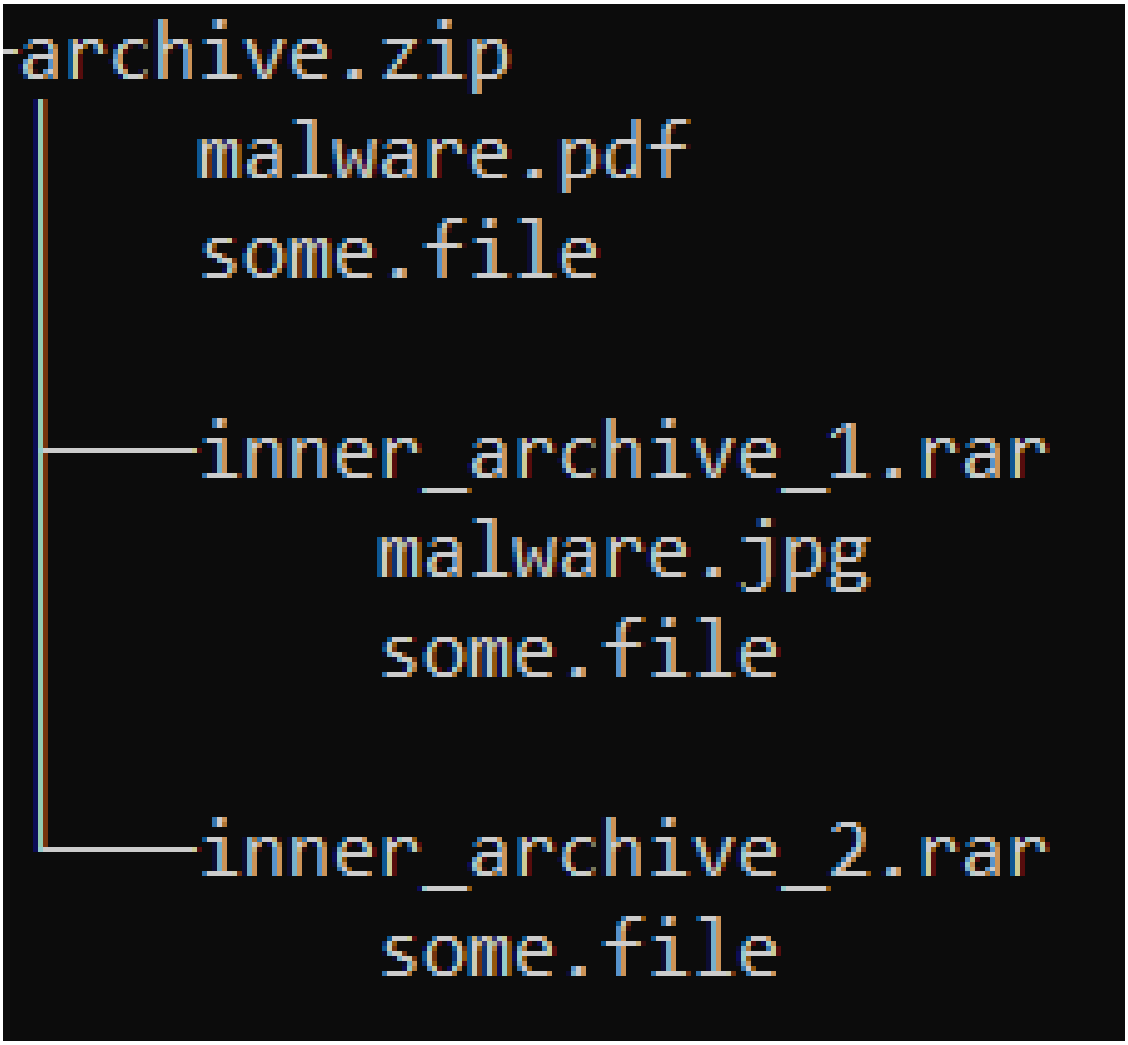
If archive sanitization is enabled for a type, then all archives within the original with the same type get sanitized as well. If there are other archives inside the original for which the sanitization is disabled and their result is not allowed then those archives won't get into the sanitized archive.

To enable the archive sanitization: **Workflow Management > Workflows > Workflow name > Compression**

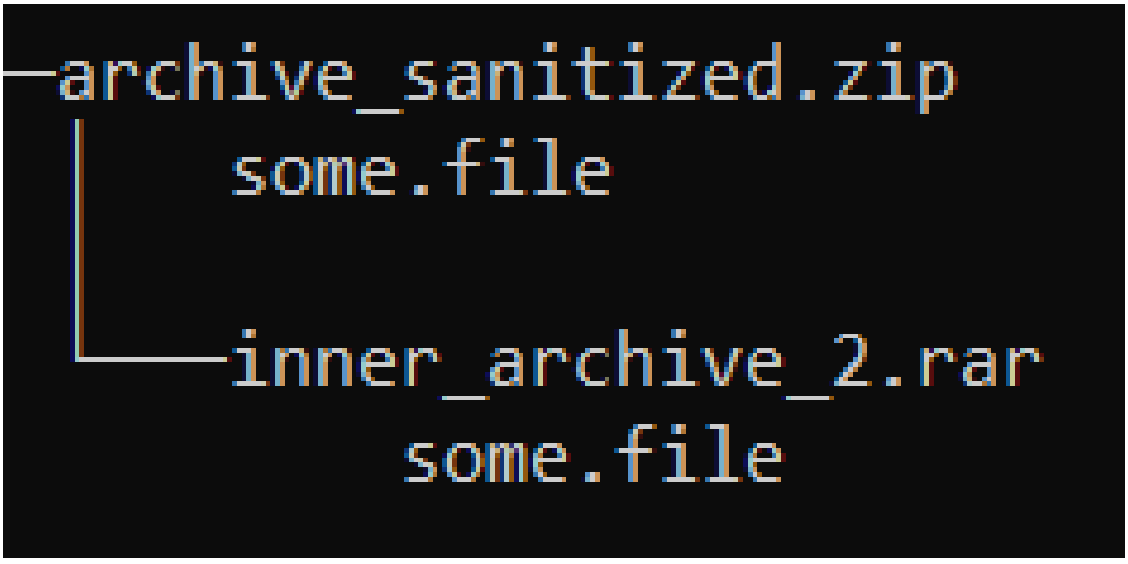


Use cases

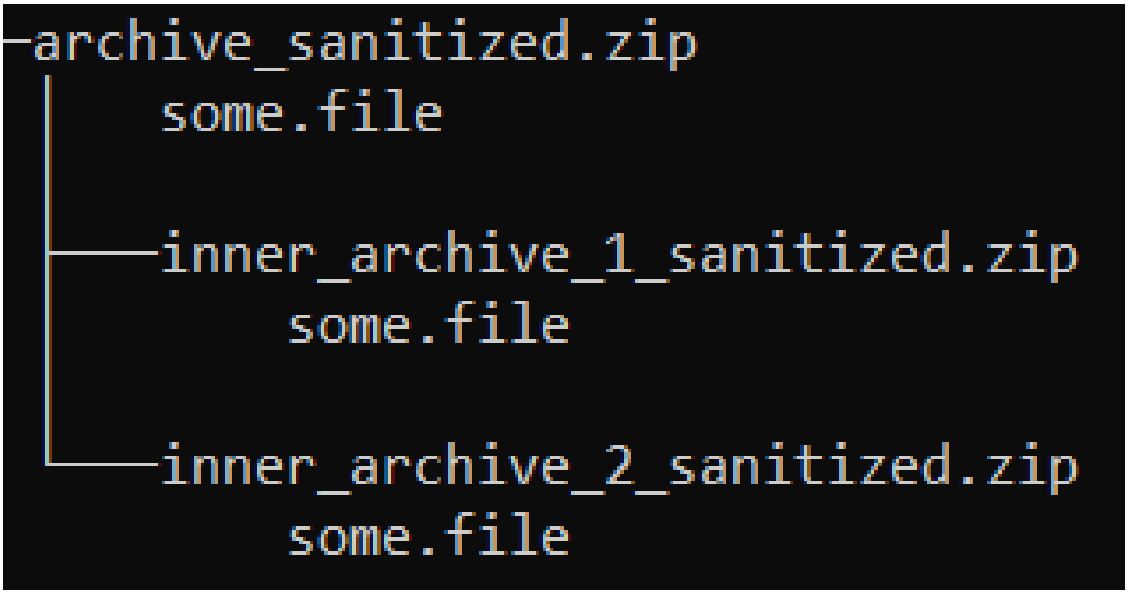
Given the following file as example:



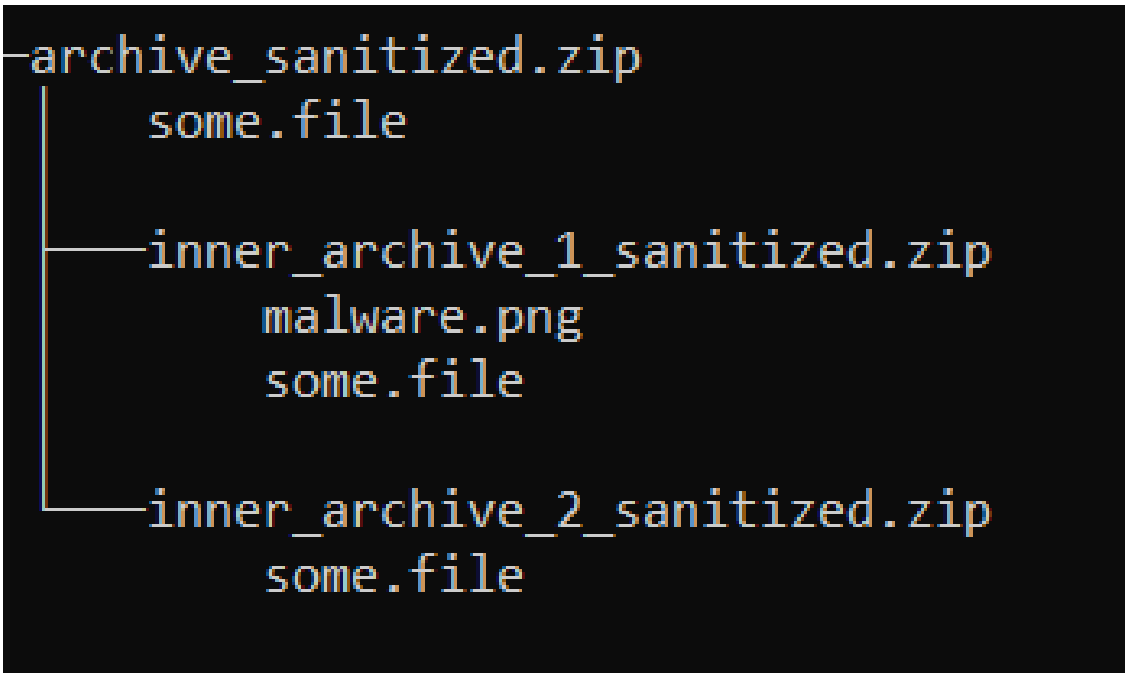
Zip to zip is enabled



Zip to Zip and Rar to Zip are enabled



Zip to Zip, Rar to Zip and JPG to PNG sanitization are enabled



Tombstone file

Since MetaDefender Core 4.19.1, whenever any child file in original archive file are not sanitized successfully, the product supports to append **tombstone file** into sanitized archive file to indicate which original child file(s) are removed from sanitized archive output file with details.

Each tombstone file will be auto created and placed at the same location as removed original child file(s) with following file name and content format:

File name format: `<original child file name>.tombstone.txt`

Code · json

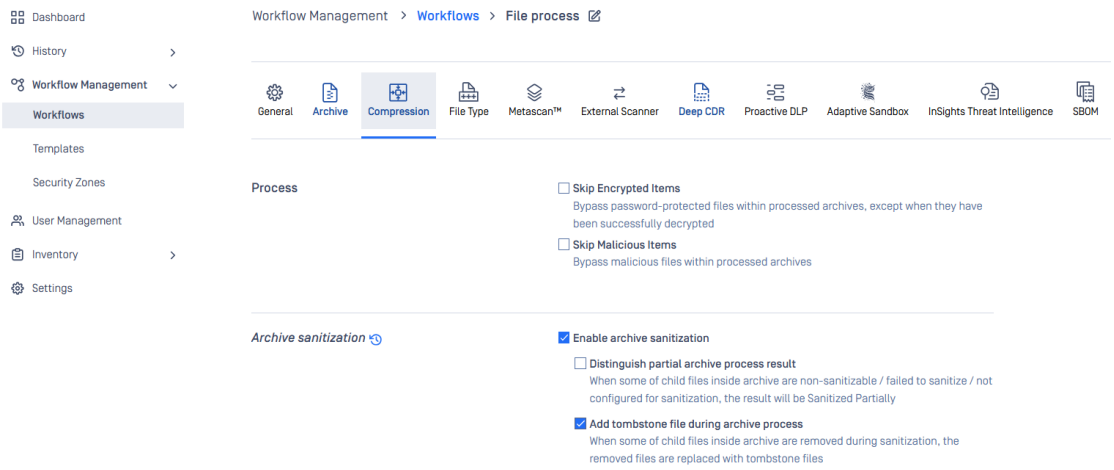
```
{
```

```
    "data_id":"<data id of original child file scan result processed by MetaDefender Core>",
    "file_info": {
        "sha256": "<SHA256 hash of original child file>"
    },
    "process_info": {
        "blocked_reason": "<blocked reason to indicate why original child file was not sanitized
successfully>"
    }
}
```

Info

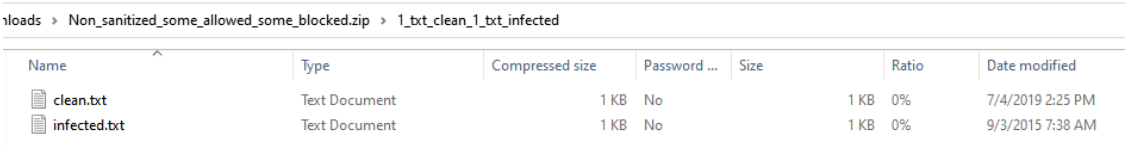
Please note that the tombstone functionality for the PST file type is supported starting from Core version 5.13.0, in combination with Archive Compression engine version 7.3.0.

A setting called “ADD TOMBSTONE FILE DURING ARCHIVE SANITIZATION” is required to be enabled (disabled by default) under each desired workflow rule to make this feature triggered:

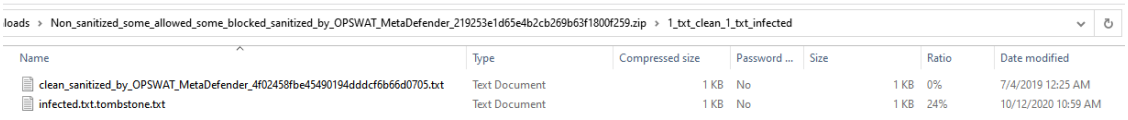


For example:

Original archive file content:



Downloading archived sanitization file and check its content:



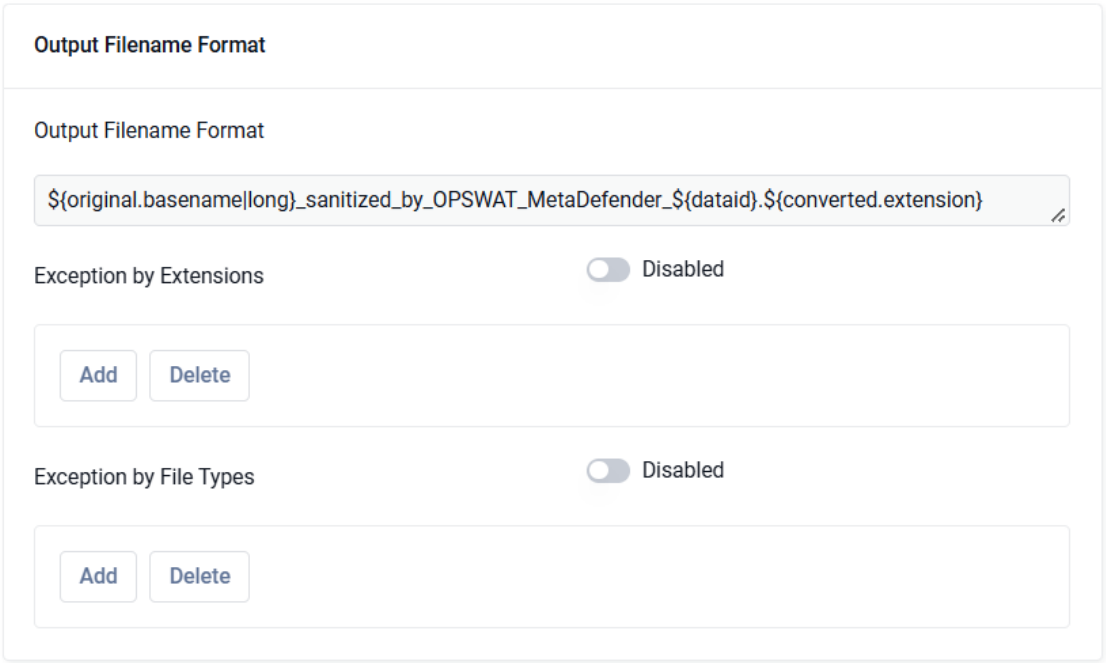
whereas a tombstone file called `infected.txt.tombstone.txt`

Code · json

```
{
    "data_id": 66aab3e361e04df7a78206e6ae5ceb15,
    "file_info": {
        "sha256" : 275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f
    },
    "process_info": {
        "blocked_reason" : Infected
    }
}
```

Setup output file name

To set up output filename for sanitized file, go to **Policy** > **Workflow rules** > choose **workflow name**



In the output filename format field, variables listed below can be used. Beside variables, any characters can be used as static text, except these:

Code

```
<>:"\|/?*$
```

Available variables:

- ${dataid} - this string identifies a file processing and can be used to retrieve results of processing
- ${datetime} - date and time at the time of sanitization
- ${original.basename} - name of the file to be sanitized, without file extension

- ${original.extension} - extension of the file to be sanitized
- ${converted.extension} - extension based on the target file type

Example usage

Code

```
${original.basename|long}_sanitized_${dataid}.${converted.extension}
```

Example output:

Code

```
testfile_sanitized_db3761f43e4545ab886f5930dbb037f3.pdf
```

Variables

Data ID

Code

```
${dataid}
```

The result is the dataid of the sanitized file.

Original filename

Code

```
${original.basename}
${original.basename|<attribute>}
```

Attribute	Description
short (default)	filename until the first '.', ie. **archive**.tar.gz
long	filename until the last '.', ie. **archive.tar**.gz

Original file extension

Code

```
${original.extension}
${original.extension|<attribute>}
```

Attribute	Description
short (default)	extension from the last '.', ie. archive.tar.**gz**
long	extension from the first '.', ie. archive.**tar.gz**

Converted file extension

Code

```
${converted.extension}
```

The result is the selected target extension used for sanitization.

Date and time

Code

```
${datetime}
${datetime|epoch}
${datetime|<format>}
```

Without any format string, the default format is 'yyyy_MM_dd-hh_mm'. With **epoch** format the result will be the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970, minus the number of leap seconds that have taken place since then. When giving custom format, the '.', '_' and '-' characters also can be used.

Expression	Output
d	the day as number without a leading zero (1 to 31)
dd	the day as number with a leading zero (01 to 31)
ddd	the abbreviated localized day name (e.g. 'Mon' to 'Sun'). Uses the system locale to localize the name.
dddd	the long localized day name (e.g. 'Monday' to 'Sunday'). Uses the system locale to localize the name.

Expression	Output
M	the month as number without a leading zero (1-12)
MM	the month as number with a leading zero (01-12)
MMM	the abbreviated localized month name (e.g. 'Jan' to 'Dec'). Uses the system locale to localize the name.
MMMM	the long localized month name (e.g. 'January' to 'December'). Uses the system locale to localize the name.
yy	the year as two digit number (00-99)
yyyy	the year as four digit number

These expressions may be used for the time:

Expression	Output
h	the hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display)
hh	the hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display)
H	the hour without a leading zero (0 to 23, even with AM/PM display)
HH	the hour with a leading zero (00 to 23, even with AM/PM display)
m	the minute without a leading zero (0 to 59)
mm	the minute with a leading zero (00 to 59)
s	the second without a leading zero (0 to 59)
ss	the second with a leading zero (00 to 59)
z	the milliseconds without leading zeroes (0 to 999)

Expression	Output
zzz	the milliseconds with leading zeroes (000 to 999)
AP or A	use AM/PM display. *A/AP* will be replaced by either "AM" or "PM".
ap or a	use am/pm display. *a/ap* will be replaced by either "am" or "pm".
t	the timezone (for example "CEST")

Deep CDR Details

Together with an error message, Deep CDR module also returns forensic info to describe what happened during the process in the case file was successfully sanitized.

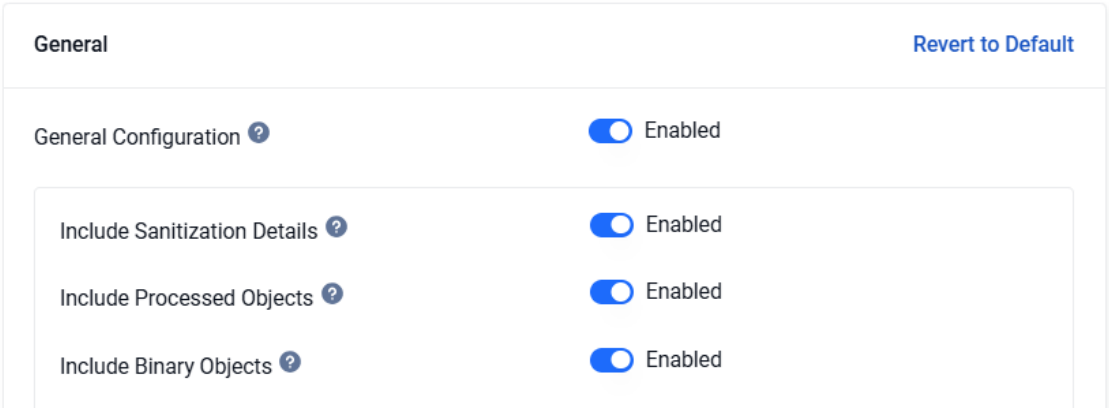
Sample forensic info · json

```
...
"post_processing": {
    "actions_failed": "",
    "actions_ran": "Sanitized",
    "converted_destination": "",
    "converted_to": "",
    "copy_move_destination": "",
    "sanitization_details": {
        "description": "Sanitized successfully.",
        "details": [
            {
                "action": "sanitized",
                "object_name": "XML content"
            },
            {
                "action": "removed",
                "count": 1,
                "object_details": [
                    "<ddeLink xmlns:r=\"http://schemas.openxmlformats.org/
officeDocument/2006/...."
                ],
                "object_name": "DDE"
            },
            {
                "action": "removed",
                "count": 2,
                "object_details": [
                    "http://metadefender.com/",
```

```
                "http://google.com/"
            ],
            "object_name": "hyperlink"
        },
    {
       "object_name":"XLS file",
       "file_name":"Microsoft_Excel_97-2003_Worksheet.xls",
       "description":"Sanitized successfully.",
       "details":[
                   {
                  "action":"removed",
                                "count": 1,
                                "object_details": [
                                        "Attribute VB_Name = \"Sheet1\"...."
                                ],
              "object_name":"macro"
            }
        ],
       "action":"sanitized"
    },
    {
           "object_name":"XLSX file",
           "file_name":"Microsoft_Excel_Worksheet.xlsx",
           "description":"invalid file structure.",
           "details":"Relationship file sharedStrings.xml does not exist",
           "action":"removed"
    },
          {
              "action": "removed",
         "binary_object_details": [
                         "TVqQAAMAAAAEAAAA..."
          ],
       "binary_object_encoding": "base64",
       "count": 1,
       "object_name": "OLE",
       "object_sha256": [
            "eec41d62ab5d2e1d880b338c47a2156a5ee7e58f3448f58cc8120392ddc8c730"
          ]
          },
          {
              "action": "sanitized",
              "count": 3,
              "object_name": "image"
          },
          {
              "action": "removed",
              "count": 1,
              "object_details": [
                      "Attribute VB_Name = \"Module1\"\r\nSub test_Macro()\r\n..."
              ],
              "object_name": "macro",
```

```
                "object_sha_256":
"3316B9DCF27981E42F98344FC680CBD2FB22DFE91E190DDECE56FA3C94EB628E"
            },
      {
               "action": "validated",
               "count": 1,
               "file_name": "\\docprops\\app.xml",
               "object_metadata": [
                    "Invalid Extended Property namely \"vt:vector\" at Line 1, Position 1066"
               ],
               "object_name": "invalid document properties",
               "reason_for_action": "Follow ISG for Microsoft Office 2007 Chapter 4.8, 4.9 and
4.10"
          }
        ],
        "sanitized_file_info": {
              "file_size": 127845,
              "sha256": "7db16cce0ea736757ebda14f64004319abdf15ad8db321bc212c03a52fee3f2d"
        }
    }
},
...
```


To enable this feature, please check "Include Sanitization Details" and "Include Processed Objects" option in Workflow Deep CDR Setting.



To include the base64 object, please check "Include Binary Objects".

The length of object details is limited to 5000 characters. If it exceeds the limit, it will be truncated.

Sanitization details info is optional, not avaialble in all file types

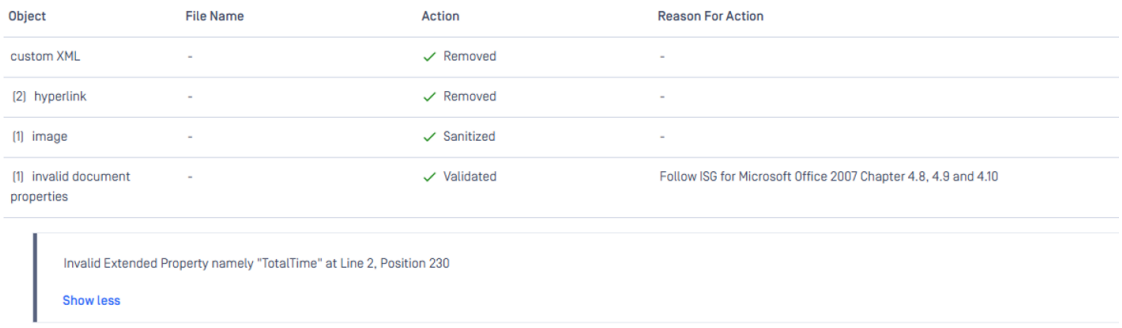
Possible action_ran values for Deep CDR	Description
Sanitized	File is sanitized successfully.
Sanitized Partially	Only applicable to the archive file sanitization, some children files are failed to sanitize.

Possible action_ran values for Deep CDR	Description
Sanitization Skipped	File is skipped because of the configurations.
Sanitization Timeout	Santization process is timeout

skipped_reason	Description
analysis_mode	Files skipped when running in the analysis mode.
unsupported_analysis_mode	Files skipped when running in the analysis mode and the file types are not supported.
signed_file	Files skipped because they contain digital signatures.
marked_as_skipped	Files explicitly marked as skipped due to configuration settings. Check "failure_category" to know the details.

Reason for Action

Some risk objects, like macros in MS Office, are easy to understand. However, there are many less obvious risk objects, such as document properties, template name, etc.... These properties allow users to input data, which can pose security risks. Deep CDR provides a "reason for action" for these objects, helping users better understand the associated risks.



Supported reasons:

Action	Reason/Concern
Validate Document Properties	Free-form text fields can potentially allow attackers to input data that does not conform to Microsoft schemas.

Action	Reason/Concern
	Reference: Inspection and Sanitization Guide (ISG) for MS Office 2007 section 4.8, 4.9, 4.10
Validate Template Name	There is a possibility that `w:attachedTemplate` refers to an invalid relationship. Reference: ISG for MS Office 2007 section 4.5
Remove Smart Tag	Smart Tags can executable code Reference: ISG for MS Office 2007 section 4.16
Validate header or footer	Headers and footers are free-form text that can provide potential data disclosure threats. Reference: ISG for Microsoft Office 2007 section 4.12
Validate structured XML	Ensure syntactic and semantic validity
Validate zip structure	Office 2007 documents are zip files that contain various files and folders. Arbitrary data can be added into the central directory without authorization. Reference: ISG for Microsoft Office 2007 Section 3.1 to 3.6

Troubleshooting sanitization failures

To distinguish different errors happened during the sanitization process, Data Sanitization engine classifies the errors to different categories The information is available in JSON output when retrieving scan result though REST API.

Sanitization failure · json

```
...
"post_processing": {
    "actions_failed": "Sanitization Failed",
    "actions_ran": "",
    "converted_destination": "",
    "converted_to": "",
    "copy_move_destination": "",
    "sanitization_details": {
        "failure_category": "invalid file structure",
        "details": "Startxref not found"
    }
},
```

```
...
```

details is an optional key *sanitization_details* is not available for all of the file types such as archive, eml file types.

Possible actions_failed values for Deep CDR	Description
Sanitization Failed	File is failed to sanitize, check *failure_category* to have more details
Sanitization Timed Out	Time to process the file is longer than the configured time

failure_category	description
invalid file structure	File is corrupted or it does not match specification
unsupported version	MS Office file is created with format before MS Office 97 or DWG file is created with AutoCAD before version 2004
exceeded file size	This file size is bigger than the configured file size
password protected	Document needs a password to unlock
processing time out	The sanitization process is timeout
failed to sanitize	Other failures during sanitization, should not happen

Supported engines-metadata JSON

Overwrite file encoding for TXT and HTML

The Deep CDR engine processes the TXT and HTML file as UTF-8 encoding by default. In several cases, the clients determine exactly the encoding of the file, so they can send this information to the engine to have a better detection. To utilize this feature, please see the bellow steps:

1. Use MetaDefender Core API to send a file

2. Use the **engines-metadata** to send the encoding

Code · json

```
{
  "deep_cdr": {
    "charset": "ISO-2022-JP"
  }
}
```

Select schema file to validate for XML

See the details here.

Code · json

```
{
    "deep_cdr": {
        "xml2xml.validate_against_schema.schema_name": "person.xsd"
    }
}
```

Overview

> **Info**
>
> Proactive DLP requires separate licensing for MetaDefender to enable this module.

With Proactive DLP module, MetaDefender can detect and then block or prevent and then allow data transferring between networks. Here are the details functions for each approach you have chosen.

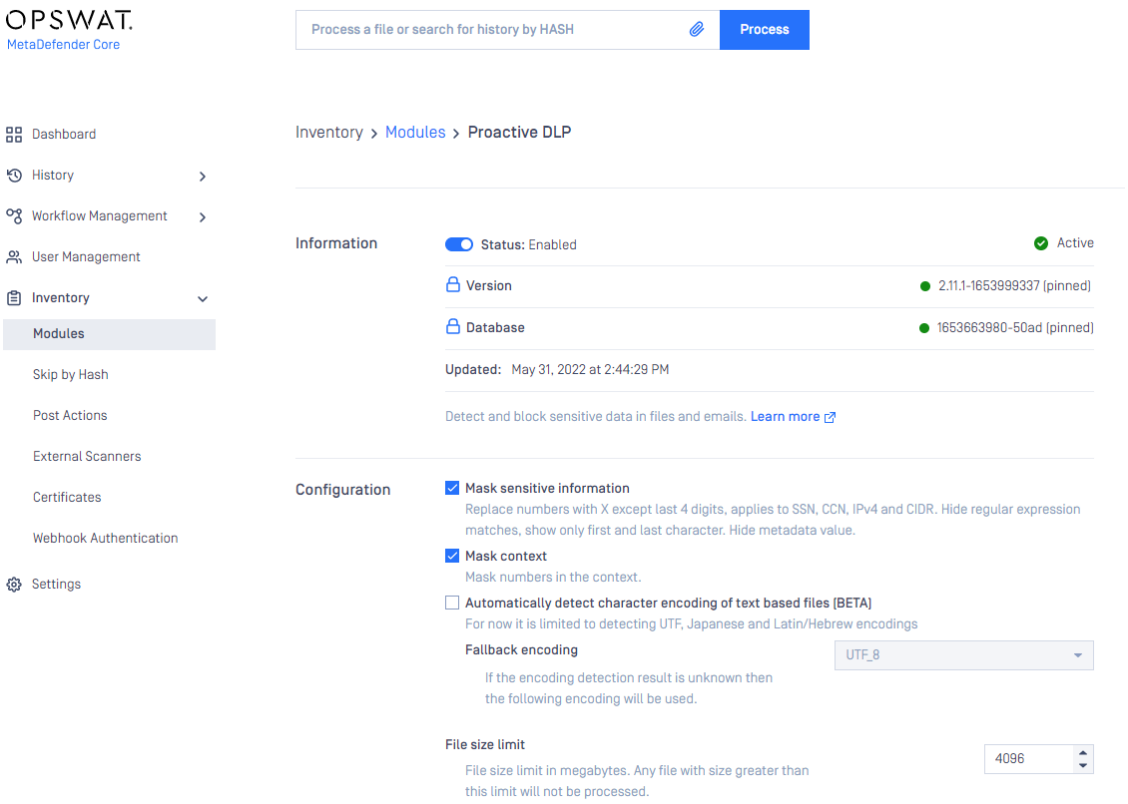
Detect and then block approach

- Detect Sensitive Information

Prevent and then allow approach

- Redact Sensitive Information
- Watermark Files
- Remove Metadata

General configuration



System requirement:

- Windows version
  - Microsoft Visual C++ 2017 Redistributable Package
  - .NET 8 dependencies
  - CPU must support AVX2 and SSE4.1 instruction set to use OCR feature
- Linux version:
  - mscorefonts
  - .NET 8 dependencies
  - CPU must support AVX2 and SSE4.1 instruction set to use OCR feature

Supported File Types

	File Type	Description	Supported Features
1	.aiml	Artificial Intelligence Markup Language	Scan | Secret detection
2	.asf | .wmv | .wma	Advanced Systems Format	Scan
3	.aspx	Microsoft ASP.NET Web Form	Scan | Secret detection
4	.atom	Atom Web Feed	Scan | Secret detection
5	.sh	Bourne Again Shell	Scan | Secret detection
6	.bmp	Windows Bitmap Image	Document Identification | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
7	.cer | .crt | .der | .pem | .key | .p12 | .pfx	Security Certificate	Scan | Secret detection
8	.crl | .pem	Certificate Revocation List	Scan | Secret detection

	File Type	Description	Supported Features
9	.csv	Comma-Separated Values	Redact | Scan | Secret detection
10	.dcm	Digital Imaging And Communications In Medicine	Anonymization | Anonymization OCR
11	.doc	Microsoft Word 97-2003 Document	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
12	.docm	Microsoft Word Macro-Enabled Document	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
13	.docx	Microsoft Word Document	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
14	.dtd	[BETA] Document Type Definition	Scan | Secret detection
15	.emf	Windows Enhanced Metafile	Scan | Watermark
16	.eml	Electronic Mail	Scan
17	.es3 | .et3 | .eak | .etv | .xml	[BETA] Microsec E-Szigno	Scan | Secret detection
18	.fdf	Adobe Forms Data Format	Redact | Scan | Secret detection

	File Type	Description	Supported Features
19	.gif	Graphics Interchange Format	Document Identification | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
20	.gpx	GPS eXchange Format	Scan | Secret detection
21	.har	HTTP Archive	Scan | Secret detection
22	.hta	Hypertext Markup Language Application	Scan | Secret detection
23	.html | .htm	Hypertext Markup Language	Document Identification | Redact | Scan | Secret detection
24	.ics	Electronic Calendaring And Scheduling (v2.0)	Scan | Secret detection
25	.inf	Windows Setup Information	Scan | Secret detection
26	.ini	Generic Configuration	Scan | Secret detection
27	.jpeg	JPEG Image	Document Identification | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
28	.jpg	JPEG Image	Document Identification | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
29	.jrxml	JasperReports Format	Scan | Secret detection

	File Type	Description	Supported Features
30	.json	JavaScript Object Notation	Scan | Secret detection
31	.jsp | .jspx | .jspf	Java Server Pages	Scan | Secret detection
32	.jspf	Java Server Pages	Scan | Secret detection
33	.jspx	Java Server Pages	Scan | Secret detection
34	.jtd	Ichitaro Document	Scan
35	.loc	GPO Locator	Scan | Secret detection
36	.log	[BETA] Log file	Redact | Scan | Secret detection
37	.mathml	Mathematical Markup Language	Scan | Secret detection
38	.mbx	Outlook Express Mailbox file	Scan
39	.mht	MIME Hypertext Markup Language	Scan | Secret detection
40	.mml	Aleph One Marathon Markup Language	Scan | Secret detection
41	.mov	Apple QuickTime Movie	Scan
42	.mp3	MP3 Audio	Scan
43	.msg	Outlook Message	Scan
44	.odf	OpenDocument XML Formula	Scan
45	.odg	OpenDocument XML Drawing	Scan
46	.odp	OpenDocument	Scan

	File Type	Description	Supported Features
		Presentation	
47	.ods	OpenDocument Sheet	Metadata removal | Scan
48	.odt	OpenDocument Text	Metadata removal | Scan
49	.odt | .ods | .odp | .ott | .ots | .otp	Open Office Document	Scan
50	.oft	Outlook Template	Scan
51		OpenSSH Private Key file	Scan | Secret detection
52	.otg	OpenDocument XML Drawing Template	Scan
53	.otp	OpenDocument Presentation Template	Scan
54	.ots	OpenDocument Sheet Template	Scan
55	.ott	OpenDocument Document Template	Scan
56	.oxps	Open XML Paper Specification	Scan
57	.pdf	Adobe Portable Document Format	Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
58	.php	PHP Hypertext Preprocessor	Scan | Secret detection
59	.png	Portable Network Graphics	Document Identification | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR |

	File Type	Description	Supported Features
			Watermark
60	.pom	Maven Project Object Model	Scan | Secret detection
61	.pot	Microsoft PowerPoint 97-2003 Template	Embedded image crop | Metadata removal | Scan
62	.potm	Microsoft PowerPoint Macro-Enabled Template	Embedded image crop | Metadata removal | Scan
63	.potx	Microsoft PowerPoint Template	Embedded image crop | Metadata removal | Scan
64	.ppk	PuTTY Private Key File	Scan | Secret detection
65	.pps	Microsoft PowerPoint 97-2003 Show	Embedded image crop | Metadata removal | Scan
66	.ppsm	Microsoft PowerPoint Macro-Enabled Show	Embedded image crop | Metadata removal | Scan
67	.ppsx	Microsoft PowerPoint Show	Embedded image crop | Metadata removal | Scan
68	.ppt	Microsoft PowerPoint 97-2003 Presentation	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR
69	.pptm	Microsoft PowerPoint Macro-Enabled Presentation	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan OCR
70	.pptx	Microsoft PowerPoint Presentation	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan

	File Type	Description	Supported Features
			metadata | Scan OCR
71	.prm	Parameter File	Scan | Secret detection
72	.ps1	PowerShell Script	Scan | Secret detection
73	.psd	[BETA] Adobe Photoshop Document	Scan
74	.py	Python Script | ASCII Text Executable	Scan | Secret detection
75	.rdf	[BETA] Resource Description Framework	Scan | Secret detection
76	.rdl | .rdlx | .rdlc	[BETA] Report Definition Language	Scan | Secret detection
77	.rels	Open Office XML Relationships	Scan | Secret detection
78	.rng	Relax NG	Scan | Secret detection
79	.rss | .xml	RSS Web Feed	Scan | Secret detection
80	.rtf	Microsoft Rich Text Format	Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
81	.sln	Microsoft Visual Studio Solution	Scan | Secret detection
82	.stc	OpenOffice XML Spreadsheet Template	Scan
83	.sti	OpenOffice XML Presentation Template	Scan
84	.stw	OpenOffice XML Text	Scan

	File Type	Description	Supported Features
		Document Template	
85	.svg	Scalable Vector Graphics	Watermark
86	.swf	Macromedia Flash Player	Scan
87	.sxc	OpenOffice XML Spreadsheet	Scan
88	.sxd	OpenOffice XML Drawing	Scan
89	.sxg	OpenOffice XML Master Document	Scan
90	.sxi | .sxd	OpenOffice XML Presentation	Scan
91	.sxw	OpenOffice XML Text Document	Scan
92	.tbm	Tableau Bookmark	Scan | Secret detection
93	.tds	[BETA] Tableau Datasource	Scan | Secret detection
94	.tif | .tiff	Tagged Image File Format	Document Identification | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
95	.tsv | .tab	Tab-Separated Values	Scan | Secret detection
96	.twb	[BETA] Tableau Workbook	Scan | Secret detection
97	.txt | .dat	ASCII Text	Document Identification | Redact | Scan | Secret

	File Type	Description	Supported Features
			detection
98	.vbs	Visual Basic Script	Scan | Secret detection
99	.vcf	Virtual Contact File	Scan | Secret detection
100	.vcs	Electronic Calendaring And Scheduling (v1.0)	Scan | Secret detection
101	.vsto	[BETA] Visual Studio Tools For Office Add-in	Scan | Secret detection
102	.wma	Windows Media Audio	Scan
103	.wmf	Windows Metafile	Scan | Watermark
104	.wmv	Windows Media Video	Scan
105	.wmv | .wma	Windows Media	Scan
106	.sdl	Web Services Description Language	Scan | Secret detection
107	.wsf | .cmd	Windows Script File	Scan | Secret detection
108	.edi	X12 Electronic Data Interchange	Scan | Secret detection
109	.xls	Microsoft Excel 97-2003 Workbook	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR
110	.xlsm	Microsoft Excel Macro-Enabled Workbook	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR
111	.xlsx	Microsoft Excel	Embedded image crop |

	File Type	Description	Supported Features
		Workbook	Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR
112	.docx | .xlsx | .pptx	Microsoft Office Document	Scan
113	.1st | .3dml | .abw | .aiml | .apml | .apt | .asc | .atom | .cfg | .classpath | .cnf | .conf | .config | .css | .err | .faq | .ficml | .fodt | .gml | .gpx | .htm | .html | .ini | .jnlp | .kml | .lxfml | .me | .musicxml | .mxl | .note | .now | .ova | .ovf | .log | .plain | .plist | .project | .psmdcp | .rdf | .readme | .rels | .resx | .rss | .scrivx | .sgml | .smil | .svg | .text | .tmp | .txt | .unx | .vml | .x3dv | .xaml | .xht | .xhtml | .xml | .xsd | .xspf | .xul | .zir	Extensible Markup Language	Redact | Scan | Secret detection
114	.xml	Microsoft Word 2003 XML Document	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR
115	.xml	Microsoft Word XML Document	Embedded image crop | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR
116	.xml	Microsoft PowerPoint XML Presentation	Scan
117	.xml	Microsoft XML	Metadata removal |

	File Type	Description	Supported Features
		Spreadsheet 2003	Redact | Scan | Scan metadata
118	.xps	Microsoft XML Paper Specification	Scan
119	.xsd	XML Schema Definition	Scan | Secret detection
120	.xslt | .xsl	Extensible Stylesheet Language Transformation	Scan | Secret detection
121	.xspf	XML Shareable Playlist Format	Scan | Secret detection
122	.zir	Compass And Ruler Geometry	Scan | Secret detection
123	.jp2 | .jpg2 | .jpf | .jpx | .mj2 | .mjp2 | .jpm | .jpgm	JPEG 2000	Document Identification | Metadata removal | Redact | Scan and redact OCR | Scan | Scan metadata | Scan OCR | Watermark
124	.pcap | .cap | .dmp | .trc	Packet Capture	Anonymization
125	.pcapng | .pcap | .cap | .dmp | .trc	Packet Capture Next Generation	Anonymization
126	.yml | .yaml	YAML Source	Scan
127	.pl	Perl Script Text Executable	Scan
128	.jnlp	Java Network Launching Protocol	Scan
129	.bat	DOS Batch File	Scan

Detect sensitive information

The main feature of the Proactive DLP engine is to detect and block sensitive data in files, including credit card numbers and social security numbers. The engine supports a wide range of file types, including Microsoft Office documents and PDF.

Sensitive Data

Financial Information	1. Credit Card Number (CCN) 2. ABA Routing Number 3. U.S. Bank Account 4. International Banking Account Number (IBAN) 5. International Securities Identification Number (ISIN) 6. SWIFT Code
Personal identity information	1. Japanese Personal My Number 2. U.S. Individual Taxpayer Identification Number (ITIN) 3. U.S. Social Security Number (SSN) 4. Turkish Identification Number (Kimlik No.) 5. Israeli National ID Number 6. Indian Aadhaar Number 7. Australian Tax File Number (TFN) 8. Indian Permanent Account Number (PAN) 9. Turkish Passport Number 10. Turkish Phone Number 11. UK Driving Licence Number 12. UK National Insurance Number 13. Ireland Personal Public Service (PPS) Number 14. Netherlands Citizen's Service Number (BSN) 15. France Social Security Number (NIR/INSEE) 16. Spain DNI Number 17. Italy Fiscal Code (Codice Fiscale)

	18. Denmark CPR Number 19. Sweden Personal Identity Number (Personnummer) 20. Finland Personal Identity Code (Henkilötunnus) 21. Poland PESEL Number 22. Bulgaria Uniform Civil Number (EGN) 23. Brazil CPF Number 24. Singapore NRIC Number 25. South Korea Resident Registration Number 26. South Africa National Identification Number 27. China Resident Identity Card Number 28. Canadian Social Insurance Number (SIN) 29. Mexico Unique Population Registry Code (CURP) 30. German Identity Card Number (Personalausweisnummer) 31. German Tax Identification Number (Steuerliche Identifikationsnummer) 32. Belgium National Register Number (Rijksregisternummer) 33. UK Unique Taxpayer Reference (UTR) 34. UK Electoral Roll Number 35. Romanian Personal Numerical Code (CNP) 36. Czech Personal Identity Number (Rodné číslo) 37. Slovenian Unique Master Citizen Number (EMŠO) 38. Slovak Personal Number (Rodné číslo) 39. Croatian Personal Identification Number (OIB) 40. Estonian Personal Identification Code (Isikukood) 41. Latvian Personal Code (Personas kods) 42. Lithuanian Personal Code (Asmens kodas) 43. Hong Kong Identity Card Number (HKID) 44. Taiwan National Identification Number 45. Thailand Population Identification Code 46. Malaysia Identification Card Number (MyKad)

	47. Austrian Tax Identification Number (Steuernummer) 48. Austrian Social Security Number (Sozialversicherungsnummer) 49. Swiss Social Security Number (AHV/AVS) 50. Chile National Identity Number (RUT/RUN) 51. German Passport Number (Passnummer) 52. Spain Social Security Number (NUSS/NAF) 53. Spain Passport Number 54. French Passport Number 55. Greek Tax Identification Number (AFM) 56. Hungarian Tax Identification Number (Adóazonosító jel) 57. Hungarian Social Security Number (TAJ) 58. Malta Tax Identification Number 59. Mexico Federal Taxpayer Registry Code (RFC) 60. Netherlands Passport Number 61. Poland Identity Card Number 62. Portugal Tax Identification Number (NIF) 63. Slovenian Tax Identification Number (Davčna številka) 64. U.S. Driver's License Number 65. U.S. Passport Number
Network and device information	1. IPv4 address or subnet mask Classless 2. Inter-Domain Routing (CIDR) 3. International Mobile Equipment Identity (IMEI/IMEISV) 4. IPv6 Address
Corporation information	1. Australian Company Number (ACN) 2. Australian Business Number (ABN) 3. Belgium VAT Number

	4. Brazil CNPJ (Legal Entity) Number 5. Japan Corporate Number
Drug and health information	1. Australian Medicare Number 2. U.K. National Health Service Number (NHS) 3. ICD-10 Code 4. ICD-9 Code 5. U.S. DEA Registration Number 6. U.S. National Drug Code (NDC)
Sensitive information (**Detect with AI**)	1. Driver's License Number 2. National ID Number 3. Passport Number 4. Date of Birth 5. Social Security Number 6. Personal Name 7. Phone Number 8. Birth Certificate Number 9. UK Tax Identification Number 10. UK Electoral Roll Number 11. UK National Insurance Number 12. Common Medical Condition 13. U.K. National Health Service Number (NHS) 14. U.S. National Drug Code (NDC) 15. Blood Type 16. Vehicle Identification Number 17. Credit Card Number

Additionally, any specific data pattern can be detected using custom regular expressions

Note

Defining so simple regexes is not the recommended way to use this engine, For example: \d , \w. _ Because

most documents contain many numbers, so the complexity and time needed to scan are going to be increased a lot when you define something so simple. Turning redaction on with this regex would cause even worse performance.

Certainty score

Certainty score is defined by the relevance of the given hit in its context . It is calculated based on multiple factors such as the number of digits, Bank Identification Number (BIN) lookup, context ...

- SSN Certainty levels
  - High
  - Medium
  - Low
- CCN/IPv4/ CIDR Certainty levels
  - Very High
  - High
  - Medium
  - Low
  - Very Low
- Custom RegEx Certainty levels
  - Medium
  - High
  - Very High
- Custom metadata Certainty levels
  - High

Supported File Types

Text and Documents

- Ansi Text (*.txt)
- ASCII Text
- CSV (Comma-separated values) (*.csv)
- Tab-separated values (*.tsv)
- iCalendar (*.ics, *.vcs)

- Microsoft Excel for Mac 2.2, 3, 4, 5, 98, 2001, X, 2004, 2008, 2011
- Microsoft Excel for Windows 2, 3, 4, 5
- Microsoft Excel 95, 97, 2000, XP, 2003, 2007, 2010, 2013, 2016 (*.xls)
- Microsoft Excel Office Open XML 2007, 2010, 2013, and 2016 (*.xlsx)
- Microsoft Office Excel XML (*.xml)
- Microsoft PowerPoint 3, 4, 95, 97, 98, 2000, 2001, 2002, 2003, 2004, 2007, 2008, 2010, 2011, 2013, 2016 (*.ppt)
- Microsoft PowerPoint Office Open XML 2007, 2010, 2013, and 2016 (*.pptx)
- Microsoft PowerPoint 97-2003 Template (*.pot)
- Microsoft PowerPoint Template (*.potx)
- Microsoft PowerPoint Show (*.ppsx)
- Microsoft PowerPoint Macro-Enabled Presentation (*.pptm)
- Microsoft PowerPoint Macro-Enabled Show (*.ppsm)
- Microsoft PowerPoint Macro-Enabled Template (*.potm)
- Microsoft PowerPoint 97-2003 Show (*.pps)
- Microsoft Office PowerPoint XML (*.xml)
- Microsoft Rich Text Format (*.rtf)
- Microsoft Word for DOS 1, 2, 3, 4, 5, 6 (*.doc)
- Microsoft Word for Mac 1, 3, 4, 5, 6, 98, 2001, X, 2004, 2008, 2011
- Microsoft Word for Windows 1, 2, 6 (*.doc)
- Microsoft Word 95, 97, 98, 2000, 2002, 2003, 2007, 2010, 2013, 2016 (*.doc)
- Microsoft Word 2003 XML (*.xml)
- Microsoft Word Office Open XML 2007, 2010, 2013, 2016 (*.docx)
- Microsoft/Open XML Paper Specification (_.xps, ._oxps)
- Ichitaro Document (*.jtd)
- OpenOffice/LibreOffice versions 1, 2, 3, 4, and 5 documents, spreadsheets, and presentations (*.sxc, *.sxd, *.sxi, *.sxw, *.sxg, *.stc, *.sti, *.stw, *.stm, *.odt, *.ott, *.odg, *.otg, *.odp, *.otp, *.ods, *.ots, *.odf) (includes OASIS Open Document Format for Office Applications)
- PDF files (*.pdf), note: Encrypted PDF files cannot be indexed, unless the PDF file can be opened without a password and the PDF file permissions allow for text extraction.
- PDF Portfolio files (*.pdf), including embedded non-PDF documents.
- Unicode (UCS16, Mac or Windows byte order, or UTF-8)
- XML (*.xml)
- XML Schema Description Files (*.xsd)
- JSON (*.json)
- Python script, ASCII text executable (*.py)

- Artificial Intelligence Markup Language (*.aiml)
- Microsoft ASP.NET Web Form (*.aspx)
- PHP Hypertext Preprocessor (*.php)
- PowerShell Script (*.ps1)
- Atom web feed (*.atom)
- GPS eXchange Format (*.gpx)
- Resource Description Framework (*.rdf)
- Open Office XML Relationships (*.rels)
- RSS web feed (*.rss)
- Visual Basic Script Files (*.vbs)
- Compass and Ruler geometry (*.zir)
- Document Type Definition (*.dtd)
- Tableau Workbook (*.twb)
- Tableau Datasource (*.tds)
- Tableau Bookmark (*.tbm)
- Windows Script File (*.wsf)
- Bourne Again Shell (*.bash) Virtual Contact File (*.vcf)

Email, HTML

- EML (emails saved by Outlook Express) (*.eml)
- MSG (emails saved by Outlook), including attachments (*.msg)
- Eudora MBX message files (*.mbx)
- HTML (*.htm, *.html)
- MIME Encapsulation of Aggregate HTML Documents (.mht)
- Hypertext Markup Language Application (.hta)

Media (Metadata check only)

- Adobe Photoshop images (*.psd)
- ASF media files (*.asf)
- MP3 (*.mp3)
- WMA media files (*.wma)
- WMV video files (*.wmv)
- GIF (*.gif)
- EMF (*.emf)

- WMF (*.wmf)
- SWF (*.swf)
- MOV (*.mov)

Supported File Types by AI

- Ansi Text (*.txt)
- ASCII Text
- Microsoft PowerPoint 3, 4, 95, 97, 98, 2000, 2001, 2002, 2003, 2004, 2007, 2008, 2010, 2011, 2013, 2016 (*.ppt)
- Microsoft PowerPoint Office Open XML 2007, 2010, 2013, and 2016 (*.pptx)
- Microsoft PowerPoint 97-2003 Template (*.pot)
- Microsoft PowerPoint Template (*.potx)
- Microsoft PowerPoint Show (*.ppsx)
- Microsoft PowerPoint Macro-Enabled Presentation (*.pptm)
- Microsoft PowerPoint Macro-Enabled Show (*.ppsm)
- Microsoft PowerPoint Macro-Enabled Template (*.potm)
- Microsoft PowerPoint 97-2003 Show (*.pps)
- Microsoft Rich Text Format (*.rtf)
- Microsoft Word for DOS 1, 2, 3, 4, 5, 6 (*.doc)
- Microsoft Word for Mac 1, 3, 4, 5, 6, 98, 2001, X, 2004, 2008, 2011
- Microsoft Word for Windows 1, 2, 6 (*.doc)
- Microsoft Word 95, 97, 98, 2000, 2002, 2003, 2007, 2010, 2013, 2016 (*.doc)
- Microsoft Word Office Open XML 2007, 2010, 2013, 2016 (*.docx)
- PDF files (*.pdf), note: Encrypted PDF files cannot be indexed, unless the PDF file can be opened without a password and the PDF file permissions allow for text extraction.
- PDF Portfolio files (*.pdf), including embedded non-PDF documents.
- Portable Network Graphics (*.png)
- JPEG Image (*.jpeg, *.jpg)
- JSON (*.json)

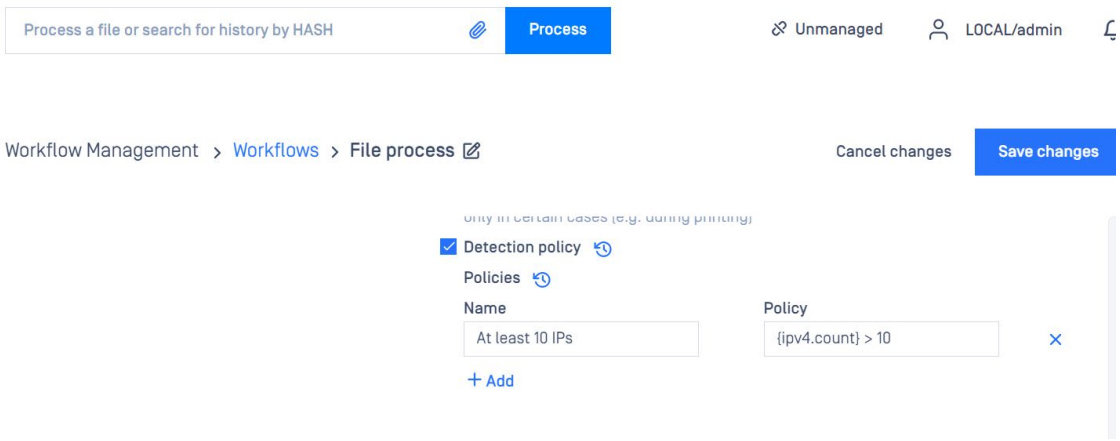
Detection Policy

The detection policy allows the users to define complicated rules to decide when files are blocked or allowed. For example, you can define a rule to block files which have more than 10 IP addresses, or even more complicated such as files that have AWS Key ID **and** AWS Secret Key

1. **For predefine sensitive info**: {predefine sensitive info.attribute} > X

Supported pre-defined sensitive info: *ccn, ssn, ipv4, cidr,...* For the full list of supported entities, please visit this page.

Example: {ipv4.count} > 10 will block the files only if the Proactive DLP module detects 10 IP address or more

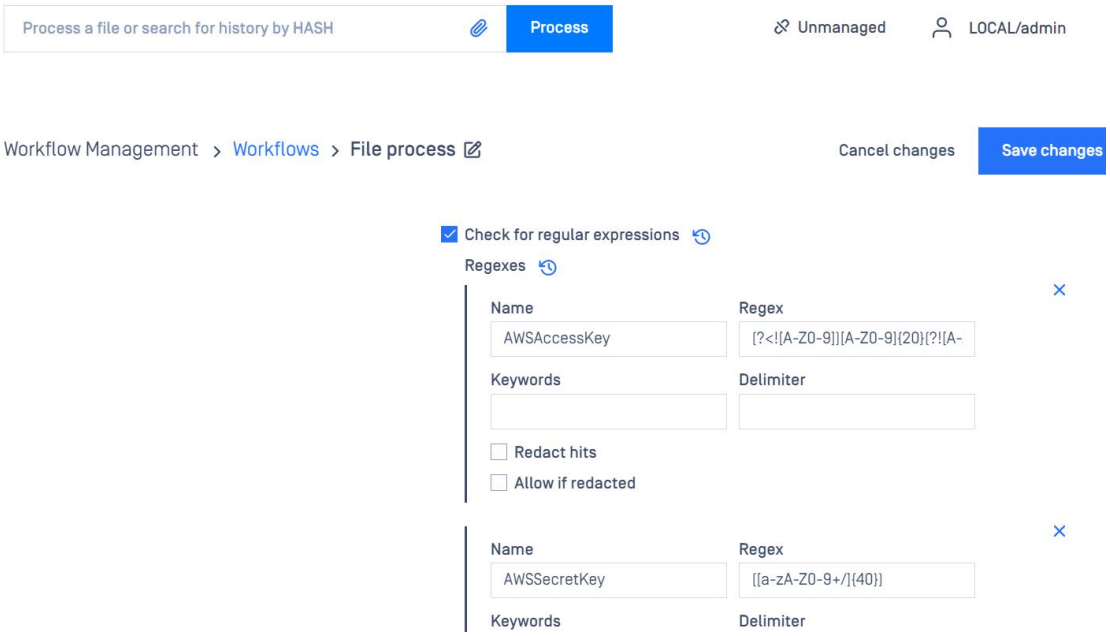


2. For Regular expression: {regex.name.atrribute} > X . The example below shows you how to block a file if it has both AWS Key ID and AWS Secret Key

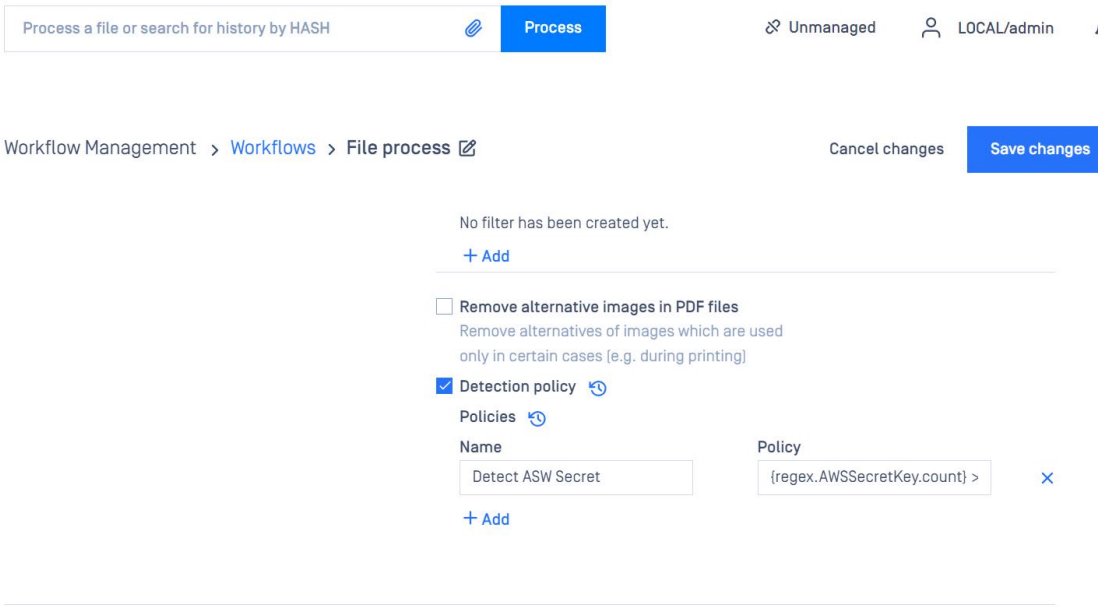
Create regular expressions to block AWS Access keys and AWS secret key

AWSAccessKey: (?<![A-Z0-9])[A-Z0-9]{20}(?![A-Z0-9])

AWSSecretKey: ([a-zA-Z0-9+/]{40})



Detection Policy: {regex.AWSAccessKey.count} > 0 and {regex.AWSSecretKey.count} > 0



3. For Metadata: {metadata.name.count} > X

Supported attribute:

- count: Only hits with equal or higher certainty than the CERTAINTY THRESHOLD will be counted

Supported arithmetic operators: <, >, =, <=, >=, <>

- Example 1: {ccn.count} >= 5
- Example 2: {ccn.count} <> {ssn.count}

Supported logical operators: and, nand, or, nor, xor

- Logical operators must be lower case
- Precedence of the operators are implemented in the following order: **nand**, **and**, **nor**, **or**, **xor**
- Example: {ccn.count} >= 5 or 1 < {metadata.Meta.count}

Supported parentheses : ()

- Always prioritizes the content to be evaluated first (**Highest precedence**)
- Example: ({ccn.count} >= 5 or {ssn.count} > 2) and {ipv4.count} > 3

Note:

- The Predefine Sensitive Types and regex name are case sensitive
- Regex name and metadata name can't have space, dot or curly brackets
- Predefine Sensitive Types should be enabled if they are used in the policy, otherwise the validation during file processing will be failed.

Detection Policy - Supported Entities

Here is the list of supported entities

Sensitive Type Name	Sensitive Type ID
Credit Card Number (CCN)	ccn
ABA Routing Number	aba_routing___number
U.S. Bank Account	us_bank___account
International Banking Account Number (IBAN)	iban
International Securities Identification Number (ISIN)	isin
SWIFT Code	swift_code
Japanese Personal My Number	jp_persmynum
U.S. Individual Taxpayer Identification Number (ITIN)	us_itin
U.S. Social Security Number (SSN)	us_ssn
Turkish Identification Number (Kimlik No.)	tr_kimlik___number
Israeli National ID Number	il_national___id
Australian Tax File Number (TFN)	au_tfn
Turkish Passport Number	tr_passport
Turkish Phone Number	tr_phone___number
IPv4 address or subnet mask	ipv4
Classless Inter-Domain Routing (CIDR)	cidr
International Mobile Equipment Identity (IMEI/ IMEISV)	imei_number
Australian Company Number (ACN)	au_acn
Australian Business Number (ABN)	au_abn

Sensitive Type Name	Sensitive Type ID
Australian Medicare Number	au_medicare
U.K. National Health Service Number (NHS)	uk_nhs
Driver's License Number detected by AI	driving_lic___by_ai
National ID Number detected by AI	national_id___by_ai
Passport Number detected by AI	passport_by___ai
Amazon Web Services (AWS)	aws_access___key_id
Amazon Web Services (AWS)	aws_secret___access_key
Amazon Web Services (AWS)	aws_temporary___access_key_id
Amazon Web Services (AWS)	aws_session___token
Amazon Web Services (AWS)	aws_sts___token
Microsoft Azure	azure_sas___signature
Microsoft Azure	azure_access___key
Microsoft Azure	azure_access___token
Microsoft Azure	azure_tenant___id
Microsoft Azure	azure_client___id
Microsoft Azure	azure_client___secret
Microsoft Azure	azure_username
Microsoft Azure	azure_password
Google Cloud Platform (GCP)	gcp_api___key
Google Cloud Platform (GCP)	gc_oauth___client_id

Sensitive Type Name	Sensitive Type ID
Google Cloud Platform (GCP)	gcp_oauth___client_secret
IBM Cloud Platform	ibm_cloud___api_key
IBM Cloud Platform	ibm_api___connect_credential_id
IBM Cloud Platform	ibm_api___connect_credential_secret
IBM Cloud Platform	ibm_cos___hmac_access_key
IBM Cloud Platform	ibm_cos___hmac_secret_key
MySQL	mysql_connection___uri
MySQL	mysql_cli___command
PostgreSQL	postgresql_connection___uri
PostgreSQL	postgresql_cli___command
Private keys	private_key___pem
Private keys	private_key___ppk
API keys (BETA)	api_key___common
Passwords (BETA)	password_generic___password

Secrets Detection

Nowadays, it is very common for people to commit secrets into repositories or even to store them in code by accident. There are many types of secrets, including keys, passwords, API tokens, and other sensitive information. If anyone had access to these secrets, they could use them maliciously. When secrets are exposed to the public, they are compromised and have to be replaced, which can be very expensive. The Proactive DLP now has the ability to help prevent secrets from being exposed. Using the Secret Detection feature, you will be able to scan your files for secrets they might contain.

Supported file types

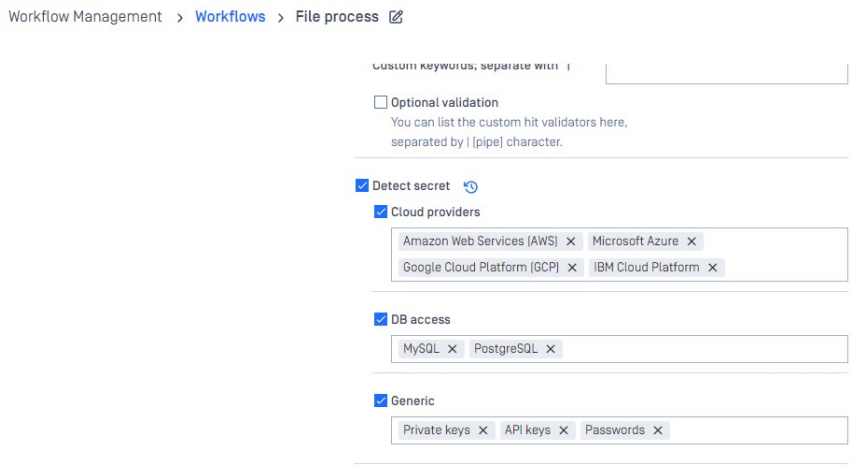
- Text based files, e.g: source code, configuration files.

Supported Secrets

- Amazon Web Services (AWS)
  - AWS Access Key ID
  - AWS Secret Access Key
  - AWS Temporary Access Key ID
  - AWS Session Token
  - AWS STS Token
- Microsoft Azure
  - Azure SAS Signature
  - Azure Access Key
  - Azure Access Token
  - Azure Tenant ID
  - Azure Client ID
  - Azure Client Secret
  - Azure Username
  - Azure Password
- Google Cloud Platform (GCP)
  - Google Cloud Api Key
  - Google Cloud Key *(We search for the base64 encoded version of the key as well)*
  - Google OAuth Client ID *(We search for reversed client ID as well)*
  - Google OAuth Client Secret
- Private keys (PEM, PPK)
- IBM keys
  - IBM Cloud key
  - IBM COS HMAC Credentials
  - IBM API Connect Credentials
- Generic passwords
- Generic API tokens
- PostgreSQL credentials
- MYSQL credentials

Enabling Secret Detection

Policies > Workflow rules > Workflow name > Proactive DLP > Detect secret



Example

Info

The following examples do not represent actual secrets.

Example
'AWS_ACCESS_KEY_ID' = 'AKIAIOSFODNN7EXAMPLE'
'aws_secret_ access_key' = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
'AWS Temporary Access Key ID' = 'ASIAIOSFODNN7EXAMPLE'
'AWS Session Token' = 'FQoGZXIvYXdzEF8aD0YjK34RwRy8tSJEXAMPLEFa3fLgDzZrW2Z8t9Z9Yg4MupJys4l+2 H4ZDQDbTdEXAMPLES5N5OesDL5x5j5fiUqhEXAMPLEJUu1QuyK0CvLln6JFV7lEXAMPLE'
'Azure SAS Signature' = '? sv=2020-08-04&ss=bfqt&srt=sco&sp=rwdlacupx&se=2023-05-01T22%3A17%3A16Z&st=2023-04-24T14%3A17%3A16Z&spr
'Azure Access Key' = 'Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsu Fq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksc
'Azure Access Token' = 'eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik1u Q19WWmNBVGZNNXBPWWlKSE1iYTlnb0VLWSIsImtpZCI6Ik1uQ19WWmNBVG ZNNXBPWWlKSE1iYTlnb0VLWSJ9.eyJhdWQiOiJodHRwczovL21hbmFnZW1lbn QuY29yZS53aW5kb3dzLm5ldC8iLCJpc3Mi zLm5ldC8wMTIzNDU2Nzg5L2I4YmY0M2YtNmVlOC00ZmUxLWEzOTEtMGRiOG I1ZDM3Y2Y2LyIsImlhdCI6MTU5Mzg5OTQ2NCwibmJmIjoxNTkzODk5NDY0LCJl

Example
eHAiOjE1OTM5MDMzNjQsImFjciI6IjEiLCJhaW8iOiJBU1FBRG8vOEhBQUFBQVFB Q2dIY0FBQUY1YmVyM0l1dCIsImFtciI6WyJwd2QiXSwiYXBwaWQiOiI5OTljZDI2M C0zZjVhLTQ1NzctYmEwZC0xMjg2MmFmMjMxODgiLCJhcHBpZGFjciI6IjAiLCJmY W1pbHlfbmFtZSI6IkJyb3duIiwgImdpdmVuX25hbWUiOiJKb2huIiwiaXBhZGRyIjoiMT IzLjQ1LjY3Ljg5IiwibmFtZSI6IkpvaG4gQnJvd24iLCJvaWQiOiI5Zjg4ZWI4YS0yYmUyLT Q5ZmEtOTk5My1kYjg4YzUyZTAzZTgiLCJwdWlkIjoiMTAwMzAwMDA4Rjg4RUI4QSIsI nNjcCI6InVzZXJfaW1wZXJzb25hdGlvbiIsInN1YiI6ImtmcGZ2eWJrVG9QZ1MzVzF0dW x0VzJYUkNkVzlpWU0zR1RtM0RtWWp0VlEiLCJ0aWQiOiIwMTIzNDU2Nzg5L2I4YmY 0M2YtNmVlOC00ZmUxLWEzOTEtMGRiOGI1ZDM3Y2Y2IiwidW5pcXVlX25hbWUiOi Jqb2huLmJyb3duQGV4YW1'
'Azure Tenant ID' = '12345678-9abc-def0-1234-56789abcdef0'
'Azure Client ID' = 'a1b2c3d4-e5f6-1234-5678-90abcdef1234'
'Azure Client Secret' = 'P@ssw0rd!a1b2c3d4e5f6g7h8i9j0k!l@Mn0'
'Azure Username' = 'john.doe@example.com'
'Azure Password' = 'Azure$tr0ngP@ssw0rd123!'
'Google Cloud API Key' = 'AIzaSyDc7A03Xm8f9gh1ij2k3456L789mNO0pqR'
'Google OAuth Client ID' = '123456789012-abcdefghijklmno1pqrstuvwxyz.apps.googleusercontent.com'
'Google OAuth Client Secret' = 'Xyz1A2B3C4d5e6F7g8H9i0'
-----BEGIN RSA PRIVATE KEY----- MIIEowIBAAKCAQEAzC0EXAMPLEKEYCONTENTVwIDAQAB -----END RSA PRIVATE KEY-----
'IBM Cloud API Key' = 'fYgHijKlmn123_OPqrst456-UvWx789zABCDEF01'
' IBM COS HMAC Access Key ID' = '1a2b3c4d5e6f7g8h9i0j1k2l3m4n5o6p'
'IBM API Connect Client Secret' = 'AbCdEfGhIjKlMnOpQrStUvWxYz123456'
Host: db.example.com Port: 5432 Database: example _ db Username: example _ user Password: P@ssw0rd!123
Host: db.example.com Port: 5432 Database: example _ db Username: example _ user Password: P@ssw0rd!123

Example
'Generic Password' = 'S3cureP@55w0rd!2023'
'Generic API Key' = '1234abcd5678efgh9012ijkl3456mnop'

Optical Character Recognition (OCR)

OCR is a commonly-used technology to recognize text inside images. It examines the text of the documents and converts the characters into code that can be used for data processing. Proactive DLP now can utilize this technology to detect and redact sensitive information.

Supported file types

Category	Formats / Extensions
Portable Document Format	PDF
Microsoft Office	doc, docx, MS Word XML, xls, xlsx, ppt, pptx, rtf
Standalone Images	jpg, png, tiff, bmp, jp2, jpg2, jpf, jpx, mj2, mjp2, jpm, jpgm

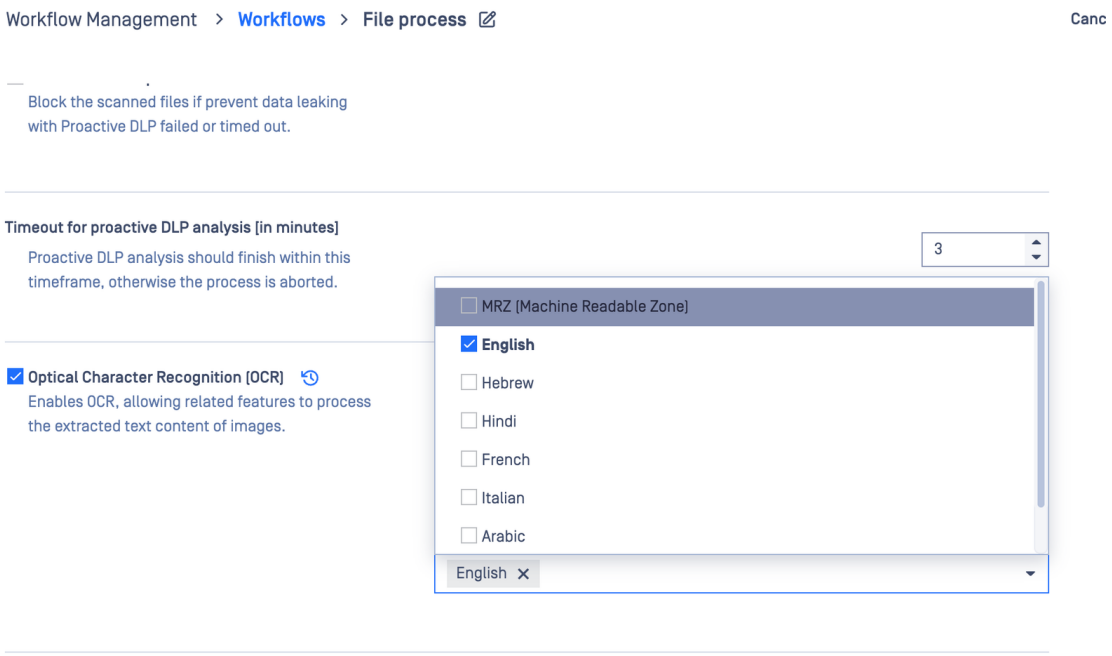
Supported languages

1. Arabic
2. English
3. French
4. Hebrew
5. Hindi
6. Italian
7. Japanese

*The OCR feature can also detect MRZ (Machine Readable Zone) data from ICAO-compliant ID cards.

Enabling OCR

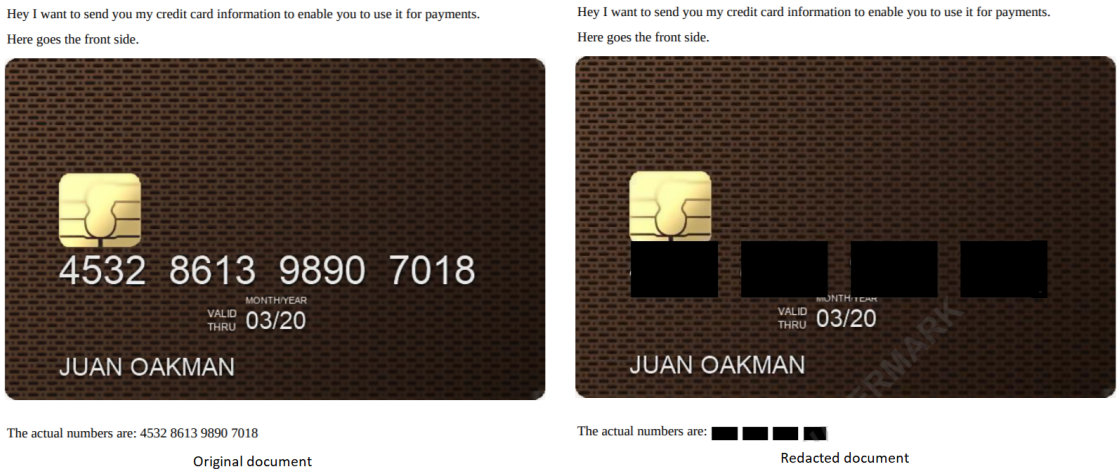
Policies > Workflow rules > "Workflow name" > Proactive DLP > Optical character recognition (OCR)



OCR Quality:

- Normal: detect the information without pre-processing images
- Best: pre-processing images before detecting the image to have a better detection rate, however, performance will be impacted

Example output



System requirements

- CPU must support AVX2 and SSE4.1 instruction set

Vectors can affect the accuracy

- Low contrast documents
- Documents with small text

- Documents with blurry images
- Colored paper or background in documents
- Handwritten text
- Unusual or script-type fonts

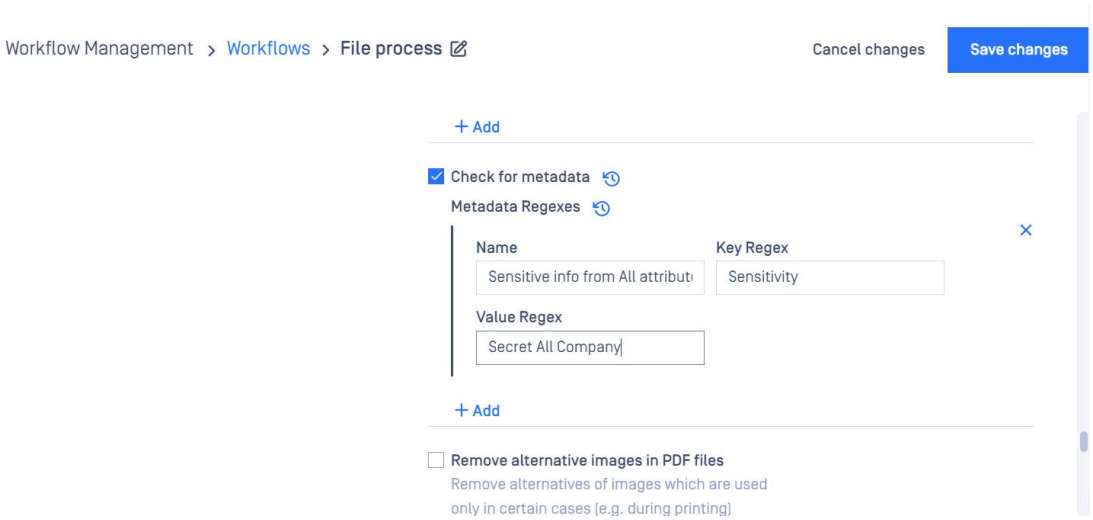
This feature uses AI functionality for Optical Character Recognition (OCR) to extract text from images and scanned documents, converting it into machine-readable text. You have the option to enable or disable the AI-powered feature within the configuration settings. By default, this feature is disabled but can be activated or deactivated at any time based on user preference. If you enable the AI-powered feature, OPSWAT will not use your content to train or fine-tune its services. You should not rely on any results generated from AI-based functionality without verifying them.

Leverage metadata info added by data classification systems

Metadata can provide information that is useful for classifying the file sensitivity. There are several systems that automatically categorize documents using custom metadata or document properties, such as Windows Azure Information Protection, Titus, etc. These systems add specific information to files' properties demonstrating how confidential the documents are. Proactive DLP now can read files' properties and block the sensitive items based on rule configurations.

To set the detection:

- *Workflow Management > Workflows > Workflow name > Proactive DLP > Check for metadata*



– –

Supported file types: pdf, xls, xlsx, MS Excel XML, doc, docx, MS Word XML, ppt, pptx, rtf, bmp, jpg, png, gif, tiff

Process hidden parts in a cropped image

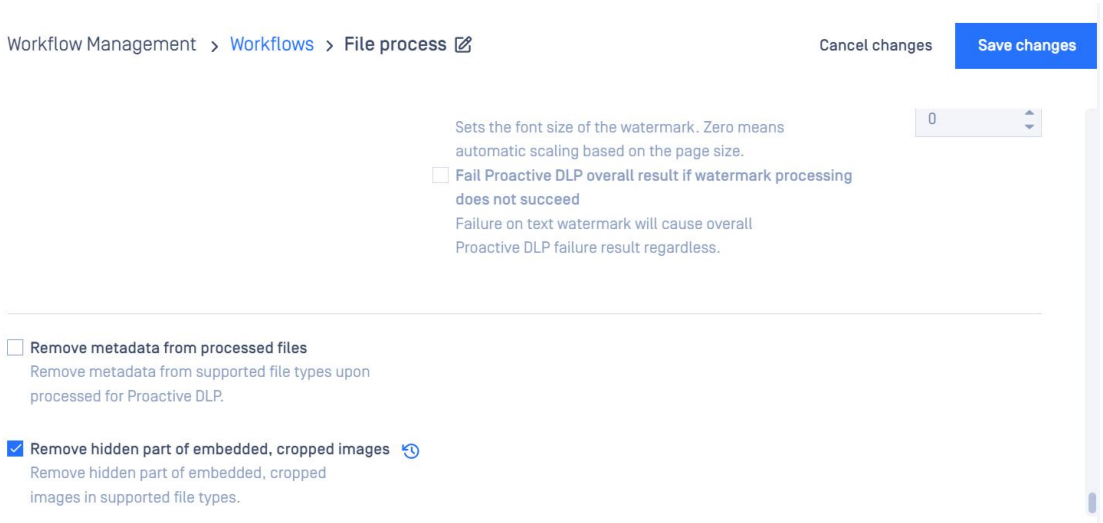
Cropping pictures inserted an Office file (such as a Word document, PowerPoint presentation, or Excel workbook) is a convenient tool in Microsoft Office apps that widely used by many people. The problem is that Office's cropping

tool only modifies how the cropped image is displayed in the body of the document. Actually, the original picture remains intact. Cropped portions of the image are not completely removed from the document and able to be seen by others. So, data leakage can occur if there is sensitive data in the trimmed areas.

Proactive DLP engine can detect sensitive data within cropped parts of images embedded in MS Word files (.doc and .docx) by leveraging Optical Character Recognition (OCR) technology. Furthermore, administrators also can configure this feature to permanently remove the cropped areas of the images.

To configure this feature:

- *Workflow Management > Workflows > Workflow name > Proactive DLP > Check for "Remove hidden part of embedded, cropped images"*



Support file types: MS Office (DOC/DOCX/XLS/XLSX/PPT/PPTX/PPS/PPSX/POT/POTX/PPTM/PPSM/POTM/ Microsoft Office Word XML)

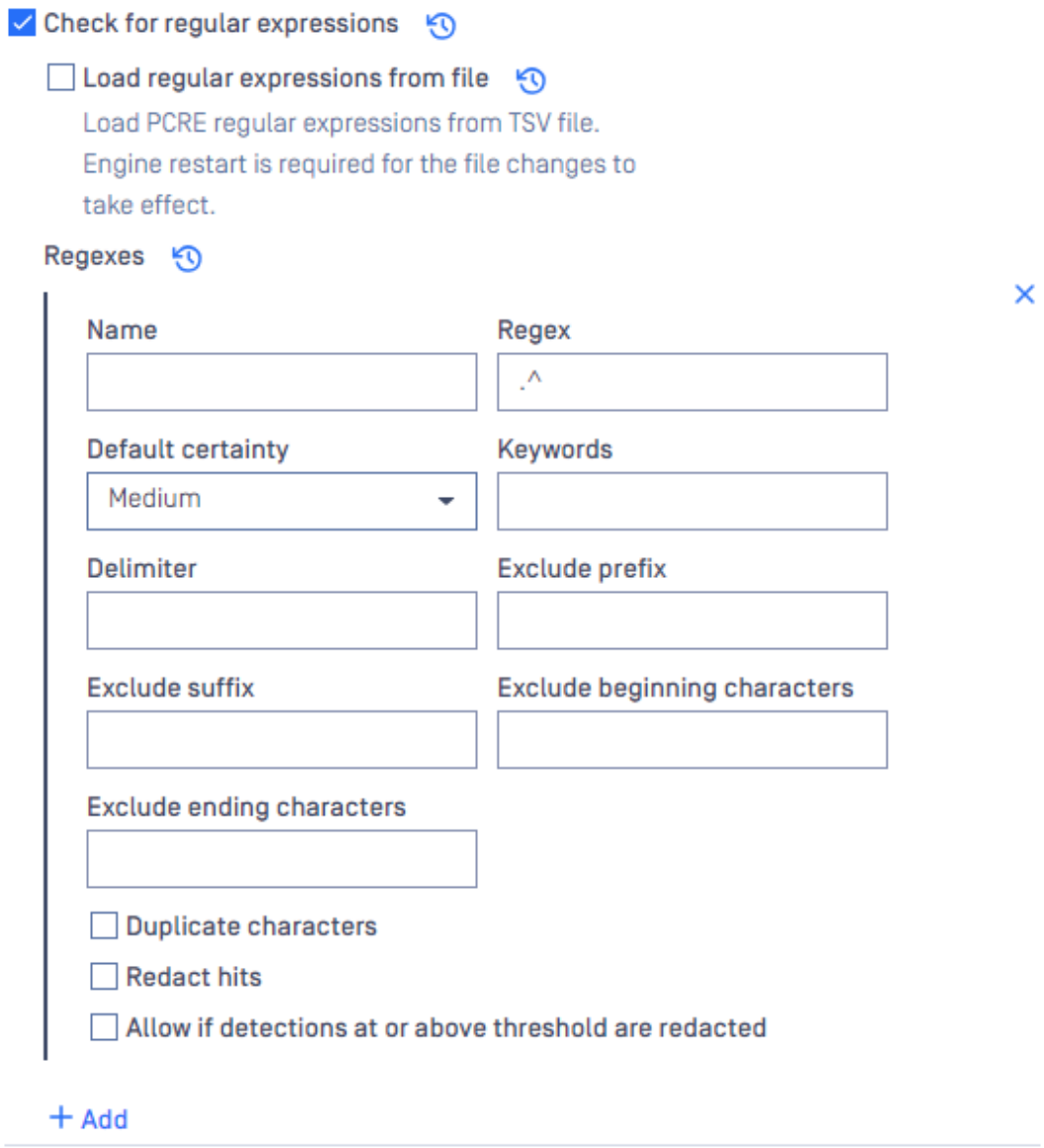
The feature is available from MetaDefender Core 4.19.2 or newer

Regular expressions

Regular expressions, or regex, are a way to search for specific patterns in text. In PDLP, they help find and protect sensitive information. If you're looking for a specific pattern, you can introduce your own regex to define additional sensitive info. This section will introduce how regular expressions work in the PDLP. We'll go over each part step by step.

Enabling regular expression

Policies > Workflow rules > "Workflow name" > Proactive DLP > Check for regular expressions



Let's discuss each section and provide an example:

- **Name**
  - The label or identifier for the regular expression pattern
  - Example: **Bank Account Number**
- **Regex**
  - The specific pattern or sequence you're trying to match in the text.
  - Example: **\d{8,17}\b**
- **Default certainty**
  - A confidence level assigned to matches of this regex pattern.
  - Example: **High**

- **Keywords**
  - Specific words or terms related to the regex pattern that might assist in its identification.
  - Example: **Bank, Account Number, Bank Account Number, USA Bank**
- **Delimiter**
  - Validates a match by checking the surrounding characters. Only match the exact standalone string.
  - Example: **;**
- **Exclude prefix**
  - Characters or patterns to ignore directly before the regex match.
  - Example: **Invalid-**
- **Exclude suffix**
  - Characters or patterns to ignore directly after the regex match.
  - Example: **-Invalid**
- **Exclude beginning characters**
  - Applied to data that matches the Data identifier Pattern. If a hit starts with the given patterns, it will be excluded.
  - Example: **{**
- **Exclude ending characters**
  - Applied to data that matches the Data identifier Pattern. If a hit ends with the given patterns, it will be excluded.
  - Example: **}**
- **Duplicate characters**
  - Ensure that a string of digits are not all the same.
- **Redact hits**
  - Determines if matches should be redact or replaced with substitute characters.
- **Allow if detections at or above threshold are redacted**
  - It allows data to pass if the match is at or above the threshold.

Samples

Below are some examples of regular expressions you can use

Name	Description	Regular Expression
Email	The regex matches on any email. For example: • jwhite@domain.com • aborden@gmail.com • jmunsch@hotmail.fr • gwatson@hotmail.co.uk	`(?:^|\s)[\w!#$%&'*+/=?^`{\|}~-](\.?[\w!#$%&'*+/=?^`{|}~-])*@\w+[.-]?\w*\.[a-zA-Z]{2,3}\b`
Email	The regex matches on email with a specific domain. For example: • jwhite@company.com	`(?:^|\s)[\w!#$%&'*+/=?^`{\|}~-](\.?[\w!#$%&'*+/=?^`{|}~-])*@company.com`

U.S. Phone Number	The regex matches on a U.S. Phone number.
	For example:
	• Local: 754-3010
	• Domestic: (541) 754-3010
	• International: +1-541-754-3010
	• Dialed in the U.S.: 1-541-754-3010
	• With extension number:
	  • (541) 754-3010 x350
	  • (541) 754-3010 Ext. 350
	  • (541) 754-3010 Extension 350
		`(?:(?:\+?1[-.\s])?\(?\d{3}\)?[-.\s])?\d{3}[-.\s]\d{4}(?:\s(?:x|#|[eE]xt[.]?|[eE]xtension){1} ?\d{1,7})?\b`

U.S. Address	The regex matches on a full U.S. Address. For example: • 1775 Washington St, Hanover, MA 02339 • 350 E Fairmount Ave, Lakewood, NY 14750	`\d{1,5}(\s[\w\-.,]*){1,6},\s[A-Z]{2}\s\d{5}\b`

Name	Description	Regular Expression
Full name	The regex matches on a string that contains “Full name“ keyword and 2 or 3 words. For example: • Full name: Lisa garrison Simpson	`Full name:\s[A-Z][a-z]+(?:[ \t]*[A-Z]?[a-z]+)?[ \t]*[A-Z][a-z]+\b`
U.S. Driver license number	The regex matches on a State Driver's license number. For example: • California: A1234567 • Texas: 1234567 or 12345678 • Florida: A123456789123	California: `\b[A-Za-z]{1}[0-9]{7}\b`
U.S. Bank Account number	The regex matches on a string that contains the “Bank Account Number“ keyword and an 8 to 17 digits number. For example: • Bank Account Number: 12345678 • Bank Account Number 12345678901234567	`Bank Account Number\W*\d{8,17}\b`
U.S. Passport number	The regex matches on a string that contains a Passport related keyword and a 9 digits number. For example: • Passport Number: 123456789 • Passport No. 123456789	`(Passport Number|Passport No|Passport #|Passport#|PassportID|Passportno|passportnumber)\W*\d{9}\b`
Date of Birth	The regex matches on a date with the YYYY/MM/DD format and a "Date of birth:" or "Birthday:" prefix (Year min: 1900, Year max: 2020).	`(Date of birth:|Birthday:)\s+(?:19\d{2}|20[01][0-9]|2020)[-/.](?:0[1-9]|1[012])[-/.](?:0[1-9]|[12][0-9]|3[01])\b`

Name	Description	Regular Expression
	For example: • Date of birth: 1900/12/01 • Date of birth: 2019.01.25 • Birthday: 2020-10-30	
Date of Birth	The regex matches on a date with the DD/MM/YYYY format and a "Date of birth:" or "Birthday:" prefix (Year min: 1900, Year max: 2020). For example: • Date of birth: 12/01/1900 • Date of birth: 01.12.2019 • Birthday: 10-10-2020	`(Date of birth:|Birthday:)\s+(?:0[1-9]|[12][0-9]|3[01])[-/.](?:0[1-9]|1[012])[-/.](?:19\d{2}|20[01][0-9]|2020)\b`
Date of Birth	The regex matches on a date with the MM/DD/YYYY format and a "Date of birth:" or "Birthday:" prefix (Year min: 1900, Year max: 2020). For example: • Date of birth: 12/01/1900 • Date of birth: 01.25.2019 • Birthday: 10-30-2020	`(Date of birth:|Birthday:)\s+(?:0[1-9]|1[012])[-/.](?:0[1-9]|[12][0-9]|3[01])[-/.](?:19\d{2}|20[01][0-9]|2020)\b`

Importing Regular Expressions via a TSV File

In this section, we will guide you through the process of importing regular expressions using a TSV (Tab-Separated Values) file and integrating them into the PDLP. We'll cover the basics of preparing your TSV file for regular expressions and the steps to import them effectively into the PDLP.

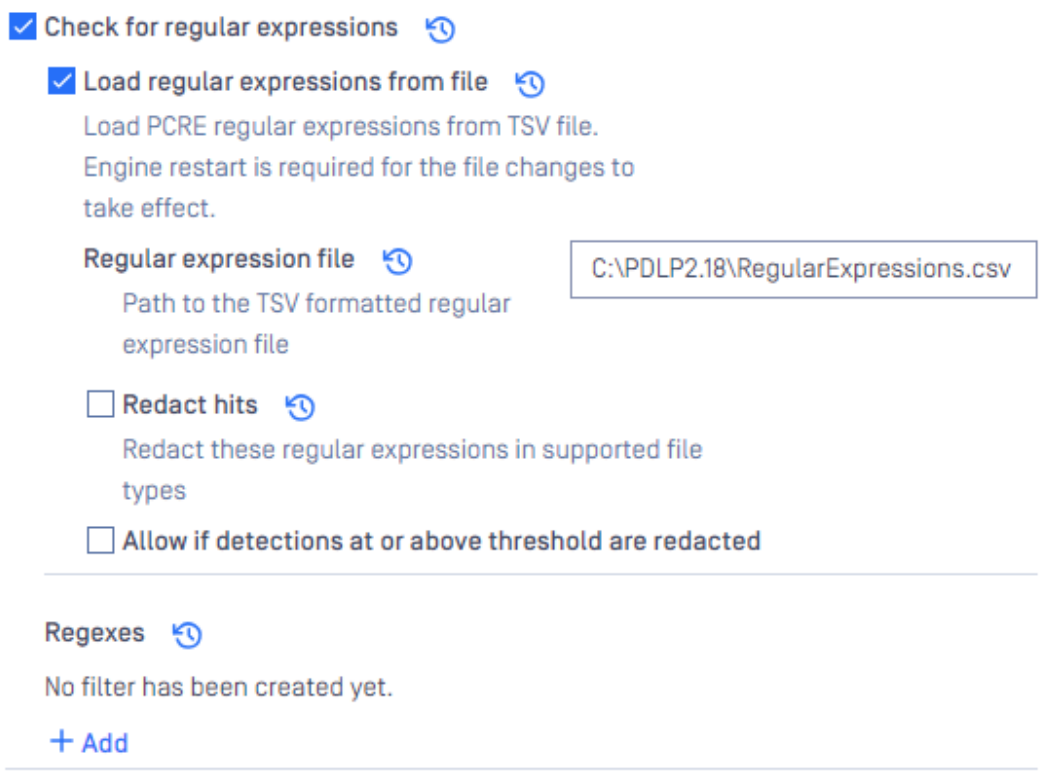
Structure of the TSV File

- Only UTF-8 file encoding is supported.
- Each regular expression should be represented on a single line. The format of each line could be:
  - <NAME> <TAB> <RegEX>
  - <RegEX>

- NAME: Should not exceed 64 characters and may contain alphanumeric characters and underscores ([a-zA-Z0-9_]).
  - If the NAME is not provided, a default will be generated in the format: CUSTOM_FILE_REGEX_<line number>.
- SEPARATOR: A TAB character will be used
- REGULAR EXPRESSION: Everything following the TAB character is treated as the regular expression. If the NAME and TAB are not present, the entire line will be considered as the regular expression.

Enabling Importing Regular Expression

Policies > Workflow rules > "Workflow name" > Proactive DLP > Check for regular expressions > Load regular expressions from file



Let's discuss each section and provide an example:

- **Regular expression file:** Path to TSV formatted regular expression file

Important Notice

- In the event of an error, non-compliance with the specified format, or incorrect regex, the engine will fail. Please consult the engine logs to determine if there are any errors in the file you provided.

Limitations

These are the current limitations related to TSV custom regex feature.

- Max TSV file size: 500 * 1024 byte (~500 Kb)
- Max TSV line number: 255 Line
- Max TSV Line characters: 300 character (Name length + tabulator + regex length)
- Max regex Name length: 64 character

Samples

Below is an example of a regular expressions file you can use

Code

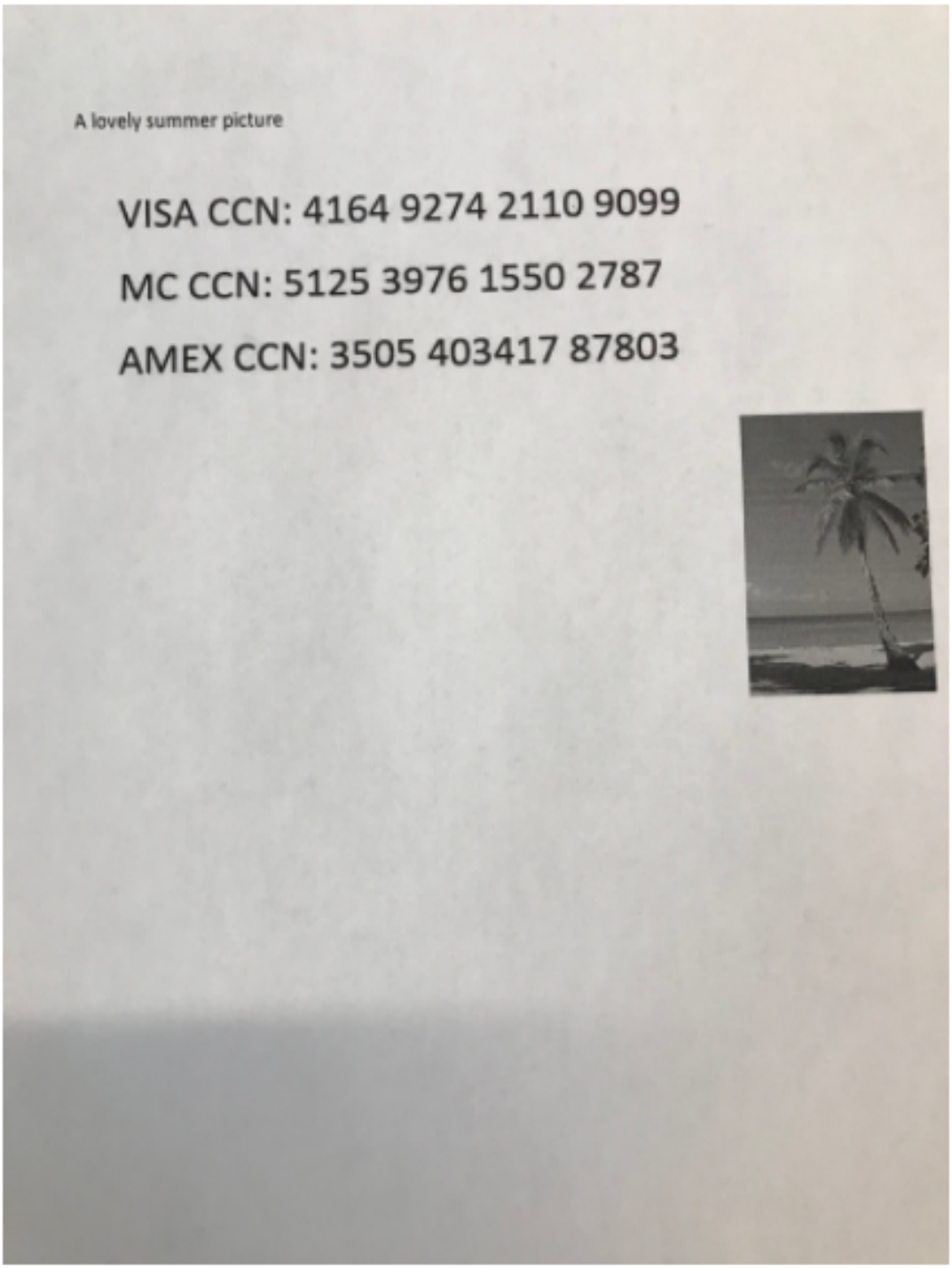
```
Email    ([a-zA-Z0-9_\.-]+)@([\da-zA-Z\.-]+)\.([a-zA-Z\.]{2,6})
URL    https?:\/\/(?:www\.)?([-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b)*(\/[\/\d\w\.-]*)*(?:
[\?])*(.+)*
US_Address    \d{1,5}(\s[\w.,]*){1,6},\s[A-Z]{2}\s\d{5}\b
California_Driver_License    California: \b[A-Za-z]{1}[0-9]{7}\b
Date_of_Birth    (Date of birth:|Birthday:)\s+(?:19\d{2}|20[01][0-9]|2020)[-/.](?:0[1-9]|1[012])
[-/.](?:0[1-9]|[12][0-9]|3[01])\b
Full name:\s[A-Z][a-z]+(?:[ \t]*[A-Z]?[a-z]+)?[ \t]*[A-Z][a-z]+\b
(Passport Number|Passport No|Passport #|Passport#|PassportID|Passportno|
passportnumber)\W*\d{9}\b
```

Remove alternative images in PDF

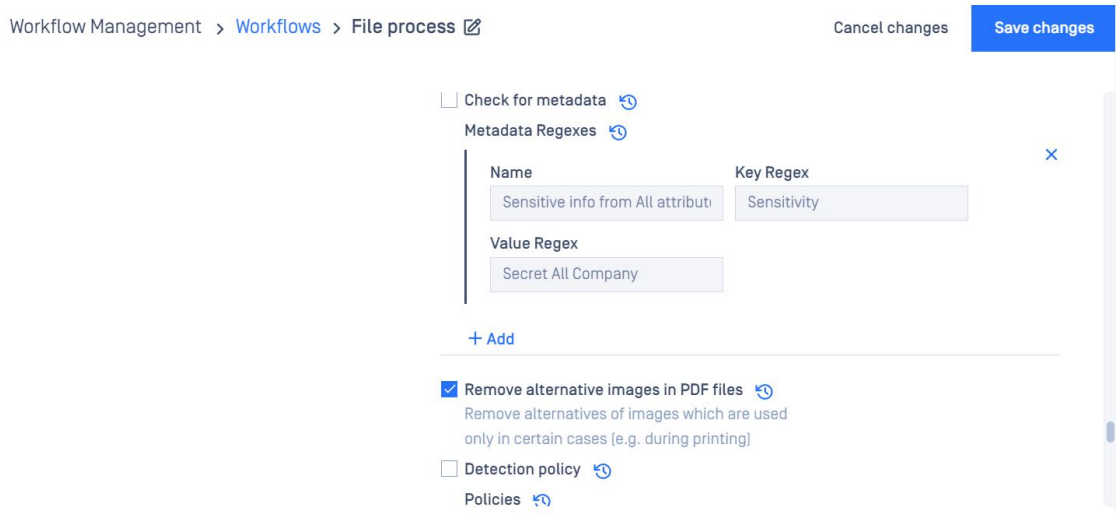
The use of alternate tags within the PDF allows authors to define what image is displayed when printed. This means if there is sensitive information in the second image, someone can pass it on to an external source and view it with a click of the print button.

A lovely summer picture





To prevent this case, go to **Workflow Management > Workflows > Proactive DLP** enable **Remove Alternative Images in PDF Files**



** **

The output file after processing contains the first image only

Supported engines-metadata JSON

Overwrite file encoding detection

The Proactive DLP engine has the capability to detect file encoding and process the content with that encoding. However, by nature, it is not 100% accuracy with file encoding detection. In several cases, the clients determine exactly the encoding of the file, so they can send this information to the engine to have a better detection. To utilize this feature, please see the bellow steps:

1. Use MetaDefender Core API to send a file

2. Use the **engines-metadata** to send the encoding

JSON · javascript

```
{
  "proactive_dlp": {
    "charset": "ISO-2022-JP"
  }
}
```

The engine uses UTF-8 as default encoding if the header is not set

Supported metadata type in PDLP

This page indicates the types of metadata supported by OPSWAT PDLP for each type of file. A list of these metadata can be found below:

	File Type	Metadata

	File Type	Metadata
1	gif	XMP
2	tiff	XMP, EXIF
3	bmp	XMP
4	jpeg*	XMP, EXIF
5	png	XMP
6	pdf	built-in properties, custom properties
7	xls	built-in properties, custom properties
8	xlsx	built-in properties, custom properties
9	xlm	built-in properties, custom properties
10	xml_xls	built-in properties, custom properties
11	ppt	built-in properties, custom properties
12	pot	built-in properties, custom properties
13	pps	built-in properties, custom properties
14	pptx	built-in properties, custom properties
15	potx	built-in properties, custom properties
16	ppsx	built-in properties, custom

	File Type	Metadata
		properties
17	pptm	built-in properties, custom properties
18	potm	built-in properties, custom properties
19	ppsm	built-in properties, custom properties
20	rtf	built-in properties, custom properties
21	doc	built-in properties, custom properties
22	docx	built-in properties, custom properties
23	docm	built-in properties, custom properties
24	odt	built-in properties, custom properties
25	ods	built-in properties, custom properties
26	xml_doc	built-in properties, custom properties
27	xml_docx	built-in properties, custom properties

The following list will provide a more detailed description of each of the metadata:

1. **XMP** (Extensible Metadata Platform): if available, except for the jpeg format, which requires some information to be retained in order to maintain the quality of the original image.
2. **EXIF** (Exchangeable Image File Format): if available, except for the jpeg format, which requires some information to be retained in order to maintain the quality of the original image.

3. **Built-in properties**: In cases where it is applicable; title, creator, subject, company, description manager, comment, producer, creationDate, modDat.
4. **Custom properties**: In cases where it is applicable; Extensible Metadata Platform such as the following: title, language, creators, contributors, rights, coverage, date, description, identifier, publisher, relation, source, subject, type, title, creator, lastModifiedBy, Company, revision, created, modified, codepage, author, keywords, comments, template, lastsaved by, *revisionnumber,* total edit_time, last_ printed, create_time, last_saved_time, num_pages, num_words, num_chars, thumbnail, creating_application, security, codepage_doc, category, presentation_target, bytes, lines, paragraphs, slides, notes, hidden_slides, mm_clips, scale_crop, heading_pairs, titles_of_parts, manager, company, links_dirty, chars_with_spaces, unused, shared_doc, link_base, hlinks, hlinks_changed, version, dig_sig, content_type, content_status, doc_version, and other fields that can be represented in xmp metadata (like user defined properties)

---

*In order to maintain the quality of the original image, some fields must be preserved. The following is a list of exceptions that are allowed for jpeg fields in metadata: quality, horizontal resolution, vertical resolution, resolution units, DPI info, bits per channel, CMYK color profile, color type, compression type, horizontal sampling, vertical sampling, palette, preblend alpha (if present), RGB color profile, sample rounding mode, color space, brightness value, contrast, compressed bpp, gain control, gamma, saturation, sharpness, orientation, white balance.

Redact/Substitution/Hash sensitive information

It is imperative to safeguard sensitive data as soon as it has been detected in order to prevent unauthorized access to it. In order to address this concern, OPSWAT Proactive Data Loss Prevention (PDLP) system comes with two distinct features: **redaction**, **substitution** and **Hash**:

- **Redaction**: As a result of redaction, sensitive information such as Credit Card Numbers (CCNs) and Social Security Numbers (SSNs) is concealed, while maintaining the overall structure and layout of the document as a whole. It is important to note that this method ensures that the private information is hidden while maintaining the readability of the remaining content.
- **Substitution:** On the other hand, the substitution feature is able to replace sensitive data with placeholders, thus adding another level of security for confidential information. OPSWAT PDLP system preserves privacy and prevents unauthorized access to sensitive data by incorporating both redaction and substitution features.
- **Hash:** This feature introduces a secure hashing mechanism for PII substitution, enabling compliant and safe data sharing. The hashing process ensures that any substituted personal information remains non-reversible and unintelligible if intercepted, providing an additional layer of security and protection against data exposure.
  - Standardized substitution format: PII is replaced with `<pii-id prefix>_<hash-value>`
  - Supported hashing algorithms: SHA-256 SHA-384 SHA-512



Supported File Types

- Portable Document Format (PDF)
- Microsoft Office Word (DOC/DOCX)
- Microsoft Office Word XML (XML)
- Microsoft Office Excel (XLS/XLSX)
- Microsoft Office Excel XML (XML)
- Microsoft Office PowerPoint (PPT/PPTX)
- Microsoft Rich Text Format (RTF)
- Extensible Markup Language (XML)
- Comma Separated Values (CSV)
- Text files (TXT, HTML)
- JPG *(Optical Character Recognition based)*
- PNG *(Optical Character Recognition based)*
- TIFF *(Optical Character Recognition based)*
- BMP *(Optical Character Recognition based)*

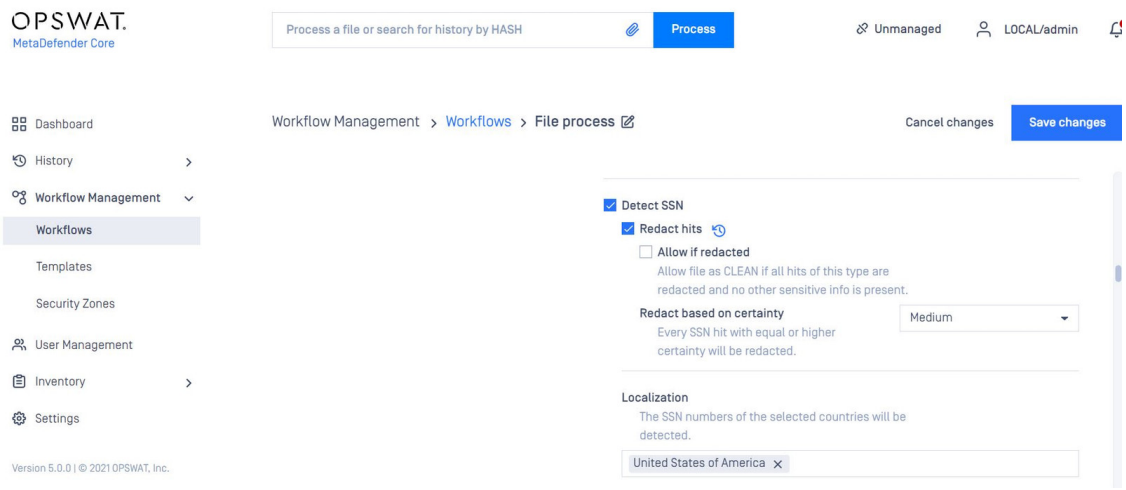
Supported Sensitive Information

- All categories under Detect Financial Information
- All categories under Detect personal identity information
- All categories under Detect network and device information
- All categories under Detect corporation information
- All categories under Detect drug and health information

- All categories under Detect sensitive information with AI (BETA)
- All categories under Detect secret
- Custom regular expression

Enabling redaction

- Workflow Management > Workflows > "Workflow name" > Proactive DLP



Enable "Allow if redacted" will mark the final result to be "Allowed" even the files have sensitive info

Sample redacted file

> Following sample does not represent an actual person.



Credit Card Authorization Form
One-Time & Repeat Gifts

CARDHOLDER INFORMATION

Name: **Brandon Patterson**

Billing Street Address: 7134 Glenridge Road

Street Address (cont.):

City: Mahwah State: NJ Postal Code: 07430

Country: U.S. Email [redacted]

Address:

Direct Telephone: ( ) -

GIFT INFORMATION

Fund Name or Gift Purpose: Give a gift to my son

x I authorize a one-time charge against my credit card for the follow amount $ 500.0

□ I authorize a recurring charge against my credit card for the following amount

CREDIT CARD INFORMATION

Credit Card Type: □ MasterCard X Visa □ American Express □ Discover Card

Number: [redacted]

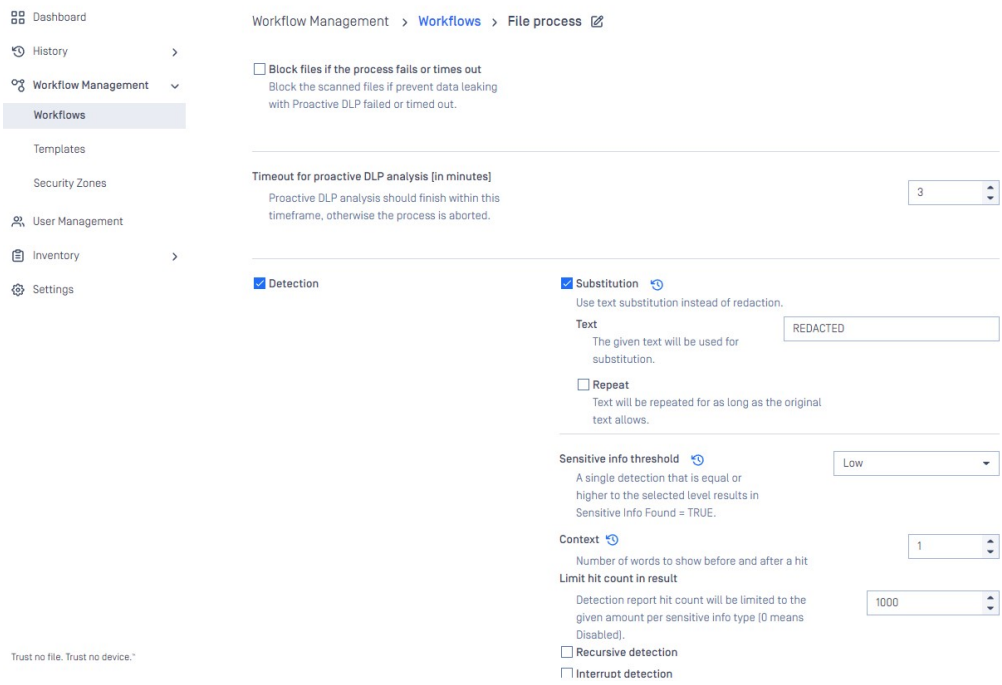
Expiration Month: 5 Expiration Year: 2020

Cardholder Signature X Date 5 /21 /2019

Security Code: 783

Enabling substitution

- Workflow Management > Workflows > "Workflow name" > Proactive DLP



By enabling the substitution feature, you can effectively replace sensitive data with alternate characters or placeholders based on your need.

> Following sample does not represent an actual person.

ACCOUNT TRANSFER REQUEST

To,
The Branch Manager
USA BANK

From:
Name:Joe Doe
Email:[redacted]

Dear Madam/ Sir,

Request for my Deposit Account Transfer
A/c No. BZ [redacted]
From SOUTH FLORIDA Code to 2ERTX

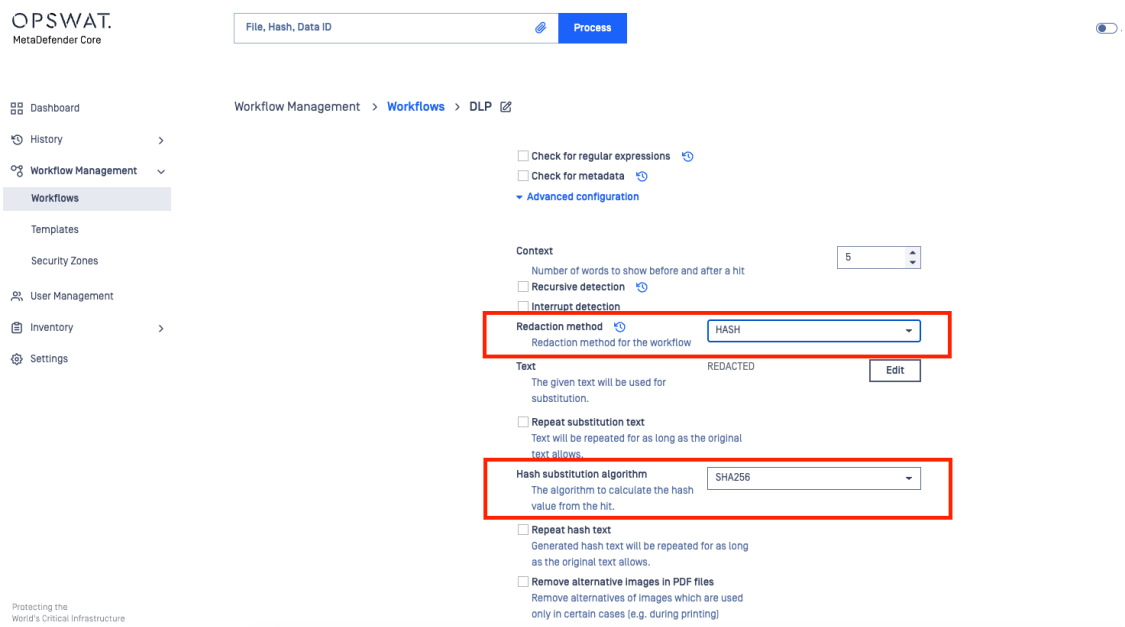
1.With your bank, I hold the following account/accounts.
2.The captioned account needs to be transferred. A copy of the new address proof is enclosed.
3.Please transfer My CIF to [redacted].
4.I understand that my Home Branch will continue to be the same if CIF is not transferred.
5.I would like to provide you with my Social Security Number, [redacted].
Please arrange accordingly.

Yours faithfully,

Joe Doe
Dated: 25th Apr, 2023

Enabling Hash

- Workflow Management > Workflows > "Workflow name" > Proactive DLP

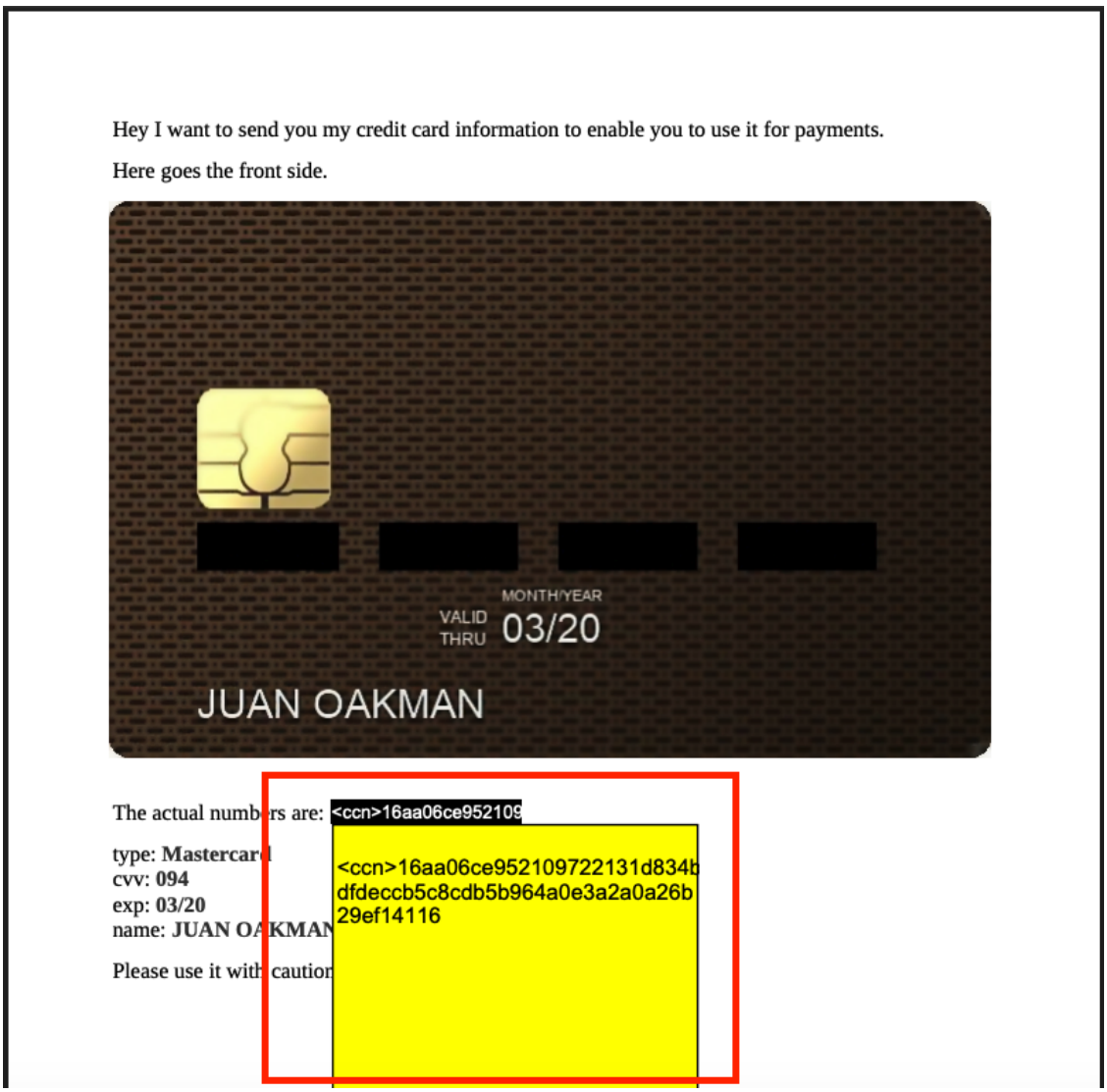


By enabling the Hash Redaction feature, the system replaces detected sensitive information with its hashed

equivalent, preserving data structure while protecting confidentiality.

Info

Following sample does not represent an actual person.



Watermark files

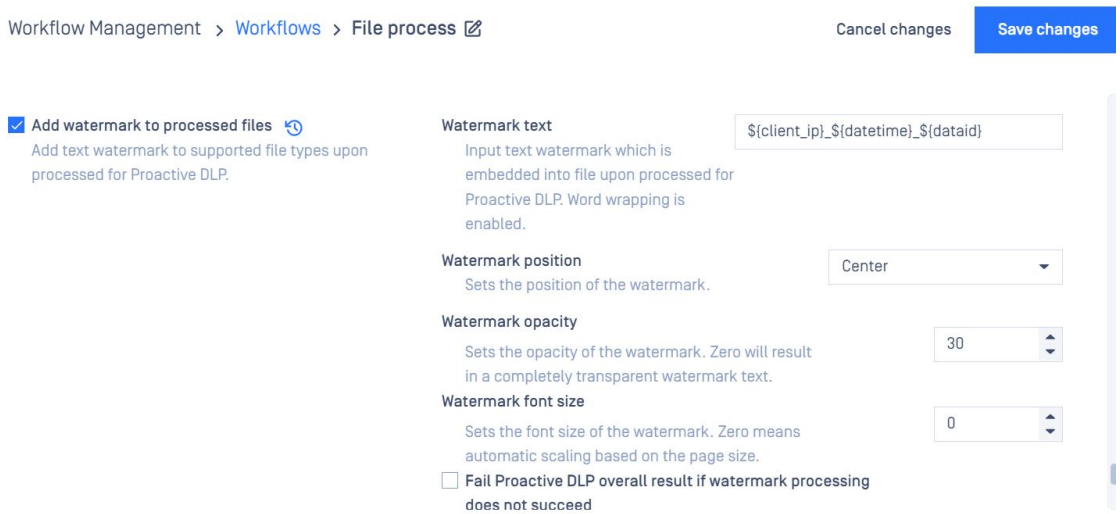
Nowadays, data leak becomes more and more popular, once the data went outside the organization, there is no way to track who caused the leak. Adding watermark to sensitive data is one of the solutions.

Supported File Types

- Images JPEG, TIFF, PNG, GIF, BMP, WMF, EMF, SVG
- Document: DOC, DOCX, RTF, PDF

To set watermark

- Policies > Workflow rules > "Workflow name" > Proactive DLP



Watermark configurations

Watermark text supports the following variables:

- ${user}: Username who submitted file (if any)
- ${client_ip}: Client IP address (version 4) where the file submission comes from
- ${filetime|epoch} : Unix/Epoch timestamp (e.g. 1561729083296) of file submission time
- ${filetime|full} : Full format timestamp of file submission time (28 06 2019 01:38:03 PM UTC)
- ${filetime|short} : Short format timestamp (e.g. 28/06/2019) of file submission time
- ${dataid} - this string identifies a file processing and can be used to retrieve results of processing
- ${original.basename} - name of the file to be processed, without file extension
- ${original.extension} - extension of the file to be processed
- ${converted.extension} - extension based on the target file type, for now, it is the same as ${original.extension}

From MetaDefender Core 4.16.3, you can define variables in metadata in the process file api and retrieve them for the "TEXT WATERMARK". For example: metadata: {"client_ip":"10.0.1.100","user_name":"john"} TEXT WATERMARK: ${ client_ip }_${ user_name}

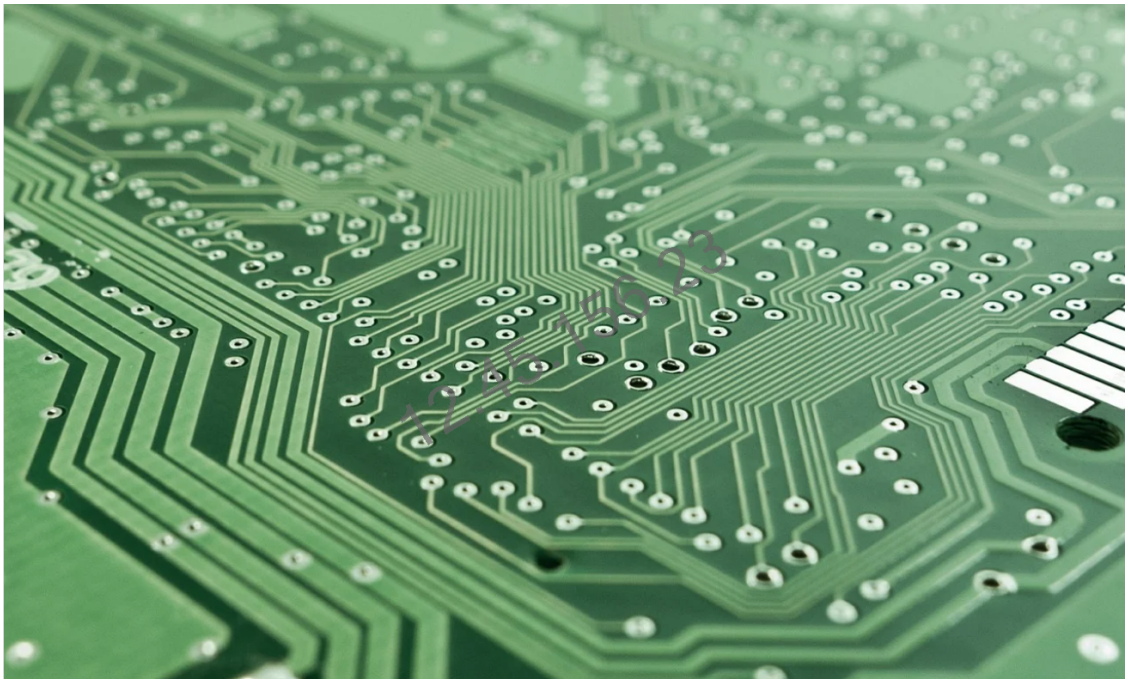
Watermark position

- Center: setting position to Center will use some rotation to make the text diagonal
- Top-Left
- Top-Right
- Bottom-Left
- Bottom-Right

Watermark opacity: the opacity of the watermark

Watermark font size: Setting font size to 0 means it will be automatically calculated

Sample output



Known issues/limitations:

- A very long Watermark text can overflow on a low resolution image in Linux version.
- Long Watermark text without whitespaces can overflow in pdf files on all supported platforms
- A non-English watermark (e.g., Hebrew, Japanese) requires additional fonts to be installed on your Linux system; otherwise, the watermark process may fail.

Watermark Text Issue

1. If the watermark text appears as '□ Instead of displaying the actual text, it may indicate that the required font is not installed on your system. In such cases, we suggest installing UNIFONT on your system to resolve the issue.
2. To fix the issue with non-English watermarks, you need to install fonts provided by Microsoft (msttcore-fonts). These fonts are free to install and use. You can follow the tutorial on the following page to learn how to install msttcore-fonts on a Linux system.

Remove metadata

Most of the files have some kind of metadata which could contain sensitive information like the author, GPS coordination, etc. which can expose the organizations to risk. Proactive DLP can remove metadata contained within the file.

Supported File Types

- Images: JPG, PNG, TIFF, GIF

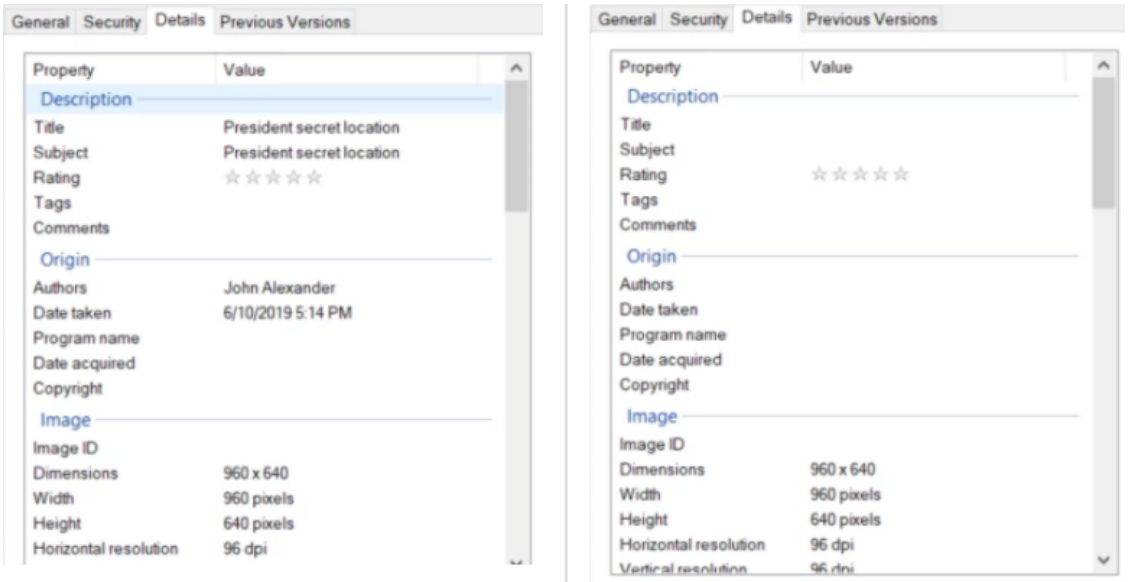
- Documents: PDF, MS Office, RTF
- OpenOffice: ODT, ODS

To set remove metadata

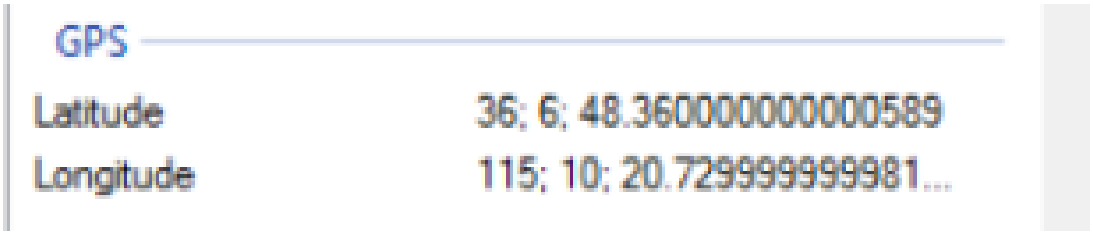
- Workflow Management > Workflows > "Workflow name" > Proactive DLP



Sample Exif info before and after processing



GPS info

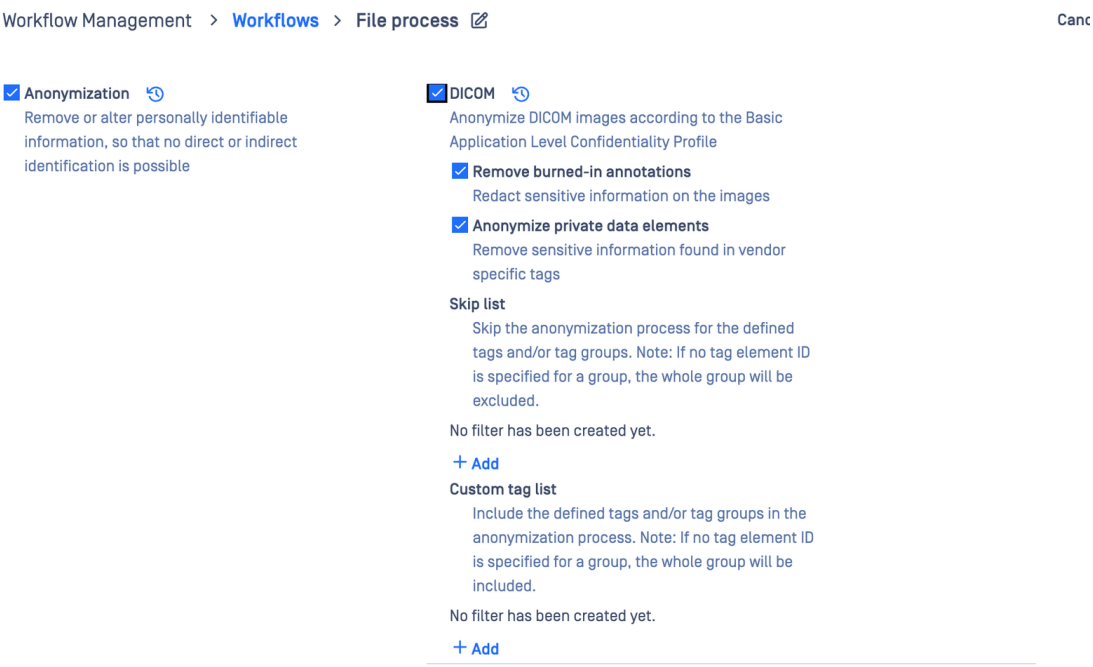


DICOM anonymization

It is now possible to easily remove personal identifying information from the headers of DICOM images as well as those data which are burned on the DICOM images by using the PDLP DICOM Anonymization feature. Among the information that can be removed from DICOM files are the data related to the patient, the data related to the examination, as well as the data that relates to the physicians who have been involved in the process of generating the DICOM files.

Enabling DICOM anonymization

Policies > Workflow rules > Workflow name > Proactive DLP > Anonymization >DICOM

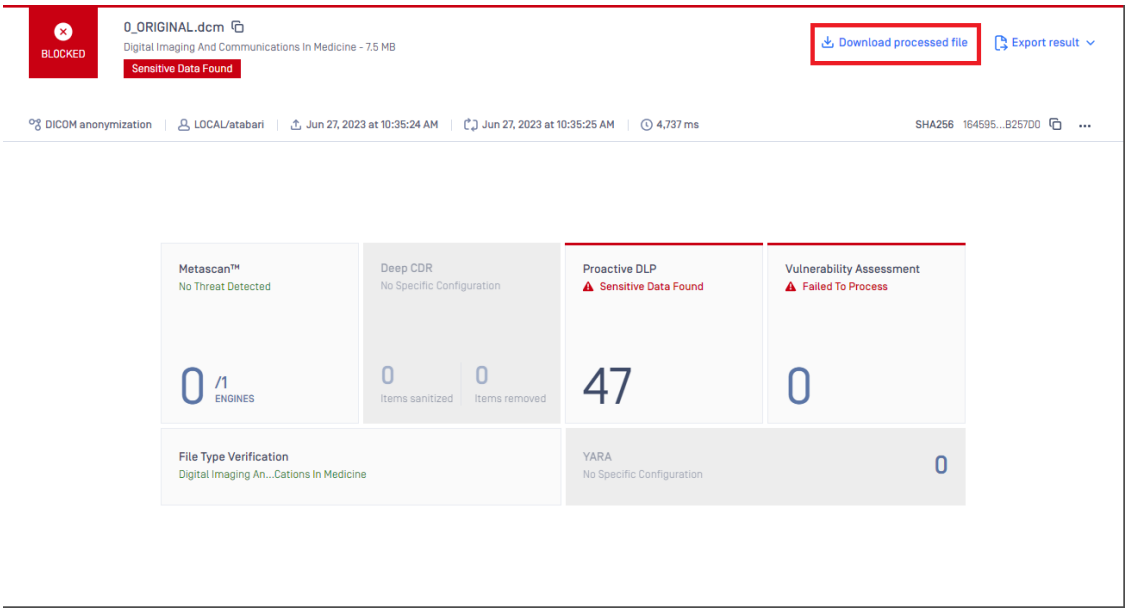


There are two options available when it comes to DICOM anonymization:

1. In order to anonymize only the DICOM header, you will have to uncheck the "Remove burned-in annotations" option when setting up the DICOM anonymization feature
2. To remove the burned-in PHI/PII from the DICOM images, you will need to make sure that the "Remove burnt-in annotations" box is checked.

3. Anonymize private data elements When enabled, the system will automatically remove vendor-specific private tags that may contain sensitive information. This ensures broader anonymization beyond standard DICOM fields.
4. Skip list: users can define specific tags or tag groups that should be excluded from the anonymization process.
5. Custom tag list: This option allows you to explicitly include certain tags or tag groups for anonymization, even if they are not part of the default DICOM confidentiality profile. Note: If no tag element is specified, selecting the group will include the entire group for anonymization.

Having set up the DICOM Anonymization, you can drop the DICOM file into your PDLP. Once the DICOM file has been dropped, PDLP will anonymize the file and you can download the anonymized version of the DICOM file at the end.



DICOM header metadata anonymization

To anonymize DICOM headers, we follow the DICOM standards so that we can be certain that all PII and PHI are anonymized according to the regulation's guidelines. In the following table you will find a list of the different actions we apply to different tag groups:

Groups	Actions
D_TAGS	Replace with a non-zero length value that may be a dummy value
Z_TAGS	Replace with a zero length value, or a non-zero length value that may be a dummy value
X_TAGS	Completely remove the tag
U_TAGS	Replace all UID's random ones. Same UID will have the same replaced value

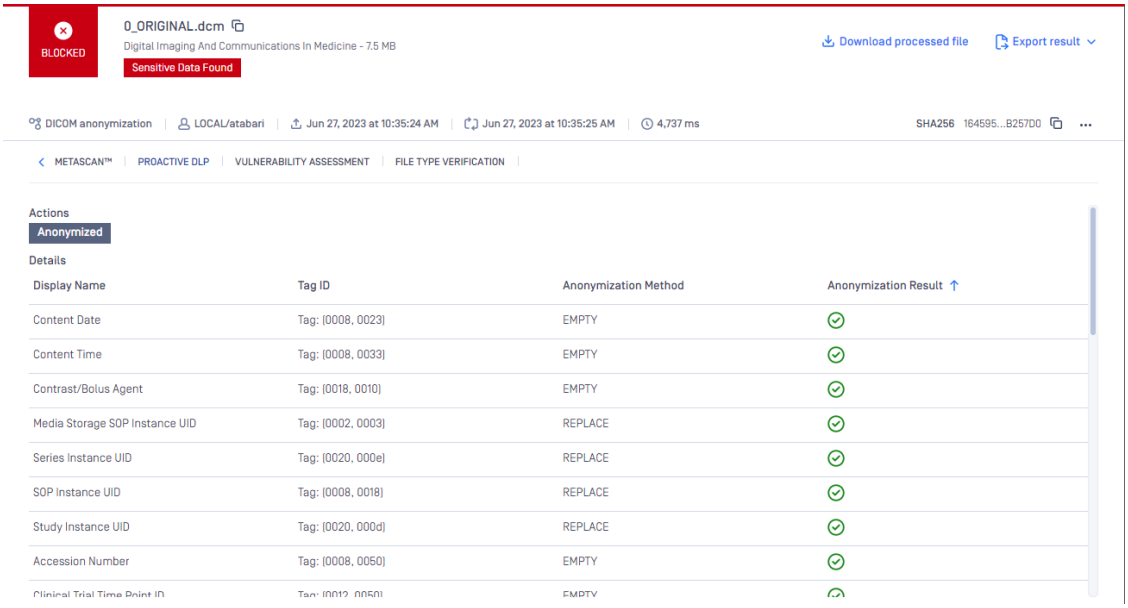
Groups	Actions
Z___D___TAGS	Replace with a non-zero length value that may be a dummy value
X___Z___TAGS	Replace with a zero length value, or a non-zero length value that may be a dummy value
X___D___TAGS	Replace with a non-zero length value that may be a dummy value
X_Z___D___TAGS	Replace with a non-zero length value that may be a dummy value
X_Z___U___STAR___TAGS	If it's a UID, then all numbers are randomly replaced. Else, replace with a zero length value, or a non-zero length value that may be a dummy value

Table reference:

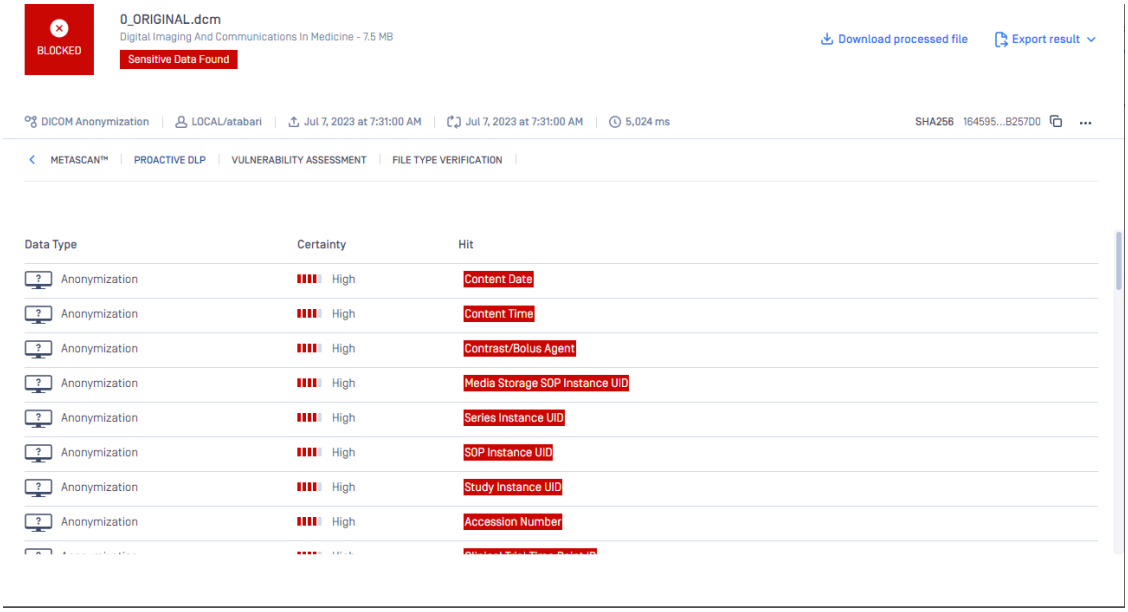
- https://dicom.nema.org/medical/dicom/current/output/chtml/part15/chapter_E.html

From MD core above 5.3.0 you can see what tags anonymized in the header of a DICOM file at the result pages. For MD core version less than 5.3.0 the DICOM Anonymization will perform the task and anonymize the file however we will not show anything in the result page.

MDCore > 5.3.0



MDCore < 5.3.0

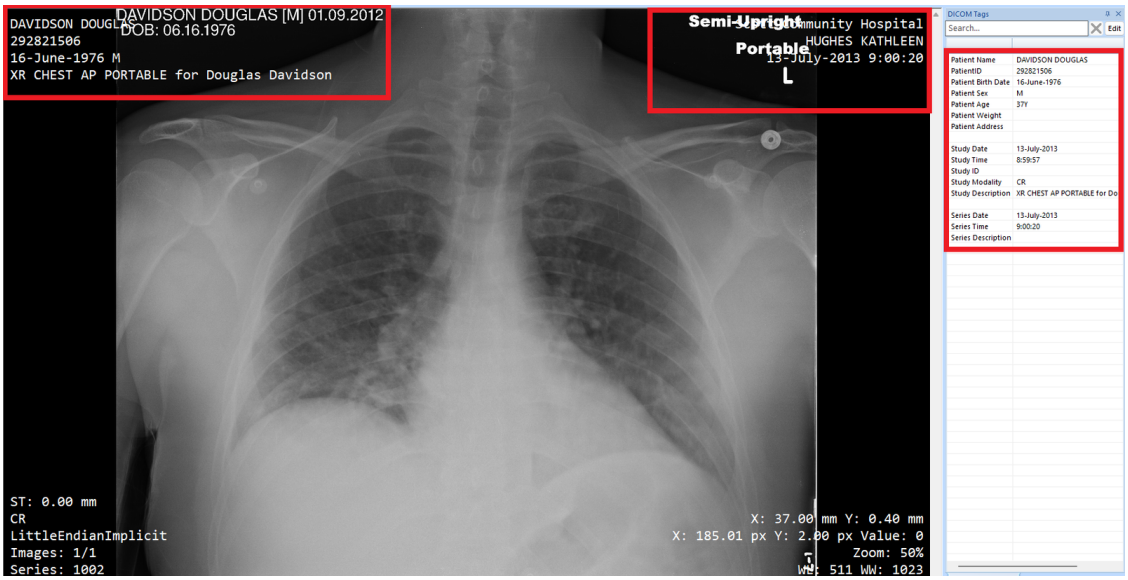


DICOM image anonymization

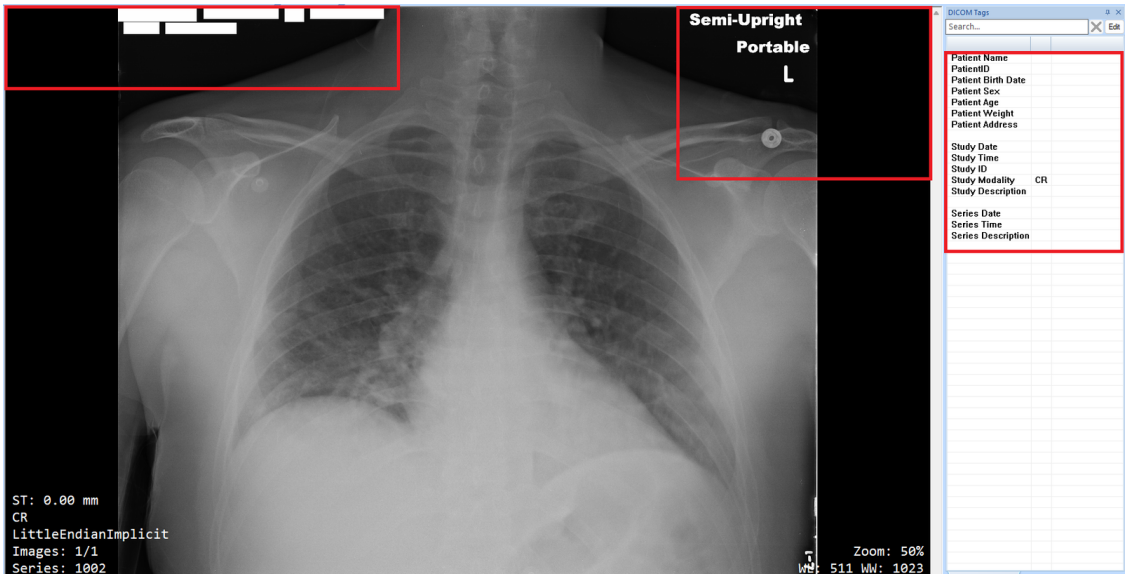
Additionally, we are also capable of redacting (removing) sensitive content from DICOM images. With the powerful AI-based identification engine that has been added to the OPSWAT PDLP, we have been able to identify and redact the PII and PHI that is burned onto the medical images.

> **Example**
>
> The information provided in these images is fake and it does not depict a real patient in any way.

Before



After



Supported image types

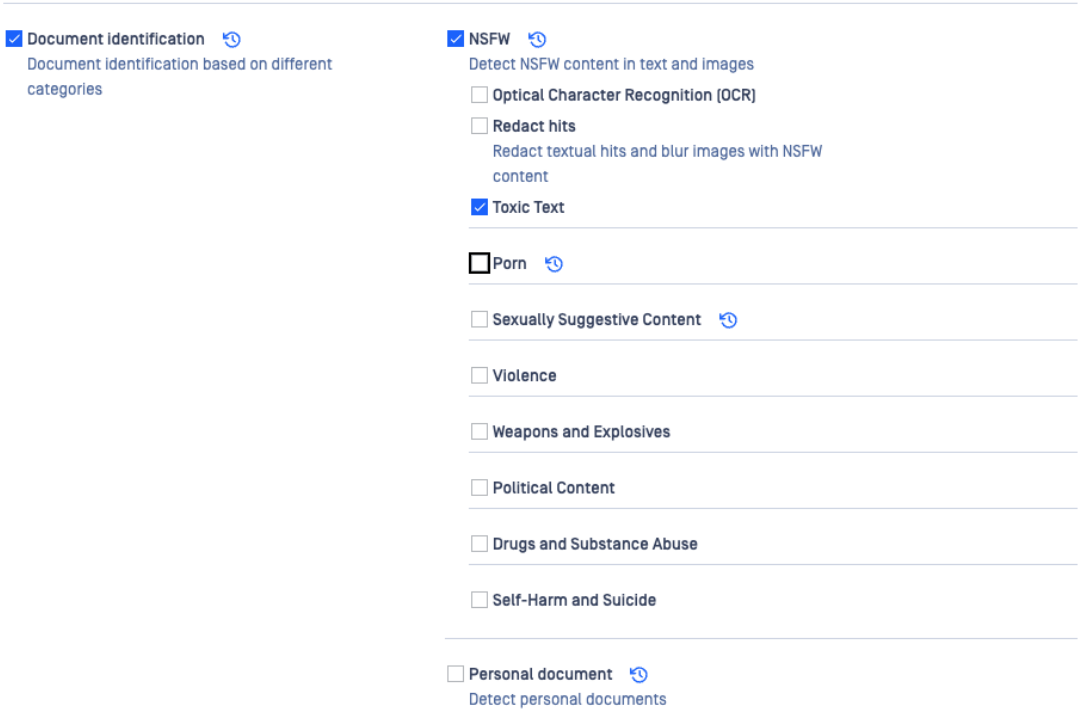
- Explicit VR Little Endian
- Implicit VR Little Endian
- Explicit VR Big Endian
- Deflated Explicit VR Little Endian
- RLE Lossless
- JPEG Baseline
- JPEG Extended
- JPEG2000 Lossless
- JPEG2000

Supported NLP entities

- **PERSON**: A full name consists of a first name, a middle name, or initials, and a last name
- **LOCATION**: Geographic or political location (cities, provinces, countries, international regions, bodies of water, mountains)
- **NRP**: Nationality, religion, or political affiliation of a person.
- **DATE_TIME**: Dates or periods smaller than a day, whether absolute or relative

This feature uses AI functionality to detect and redact PII (Personally Identifiable Information) and PHI (Protected Health Information) from DICOM file types. You have the option to enable or disable the AI-powered feature within the configuration settings. By default, this feature is disabled but can be activated or deactivated at any time based on user preference. If you enable the AI-powered feature, OPSWAT will not use your content to train or fine-tune its services. You should not rely on any results generated from AI-based functionality without verifying them.

Document Identification



I. NSFW (Not Safe for Work)

This section provides the controls and settings related to the detection and management of content that is categorized as NSFW within text and images.

Detection categories: Here is a list of categories OPSWAT PDLP can detect:

- **Toxic Text:** Detects toxic or harmful language in text content, helping prevent the transfer of offensive, abusive, or inappropriate communication.
- **Suggestive Content:** Images that are sexually suggestive but do not depict explicit sexual acts.
  - **Explicit Content**: Images depicting explicit sexual acts or pornography.
  - **Animated Adult Content**: Hentai and other animated images with pornographic themes.
- **Violence:** Detects imagery containing acts of aggression, physical harm, or graphic violence. This includes scenes depicting injury, assault, or excessive brutality.
- **Weapons and Explosives:** Identifies images containing firearms, bombs, or explosive materials. This detection helps mitigate the risk of circulating or exfiltrating content related to weaponry or potential threats.
- **Political Content:** Flags visual materials containing political symbols, campaigns, figures, or protest imagery. This ensures that communications remain compliant with internal neutrality and non-political content policies.
- **Drugs and Substance Abuse:** Detects the presence or depiction of illegal drugs, paraphernalia, or substance use. This category supports compliance with workplace conduct standards and helps prevent the distribution of

inappropriate imagery.

- **Self-Harm and Suicide:** Identifies visual content that depicts or suggests self-harm, suicidal behavior, or related paraphernalia. This detection enables early identification of distressing content and promotes responsible content handling.

Functionality: Enables or disables the detection of NSFW content in both textual and visual data.

- **Optical Character Recognition (OCR):** Scans and recognizes text within images. This helps in identifying NSFW text that might be embedded in graphics
  - **OCR Quality:** Determines the accuracy and efficiency of the OCR process
    - *Normal*: detect the information without pre-processing images
    - *Best*: pre-processing images before detecting the image to have a better detection rate, however, performance will be impacted

- **Redact Hits:** Automatically redacts (blurs & Blackout) the detected NSFW content
  - **Allow if detections at or above threshold are redacted**: This setting ensures that there is no interruption in the flow of content after making changes and redacting some portion of the content
    - *Redaction Threshold:* Defines the sensitivity level for redaction

Example

Here is an example of how the toxic text can work in action and detect the words which is not okay for work

```
Subject: Re: Project Submission

Dear Joe,

I have reviewed the project submission and I must say, the idea proposed seems
quite [redacted] I can't believe that this was the best concept you could come up with
after all the time spent in brainstorming sessions.

Moreover, I found the execution of the tasks to be done in a very lazy manner. It
appears that there was a lack of effort and diligence in completing the assignments
on time and according to the set standards.

Also, there were numerous mistakes throughout the project. It's baffling how such
[redacted] mistakes were overlooked. It’s crucial to pay attention to details to ensure
the accuracy and quality of the work delivered.

Lastly, sometimes the decisions made during the project phase were not well thought
out. It seemed as though an [redacted] was making these decisions without considering
the implications they might have on the overall project outcome.

I expect a thorough review and correction of the project. Please submit the revised
version by the end of this week.

Best Regards,

Jonathan Doe
Project Manager
jonathan.doe@example.com
```

This feature uses AI functionality to analyze both images and text to determine whether the content is toxic or not safe for work. You have the option to enable or disable the AI-powered feature within the configuration settings. By default, this feature is disabled but can be activated or deactivated at any time based on user preference. If you enable the AI-powered feature, OPSWAT will not use your content to train or fine-tune its services. You should not rely on any results generated from AI-based functionality without verifying them.

II. Personal Document Configuration

This section focuses on identifying documents, determining if uploaded items are personal documents, like passports or ID cards or Driver License. Based on the detection, customers can choose to block or allow the content.

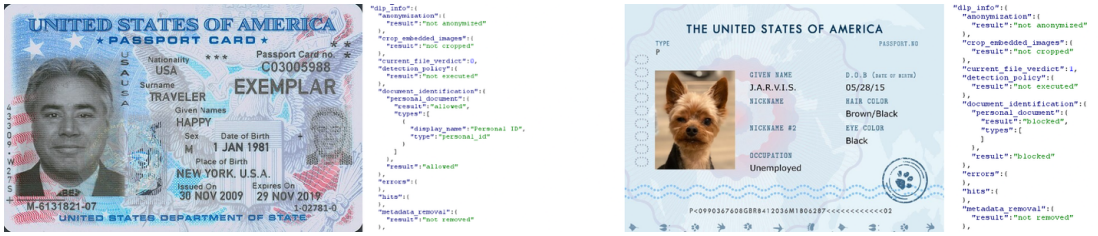
Functionality: Enables or disables the automatic detection of personal documents, such as IDs, driver license, and other sensitive ID cards.

- **Default Behaviour:** Defines the action taken when a personal document is detected
  - *Allow*: If a customer wishes to permit identified images, this setting can be activated, but only images verified as personal documents will be allowed.
  - *Block*: If a customer wants to restrict identified images, enabling this feature will ensure those images are blocked.

Below are the supported personal document identification categories:

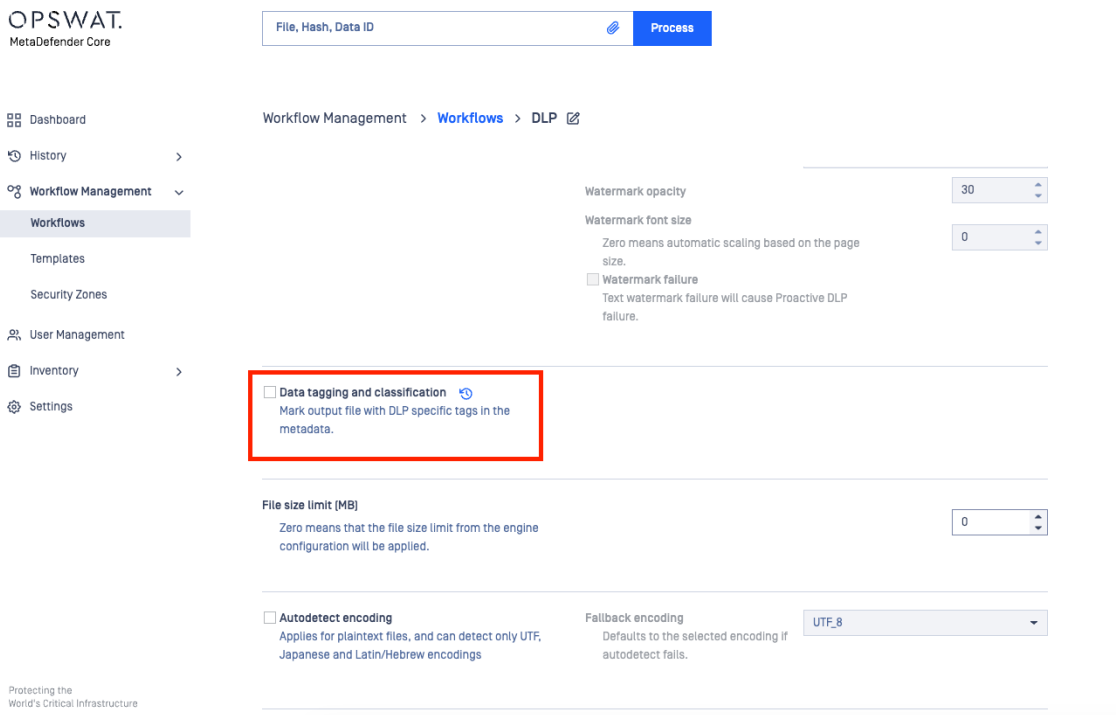
- **Identification Documents:** Detects documents containing personal identification details or official ID formats. This includes passports, driver's licenses, ID cards, social security cards, and similar records used to verify individual identity.
- **Financial Documents:** Identifies materials containing banking, investment, or financial account information. Includes bank statements, loan documents, tax forms, invoices, receipts, and payment confirmations.
- **Animal and Pet Documents:** Detects veterinary records, pet ownership certificates, vaccination cards, and registration documents related to animals.
- **Energy and Utility Documents:** Recognizes documents issued by energy, water, or telecommunications providers. Includes bills, statements, and contracts that often contain addresses, account numbers, and payment information.
- **Employment Documents:** Identifies HR-related documents such as employment contracts, offer letters, pay slips, performance evaluations, and resignation forms. These often include sensitive employee identifiers and confidential company data.
- **Government and Tax Documents:** Detects documents issued by government agencies or containing regulatory, tax, or filing information. Includes tax returns, government correspondence, permits, and legal filings.

Example



This feature uses AI functionality to differentiate between various documents and identify whether the document contains any government issued identification documentation of any person. You have the option to enable or disable the AI-powered feature within the configuration settings. By default, this feature is disabled but can be activated or deactivated at any time based on user preference. If you enable the AI-powered feature, OPSWAT will not use your content to train or fine-tune its services. You should not rely on any results generated from AI-based functionality without verifying them.

Data tagging and classification



OPSWAT Proactive DLP supports embedding metadata tags into PDF, Word, PowerPoint, and common image file formats. These tags serve as a mechanism for conveying detailed information about the results and outcomes of the PDLP content analysis. The metadata is attached directly to the processed and downloadable file, allowing downstream systems, such as document management platforms, SIEM solutions, and custom automation workflows, to automatically detect, interpret, and act on this information. By enabling consistent identification and traceability of sensitive content, this capability enhances end-to-end policy enforcement, auditing, and content governance across the organization.

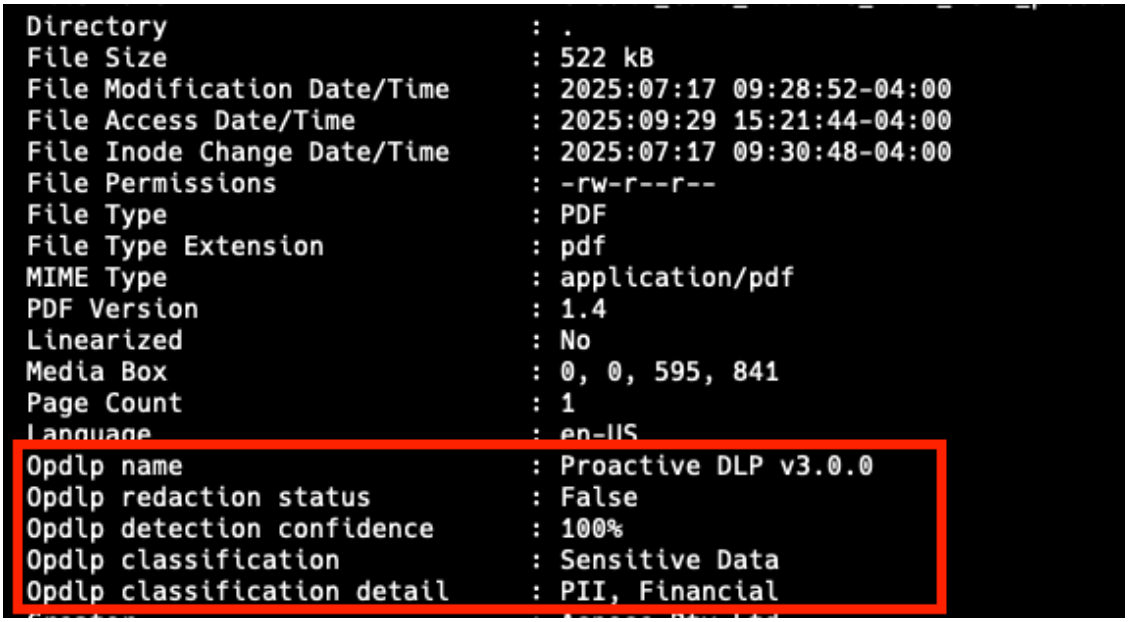
Below is the list of metadata tags currently supported by PDLP:

1. **OPSWAT-PDLP-CLASSIFICATION (opdlp classification):** This tag is designed to convey information about the file

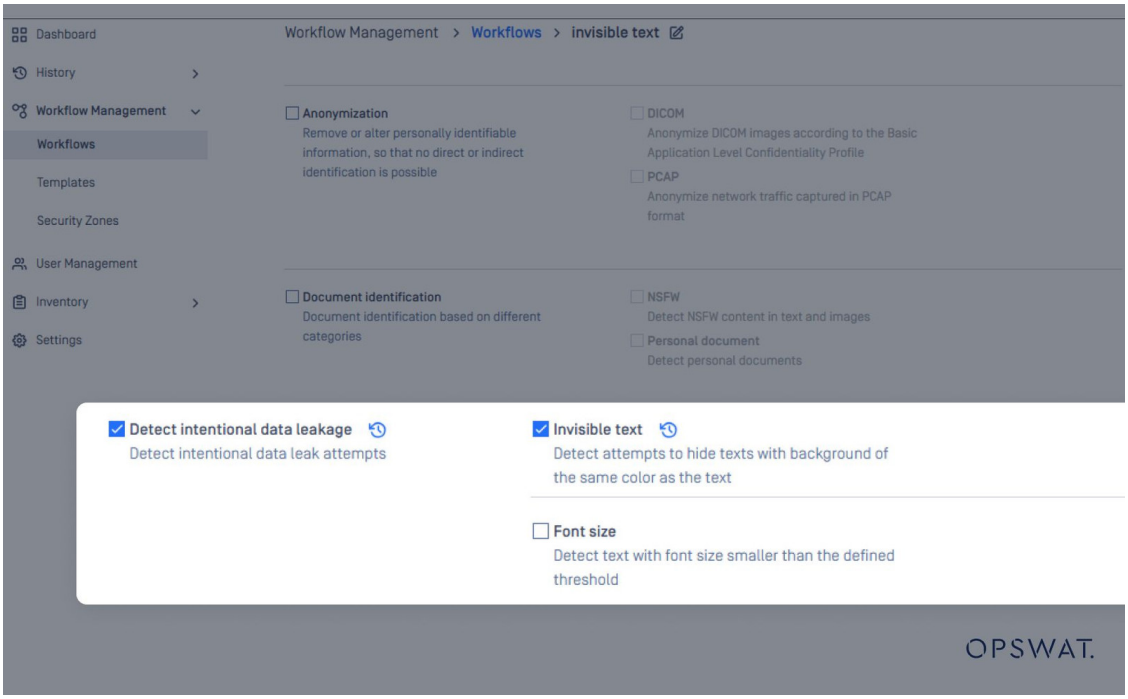
classification outcome by summarizing the reason a file or event triggered DLP controls. Examples include cases where the file contains sensitive data, the file or associated user poses potential risks, or a specific DLP policy has been violated. The tag also provides relevant business context to support efficient alert triage and remediation.

2. **OPSWAT-PDLP-CLASSIFICATION-DETAILS (opdlp classification details):** This tag is intended to add more specific details to those classification categories listed in *OPSWAT-PDLP-CLASSIFICATION*
3. **OPSWAT-PDLP-REDACTED (opdlp redacted):** This tag is intended to deliver information about the result of the redaction.
4. **OPSWAT-PDLP-CONFIDENCE-SCORE (opdlp confidence score):** This tag represents the PDLP confidence score assigned to the processed file. The DLP confidence score is a numerical value, typically ranging from 0 to 1 (or expressed as 0–100%), that quantifies the likelihood that a file contains sensitive information such as PII, PHI, PCI, or intellectual property. Higher scores indicate greater confidence that the file meets the criteria for a policy violation.

Example:



Detect intentional data leakage



Malicious insiders often employ subtle techniques to evade DLP controls and conceal sensitive information within files or documents. Two common evasion methods include **invisible text** and **small font manipulation**, both of which are now detectable through OPSWAT Proactive DLP's advanced content inspection.

- **Invisible Text Detection:** Detects hidden or white-colored text embedded within documents or images that is not visible to human reviewers but can still be processed by digital systems. This technique is often used to conceal confidential data, trigger automated workflows, or bypass keyword-based inspection mechanisms.
- **Small Font Detection:** Identifies attempts to hide sensitive data by using extremely small font sizes that make the content unreadable to the human eye but still machine-detectable. This approach can be used to disguise regulated information such as PII, credentials, or source code snippets within otherwise benign files.

By identifying and flagging hidden content manipulation techniques, OPSWAT Proactive DLP enhances detection accuracy and mitigates the risk of data exfiltration resulting from concealed or obfuscated information within files.

Example:

Summary of Achievements

- Customer satisfaction increased by 15% this quarter, with overwhelmingly positive feedback on support services.
- Delivery timelines improved by 10% compared to Q3, enhancing customer retention.
- Secured two new enterprise clients in the technology sector, contributing to long-term growth.

New Initiatives

- Launching employee engagement programs in Q1 to strengthen workplace culture and productivity.
- Expanding operations in the Southeast region, targeting high-growth markets.
- Exploring AI-driven solutions to enhance efficiency and reduce operational costs.

Team Highlights

- HR onboarded 15 new team members across multiple departments.
- The marketing team launched a successful campaign, generating a 20% increase in web traffic.
- R&D achieved a key milestone in product development, with a beta version ready for testing.

Upcoming Challenges

- Supply chain optimization remains a focus area for Q1.
- Addressing new compliance regulations with enhanced internal policies.
- Ensuring seamless integration of new technologies into existing workflows.



Company X External Quarterly Report

Date: December 20, 2024

Prepared by: John Doe, CEO

Summary of Achievements

- Customer satisfaction increased by 15% this quarter, with overwhelmingly positive feedback on support services.
- Delivery timelines improved by 10% compared to Q3, enhancing customer retention.
- Secured two new enterprise clients in the technology sector, contributing to long-term growth.

New Initiatives

- Launching employee engagement programs in Q1 to strengthen workplace culture and productivity.
- Expanding operations in the Southeast region, targeting high-growth markets.
- Exploring AI-driven solutions to enhance efficiency and reduce operational costs.

Team Highlights

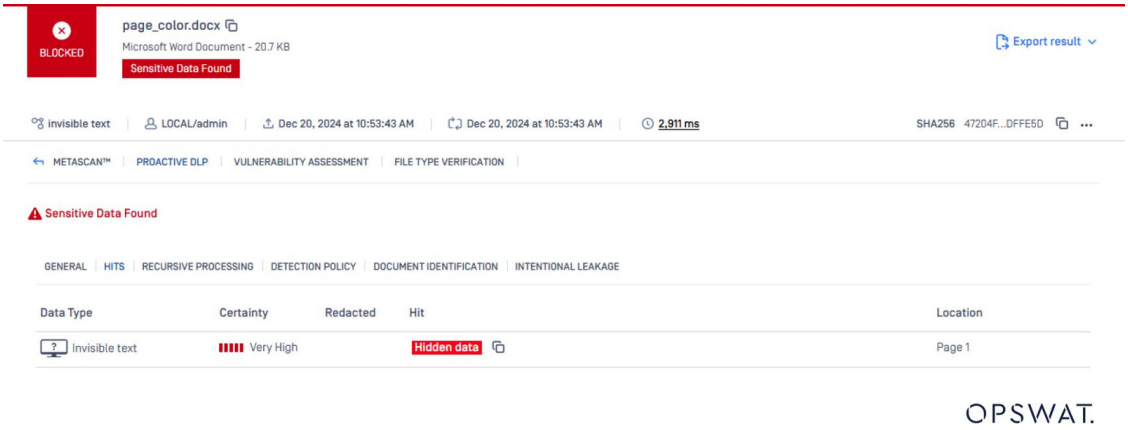
- HR onboarded 15 new team members across multiple departments.
- The marketing team launched a successful campaign, generating a 20% increase in web traffic.
- R&D achieved a key milestone in product development, with a beta version ready for testing.

Upcoming Challenges

- Supply chain optimization remains a focus area for Q1.
- Addressing new compliance regulations with enhanced internal policies.
- Ensuring seamless integration of new technologies into existing workflows.

Confidential: SSN: 123-45-6789 | Project Code: AlphaX2025

Executive Bonuses (Confidential): $75,000 each



Performance (Throughput)

Disclaimer

We do not guarantee the same performance in your environment. Performance can vary significantly

depending on data sets and systems used when running the tests. The sole purpose of this section of the User Guide is to provide a high-level indicator of performance impact when enabling Proactive DLP in your business logic.

Windows System Info

- RAM: 32GB
- CPU: Intel® Core™ i7-6700 CPU @ 3.40GHz × 8
- OS: Windows Server 2019
- Disk Drive: 256GB SSD

Linux System Info

- RAM: 32GB
- CPU: Intel® Core™ i7-4790 CPU @ 3.60GHz × 8
- OS: Ubuntu 20.04.1 LTS
- Disk Drive: 256GB SSD

Resources

- Windows: MetaDefender Core v4.19.0 with 8 engines
- Linux: MetaDefender Core v4.19.0 with 5 engines

Test Results

File type	Average file size (KB)	Total files	Sensitive files	Detection (s/file)		Detect + Redaction (s/file)	
				Windows	Linux	Windows	Linux
Text	170	1456	728	0.04	0.05	0.04	0.05
Text	3174	864	592	0.25	0.21	0.25	0.22
Pdf	392	1572	785	0.28	0.28	0.44	0.47
Pdf	6553	846	242	2.5	2.72	3.0	4.9

File type	Average file size (KB)	Total files	Sensitive files	Detection (s/file)		Detect + Redaction (s/file)	
Word	224	629	314	0.14	0.16	0.15	0.16
Word	2969	808	128	0.26	0.26	0.36	0.35
Excel	191	2942	1471	0.51	0.43	0.52	0.46
Excel	1904	840	192	3.3	2.8	3.3	2.7

Overview

Warning

As the sandbox product has undertaken a variety of product name changes ("OPSWAT Filescan", "OPSWAT Filescan Sandbox" and now "MetaDefender Sandbox", you may find artefacts in the product or slightly outdated screenshots in the documentation). Please note that "**MetaDefender Aether for Core**" and " **Adaptive Sandbox**" are terms both referring to the same product.

What is MetaDefender Aether for Core (Adaptive Sandbox)?

The MetaDefender Aether for Core in MetaDefender Core is a next-gen sandbox and threat intelligence solution that utilizes Adaptive Threat Analysis (ATA) of the OPSWAT Filescan platform. It binds together dozens of state-of-the-art tools, services, and proprietary algorithms with the focus on zero-day malware detection and Indicators of Compromise (IOCs) extraction from files, documents, scripts at speed and scale. Using a technology called Adaptive Threat Analysis (ATA), OPSWAT Sandbox is a solution that is able to provide actionable intelligence in many more cases. Combined with its unmatched speed, it becomes possible to significantly reduce the number of artefacts needing a manual analysis.

For information of the standalone Filescan product, visit our product documentation site.

Sandbox Engines in MetaDefender Core

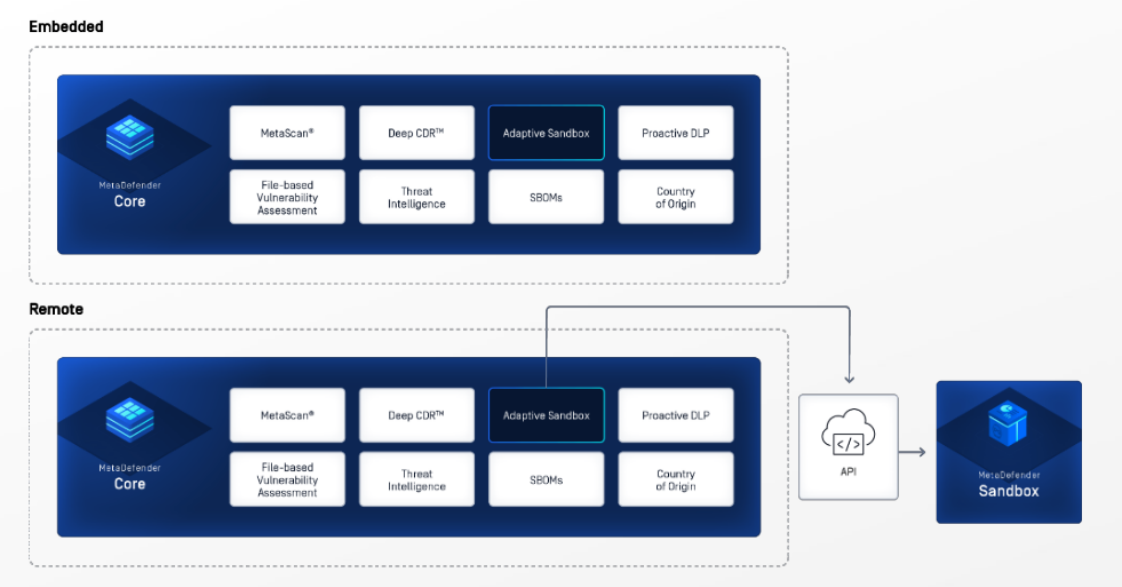
For using Filescan functionality through MD Core, there is 2 types of engines to choose from:

- **Embedded** - this contains a bundled Adaptive Threat Analysis (ATA) engine, and is an optimal choice for off-line environments
- **Remote** - this utilizes a deployed standalone instance of an MetaDefender Sandbox platform instance via the API (on premise or in the cloud). This is the recommended engine for most cases.

Info

A MetaDefender Core license can contain only one version of the Sandbox Engine.

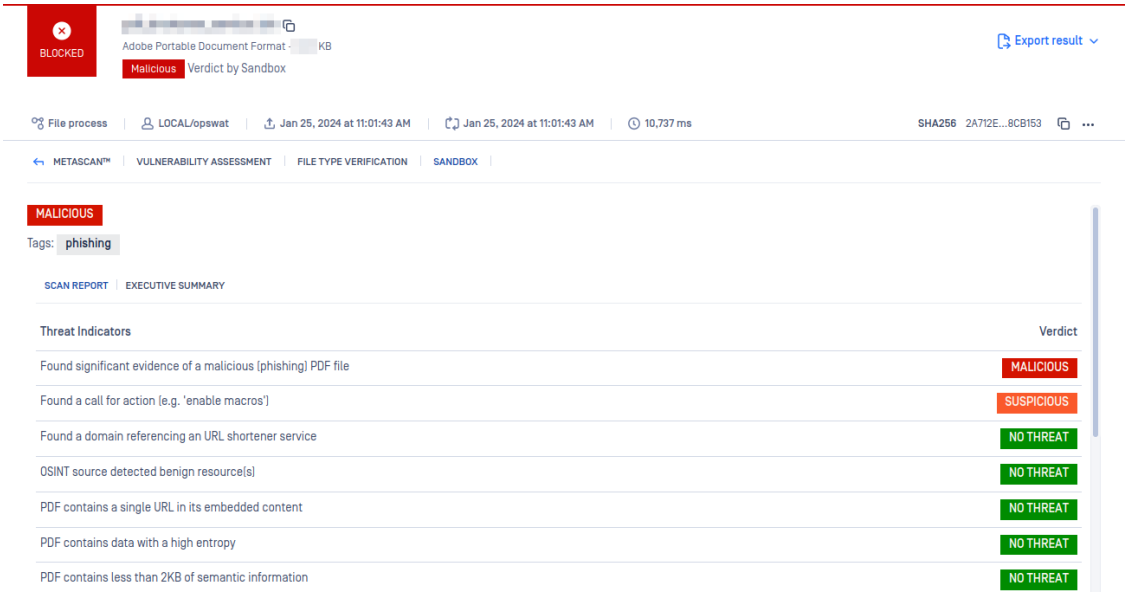
The following diagram illustrates the different engine setups:



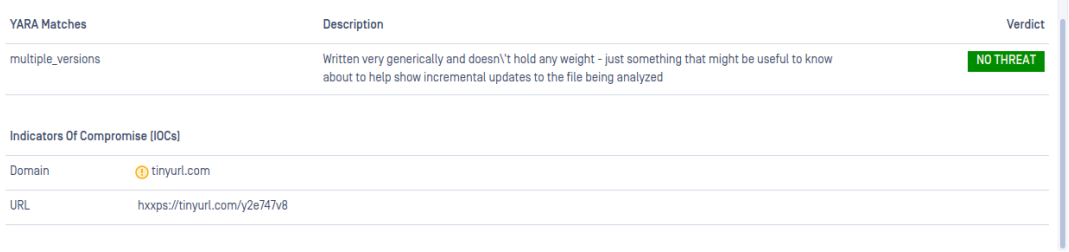
More details on the exact feature sets can be found here.

Example Sandbox Engine results

Verdict and Threat Indicators



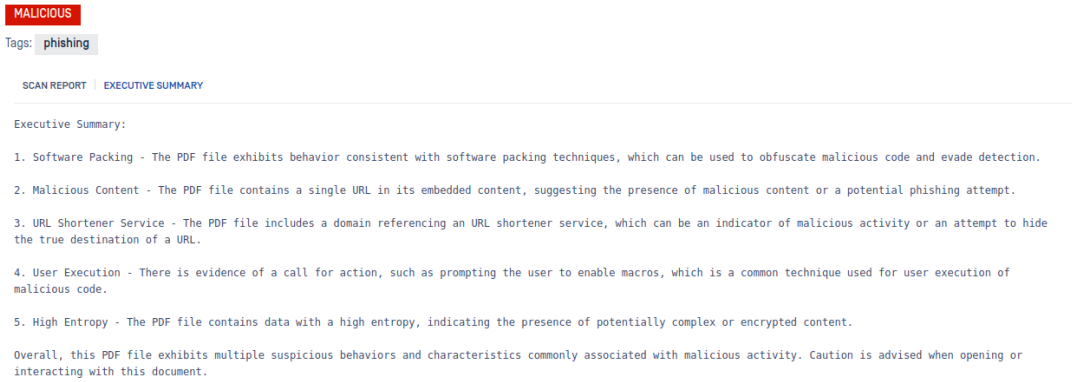
YARA matches and detected IOCs



Executive Summary

Info

Only available for the Embedded Sandbox Engine


MALICIOUS

Tags: phishing

SCAN REPORT | EXECUTIVE SUMMARY

Executive Summary:

1. Software Packing - The PDF file exhibits behavior consistent with software packing techniques, which can be used to obfuscate malicious code and evade detection.

2. Malicious Content - The PDF file contains a single URL in its embedded content, suggesting the presence of malicious content or a potential phishing attempt.

3. URL Shortener Service - The PDF file includes a domain referencing an URL shortener service, which can be an indicator of malicious activity or an attempt to hide the true destination of a URL.

4. User Execution - There is evidence of a call for action, such as prompting the user to enable macros, which is a common technique used for user execution of malicious code.

5. High Entropy - The PDF file contains data with a high entropy, indicating the presence of potentially complex or encrypted content.

Overall, this PDF file exhibits multiple suspicious behaviors and characteristics commonly associated with malicious activity. Caution is advised when opening or interacting with this document.

Supported File Types

File types

The Adaptive Sandbox engine supports a wide variety of file types, organized by category below:

Application types

File Type	OS	Comment
A3X	Windows	AutoIt v3 Script
AU3	Windows	AutoIt v3 Script

Office Documents

File type	OS	Comment
DOCX	Windows	Microsoft Word Document
DOCM	Windows	Microsoft Word Macro-Enabled Document
DOTX	Windows	Microsoft Word Template

File type	OS	Comment
DOTM	Windows	Microsoft Word Macro-Enabled Template
DOC	Windows	Microsoft Word Document
DOT	Windows	Microsoft Word Template
XLSX	Windows	Microsoft Excel Workbook
XLSM	Windows	Microsoft Excel Macro-Enabled Workbook
XLS	Windows	Microsoft Excel Workbook
XLTX	Windows	Microsoft Excel Template
XLTM	Windows	Microsoft Excel Macro-Enabled Template
PPTX	Windows	Microsoft PowerPoint Presentation
PPTM	Windows	Microsoft PowerPoint Macro-Enabled Presentation
PPSX	Windows	Microsoft PowerPoint Slide Show
PPT	Windows	Microsoft PowerPoint Presentation
PPAM	Windows	Microsoft PowerPoint Add-in
POTX	Windows	Microsoft PowerPoint Template
POTM	Windows	Microsoft PowerPoint Macro-Enabled Template
POT	Windows	Microsoft PowerPoint Template
ODT	Windows	OpenDocument Text

File type	OS	Comment
ODS	Windows	OpenDocument Spreadsheet
RTF	Windows	Rich Text Format
HWP	Windows	Hangul Word Processor
PUB	Windows	Microsoft Publisher
ONE	Windows	Microsoft OneNote exported section

Executable Files

File type	OS	Comment
PE (EXE/DLL)	Windows	PE unpacking & re-analysis
PE (dotnet_pe)	Windows	PE.NET decompilation & re-analysis
PE (other)	Windows	Other specific PE types (SFX, Golang, Rust, etc.)
APK	Android	Android Application Package
CPL	Windows	Control Panel Extension
ELF	Unix	Executable and Linkable Format
LNK	Windows	Windows Shortcut
MSI	Windows	Windows Installer Package

Image Files

File type	OS	Comment
DWG	Windows	VBA macro extraction

File type	OS	Comment
SVG	Windows	Scalable Vector Graphics

Media Files

File type	OS	Comment
ASF	Windows	Windows Media Video
MP3	Multi-OS	Digital Audio

Other

File type	OS	Comment
Java	Multi-OS	Java Decompilation & re-analysis
MSC	Windows	Microsoft Management Console
OLE	Windows	Object Linking and Embedding
SCT	Windows	Windows Scriptlet

Adobe Files

File type	OS	Comment
PDF	Multi-OS	Portable Document Format

Text Files

File type	OS	Comment
HTML	Windows	Hypertext Markup Language
TXT	Windows	Plain Text

File type	OS	Comment
HTA	Windows	HTML Application
WSF	Windows	Windows Script File
BAT	Windows	Batch Script
JScript	Windows	Microsoft's JavaScript
JSE	Windows	JScript Encoded
Powershell	Windows	PowerShell Script
VBScript	Windows	Visual Basic Script

Email Files

File type	OS	Comment
EML	Windows	Electronic Mail
MBOX	Windows	Mailbox Format
RFC822	Windows	RFC822 Email Format

Archive types

The Adaptive Sandbox engine analyzes various archive file types, with certain types being processed through the Archive Engine, as listed below:

Archive Type	Type-specific analysis	Comment
7z	N/A	Extracted by Archive engine
ACE	N/A	Extracted by Archive engine
BZIP2/BZ2	N/A	Extracted by Archive engine
CAB		Extracted by Archive engine

Archive Type	Type-specific analysis	Comment
CHM	YES	
DEB	NO	Extracted by Archive engine
ISO	NO	Extracted by Archive engine
PKG	NO	
RAR	N/A	Extracted by Archive engine
SFX (PEEXE)	YES	
TBZ2	N/A	Extracted by Archive engine
TAR	N/A	Extracted by Archive engine
TGZ	N/A	Extracted by Archive engine
VHD	N/A	Extracted by Archive engine
XPI	NO	Extracted by Archive engine
ZIP	N/A	Extracted by Archive engine

System requirements

Dependencies

Embedded Engine	Remote Engine
See: How to upgrade to Java 17 **Linux:** openjdk-17-jre **Windows:** Amazon Corretto 17	No additional dependencies are required.

Note

Before version 2.2.0 Visual C++ Redistributable for Visual Studio is required for Windows.

Warning

The required Java version for the embedded Aether(adaptive sandbox) is Java 17. While Java 8 or 11 may still function temporarily, they will no longer work at all after March 2025. Please plan to upgrade to Java 17 before March 2025 to avoid any disruption

The default system Java version should be set correctly; please check if the correct version is used. The computer should be restarted after Java installation.

Check Java version · bash

```
java -version
openjdk version "17.0.12" 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu222.04, mixed mode, sharing)
```

System requirements

Embedded Sandbox Engine

System requirements depend on the number and type of submitted files, as well as their content and size. The performance also varies based on the configuration of the system and engine workflow. Below, you will find benchmark measurements using different CPU, RAM, and Sandbox parallel processing configurations on various sample sets. These measurements serve as a guideline to help estimate the required resources for your use case.

Warning

These are additional requirements on top of the MetaDefender Core recommended system configuration

Note

These measurements are done with default engine workflow settings on Ubuntu Linux

Example Configurations

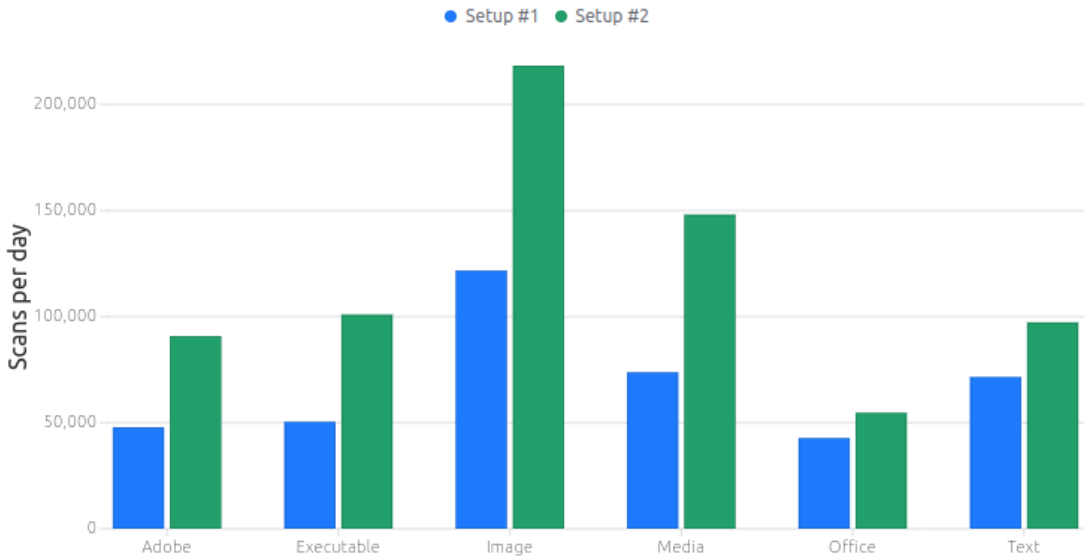
	CPU Cores	RAM	Parallel Count
Config #1	4	8	5 (default)
Config #2	8	16	10

Measured daily throughput

Warning

Throughput can be limited by license restrictions! See Licensing

File type	Avg size (MB)	Throughput #1	Throughput #2	#1 (MB/s)	#2 (MB/s)
Adobe	0.93	47856	90816	0.52	0.98
Executable	0.16	50424	101040	0.09	0.19
Image	1.6	121728	218328	2.25	4.04
Media	0.13	73776	148176	0.11	0.22
Office	1.6	42744	54720	0.79	1.01
Text	8.8	71520	97344	7.28	9.91



Remote Aether (Adaptive Sandbox) Engine

For MetaDefender Aether (Standalone) system requirements please refer to our Technical Datasheet and Performance Measurement page.

Upgrade to Java 17

Warning

Support for Java 8 and Java 11 for the embedded Aether (Adaptive Sandbox) engine ends in March 2025.

Update the Java version on your system to continue receiving engine updates after this date.

Upgrade on Windows

Install the recommended Java version with the MSI installer which is documented on our system requirements page and restart your machine.

Upgrade on Linux

Install the recommended java package.

Example for Ubuntu 22.04 · bash

```
sudo apt install openjdk-17-jre
```

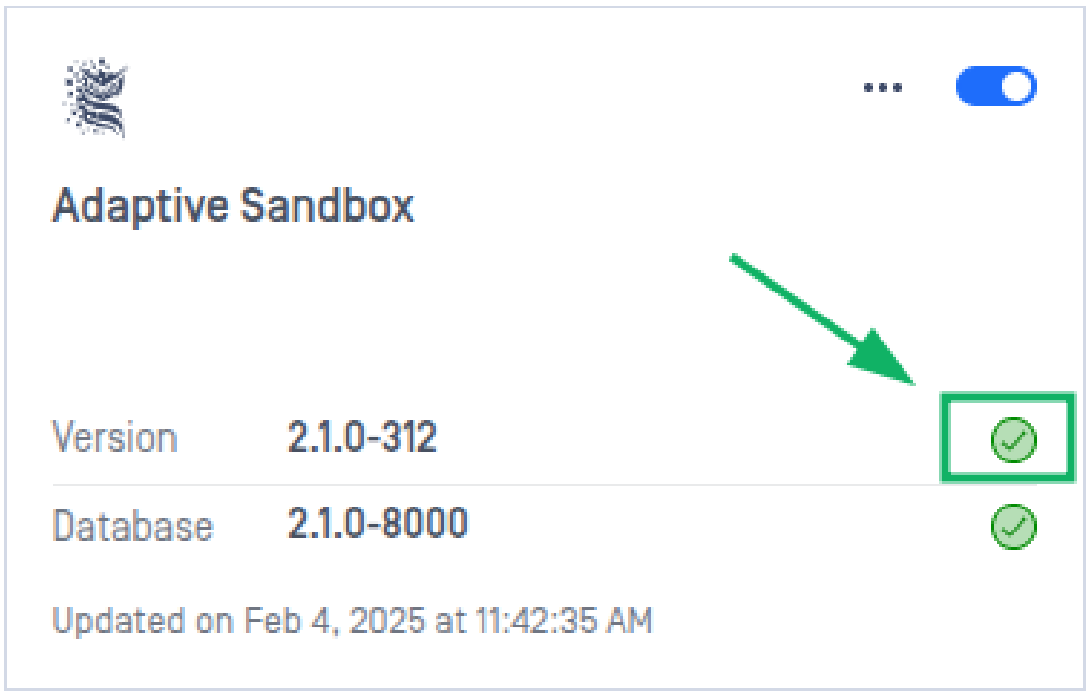
Verify the Java version

Make sure that the Java version is correctly updated:

Command · bash

```
java -version
openjdk version "17.0.13" 2024-10-15 LTS
OpenJDK Runtime Environment Corretto-17.0.13.11.1 (build 17.0.13+11-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.13.11.1 (build 17.0.13+11-LTS, mixed mode, sharing)
```

After installing the correct Java version, the engine should start without errors and warnings.



Installation

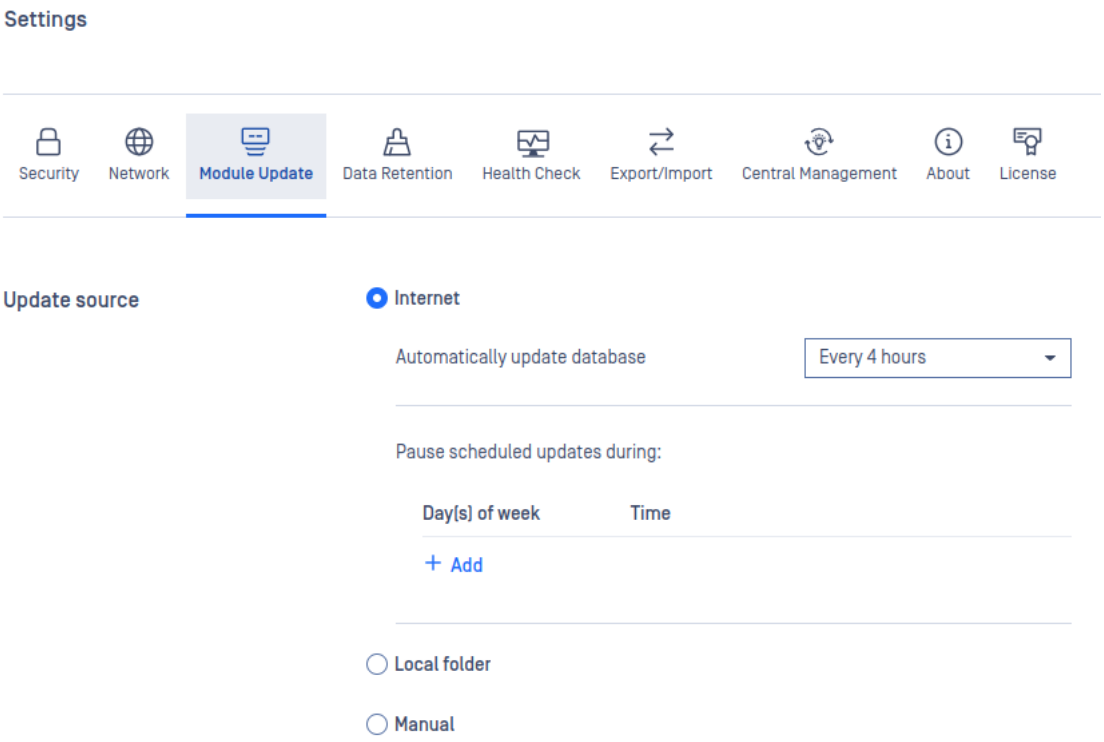
Pro Tip

The MetaDefender Aether (Adaptive Sandbox) Engine supports engine updates in either online or offline environments

Engine updates

Go to **Settings > Module Update**

MetaDefender Core will automatically download all the Engines available in your license from our online update infrastructure, and download new Engine versions automatically based on your **Module Update** configuration.



Offline Engine updates

Please see the links below for more details.

How do I deploy MetaDefender Core to an offline Windows environment

How do I deploy MetaDefender Core to an offline Linux environment

Offline MetaDefender Aether environment

You might want to use your Remote Sandbox Engine with a standalone MetaDefender Aether instance running in an offline environment. For more details on how to use your MetaDefender Aether instance offline, please see the link below.

How do I use MetaDefender Aether in an offline environment

Licensing

> **Note**
>
> This page applies to the **Embedded Aether (Adaptive Sandbox)** engine version **2.2.0** and above.

License Usage and Execution Limits

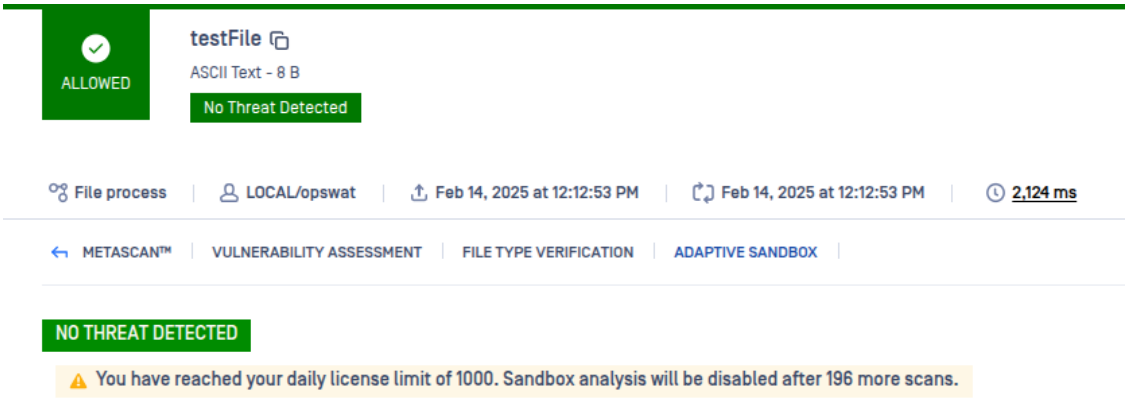
Your MetaDefender Core license includes a **daily execution limit** for the **Embedded Aether (Adaptive Sandbox)** engine, which determines the maximum number of scans that can be performed within a single day. The execution count resets at midnight (UTC).

Warning

Restart the engine after activating a new license for the changes to take effect. Current scan count will not reset on license actions.

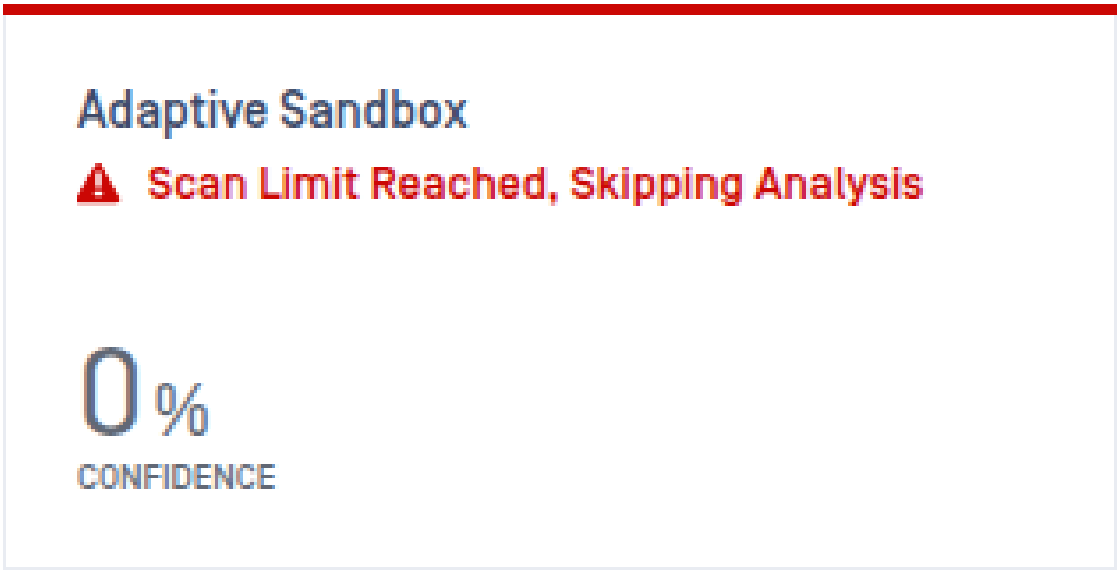
Soft Limit (Warning)

Your license allows exceeding the limit by 20%. During this period, known as the **soft limit**, the system will continue to allow scans but will display a warning indicating the remaining available executions.



Hard Limit (Enforcement)

If the number of executed scans exceeds 120% of the daily limit, the **hard limit** is enforced, and further executions will be blocked until the next day. A license error will be returned, preventing additional scans.



Overuse Adjustment

If the soft limit is reached for 7 consecutive days, the soft limit will become the hard limit, effectively reducing the maximum number of scans allowed per day. The original soft limit is reverted the next day if the scan limit is not reached.

Example Scenarios:

License limit: 100 scans/day

Soft limit: 101-120 scans Warning displayed, but scans continue.

Hard limit: 121+ scans Execution blocked until midnight.

If soft limit is reached 7 consecutive days New hard limit set to 100 scans/day, and the 20% buffer is no longer available. Warning zone starts with scan number 81.

Ensure that your scanning usage stays within acceptable limits to avoid disruptions. If you require more executions, consider upgrading your license.

Configuration

The MetaDefender Aether for Core (Adaptive Sandbox) Engine offers two integration modes:

1. **Inline:** working as a part of MetaDefender Core processing workflow (real-time processing). Allowing users to block entire processing based on MetaDefender Aether for Core engine outcome and decision.
2. **Out of band:** working as a part of MetaDefender Core quarantine (post-investigation processing). Providing additional option to analyze quarantined items along with the existing Threat Intelligence technology.

Global configuration

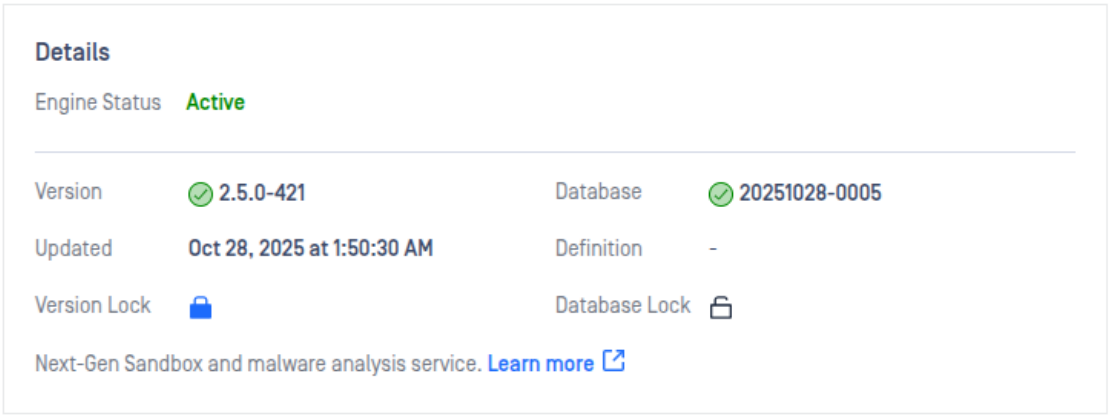
Go to **Inventory > Modules** and select the **MetaDefender Aether for Core (Adaptive Sandbox)** module

> **Pro Tip**
>
> Use your OPSWAT Reputation secret to further enhance your MetaDefender Aether for Core (Adaptive Sandbox) Engine capabilities.

The global configuration is used by the out of band MetaDefender Core quarantine process and also as the default setting for new workflow configurations.

Engine details

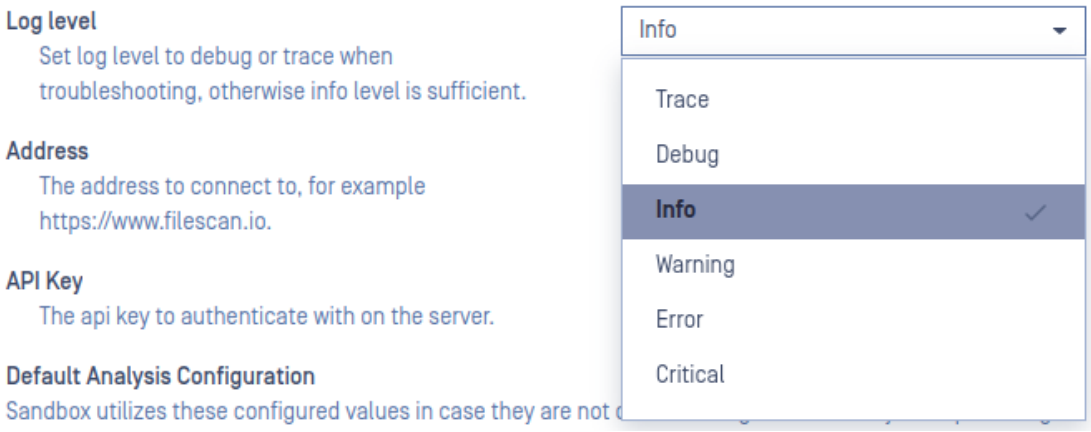
See details like engine version and status.



Configuration

Log level

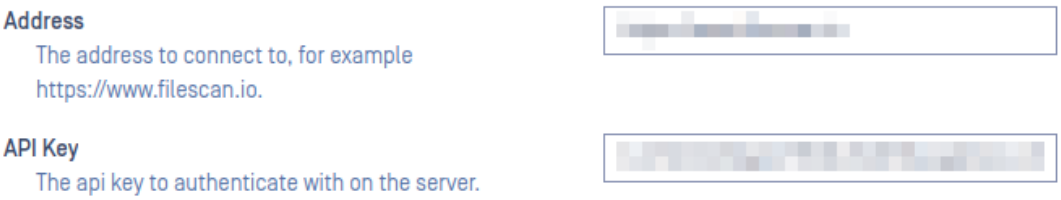
Configure the log level of the Sandbox Engine. In case of issues a lower log level (**Debug**) might help the support team.



Remote server address

Configure the address and secret of your standalone MetaDefender Sandbox instance.

> **Info**
>
> Only available for the Remote Sandbox Engine



OPSWAT Reputation configuration

Configure your **OPSWAT Reputation** secret to use the OPSWAT Reputation OSINT scan task.

> **Info**
>
> Only available for the Embedded Sandbox Engine. For the Remote Sandbox Engine the secret must be configured for the standalone MetaDefender Sandbox instance.



OpenAI configuration

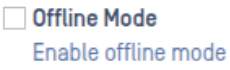
Configure your OpenAI secret and preferred ChatGPT model for the **Executive Summary** feature. For more details see the workflow configuration section below on this page.

> **Info**
>
> Only available for the Embedded Sandbox Engine

Offline Mode

Disable online services.

> **Info**
>
> Only available for the Embedded Sandbox Engine



Default Analysis configuration

The global configuration stores the default configuration settings for default workflow and quarantine processing. For more details see the workflow configuration section below on this page.

Workflow configuration

Go to **Workflow Management > Workflows**, select your workflow and select the **MetaDefender Aether for Core (Adaptive Sandbox)** module

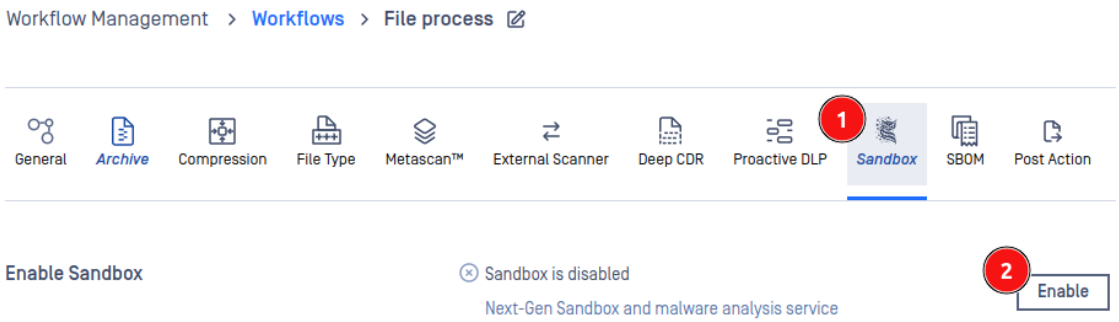
Pro Tip

Customize your workflows based on your specific usage scenario to enhance performance. For example turn off *file types* and update *engine results filtering*.

Setup several workflows tailored to different use cases, if required.

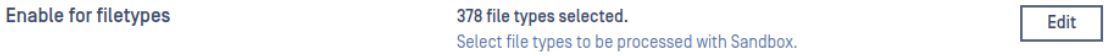
Activate `Rapid mode` when there's a need for higher throughput.

Enable the **MetaDefender Aether for Core (Adaptive Sandbox)** Engine in your workflow



Filetype filtering

Configure the **MetaDefender Aether for Core (Adaptive Sandbox)** Engine to run only for the selected file types. All the supported file types are selected by default.

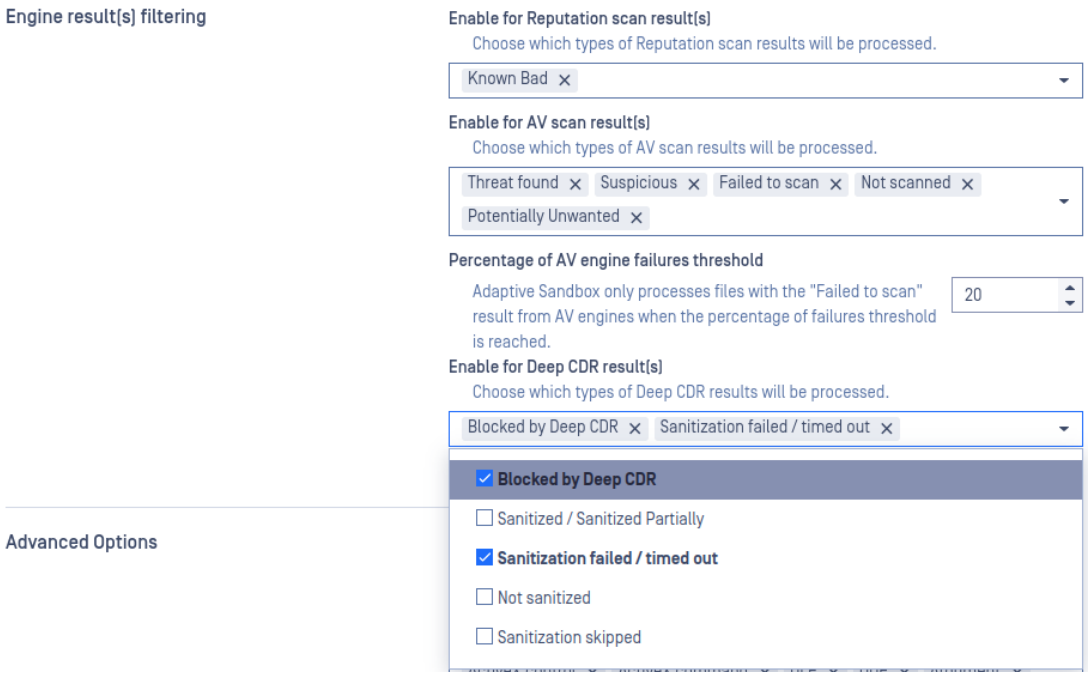


Warning

It's not recommended to choose file types that aren't included in the default configuration.

Engine result filtering

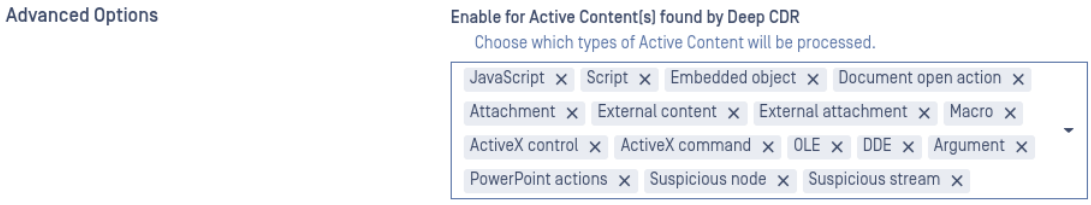
Configure the Sandbox Engine to run conditionally based on **Reputation** engine results, **Metascan** AV engine results or **Deep CDR** sanitization results.



Advanced options

Deep CDR filtering

Configure the Sandbox Engine to run conditionally based on Active Content(s) found with Deep CDR processing.



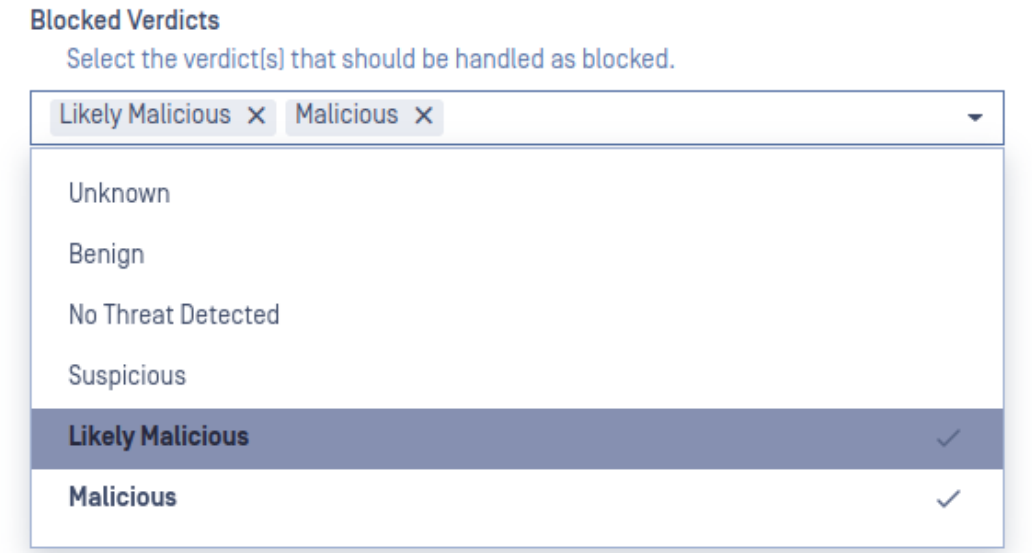
Scan timeout

Configure a timeout value in seconds which will be applied to each scan.

> **Warning**
>
> A high timeout value might significantly degrade engine performance

Blocked verdicts

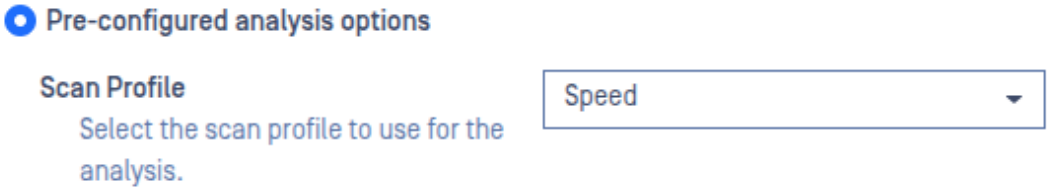
MetaDefender Core will block files if the Sandbox Engine produces a blocked verdict.



Scan tasks

Configure which tasks should be executed. Different configuration settings available for the Embedded and Remote Sandbox Engine.

Pre-configured analysis options

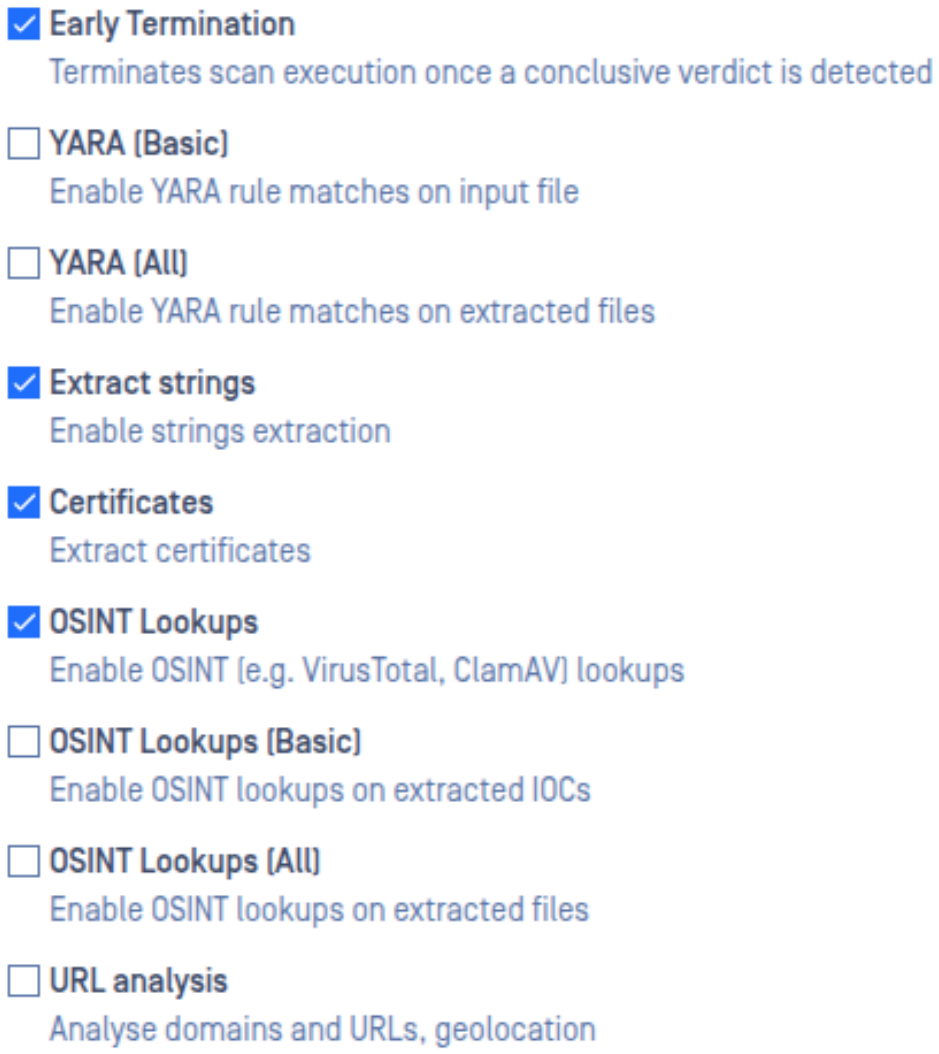


The following profiles are available in both Engines:

- **Speed:** This mode focuses on generating a final Verdict within the shortest time. It is likely that the list of IOCs is only partially complete.
- **Analysis:** This mode prioritizes extracting the most amount of IOCs over scan time.

Custom analysis options

Embedded Engine



Remote Engine

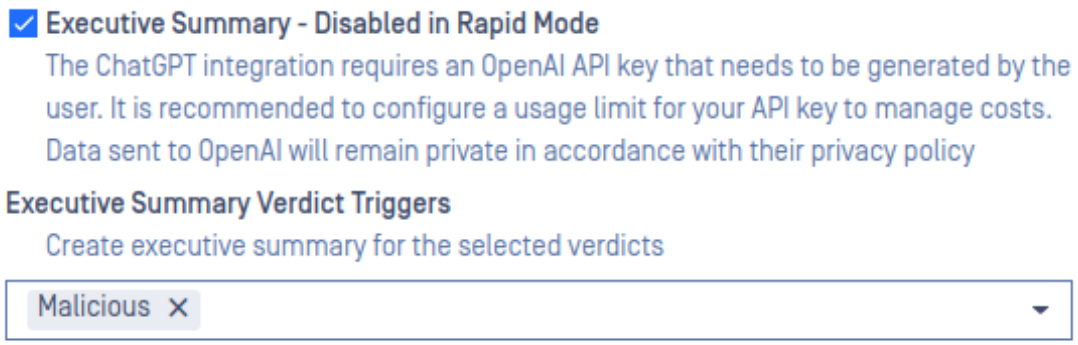
Warning!

The following options shouldn't be modified for the Remote Sandbox Engine if the *filescan.io* community site is used or **Advanced scanning** option is not enabled for the user.

- [ ] **Early Termination**
  Terminates scan execution once a conclusive verdict is detected
- [ ] **YARA (Basic)**
  Enable YARA rule matches on input file
- [x] **YARA (All)**
  Enable YARA rule matches on extracted files
- [x] **File visualization**
  Enable file visualization (e.g. PDF rendering)
- [x] **File downloads**
  Enable downloading files from extracted URLs
- [x] **OCR QR**
  Extract string with OCR and process QR codes
- [x] **Extract strings**
  Enable strings extraction
- [x] **Certificates**
  Extract certificates
- [x] **OSINT Lookups**
  Enable OSINT (e.g. VirusTotal, ClamAV) lookups
- [ ] **OSINT Lookups (Basic)**
  Enable OSINT lookups on extracted IOCs
- [ ] **OSINT Lookups (All)**
  Enable OSINT lookups on extracted files
- [x] **URL analysis**
  Analyse domains and URLs, geolocation
- [x] **Phising detection**
  Phishing detection for URLs, emails and HTML pages

Executive summary

Create an executive summary for the selected verdicts, powered by ChatGPT.



Scan Profiles

Info

Scan Profiles are available from Adaptive Sandbox 2.3.0.

When using **MetaDefender Aether for Core (Adaptive Sandbox) Remote** Engine, make sure the connected Standalone Instance is also at least version 2.2.0 for the best results.

Analysis Profile

This is the default profile for **Remote Aether engines**, designed to perform a comprehensive analysis. It executes an extensive set of tasks to extract the maximum number of Indicators of Compromise (IOCs), ensuring deep visibility into potential threats.

Speed Profile

Optimized for **Embedded Aether engines**, this profile prioritizes rapid verdict delivery. It runs a streamlined set of essential scan tasks to minimize analysis time while still providing reliable threat detection.

Feature Comparison

Scan Task	Speed	Analysis
Early Termination	Yes	No
Yara (Basic)	Yes	Yes
Yara (All)	No	Yes

Scan Task	Speed	Analysis
File visualization	No	Yes
File downloads	No	Yes
OCR QR	Yes	Yes
Extract strings	Yes	Yes
Certificates	Yes	Yes
OSINT Lookups	Yes	Yes
OSINT Lookups (Basic)	Yes	Yes
OSINT Lookups (All)	Yes	Yes
URL Analysis	No	Yes
Phishing detection	No	Yes

Load Balancing / Failover

> **Info**
> Only available in Remote Adaptive Sandbox version 2.4.0 and higher.

Enable Load Balancing

Load balancing / Failover can be enabled in **Inventory -> Modules -> Adaptive Sandbox -> Enable load balancing/ failover**

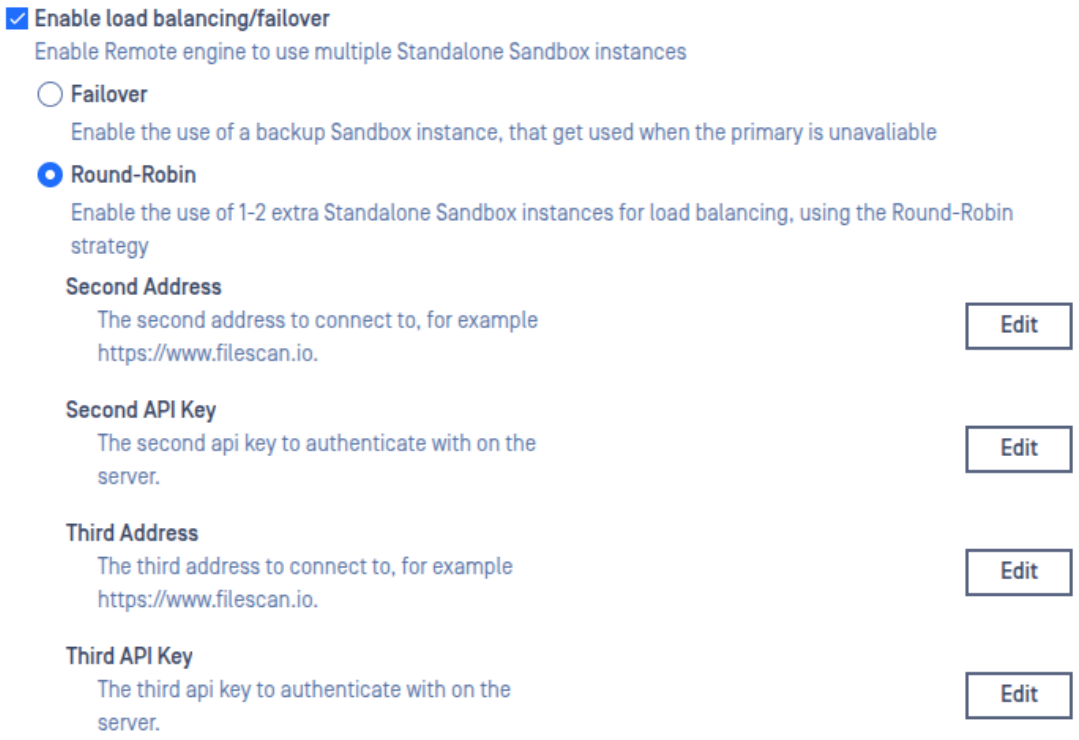


Round Robin

The Round Robin strategy can be used to add 1-2 extra Standalone Sandbox instances, for better scanning performance in high demand scenarios

Round Robin will cycle through the servers trying to distribute the load between them (Server 1 -> Server 2 -> Server 3

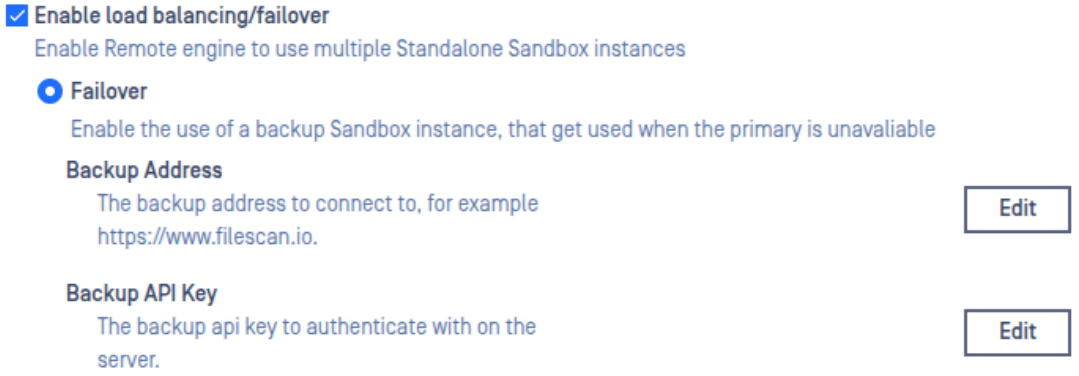
-> Repeat)



Failover

The Failover strategy can be used to register a second Standalone Instance for backup in case of High Availability setups.

The failover server will only be used when the first server is unresponsive.



Release notes

Our release notes were moved here.

Knowledge Base

A collection of information regarding the MetaDefender Aether for Core (Adaptive Sandbox) module in MetaDefender Core. For an FAQ of MetaDefender Aether, please see this page.

Internet Connection

The following features require a working internet connection

OSINT lookups

Affected features:

- OSINT lookups (Basic)
- OSINT lookups (Full)

Accessed URL: `https://api.metadefender.com/v4/`

Executive summary

Affected features:

- Executive Summary

Accessed URL: `https://api.openai.com/v1/chat/completions/`

Digital signature verification

Affected features:

- Digital signature whitelist

Sandbox needs to access the Certificate Revocation List (CRL) to verify digital certificates revocation status. Without this, whitelisting based on certificate is not working.

Common CRL distribution points:

- crl.microsoft.com
- crl.verisign.com
- crl.globalsign.com
- crl.digicert.com
- crl.entrust.net
- crl.sectigo.com

Enhancing Threat Detection with Deep CDR & MetaDefender Aether for Core (Adaptive Sandbox)

Check your version

This article applies to MetaDefender Core 5.12.1 (or higher) and Adaptive Sandbox 2.1.0 (or higher) releases deployed on Windows or Linux systems.

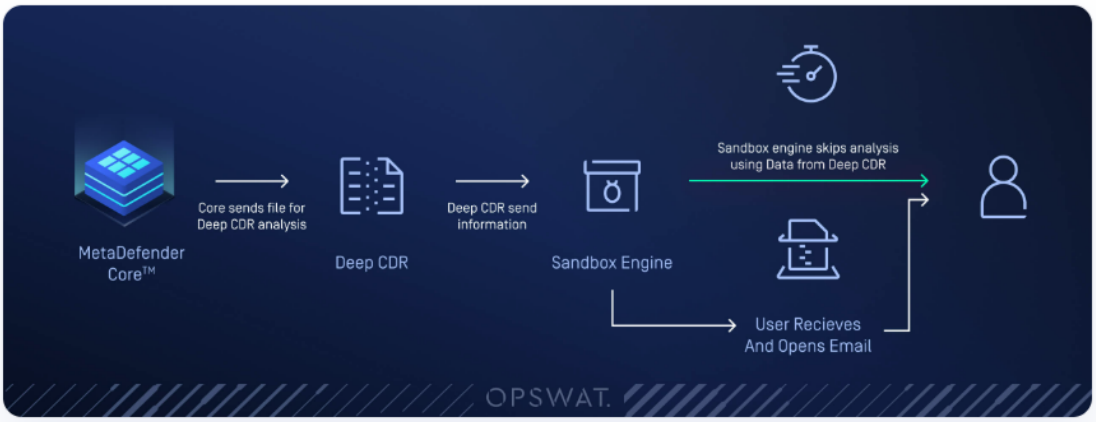
Integrating Deep CDR Analysis Mode with Adaptive Sandbox provides a layered approach to security, combining content disarmament (CDR) for immediate risk reduction with sandbox analysis for advanced zero-day detection.

- **Deep CDR acts as a pre-filter** by stripping active content (e.g., macros, scripts) to mitigate risk but does **not** determine if the file was malicious.
- **Adaptive Sandbox is required for full threat analysis**, identifying **zero-day threats, hidden malware, and evasive techniques** that CDR alone cannot detect.
- **Using CDR to trigger sandboxing** ensures that high-risk files (those containing active content) are analyzed more deeply while reducing unnecessary sandbox scans for benign files.

I. Why This Matters

Stripping active content is a foundational step in securing a file. If a file was originally malicious, removing certain elements merely disarms the malicious content, **its inherent untrustworthiness remains.**

Sandbox analysis complements CDR to uncover the true threat and potential scale of an attack, allowing for a **deeper inspection of Obfuscation, Behavioral Patterns and the identification of unknown malware.**



II. How to setup

Files are routed to the Sandbox based on multiple independent filters. A file that matches **any** of below filters will be sent Sandbox analysis.

- **Enable for Deep CDR result(s)**
- **Enable for Active Content(s) found by DeepCDR**

1. Setup Deep CDR Analysis mode

If you prefer to retain your files without modifications while still leveraging the benefits of CDR integration, enable the **CDR Analysis Mode**. For more information, visit the Deep CDR analysis mode page.

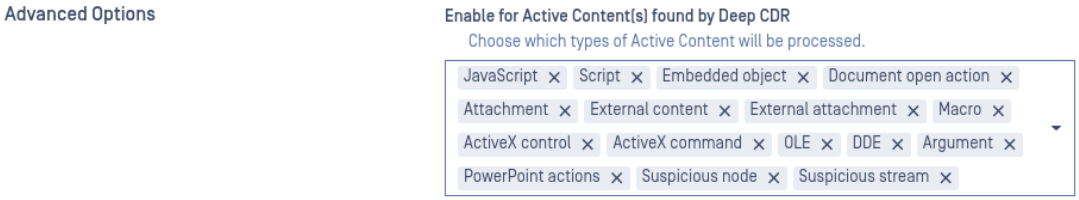
Enable Deep CDR for archives

To submit archives for processing, ensure that **Deep CDR Archive Processing** is enabled in the **Archive Compression Engine Workflow Settings**. Additionally, activate the required archive types for your use case. For detailed

instructions, refer to the Deep CDR advanced configuration page.

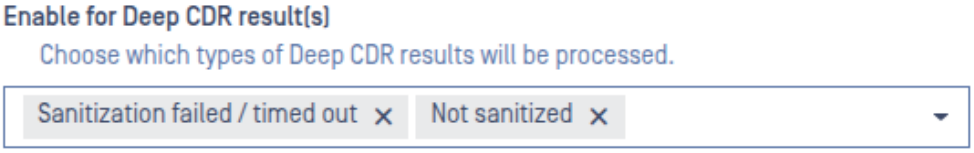
2. Setup Active Content trigger

Under **Workflow Management** -> **Workflows** -> **(Name of the workflow)** -> **Adaptive Sandbox** -> **Advanced configuration** There is an option "Enable for Active Content(s) found by Deep CDR":



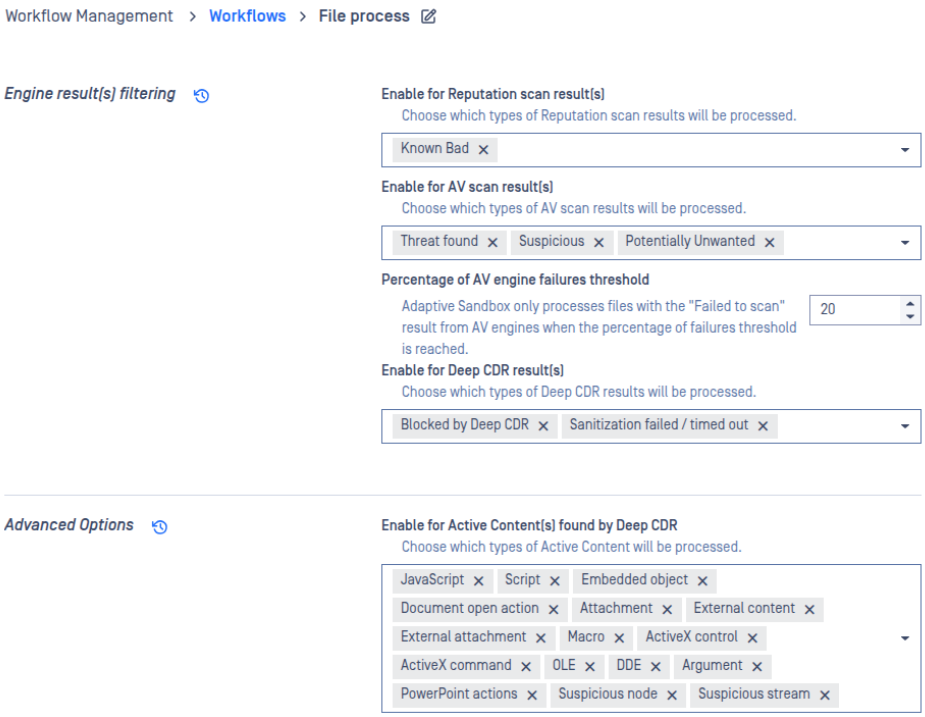
When enabled and specific active content types are selected, files in which Deep CDR detects those types (e.g., JavaScript, scripts, embedded objects) are automatically routed to the Sandbox.

Note: In Analysis Mode, Deep CDR does not generate sanitized files. As a result, the Deep CDR result will always be “Sanitization skipped” or “No Specific Configuration” (for unsupported file types). To prevent unnecessary files from being sent to the Sandbox, remove “Sanitization skipped” and “No Specific Configuration” from the **Enable for Active Content(s) found by Deep CDR** filter.

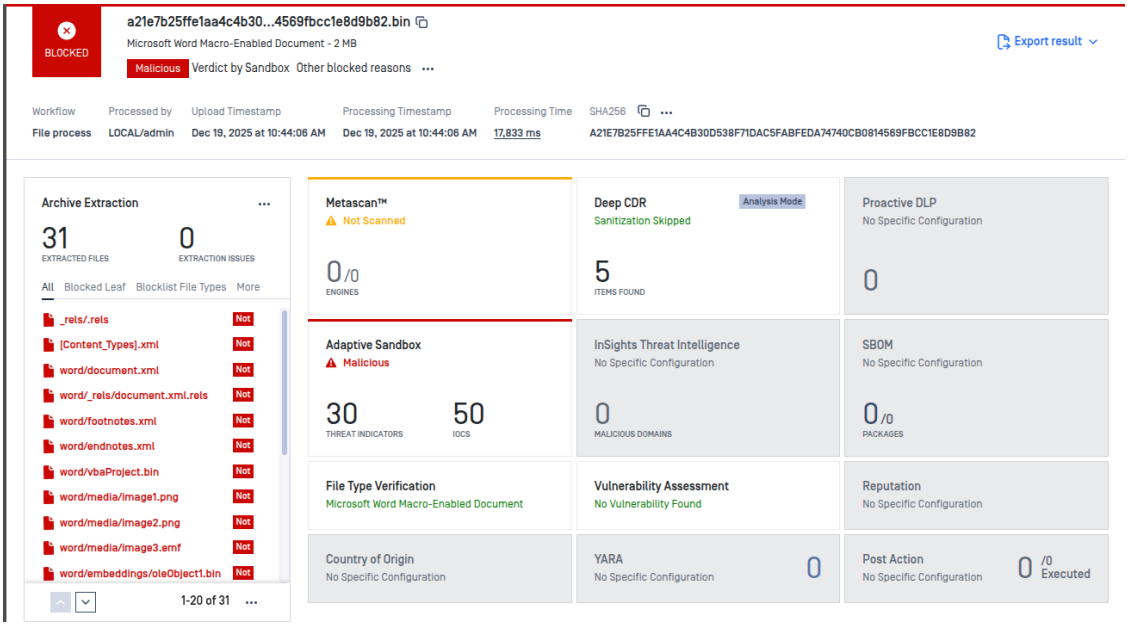


III. Example configurations

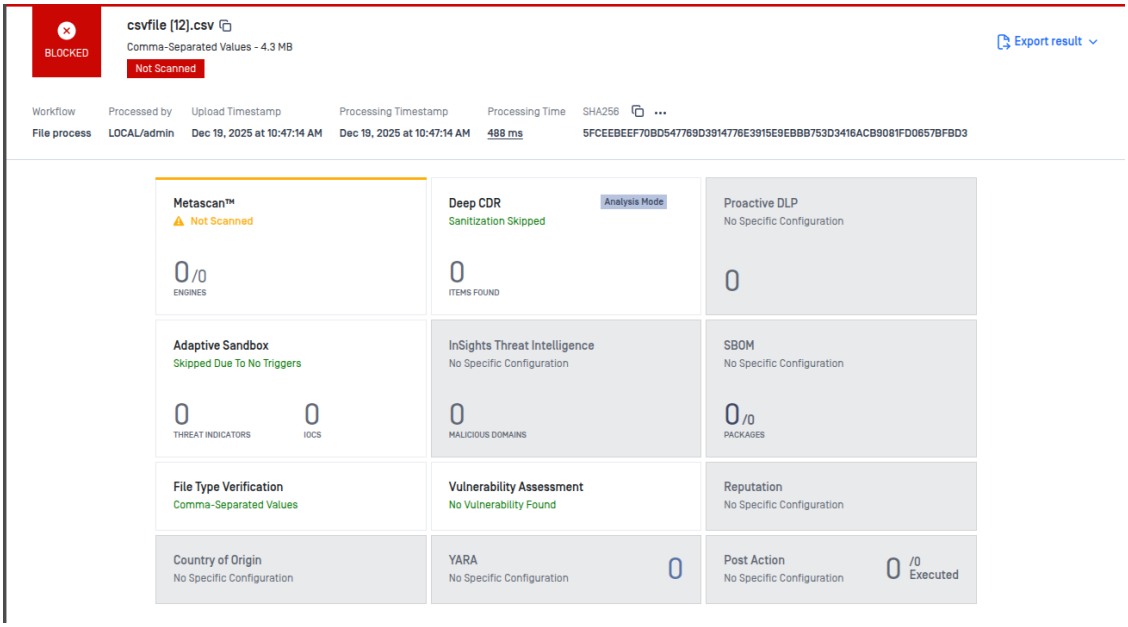
With below configurations:



Although Deep CDR detected active content, no sanitized file was generated because the system was running in **CDR Analysis Mode**. The final verdict was determined by **Adaptive Sandbox**, which identified malicious behavior and blocked the file.



In the example below, the CSV file does not contain active content. As a result, Deep CDR reported **"Sanitization skipped"** with no findings, and **Adaptive Sandbox** was skipped because no execution triggers were detected.



IV. Key Takeaway

Deep CDR **removes risky elements but does not determine trustworthiness**—it is not a replacement for threat detection. **Adaptive Sandbox is required to detect malware and unknown threats, ensuring true zero-trust security.**

Support

If **Further Assistance** is required, please follow the instructions on How To Create a Support Package, then proceed to create a support case or chat with our support engineer.

How do I obtain my OPSWAT MetaDefender Cloud Reputation API Key?

OPSWAT MetaDefender Cloud offers reputation analysis of files, URLs, and IP addresses. To access these services, you'll need an API secret, which is a unique key that authorizes your requests.

Here is a detailed guide on how to obtain the **MetaDefender Cloud Reputation API Key**:

In your MetaDefender Cloud dashboard:

1. Look for the **Licenses** section in the navigation menu.

2. This section will allow you to generate MetaDefender Cloud API key, including your **Reputation API Key**.

Find more information at the Licenses and Usage Limits documentation of MetaDefender Cloud.

To enable MetaDefender Core to utilize MetaDefender Cloud for reputation checks, you need to configure it to use the API secret you obtained from the MetaDefender Cloud service.

Step 1: Access MetaDefender Core Configuration

1. Log in to MetaDefender Core

2. Ensure you have Admin Access

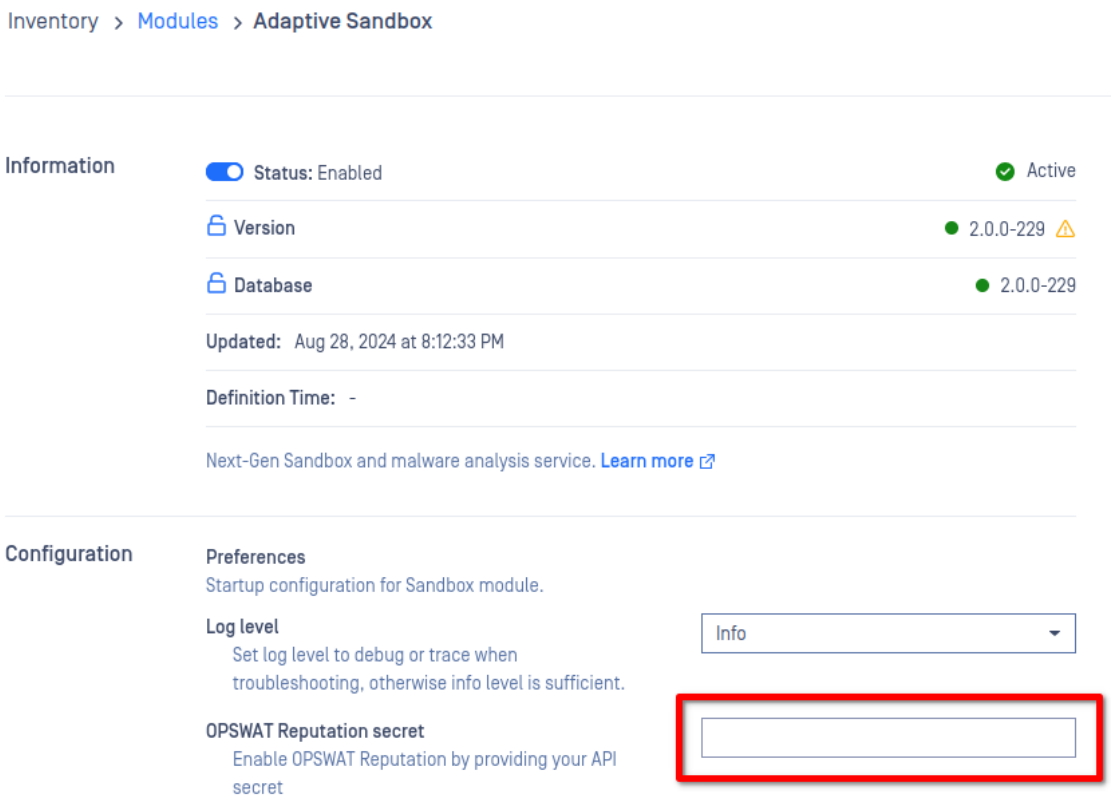
Step 2: Locate the API Configuration Section

In the MetaDefender Core dashboard:

1. **Go to the Inventory page**
2. **Navigate to the Modules Section and then find Adaptive Sandbox**:
   - Look for an option called **Configuration**.
   - This section is where you can add your MetaDefender Cloud API Key

Step 3: Enter MetaDefender Cloud Reputation API Key

1. Input Your API Secret into **OPSWAT Reputation secret** field as shown below



What is displayed on the process result tile?

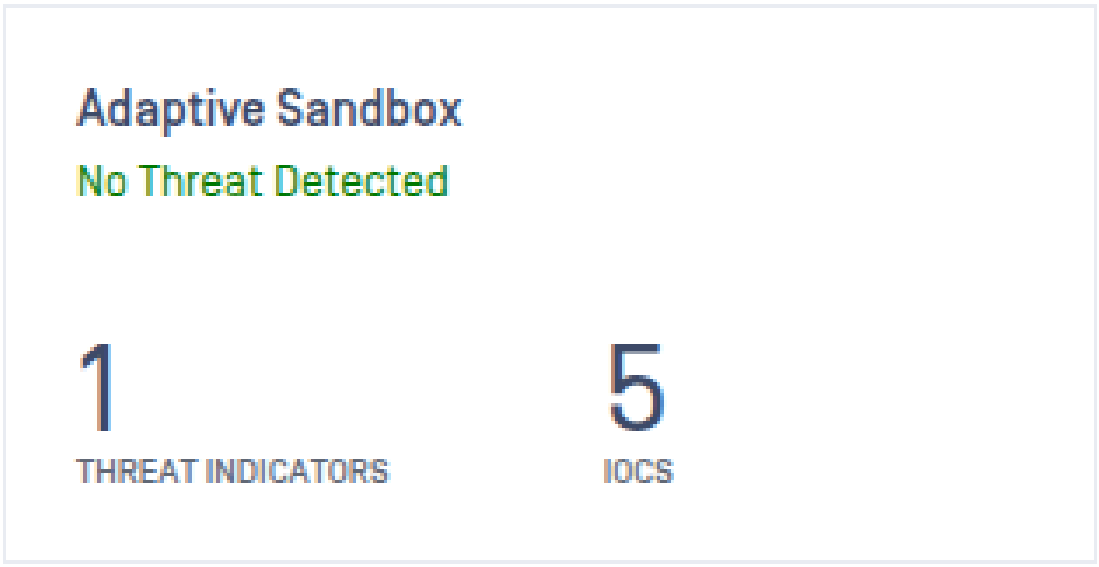
Check you version

This article applies to MetaDefender Core 5.13.0 or higher

MetaDefender Core provides a quick overview of scan results using compact tiles that summarize results for each engine. This page explains the information displayed on the Adaptive Sandbox tile and details how the tile operates.

Successful scan

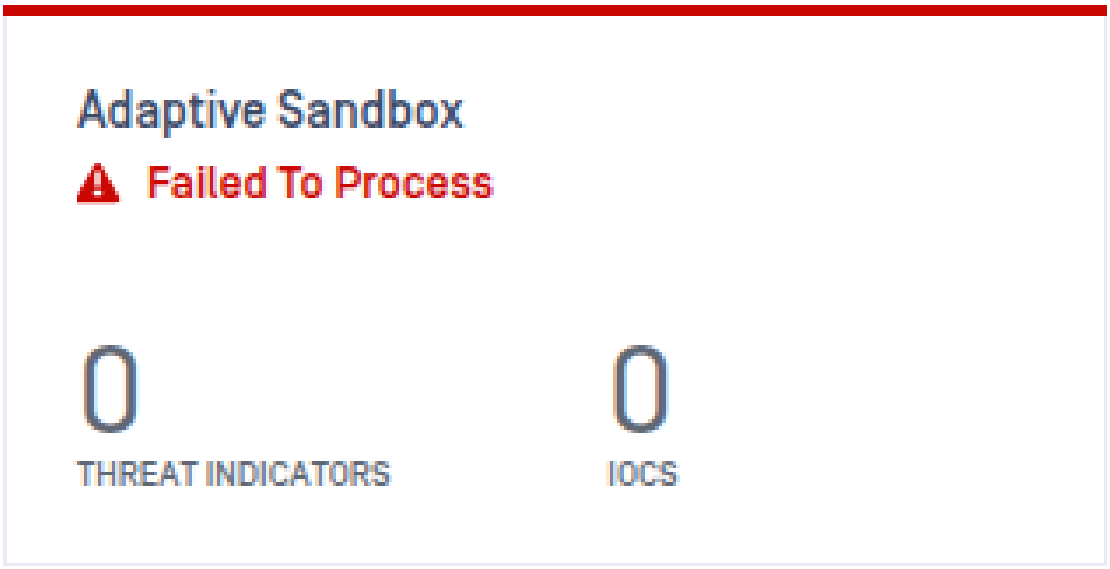
For successful scans, the tile displays the MetaDefender Aether fore Core (Adaptive Sandbox) verdict, the number of threat indicators, and the IOCs (Indicators of Compromise) identified during the scan. The tile's color corresponds to the verdict, providing a quick visual summary.



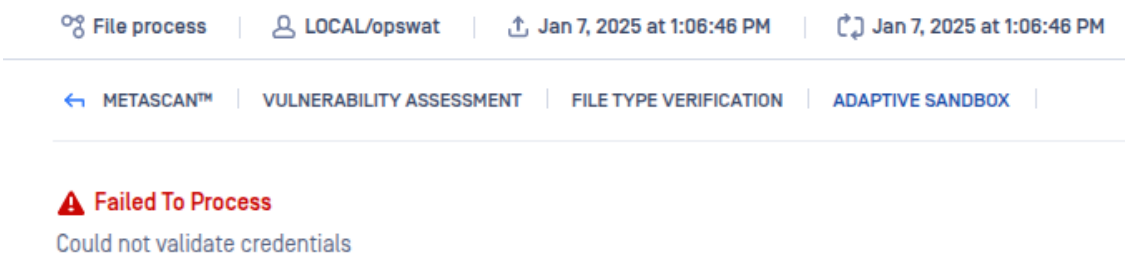
Clicking on the tile opens the detailed report, allowing users to review in-depth information about the scan results.

Failed scan

Failed scans are displayed with the error message "**Failed to process**". This indicates an issue with the scan, and users may need to check the system logs or settings for more details.

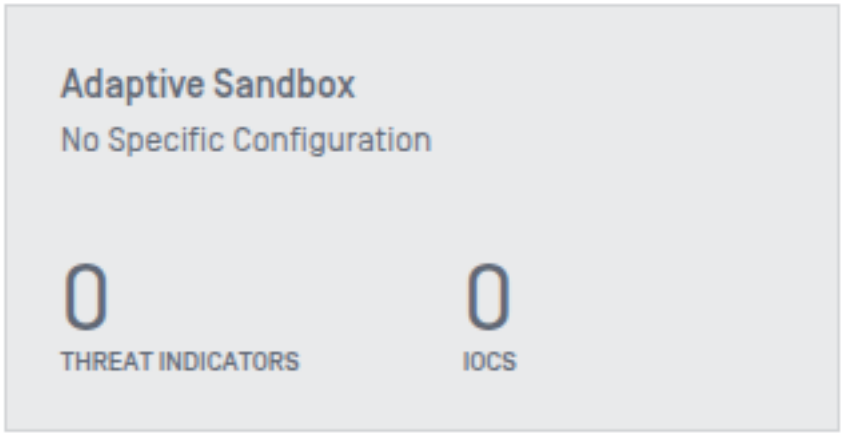
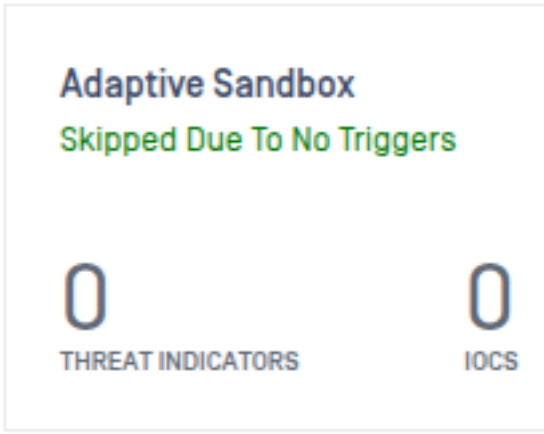


Clicking on the tile opens the error report, where users can review the error details to understand the cause of the failure.



Skipped scan

Depending on the workflow configuration, the Sandbox scan might be skipped. In this case, the tile displays the message "**No specific configuration**" with a grey accent or "**Skipped due to no triggers**" , indicating that the scan was not configured to run.

Skipped by MDCore	Skipped by Sandbox CDR workflow
	

Enhancing Threat Detection with YARA & MetaDefender Aether for Core (Adaptive Sandbox)

Check your version

This article applies to MetaDefender Core 5.14.0 (or higher) and Adaptive Sandbox 2.3.0 (or higher) releases deployed on Windows or Linux systems.

Integrating YARA Scanning with Sandboxing provides a layered approach to threat detection, combining rule-based static analysis for rapid identification of known threats with dynamic sandbox analysis for uncovering advanced, evasive, or zero-day malware.

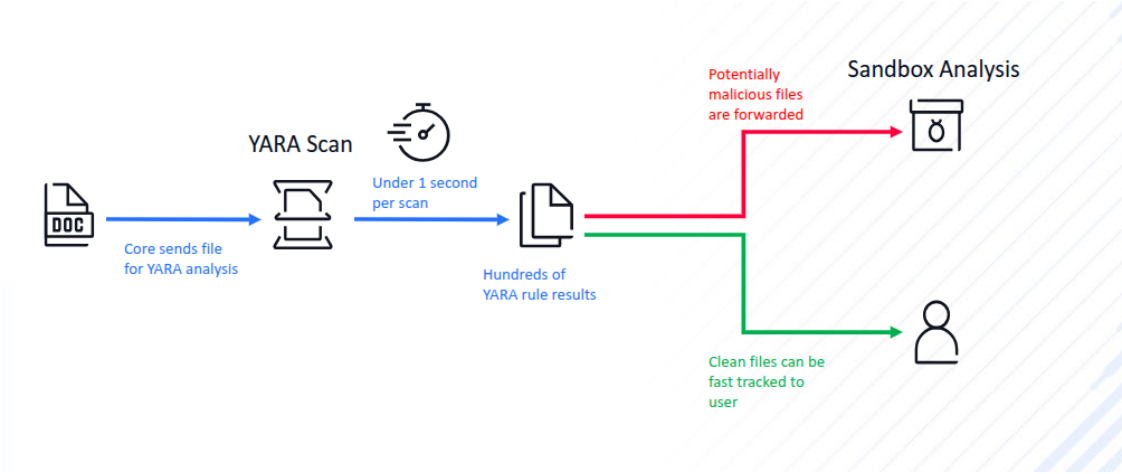
- **YARA scanning** provides rule-based pattern matching to detect known malware signatures, suspicious behaviors, or file characteristics. It acts as a fast, lightweight pre-filter for identifying potentially malicious files
- **Layered detection**: YARA rules can be tailored to flag specific threat indicators (e.g., obfuscated scripts, known malware families), while sandboxing validates and expands on these findings through behavioral analysis.

- **Improved detection accuracy**: Combining static (YARA) and dynamic (sandbox) techniques reduces false positives and enhances detection of sophisticated threats that may evade one method alone.
- **YARA as a trigger for sandboxing** enables a more efficient workflow: files that match YARA rules are escalated for deeper sandbox analysis, while clean files bypass the sandbox, reducing resource consumption and analysis time.

Why this Matters

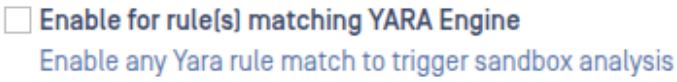
YARA rules are powerful for identifying specific patterns and characteristics of malware. By using YARA, you can detect known threats and suspicious behaviors within files or processes. This initial detection helps in identifying potential attack vectors that might be missed by traditional antivirus solutions.

Combining YARA detection with Sandbox analysis improves the accuracy of threat identification. YARA can flag potential threats based on patterns, while the Sandbox can confirm whether the flagged items exhibit malicious behavior, reducing false positives.



Setup Adaptive Sandbox YARA trigger

Under **Workflow Management -> Workflows -> (Name of the workflow) -> Adaptive Sandbox -> Advanced configuration,** there is an option "Enable for Adaptive Sandbox YARA rules":



Key Takeaway

Identifying attack vectors with YARA and triggering Sandbox scans based on the results offers a comprehensive approach that enhances threat detection and provides in-depth analysis. To build on this, refer to the **Attack Vector Detection Coverage** section to see the range of attack vectors the YARA engine can flag early in the threat path, creating an ideal trigger point for deeper Sandbox analysis. It gives a clearer view of how YARA-driven detection strengthens and streamlines the entire analysis workflow.

Best practices for API keys in MetaDefender Core Sandbox integrations

Best practices for API keys in MetaDefender Core Sandbox integrations (least privilege)

Overview

When MetaDefender Core integrates with a remote Sandbox, Core authenticates using a Sandbox API key. The API key inherits the permissions of the Sandbox user that owns it. Therefore, the owning user must have the minimal capabilities required to submit samples for detonation and read analysis results.

Scope and applicability

- Core submitting files or URLs to a Sandbox for detonation/analysis and retrieving verdicts and reports.
- Environments spanning dev, test, staging, and production where Core talks to one or more Sandbox tenants.

The API key itself does not hold permissions; it reflects the privileges of the Sandbox user. Adjust the user's role to change effective permissions.

Goal

Enable MetaDefender Core to submit items to the Sandbox and retrieve results using a least-privilege, auditable service account model.

Recommended account model (Sandbox side)

- Create a dedicated integration user (service account), for example `svc_core_to_sandbox`.
- If supported, mark it as a service account to improve audit clarity. This label does not grant permissions; role assignment does.

Minimum Sandbox permissions

- Ability to upload/submit files or URLs for detonation.
- Ability to access/read analysis results and verdicts.

In many RBAC models, an Analyst-type or equivalent operational role covering "submission + results access" is sufficient. Avoid administrative roles unless required for tenant configuration, user management, or global policy changes.

Principle of least privilege: grant only submission and read-results scopes to reduce blast radius if the key is exposed.

API key hygiene and security best practices

- Use a dedicated integration user, never a personal account.
- Scope access to the smallest organizational boundary needed (specific tenant, org, or project).
- Issue separate API keys per integration and per environment (dev/test/stage/prod) to isolate risk and simplify revocation.
- Store keys in a secure secret manager; never embed in source control, CI logs, or config files in plaintext.
- Minimize key exposure in telemetry: redact keys from logs, traces, and error messages.
- Rotate keys regularly and on personnel or role changes; automate rotation where possible.
- Constrain network paths so only the Core system can reach the Sandbox API (IP allowlists, private connectivity, or mTLS where available).

Configuration guidance (high level)

1. Create the Sandbox service account `svc_core_to_sandbox` and assign the lowest role that still permits submissions and results access.
2. Generate an API key for that account; store it in your organization's approved secret manager (for example, a vault) and reference it from Core's configuration.
3. Restrict network egress/ingress to the Sandbox API endpoint from the Core host or cluster only; verify TLS and certificate pinning options if supported.
4. Validate end-to-end by submitting a benign sample and confirming retrieval of the corresponding analysis result.

Operational checklist

- Sandbox integration user created (`svc_core_to_sandbox`)
- User assigned the lowest role that still permits submit + results access
- API key stored in an approved secret manager and referenced by Core without plaintext exposure
- Key rotation cadence, ownership, and revocation procedure documented

Common pitfalls to avoid

- Using admin or global roles for routine submissions, expanding attack surface unnecessarily.
- Reusing one API key across multiple environments or integrations, complicating audit and revocation.
- Leaving keys in application logs or build artifacts where they can be harvested.

Support

If further assistance is required, please proceed to log a support case or chat with our support engineer.

How to configure Offline Reputation Service allowlisting in MetaDefender Core Adaptive Sandbox.

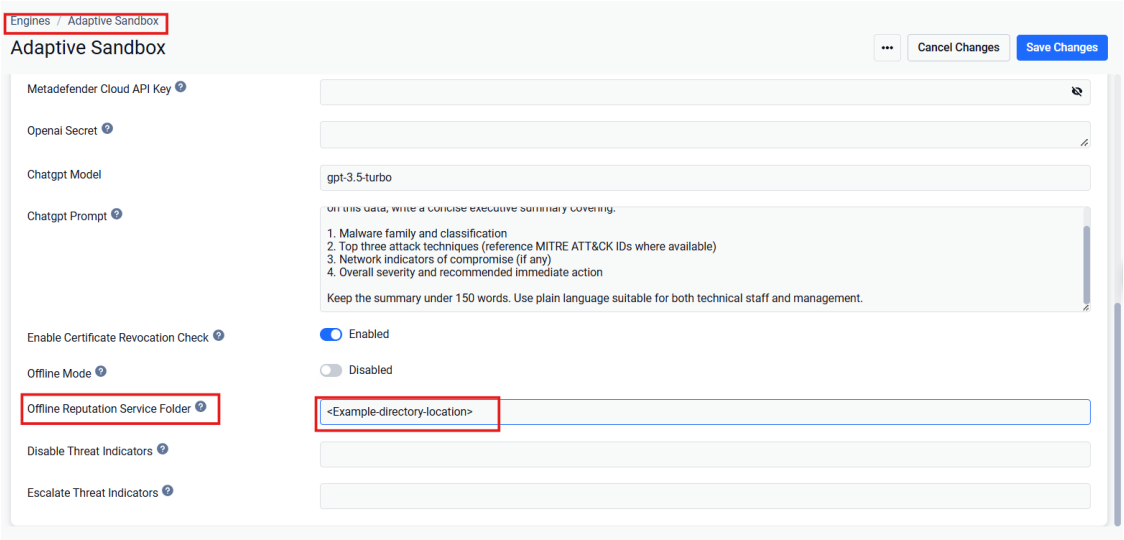
Check Your Version:

This article applies to MetaDefender Core with the embedded Adaptive Sandbox module, Adaptive Sandbox version **3.3.0** or later, deployed on Windows and Linux systems.

Overview

The Offline Reputation Service allows you to supply local reputation entries for MetaDefender Core's Adaptive Sandbox. Use an allow list to trust selected indicators, such as a URL extracted from a document, when you have reviewed the detection and determined that an exception is appropriate.

Configure a permanent reputation folder in the Core interface and place the applicable reputation files in that folder.



Applies to

- MetaDefender Core with the embedded Adaptive Sandbox module on Windows or Linux.
- Adaptive Sandbox version **3.3.0** or later, with a Core build that exposes the **Offline Reputation Service folder** setting.

Configure URL allowlisting

1. Create a permanent reputation folder

On the server running Adaptive Sandbox, create a dedicated folder outside the MetaDefender Core installation and engine directories. For example:

Platform	Example folder
Windows	`C:\Users\CoreAdmin\SandboxReputation`
Linux	`/home/coreadmin/SandboxReputation`

These paths are examples; choose a path appropriate for your deployment. Avoid special characters in the path. A simple path without spaces is recommended. Ensure that the account running the module can read the folder and its files, including access through the parent directories.

Important: Do not store customer reputation files inside Core's installation or engine directories. Files in those locations can be removed or replaced during updates or restarts.

2. Create the URL allow list

Inside the folder, create a plain-text file named exactly:

```
Code

url_allowed.txt
```

Add one URL or domain per line. For example:

```
Code

https://portal.example.com
downloads.example.org
```

The protocol is optional. The reputation service ignores the protocol when matching an entry, so using `http://`, `https://`, or no protocol does not change the allowlisting behavior.

You may add an optional description after a comma:

```
Code

portal.example.com,Approved internal portal
```

URL matching is domain based. Review the scope of the exception before adding an entry; do not assume that a full URL limits the exception to a single page or path.

3. Configure the folder in MetaDefender Core

1. Sign in to the MetaDefender Core management console.
2. Go to **Inventory** → **Modules** → **Adaptive Sandbox**.
3. Locate **Offline Reputation Service folder**.
4. Enter the full path to the folder created in step 1. Enter the folder path, rather than the path to `url_allowed.txt`.

5. Save the module configuration.

You do not need to enable **Offline mode** for this feature. A **manual restart** of the MetaDefender Core service is **not required** and the engine will restart the **first** time this feature is enabled and the settings are saved; afterwards, the engine has to be **restarted** for any new changes to take effect.

Other supported reputation file names

The linked Offline Reputation Service reference defines the following allow-list files:

Entry type	Allow-list file
MD5 hash	`md5_allowed.txt`
SHA-256 hash	`sha256_allowed.txt`
IP address	`ip_allowed.txt`
Domain	`domain_allowed.txt`
URL	`url_allowed.txt`
Digital certificate owner	`digicert_owner_allowed.txt`

Certificate-owner entries have separate formatting requirements and do not support optional descriptions.

File formats

Each file - except `digicert_owner_allowed.txt` - can contain single-line entries as comma separated data and optional description pairs.

Code · csv

```
<data>,<optional description>
```

Examples: **md5_allowed.txt**

Code · csv

```
d41d8cd98f00b204e9800998ecf8427e,Known good from internal tool, empty file
d41d8cd98f00b204e9800998ecf8427d
```

Example:**digicert_owner_allowed.txt**

Code · csv

```
CN="Trusted .Inc", O="Trusted, Inc.", L=San Francisco, ST=California, C=US
```

Import Behavior

- Data is **imported at application startup, if a file has changed**.
- If a file changes, **existing data for that type and list is replaced entirely**.
- All internal (system-managed) reputation data is updated after installation.
- External (customer) reputation data can be modified and re-imported at any time.

Frequently asked questions

Q: Which file path should I use for the allowlist file?

You can store the allowlist file in any suitable location, subject to the following restrictions:

- Do not store it in the MetaDefender Core installation folders, as files in these folders may be removed or replaced during updates or service restarts.
- Ensure the path contains no spaces or special characters.

A dedicated folder under your user profile is a suitable location.

Q: Do I need to enable Offline mode?

No. This feature works regardless of whether Offline mode is enabled.

Q: Should I include the protocol, such as HTTP or HTTPS, when adding a URL allow/block?

The protocol is optional. The reputation service ignores it, so allowlist and blocklist entries apply whether the protocol is included, omitted, or different from the protocol used in the request.

Q: Do I need to restart the MetaDefender Core service after making these changes?

No. A manual service restart is not required. The engine will restart the **first** time the settings are saved, and afterwards the engine has to be restarted for any new changes to take effect.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Sandbox-Core High Availability Setup

Setting up Core-Sandbox pairs (comprising essential system components) in each Availability Zone (AZ) or data center. It helps ensure high availability (HA) across multiple locations.

Limiting cross-data center traffic

It is recommended to limit cross-data center traffic as much as possible. The goal is to minimize the communication between different data centers. This reduces the risk of issues that can arise from data traveling across long

distances.

Using a Load Balancer (LB) on top of each Core-Sandbox pair

To have a proper High Availability (HA) configuration, it is recommended to deploy a Load Balancer on top of a Core-Sandbox pair following the Using external Load Balancer guide.

Isolating communication and ensuring each AZ can function independently

A given Core-Sandbox pair should only communicate with each other. There should be no need for any cross-communication between AZs.

If AZ#1 goes down (with all its servers), then AZ#2 must be able to fully take over the processing of incoming requests.

The key point is that each AZ must be able to function independently. It cannot rely on any resources in another AZ. A proper High Availability (HA) configuration retains its full functionality even if an entire AZ is unavailable.

Monitoring system resources

In general, the end user should monitor the usage of system resources for both Core and Sandbox in each AZ to see if there is a performance bottleneck in the processing pipeline.

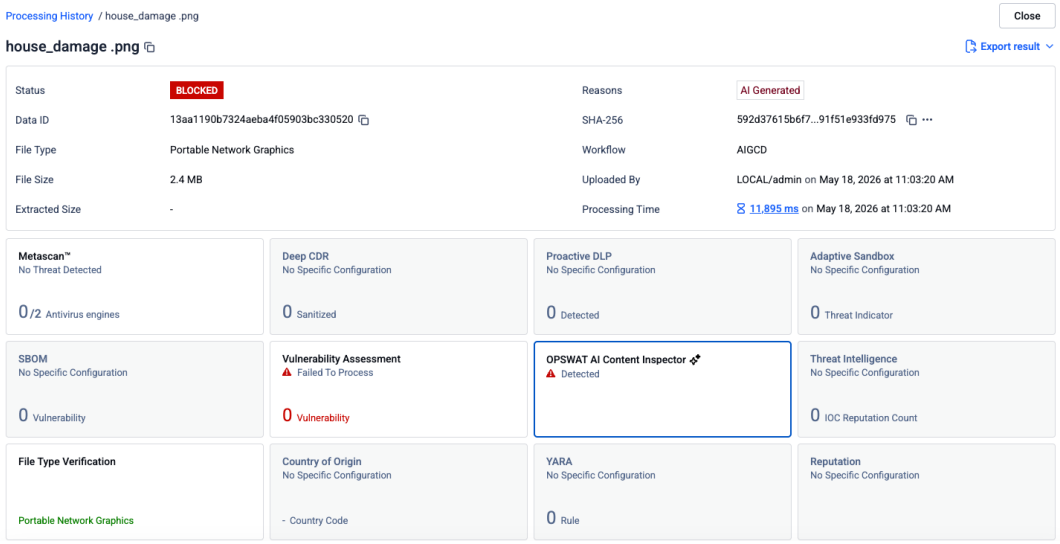
To handle a failover scenario, it is sufficient to monitor the health status of Core using this API endpoint: https://docs.opswat.com/mdcore/metadefender-core#healthcheck

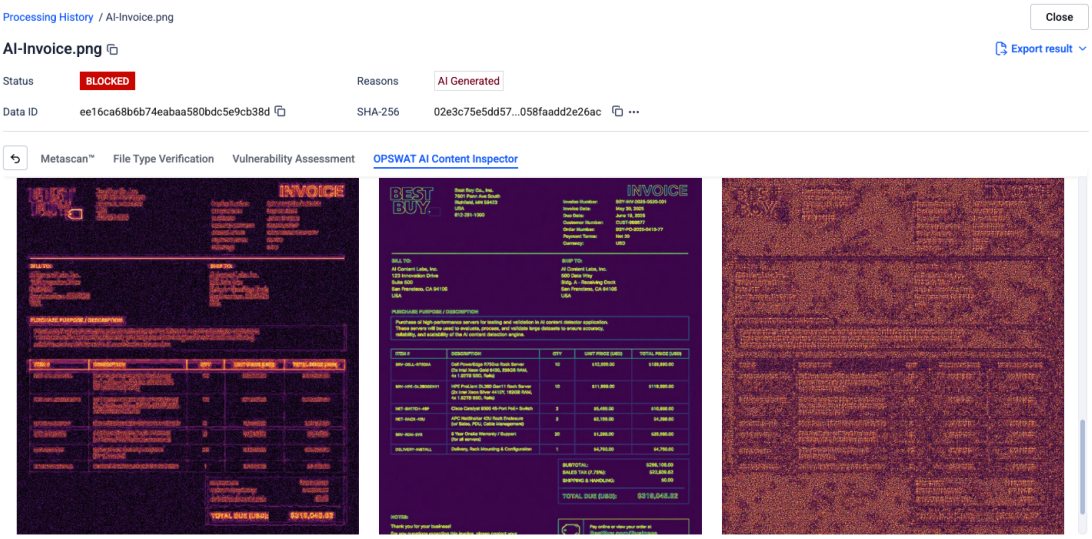
If the Core instance in AZ#1 is not responding or its queue is getting full (perhaps due to a Sandbox issue), then the application load balancer can decide to send all new requests to Core in AZ#2.

Overview

> **Info**
>
> OPSWAT AI Content Inspector requires separate licensing for MetaDefender to enable this module.

OPSWAT AI Content Inspector is an AI-driven content authenticity and document fraud detection engine for MetaDefender. It analyzes visual, textual, and structural signals to help organizations identify suspicious invoices, claim images, accident photos, and other high-risk content before review, approval, or payout. It also identifies whether digital content, including text and images, was created by a human or generated by an AI system.





The engine uses a hybrid detection methodology that combines metadata analysis, content fingerprinting, and multimodal AI classification to evaluate both content origin and document authenticity. By correlating signals across these layers, OPSWAT AI Content Inspector can distinguish AI-generated content from authentic material and surface fraud indicators in invoices, claim evidence, and identity documents. Detection results include confidence scores and explainability data, enabling administrators to understand the basis for each classification and fraud

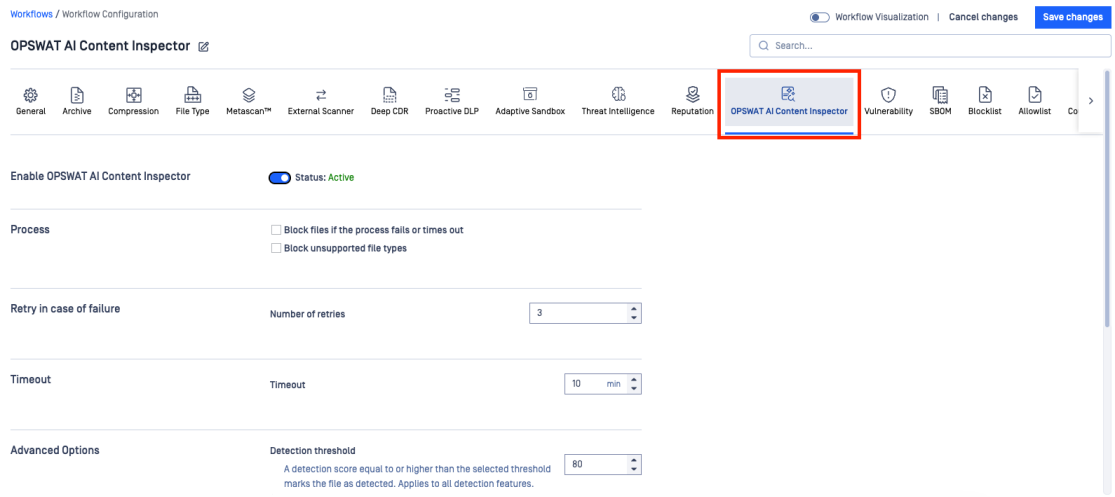
verdict.

Key Capabilities

- **Detect AI-generated text and images, and flag AI-generated document-fraud indicators** across commonly used file formats found in enterprise workflows such as claims intake, AP/invoice processing, and etc.
- **Analyze metadata patterns and content fingerprints** to identify synthetic generation, tampering, and forgery signals before content reaches downstream systems.
- **Enforce organizational content integrity and fraud-screening policies** based on detection outcomes , allow, flag, block, or route for human review at scale.

General Configuration

Policies > Workflow rules > "Workflow name" > OPSWAT AI Content Inspector



Supported file types

	File Type	Description	Supported Features
1	.pcx	Personal Computer eXchange	AI-generated image detection by AI and by metadata inspection
2	.fpx	FlashPix bitmap	AI-generated image detection by AI and by metadata inspection
3	.jpg	JPEG Image	AI-generated image detection by AI and by metadata inspection

	File Type	Description	Supported Features
4	.bmp	Windows Bitmap Image	AI-generated image detection by AI and by metadata inspection
5	.emf	Windows Enhanced Metafile	AI-generated image detection by AI and by metadata inspection
6	.avif	AV1 Image File Format	AI-generated image detection by AI and by metadata inspection
7	.apng	Animated Portable Network Graphics	AI-generated image detection by AI and by metadata inspection
8	.png	Portable Network Graphics	AI-generated image detection by AI and by metadata inspection
9	.tga	Truevision Graphics Adapter	AI-generated image detection by AI and by metadata inspection
10	.webp	Web Picture	AI-generated image detection by AI and by metadata inspection
11	.ico	Microsoft Icon file	AI generated image detection by AI and by metadata inspection
12	.jpx | .j2k	JPEG 2000	AI-generated image detection by AI and by metadata inspection
13	.eps	Encapsulated PostScript	AI-generated image detection by AI and by metadata inspection
14	.cur	Microsoft Windows cursor image	AI-generated image detection by AI and by

	File Type	Description	Supported Features
			metadata inspection
15	.psd	Adobe Photoshop Document	AI-generated image detection by AI and by metadata inspection
16	.wmf	Windows Metafile	AI-generated image detection by AI and by metadata inspection
17	.txt	plain text	AI-generated text detection by AI
18	.md | .markdown	Markdown markup language	AI-generated text detection by AI
19	.pdf	Adobe Portable Document Format	AI-generated text and Image detection by AI
20	.msg	Outlook Message	AI-generated text and Image detection by AI
21	.eml	Electronic Mail	AI-generated text and Image detection by AI
22	.docx	Microsoft Word Document	AI-generated text and Image detection by AI
23	.docm	Microsoft Word Macro-Enabled Document	AI-generated text and Image detection by AI
24	.doc	Microsoft Word 97-2003 Document	AI-generated text and Image detection by AI
25	.rtf	Rich Text Format	AI-generated text and Image detection by AI
26	.pptx	Microsoft PowerPoint Presentation	AI-generated text and Image detection by AI
27	.pptm	Microsoft PowerPoint Macro-Enabled	AI-generated text and Image detection by AI

	File Type	Description	Supported Features
		Presentation	
28	.potx	Microsoft PowerPoint Template	AI-generated text and Image detection by AI
29	.potm	Microsoft PowerPoint Macro-Enabled Template	AI-generated text and Image detection by AI
30	.ppsx	Microsoft PowerPoint Slide Show	AI-generated text and Image detection by AI
31	.ppsm	Microsoft PowerPoint Macro-Enabled Slide Show	AI-generated text and Image detection by AI
32	.ppt	Microsoft PowerPoint 97-2003 Presentation	AI-generated text and Image detection by AI
33	.pot	Microsoft PowerPoint 97-2003 Template	AI-generated text and Image detection by AI
34	.pps	Microsoft PowerPoint 97-2003 Slide Show	AI-generated text and Image detection by AI
35	.xlsx	Microsoft Excel Workbook	AI-generated text and Image detection by AI
36	.xlsm	Microsoft Excel Macro-Enabled Workbook	AI-generated text and Image detection by AI
37	.xltx	Microsoft Excel Template	AI-generated text and Image detection by AI
38	.xltm	Microsoft Excel Macro-Enabled Template	AI-generated text and Image detection by AI
39	.xlam	Microsoft Excel Macro-Enabled Add-in	AI-generated text and Image detection by AI
40	.xlsb	Microsoft Excel Binary Workbook	AI-generated text and Image detection by AI

	File Type	Description	Supported Features
41	.xls	Microsoft Excel 97-2003 Workbook	AI-generated text and Image detection by AI
42	.xlt	Microsoft Excel 97-2003 Template	AI-generated text and Image detection by AI
43	.odt	OpenDocument Text	AI-generated text and Image detection by AI
44	.ods	OpenDocument Spreadsheet	AI-generated text and Image detection by AI
45	.ots	OpenDocument Spreadsheet Template	AI-generated text and Image detection by AI
46	.odp	OpenDocument Presentation	AI-generated text and Image detection by AI
47	.otp	OpenDocument Presentation Template	AI-generated text and Image detection by AI

Detection

OPSWAT AI Content Inspector applies a hybrid detection methodology across three specialized classifiers, each tuned to a distinct content-risk surface: AI-generated text, AI-generated images, and document fraud. Each classifier produces an independent verdict with a confidence score, and verdicts can be combined into a single policy decision, allow, flag, block, or route for review.

This page introduces the three detection types, the shared methodology that powers them, and how their verdicts surface inside MetaDefender workflows.

The three detection types

Detection type	What it identifies	Primary inputs
AI-Generated Text Detection	Text passages produced or substantially rewritten by generative language models	.txt, .md, .markdown, text layers in .pdf
AI-Generated Image Detection	Images produced by diffusion,	All supported image formats (see

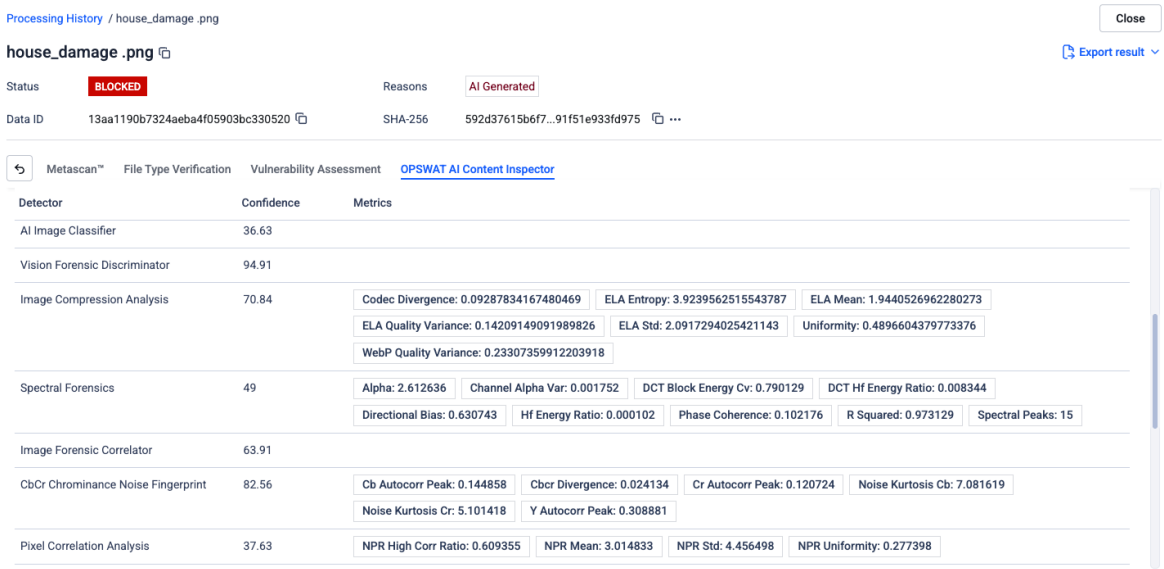
Detection type	What it identifies	Primary inputs
	GAN, or other generative image models	Supported file types)
Document Fraud Detection	Detection of forgery, tampering, and fabrication indicators in business documents using AI models, including fake invoices, insurance fraud, and others	.pdf, text-bearing documents, embedded images

Each classifier ships with its own quality gates and false-positive benchmarks, so verdicts in one category never compromise the threshold integrity of another.

How verdicts are returned

Every scan returns a structured result per classifier:

- **Verdict** - one of `Not Detected`, `Uncertain`, `Detected`
- **Confidence score** - (0–100)
- **Contributing detectors and metrics** - All detected signals and detailed information related to their detection results



AI-Generated Text Detection

AI-Generated Text Detection identifies text passages produced or substantially rewritten by generative language models. The classifier evaluates stylometric features, statistical token patterns, and structural cues that distinguish model-generated prose from human-written content, then returns a verdict with a confidence score and the contributing signals.

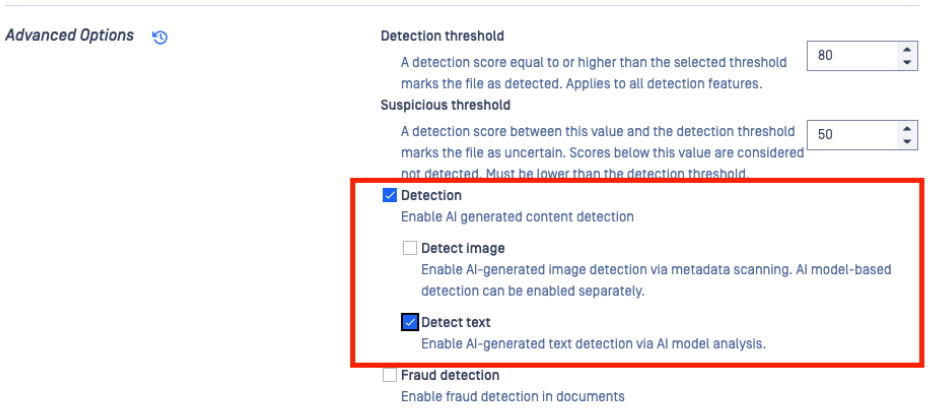
This classifier is designed for high-volume text intake pipelines where authenticity matters, including applications, support submissions, contract reviews, and any workflow where AI-generated content must be flagged, attributed, or routed for human review.

Text detectors

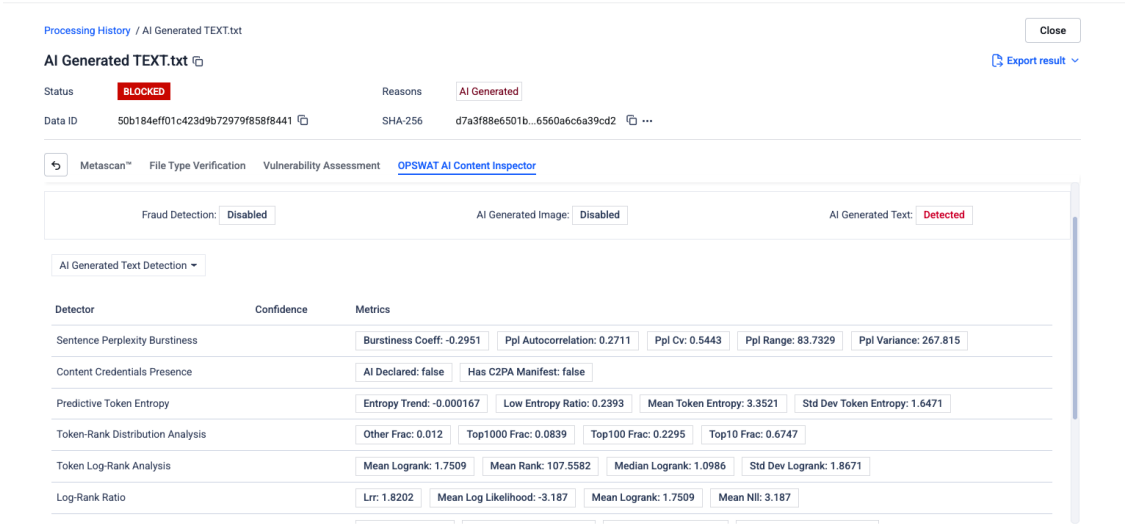
This section explains how each detector analyzes written content to distinguish AI-generated text from human-written text. The detectors evaluate linguistic, structural, and contextual patterns to identify signals commonly associated with AI-generated writing and help organizations assess the authenticity of textual data.

How to Configure

Policies > Workflow rules > "Workflow name" > OPSWAT AI Content Inspector > Advanced Options



- **Detect text:** When enabled, the text is analyzed using the full AI detection pipeline, including forensic analysis, linguistic pattern evaluation, contextual analysis, and machine learning models designed to identify signals commonly associated with AI-generated content.



AI-Generated Image Detection

AI-Generated Image Detection identifies images created or edited by generative AI, including current diffusion, flow-

matching and autoregressive image models. It combines deep-learning pixel analysis with verification of C2PA Content Credentials and metadata inspection, and returns a verdict, an AI probability and the contributing detectors.

It targets image-fraud entry points in enterprise workflows: fabricated accident photos in auto insurance claims, synthetic damage evidence in home insurance, and AI-generated images in onboarding and e-mail.

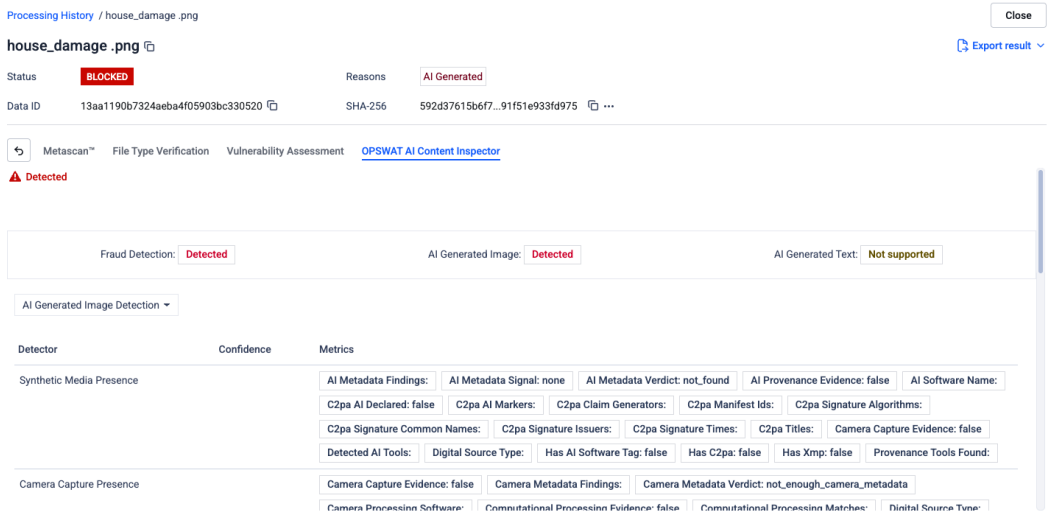
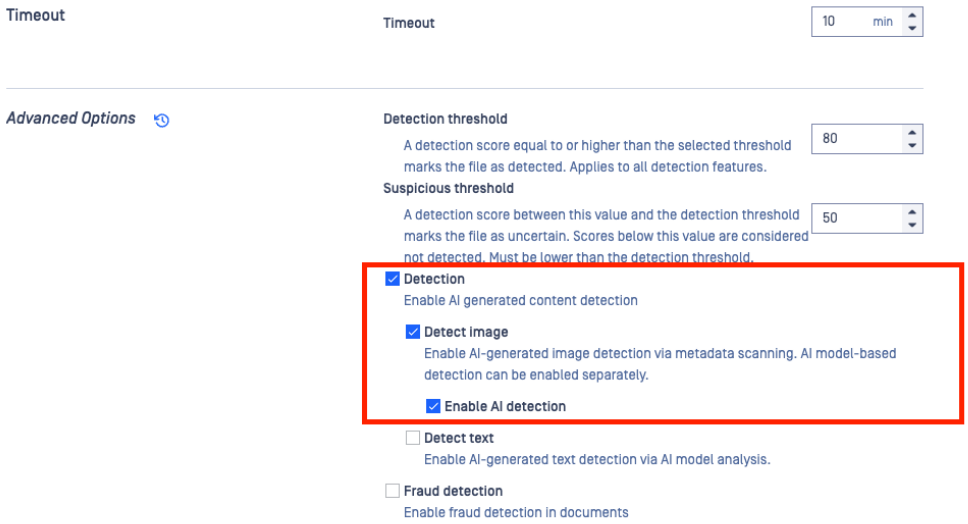


Image detectors

This section explains how each detector analyzes image content to distinguish AI-generated or manipulated images from authentic camera-captured photos. The detectors evaluate a range of visual, statistical, and metadata-based signals within the image to identify patterns commonly associated with synthetic or edited media, helping organizations assess the authenticity and integrity of visual data.

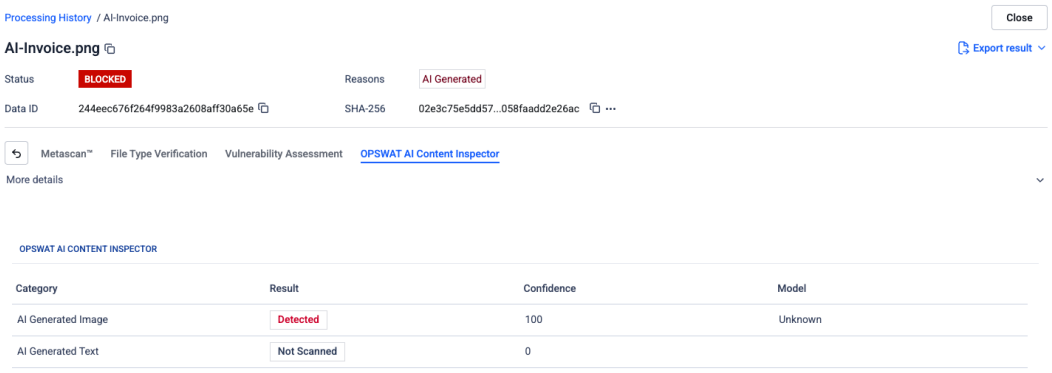
How to Configure

Policies > Workflow rules > "Workflow name" > OPSWAT AI Content Inspector > Advanced Options

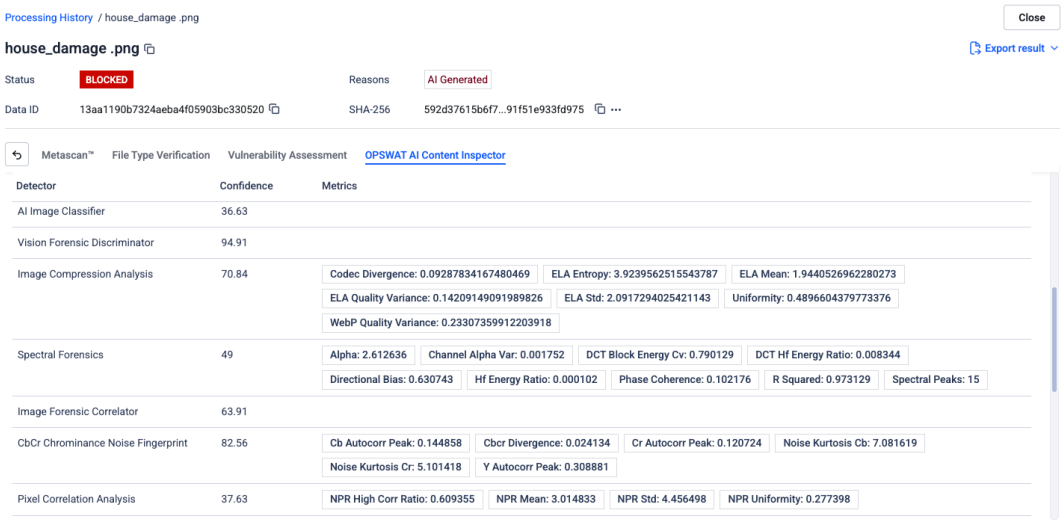


- **Detect Image:** When enabled, the uploaded image is scanned for metadata indicators associated with AI-generated or edited media. This includes checking for software tags, missing camera information, provenance

data, and other file-level signals that may suggest synthetic content. (Only metadata-based analysis is performed when this option is enabled by itself.)



- **Enable AI Model:** When enabled, the image is analyzed using the full AI-detection pipeline, including forensic analysis, machine learning models, metadata inspection, and visual artifact detection. This option performs a comprehensive scan using all available detectors and AI models to provide a higher-confidence authenticity assessment.



Overview

Reputation Engine

Faster Remediation For False Detections

Reputation Engine matches file hashes against our global database of known good and bad files; and leverages advanced analyses to remediate false detections faster.

Benefits of the Reputation Engine:

- Advanced threat detection - Identify uploaded files as `Known Good`, `Known Bad` by matching hashes against our database of known malicious or safe files. If the file is not yet part of our database, the engine will identify it as `Unknown`.
- Minimized false positives/negatives - Leverage advanced analyses conducted by malware analysts and reverse engineering experts for faster verdicts on suspicious files.

Layered security powered by Reputation Engine

The Reputation Engine, while formidable in its own right, truly shines when incorporated into a comprehensive cybersecurity strategy that emphasizes defense-in-depth.

OPSWAT's MetaDefender platform is designed to safeguard file uploads with a multi-layered defense approach. It leverages a robust reputation engine alongside other complementary capabilities:

1. **Reputation Inspection:** This feature draws upon a real-world database to assess the trustworthiness of files.
2. **Static File Analysis:** Utilizing advanced algorithms, this component identifies potentially suspicious patterns and attributes within files.
3. **Dynamic Analysis with Adaptive Sandbox:** By observing files in a controlled, isolated environment, this tool can monitor their behavior in real-time, providing valuable insights into potential threats.

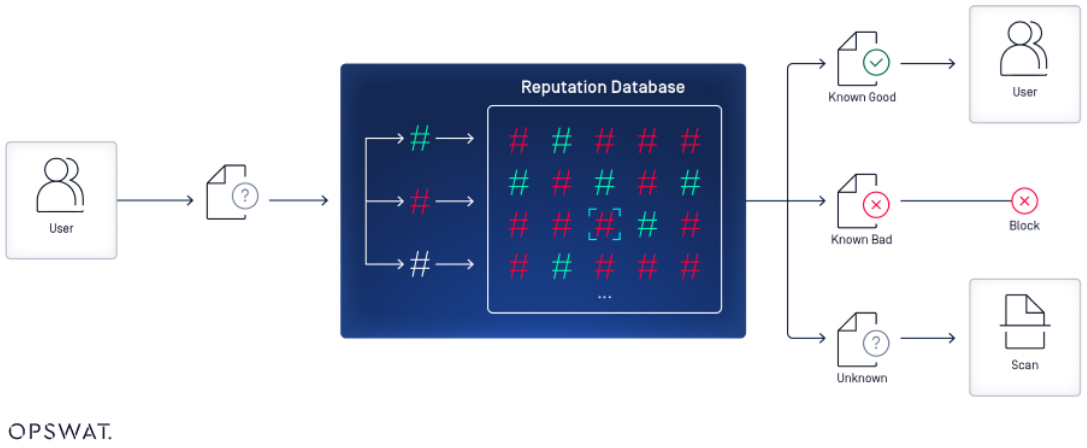
By combining these elements, MetaDefender Core offers comprehensive protection that addresses multiple vectors of attack, ensuring a more resilient defense against cyber threats.

Mechanism of Reputation Engine

The Reputation Engine serves as the primary defense for file upload security by cross-referencing against a real-world database. When users submit their files to MetaDefender Core, it generates hashes and it assess the uploaded file against the Reputation Database, our engine rapidly categorizes hashes as "Known Good," "Known Bad," or "Unknown." This capability enables users to promptly decide whether to allow, block, or scan files with other detection engines based on their predefined security policies. This prevents threats from entering your environment.

Warning

In case of Known Good or Known Bad verdict, the user can choose whether it is the final verdict for the uploaded file or user chooses to run the other engines of MetaDefender Core.

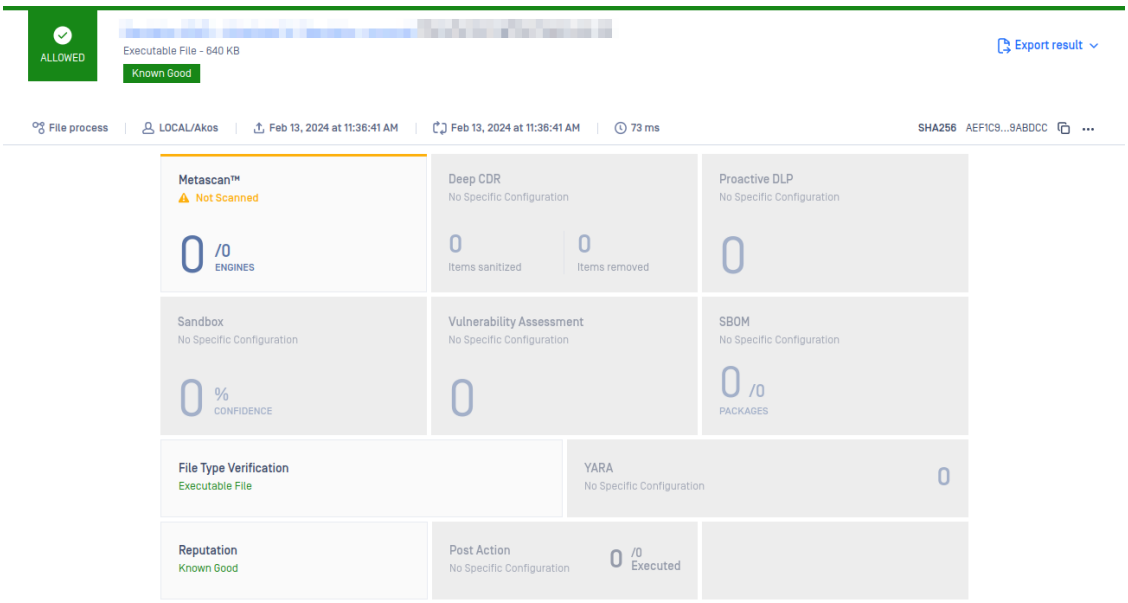


Displaying engine results

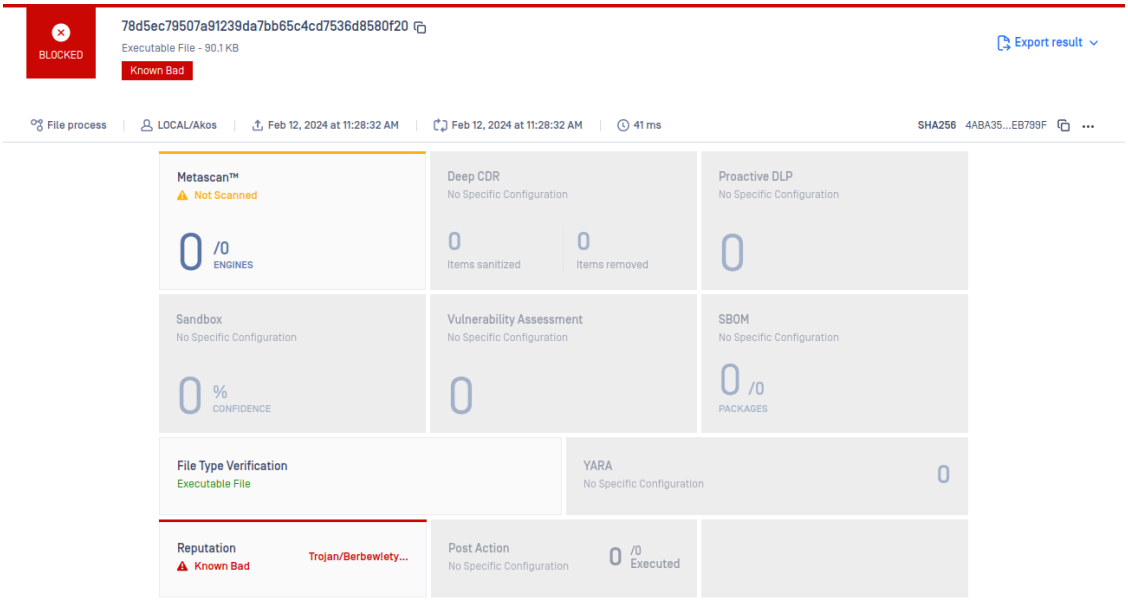
Info

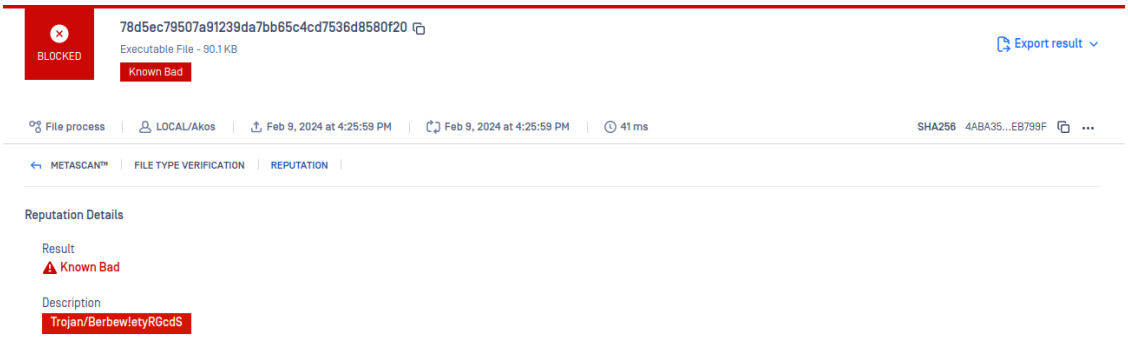
Below you will be shown examples of how the three types of file reputation results are displayed inside MetaDefender Core

Known Good: The hash matches an entry of known good files in OPSWAT's the real world database.

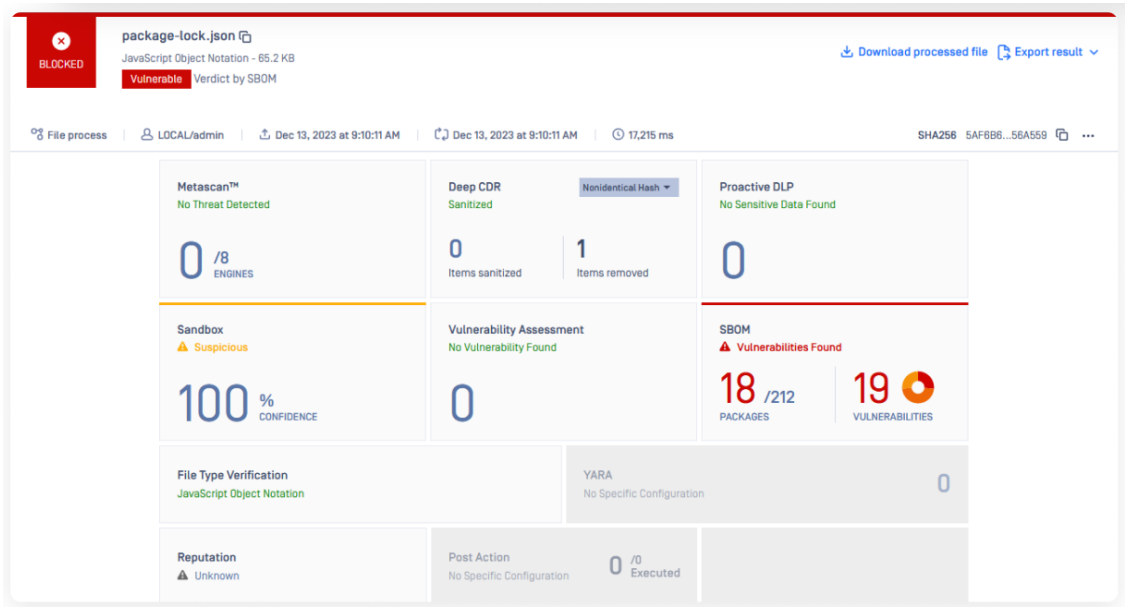


Known Bad: The hash matches an entry of known malicious files in OPSWAT's real-world database.





Unknown: There aren't any matches on this hash in OPSWAT's real-world database.



Licensing

Availability of Reputation Engine for its users:

Free of charge

All MetaDefender Core users will automatically receive **free access** to the Reputation Engine

Reporting False Detection

Enable Reputation Engine in Workflow

In order to see submitted False Detection during the next scanning session, the engine needs to be enabled under the Workflow Management.

How do I get a False Detection (False Positive or False Negative) corrected?

System requirements

Dependencies

Windows dependencies

- Visual C++ Redistributable for Visual Studio version 14.38 or higher.

Hardware dependencies

- CPU that supports AVX2 (BMI2) instruction set

Installation

Info

Reputation engine is officially supported starting from Core 5.9.0.

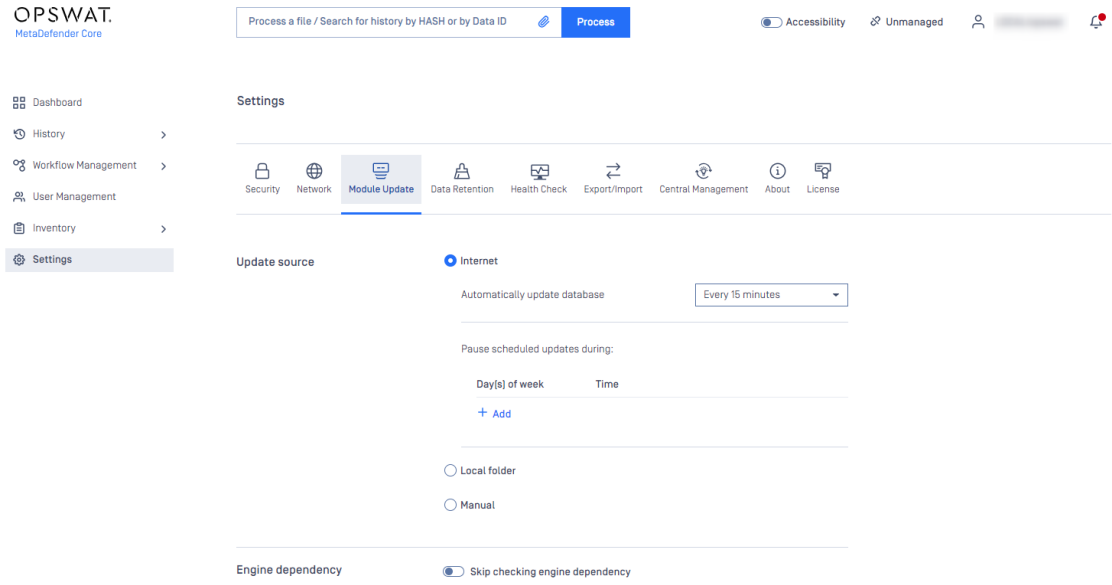
Pro Tip

The Reputation Engine supports engine updates in either online or offline environments

Engine updates

Go to **Settings > Module Update**

MetaDefender Core will automatically download all the Engines available in your license from our online update infrastructure, and download new engine versions automatically based on your **Module Update** configuration.



Offline Engine Updates

Please see the links below for more details.

How do I deploy MetaDefender Core to an offline Windows environment

How do I deploy MetaDefender Core to an offline Linux environment

Configuration

Info

Reputation Engine processing is disabled by default in the Workflow Management.

Users will need to enable it first during the Workflow Configuration.

Integration mode of Reputation Engine:

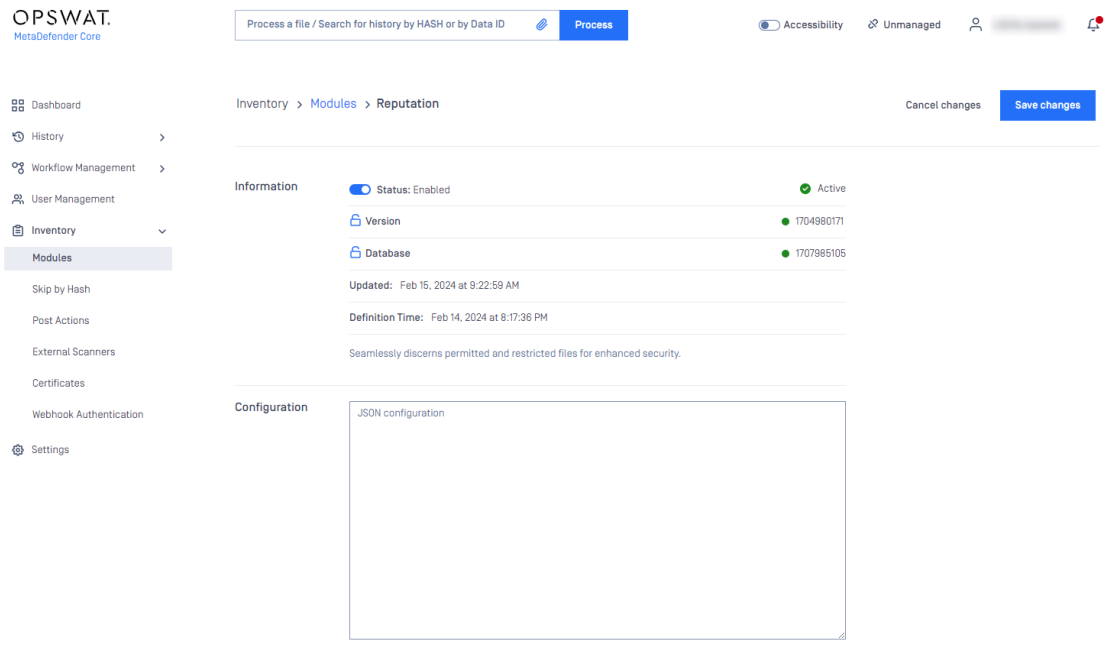
Inline: working as a part of MetaDefender Core processing workflow (real-time processing). Allowing users to block entire processing based on the outcome and decision of the Reputation Engine.

Global configuration

Go to **Inventory > Modules** and select the **Reputation** module

Enabling the engine

See details like engine version and status.



Workflow Configuration

Go to **Workflow Management > Workflows** and select your workflow and select the **File Process** module.

In the carousal below, select **Reputation**.

The Reputation Engine can be configured in the MetaDefender platform under Workflow Management section, as shown in the image below:

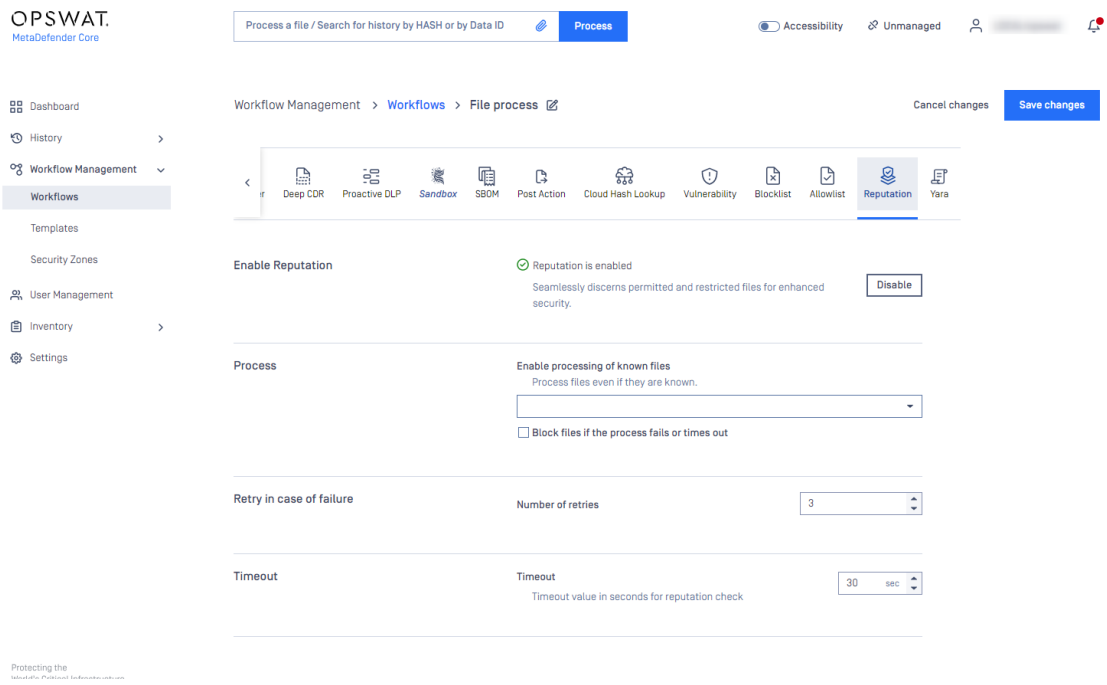
Configuration steps

- First, enable Reputation Engine in your workflow
- Optionally, modify the default workflow configuration, as described below

> **Warning**
>
> **These settings might impact performance**

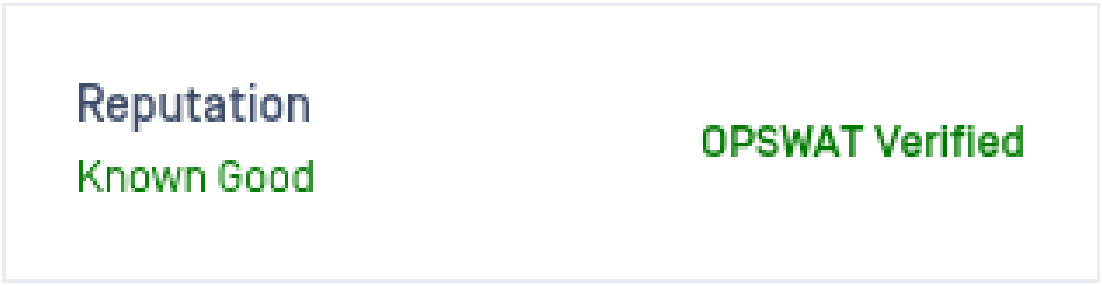
Configuration options:

- **In case of Known bad or Known good verdict, the user can choose whether it is the final verdict for the uploaded file or to run the other engines of MetaDefender Core.**
- Block files if the process fails or times out
- Determine the number of retries in case of failure
- Set timeout value for reputation check



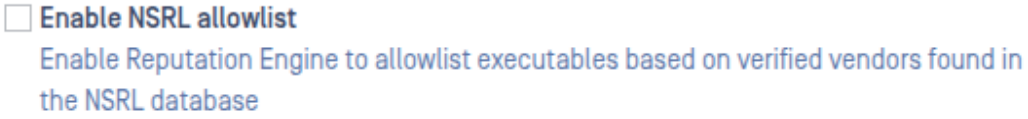
OPSWAT Verfied

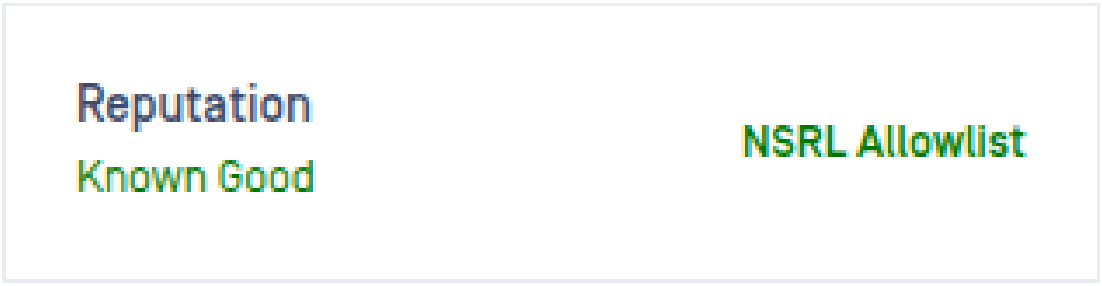
OPSWAT-verified files are included by default and cannot be changed. These files have been manually checked in-house to ensure they are safe and accurately assessed.



NSRL Allowlist

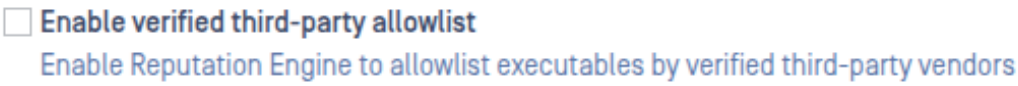
Enable to includes hashes from the official **National Software Reference Library (NSRL) hash list**, which contains widely used and trusted software hashes. The hashes have been verified and filtered by OPSWAT.

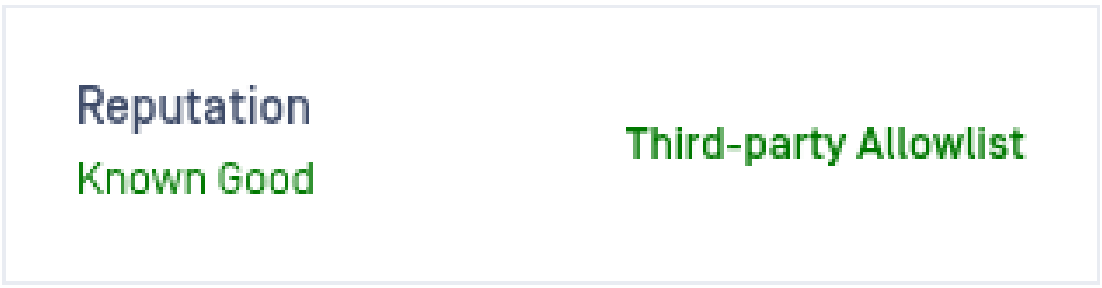




Third-party Allowlist

Enable to includes files that have been **verified and curated in-house** by OPSWAT. These files are checked against internal sources to ensure trust and reduce unnecessary scanning.





Release notes

Our release notes have been moved here.

Knowledge Center

You will find here answers for FAQs about Reputation Engine.

What is the minimum version of MetaDefender Core required for compatibility with the engine?

MetaDefender core version: **5.9.0**

How will the false detection submission impact the AV vendor's verdict?

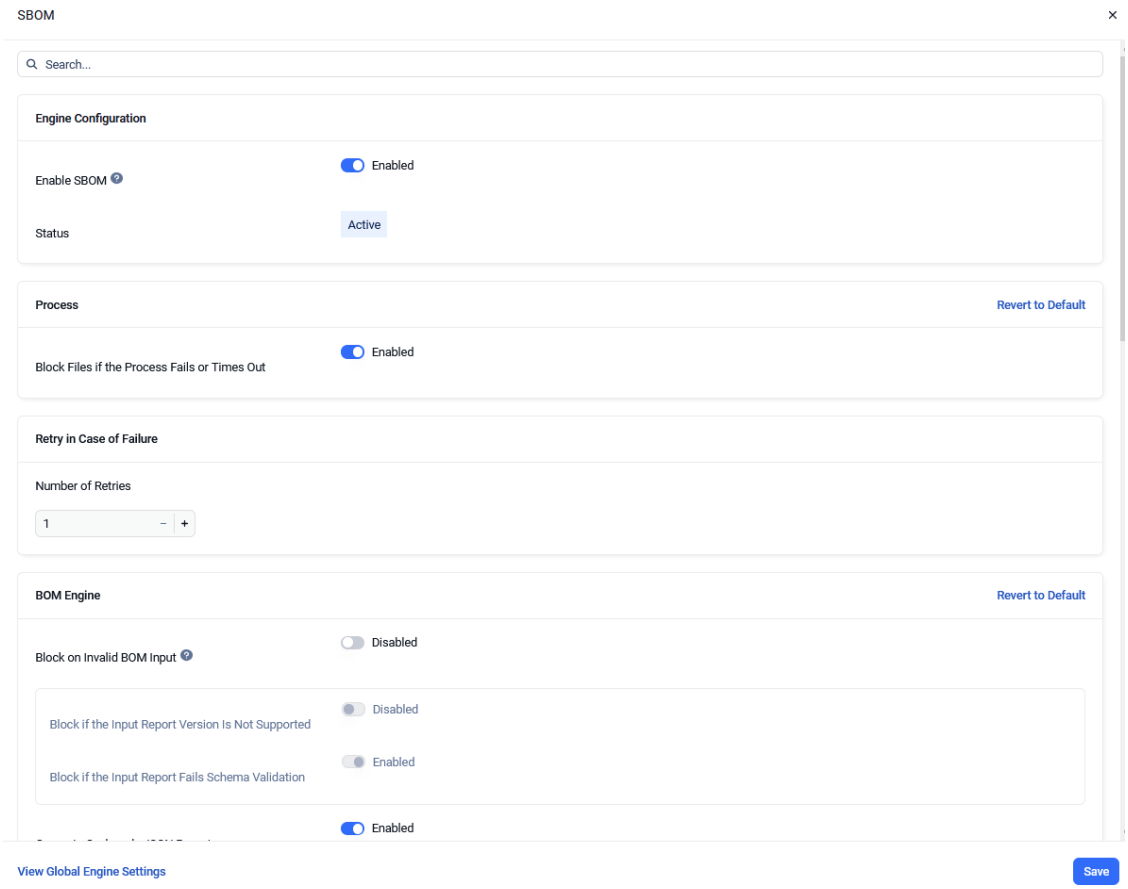
AV vendors' scan results are immutable; however, if the sample hash is incorporated into the Reputation Engine's database, the AV scan will be bypassed entirely.

Software Bill of Materials (SBOM)

OPSWAT SBOM analyzes third-party packages in source code, package archives and binaries, and container images. It generates an inventory of the components and identifies known vulnerabilities and license risks. It can also export the result as a CycloneDX or SPDX report. On MetaDefender Core 5.23 or later, the same scan can also produce a CBOM and an AIBOM.

To configure the engine, go to Workflow > [workflow name] > SBOM. Settings are grouped into Input and Output, SBOM Module, CBOM Module and AIBOM Module. Some options are shown only when the feature is licensed or the MetaDefender Core version supports it.

The engine also produces a CBOM (CBOM Module) and an AIBOM (AIBOM Module), both enabled by default on Core 5.23+. Packages are also identified in binaries (binary mode), and reports can be exported as JSON or PDF.



Support:

- Source code
- Container
- CycloneDX and SPDX reports from external tools.

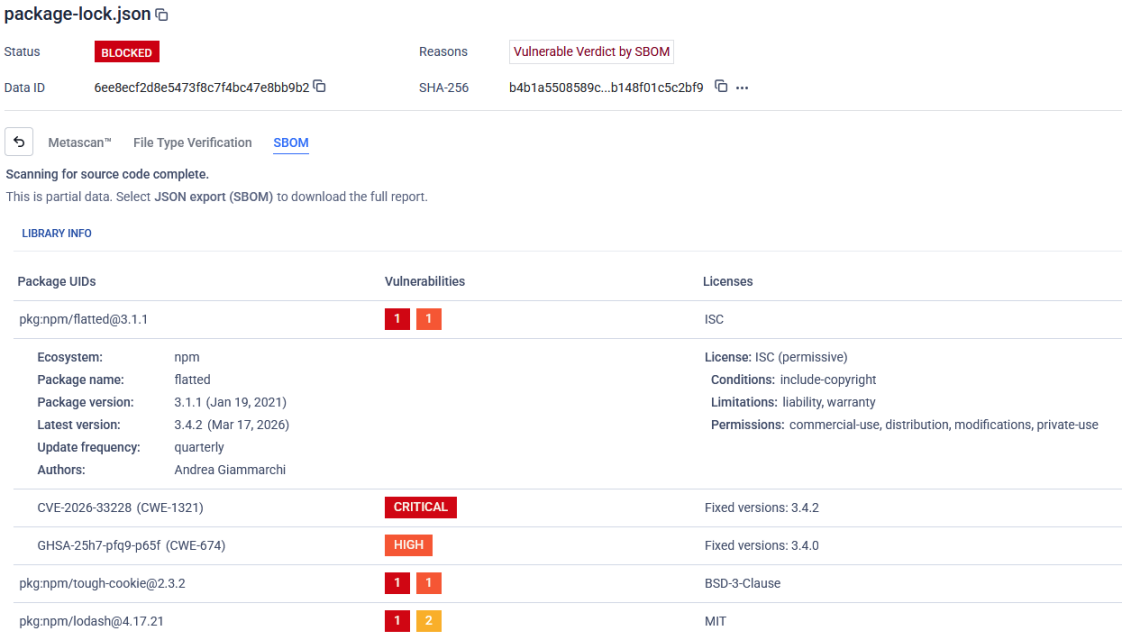
Source code

Since each programming language has its declaration files for the libraries being used, the SBOM engine only

analyzes the files with these specific filenames to avoid false positives or performance downgrades.

Programming language	Manifest and lock files	Package archives and binaries
Ruby	Gemfile.lock	.gem, .tar.gz
Python	Pipfile.lock, poetry.lock, requirements*.txt, METADATA, PKG-INFO setup.py, pyproject.toml, version.py	.whl, .tar.gz (metadata read from the archive), .egg, .zip
PHP	composer.lock, composer.json	.zip
Node.js	package.json, package-lock.json, yarn.lock, pnpm-lock.yaml	.tgz
TypeScript / CoffeeScript	package.json, package-lock.json, yarn.lock, pnpm-lock.yaml	
Java	pom.xml, pom.properties, build.gradle(.kts), settings.gradle(.kts), gradle.lockfile, libs.versions.toml	.jar, .zip, .tar.gz, source archives (src.zip, sources.zip, src.tar.gz, sources.tar.gz
Kotlin / Groovy	build.gradle(.kts), settings.gradle(.kts), gradle.lockfile, libs.versions.toml, pom.xml	.jar
Scala / Clojure	pom.xml	.jar
Go	go.mod	
Rust	Cargo.lock	
Dart	pubspec.lock	
.NET	packages.lock.json, packages.config, *.deps.json, *.nuspec, *.csproj	.nupkg, .dll
Elixir	mix.lock	

Programming language	Manifest and lock files	Package archives and binaries
Swift	Podfile.lock, Package.resolved	
C/C++	vcpkg.json, conan.lock	.exe, .dll
Julia	[Pp]roject.toml, [Mm]anifest.toml	
R	renv.lock	

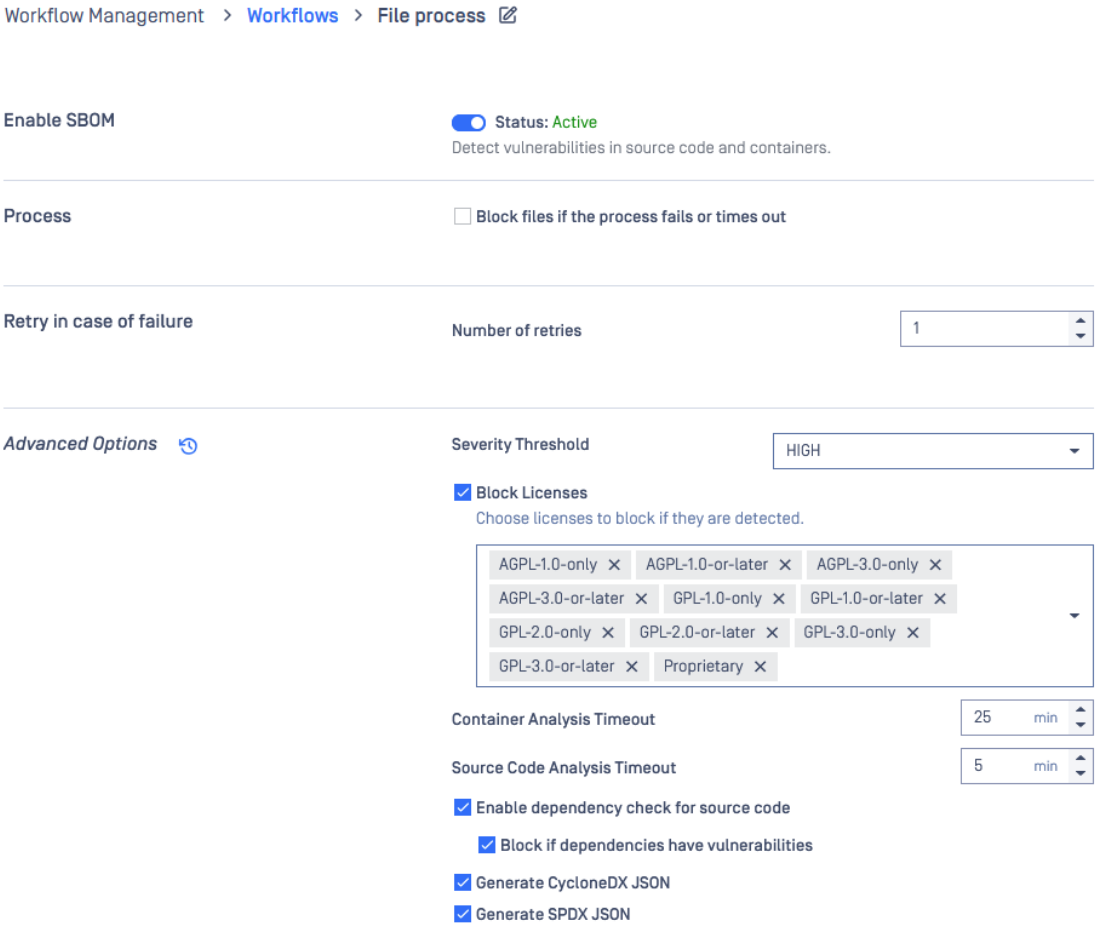


Recursive check

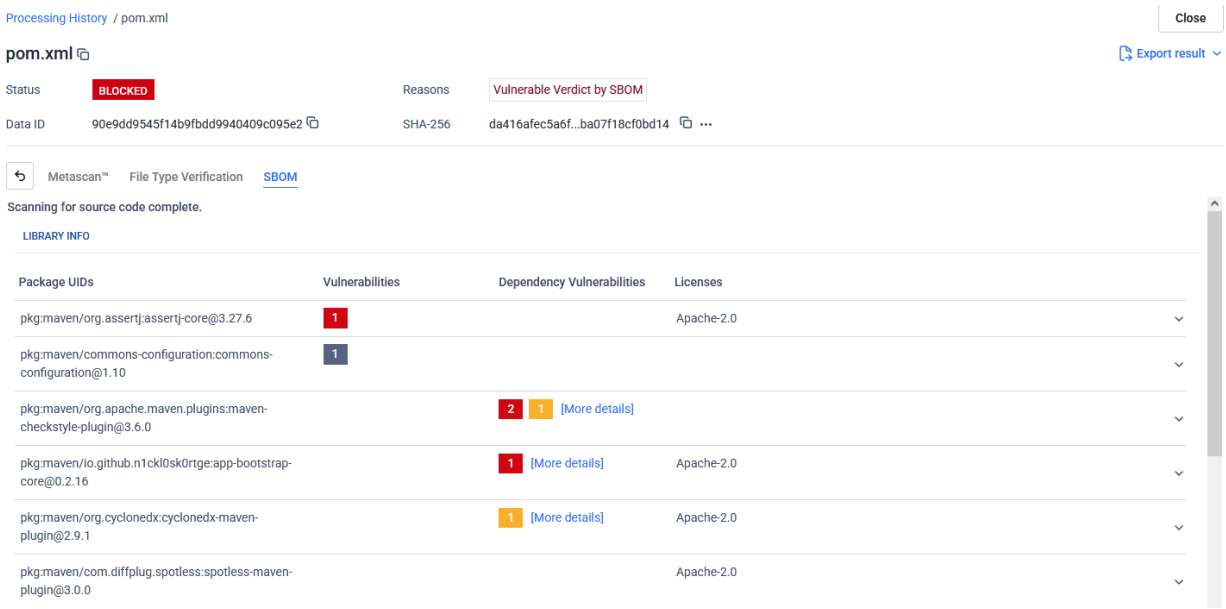
SBOM scans (nested) dependencies used in open source libraries to detect known vulnerabilities from potentially harmful packages.

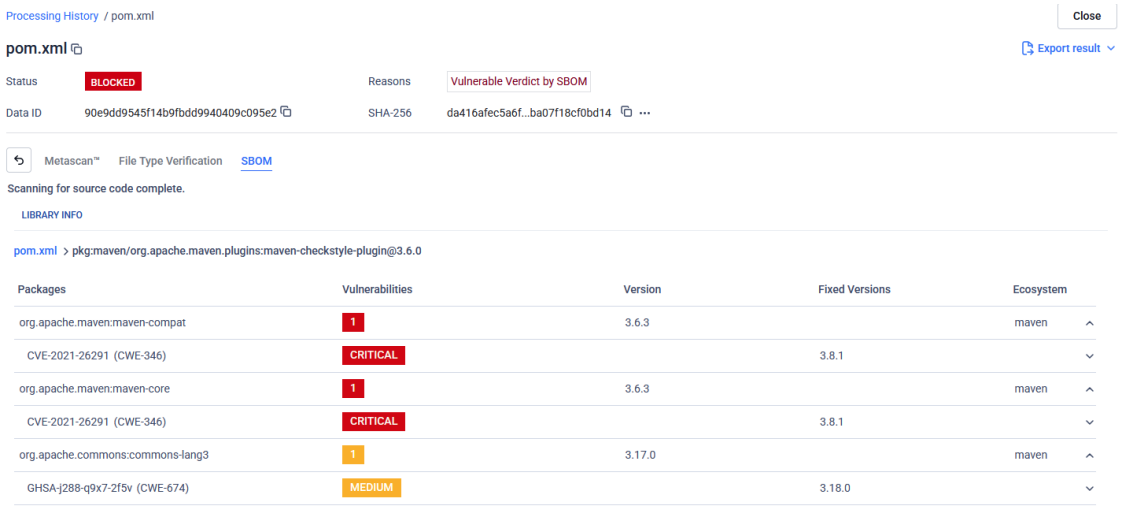
To enable recursive check: Workflow > Workflow name > SBOM > Enable dependency check for source code.

Two sub-options refine the check (both off by default):



Example scan result



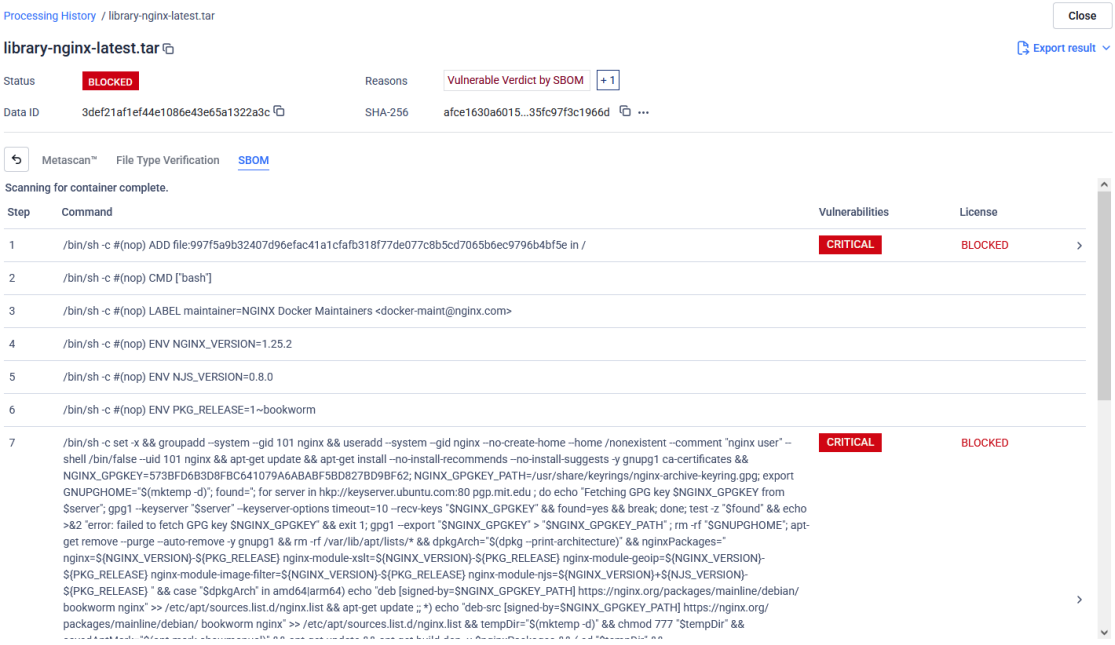


- Supported ecosystems: Python (PyPI), Node.js (npm, including TypeScript and CoffeeScript projects), Java and other Maven/Gradle languages, .NET (NuGet), Ruby, PHP. Dependency data is not available for C/C++, Go, Rust, Dart, Elixir, Swift, Julia or R.
- Dependency depth: 1. The engine lists the direct dependencies of each detected package. It does not follow them further.

Note: dependency check applies to source code only. For container images, the engine reports the operating-system packages installed in the image, not application packages.

Container image

Upload the .tar file written by docker save as is. Do not compress it. The engine also accepts a single-layer tar containing layer.tar, manifest.json and sbom_metadata.json. Only operating-system packages are reported. Application packages inside the image (npm, PyPI, Maven, …) are not scanned; to cover them, scan the application's source code or lockfile.



Supported OS package sources:

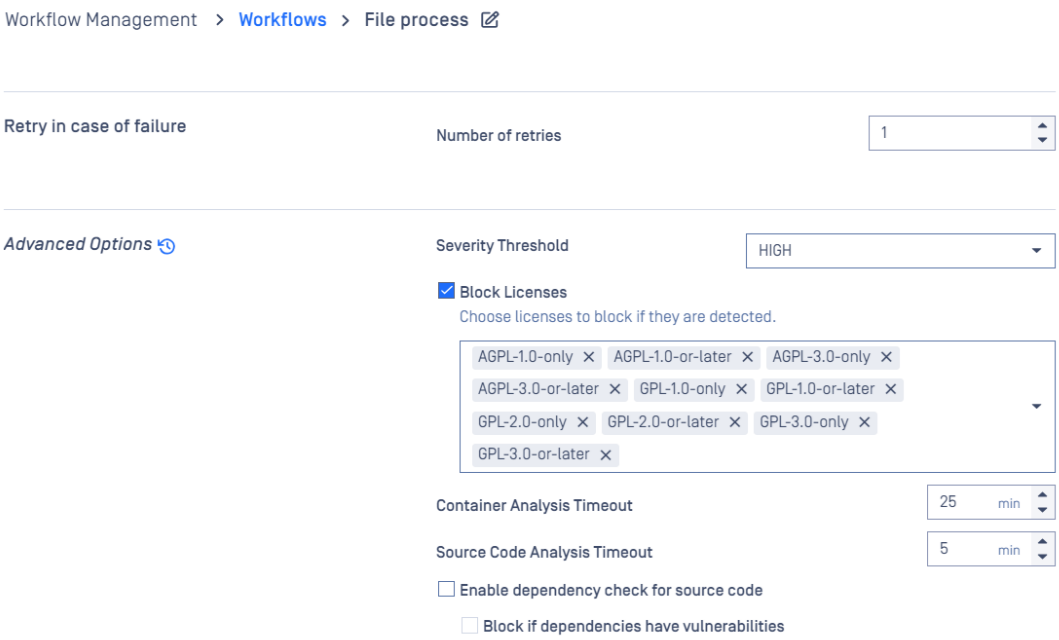
- Alpine Linux
- Red Hat Enterprise Linux
- CentOS
- Rocky Linux
- Oracle Linux
- Ubuntu
- Debian
- SUSE Linux Enterprise / openSUSE
- Amazon Linux
- AlmaLinux
- Fedora

License risk detection

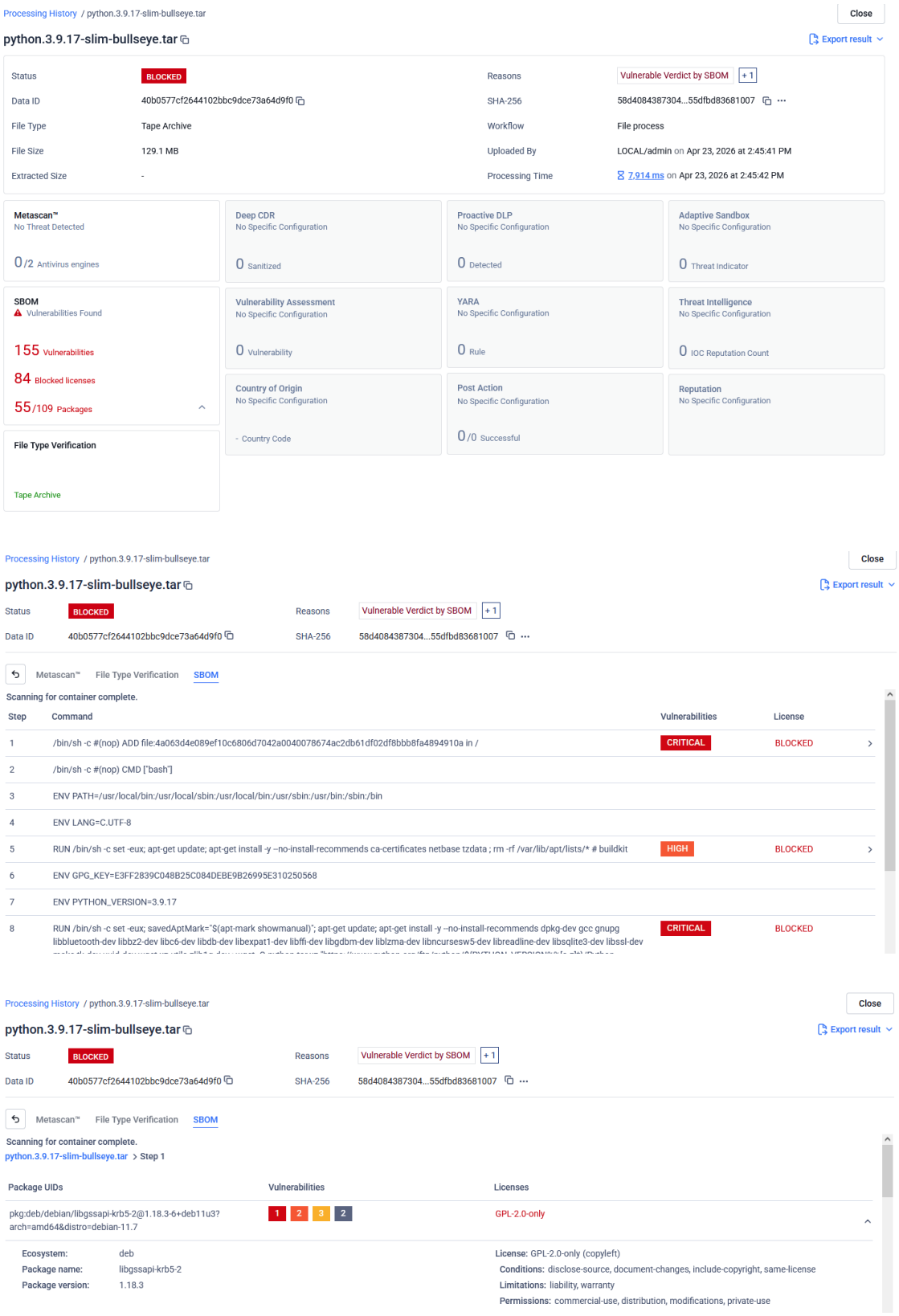
Some open-source licenses include clauses — such as copyleft provisions — that can require you to open-source your own code. Verifying that your dependencies use licenses compatible with your business model is essential to protecting your intellectual property.

The SBOM engine includes license detection that examines each open-source component detected in a scan. Beyond identifying the license type, it reports detailed license information for your open-source dependencies, helping you assess compliance obligations before they become a risk.

To block licenses, go to Workflow > [workflow name] > SBOM and enable **Block Licenses**, then choose the licenses to block. A scan that finds a blocked license is blocked, and its verdict is **License Risk Found**, unless vulnerabilities are also found, in which case the verdict is **Vulnerabilities Found**.



Process result:



License information is reported for packages from Python (PyPI), JavaScript (npm), Java (Maven), .NET (NuGet), PHP (Packagist), Ruby (RubyGems), C/C++, and identified software applications. It is also reported for operating-system packages in container images, and for license data already present in scanned CycloneDX and SPDX reports. AIBOM uses its own license list (see AIBOM).

SBOM JSON response

Source code:

- If the "Enable dependency check for source code" option is not enabled, please ignore all "dependencies" attributes

Code · json

```
{
  "final_verdict":{//the aggregated section for both direct and dependent libs
    "blocked": "<boolean>",     //true when a CVE is at or above the severity threshold, a package licence is blocked, or ("Block if dependencies have vulnerabilities" on) a dependency is vulnerable
    "verdict": "<string>",
        //<"No Vulnerabilities Found" | "Vulnerabilities Found" | "License Risk Found">
    "verdict_explanation": ["<string>"],
    "vulnerable_package_count": "<int>",     //total direct and dependent vulnerable libs
    "total_package_count": "<int>",  //total direct and dependent libs
    "vulnerability_counts":{ // total unique direct and dependent vulnerabilities
          "critical": "<int>",         //total critical CVE
      "high":         "<int>",    //total high CVE
      "medium":     "<int>",    //total medium CVE
      "low":            "<int>",    //total low CVE
      "unknown":     "<int>"       //total unknown CVE, do not know the severity
    },
    "severity": "<string>", //highest CVE severity from direct and dependent libs; omitted when no vulnerability is found
        "licensing_package_counts": {
        "allowed":             "<int>",
        "blocked":             "<int>",
        "unknown":             "<int>",
        "unsupported":         "<int>"
      }
  },
    "other_report_formats": [  // report types that could be exported
      "cyclonedx",
      "spdx"
    ],
    "overview_report": {
    "author": {
        "name": "OPSWAT Inc.",
        "tool": "BOM-<engine version>" // e.g. "BOM-5.0.1"
      },
    "scan_mode": "source_code", //source code or container
    "severity": "<string>", //the highest CVE severity from the direct libs
    "is_partial_report": "<boolean>", //if yes, please download the full SBOM report to see all packages info, by default, this JSON only returns 100 package info
    "package_count": "<int>", //total direct libs
    "vulnerable_package_count": "<int>", //total direct vulnerable libs
    "vulnerability_count": "<int>", //total direct vulnerabilities
```

```
    "vulnerability_counts":{ //direct vulnerabilities
        "critical": "<int>",
        "high":         "<int>",
        "medium":     "<int>",
        "low":             "<int>",
        "unknown":     "<int>"
      },
    "dependencies_vulnerability_count": "<int>", //total vulnerabilities from the dependent libs
    "dependencies_package_count": "<int>", //total dependent libs
    "vulnerable_dependencies_package_count": "<int>", //total vulnerability in the dependent
libs
    "dependencies_severity": "<string>", //highest CVE severity from the dependent libs
    "dependencies_vulnerability_counts":{
        "critical": "<int>",
        "high":     "<int>",
        "medium":   "<int>",
        "low":      "<int>",
        "unknown":  "<int>"
      },
    "packages": [//package details
      //the below JSON is the info of one package, including the dependencies info of the
package
    {
      "uid": "<string>",
      "ecosystem": "<string>", //e.g. npm, pypi, maven, julia, r, ...
      "group": "<string> ['lang_pkg']",
      "package_name": "<string>",
      "package_version": "<string>",
      "type": "<string>",            //component type
      "description": "<string>",
      "licenses": { ... },           //licences found for the package; {} when none
      "original_purl": "<string>",   //the package's purl
      "release_date": "<string>",
      "target": "<string>", //the input file name (e.g. pom.xml, requirements.txt, Project.toml,
renv.lock, report.spdx.json)
      "vulnerabilities": [
        {
          "aliases": ["<string>"],
          "id": "<string>",
          "severity": "<string>",
          "source": "<string>", //vulnerability data source; for SPDX 3.0.1 report scans,
results are merged from both the report and SBOM database
          "cwes": ["<string>"],
          "epss": {
                "date": "<string>",
                "percentile": "<float>",
                "score": "<float>"
             },
          "ratings": [
                {
                  "method": "<string>",
```

```
                  "score": "<float>",
                  "severity": "<string>",
                  "source": {
                    "name": "<string>"
                  }
                }
          ],
          "fixed_versions": "<string>", //all fixed version > installed version
          "type": "<string>, ['vuln', 'potential_vuln']" //potential_vuln is returned when the
libs in the input is not clearly defined, i.e: http-client: >2.0
        }
      ],
      "vulnerability_counts": {
        "critical": "<int>",
        "high":         "<int>",
        "low":             "<int>",
        "medium":     "<int>",
        "unknown":     "<int>"
      },
      "archive": "<boolean>",
      "authors": "<string>",
      "latest": {
         "release_date": "<string>",
         "version": "<string>"
          },
      "update_frequency": {
            "label": "<string>"
          },
      "url": "<string>",
      "dependencies_package_count": "<int>",// all detected dependencies for this package
      "dependencies":
      [ //empty if disable
        {
          "package_name": "<string>", // only show if there is a vulnerability
          "package_version": "<string>",
          "ecosystem": "<string>",
          "vulnerabilities": [
          {
            "id": "<string>",
            "severity": "<string>",
            "cwes": ["<string>"],
            "source": "<string>",
            "fixed_versions": "<string>",
            "type": "<string>", ["vuln", "potential_vuln"]
          }
          ],
          "vulnerability_counts": {
            "critical": "<int>",
            "high":         "<int>",
            "medium":     "<int>",
            "low":             "<int>",
```

```
            "unknown":     "<int>"
          }
        }
      ],
      "dependencies_severity": "<string>",
      "dependencies_vulnerability_counts": { // the aggregated result of the vulnerable libs,
empty if disable or no dependencies detection
          "critical": "<int>",
          "high":          "<int>",
          "low":             "<int>",
          "medium":     "<int>",
          "unknown":     "<int>"
      }
    }
    ]
  }
}
```

Container:

Code · json

```
{
  "final_verdict":{
    "blocked": "<boolean>", //true when a CVE is at or above the severity threshold or a package
licence is blocked
    "verdict": "<string>",  //<"No Vulnerabilities Found" | "Vulnerabilities Found">
    "verdict_explanation": ["<string>"],
    "vulnerable_package_count": "<int>",
    "total_package_count": "<int>",
    "vulnerability_counts":{
      "critical": "<int>",
      "high": "<int>",
      "medium": "<int>",
      "low": "<int>",
      "unknown": "<int>"
    },
     "licensing_package_counts": {
      "allowed": "<int>",
      "blocked": "<int>",
      "unknown": "<int>",
      "unsupported": "<int>"
    },
    "severity": "<string>" //omitted when no vulnerability is found
  },
  "scan_mode": "container",
  "os_info": {
    "os_name": "<string>",
    "os_version": "<string>"
  },
```

```
  "steps": [
    {
      "step": "<int>",
      "command": "<string>",
      "sha256": "<string>",
      "severity": "<string>",
      "vulnerability_count": "<int>",
      "packages": [
        {
          "arch": "<string>",
          "ecosystem": "<string>",
          "epoch": "<string>",// epoch of the OS package (e.g. "1"); empty string if not
applicable
          "group": "<string>", // ["os_pkg"]
          "os_name": "<string>",
          "os_version": "<string>",
          "package_name": "<string>",
          "package_version": "<string>", // base version (e.g. "3.0.7"); full version is
epoch:version-release
          "release": "<string>",       // release of the OS package (e.g. "18.el9_2"); empty
string if not applicable
          "vulnerabilities": [
            {
              "id": "<string>",
              "severity": "<string>",
              "source": "<string>",
              "cwes": ["<string>"],
              "fixed_versions": "<string>"
            }
          ],
          "vulnerability_counts": {
            "critical": "<int>",
            "high": "<int>",
            "low": "<int>",
            "medium": "<int>",
            "unknown": "<int>"
          }
        },
        {
          "ecosystem": "<string>",
          "group": "<string>",         // ["lang_pkg"]
          "package_name": "<string>",
          "package_version": "<string>",
          "vulnerabilities": [
            {
              "id": "<string>",
              "severity": "<string>",
              "source": "<string>",
              "cwes": ["<string>"],
              "fixed_versions": "<string>"
            }
```

```
          ],
          "vulnerability_counts": {
            "critical": "<int>",
            "high": "<int>",
            "low": "<int>",
            "medium": "<int>",
            "unknown": "<int>"
          }
        }
      ]
    }
  ]
}
```

Note: severity is present only when at least one vulnerability is found. licensing_package_counts is always present.

Cryptographic Bill of Materials (CBOM)

Applies to: *SBOM engine 5.0.0+ and MetaDefender Core 5.23+.*

What a CBOM is

A Cryptographic Bill of Materials (CBOM) is an inventory of the cryptography a file actually contains: the algorithms it uses, the libraries that implement them, the certificates it carries, the protocols it negotiates, and the key material it references.

An SBOM answers *which third-party packages are in this artifact*. A CBOM answers a different question: *which cryptography am I relying on, and will it survive a quantum adversary?* Government post-quantum mandates such as NSA CNSA 2.0, NIST IR 8547 and the EU Cyber Resilience Act all start with the same requirement: produce an inventory before planning a migration. The CBOM is that inventory.

Asset types

Each cryptographic item CBOM finds is reported as an asset of one of five types:

Type	What it covers
Algorithm	A cryptographic algorithm, such as AES-GCM, ECDSA or MD5, with details like mode, padding, curve and key length
Library	A cryptographic library the file depends on or links against, such as OpenSSL, BouncyCastle or PyCryptodome
Certificate	An X.509 certificate, including its signature and

Type	What it covers
	public-key algorithms
Protocol	A negotiated protocol such as TLS, including its cipher suites
Related Crypto Material	Keys and similar material that use an algorithm rather than being one, such as an RSA private key

Supported detection

CBOM finds cryptography in four kinds of input: key files, package manifests, source code, and binaries. The screenshots below show a sample result for each.

Supported input files

1. Source code and compile-time configuration

NAME	Files
PYTHON	`*.py`
C / C++	`*.c` `*.cc` `*.cpp` `*.cxx` · `*.h` `*.hpp` `*.hh` `*.hxx`
JAVA	`*.java`
GO	`*.go`

2. Declared dependencies and build configuration

NAME	Files
CMAKE	`CMakeLists.txt`, `*.cmake`
MAKE	`Makefile`, `GNUmakefile`, `*.mk`
PYTHON	`requirements.txt`, `setup.py`, `pyproject.toml`
MAVEN / GRADLE	`pom.xml` · `*.gradle`, `*.gradle.kts`

NAME	Files
GO	`go.mod`
NPM	`package.json`
RUBY	`Gemfile`
RUST	`Cargo.toml`
PHP	`composer.json`
CONAN	`conanfile.txt`
VCPKG	`vcpkg.json`
MESON	`meson.build`
SCONS	`SConstruct`

3. Runtime configuration, material at rest, document import, binaries

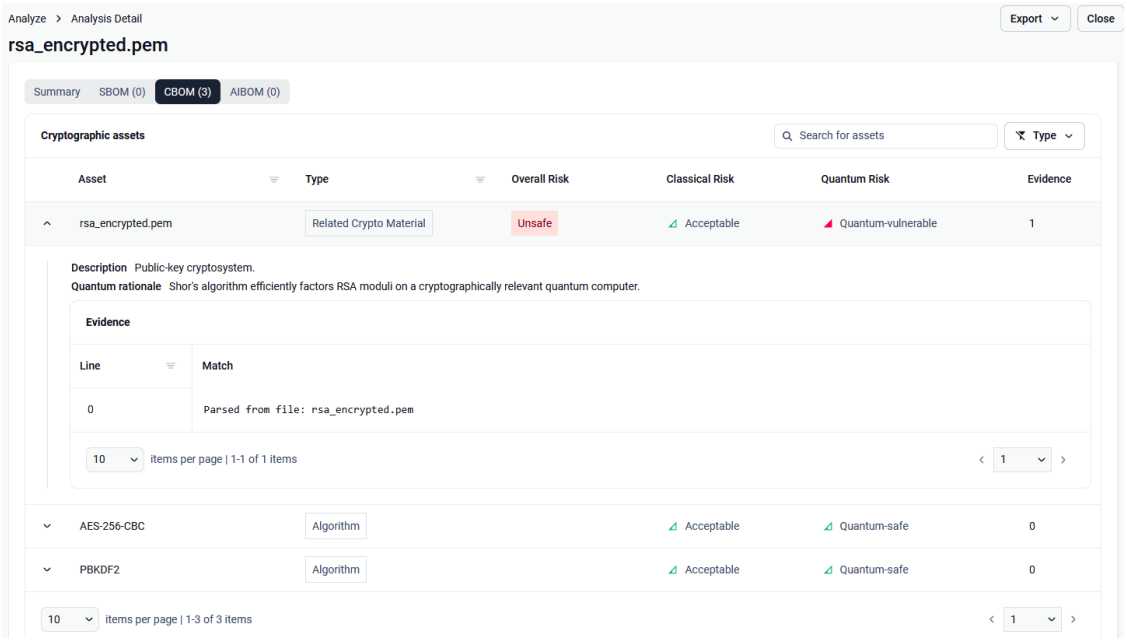
NAME	Files
CONFIG	`*.ini`, `*.cfg`, `*.conf`, `*.properties`: cipher suites, version pins, cert and key paths
CERTIFICATE	`*.crt`, `*.cer`, `*.pem`, `*.der`, PEM chains and CA bundles `*.p7b`, `*.p7c` (PKCS#7 bundles)
KEY	`*.key` · `*.pub`
KEY CONTAINER	`*.p12`, `*.pfx`, `*.jks`, `*.jceks`, `*.bcfks`, `*.keystore` · encrypted PEM private keys
WINDOWS PE	`*.exe`, `*.dll`, `*.sys` and any PE image: Authenticode signing certificates

4. Infrastructure as code

NAME	Files
TERRAFORM	`*.tf`, `*.tfvars`: includes the ALB `ssl_policy`
CLOUDFORMATION	`*.yaml`, `*.yml`, `*.template`, `*.json` with `AWSTemplateFormatVersion` or `AWS::`, up to 1 MB

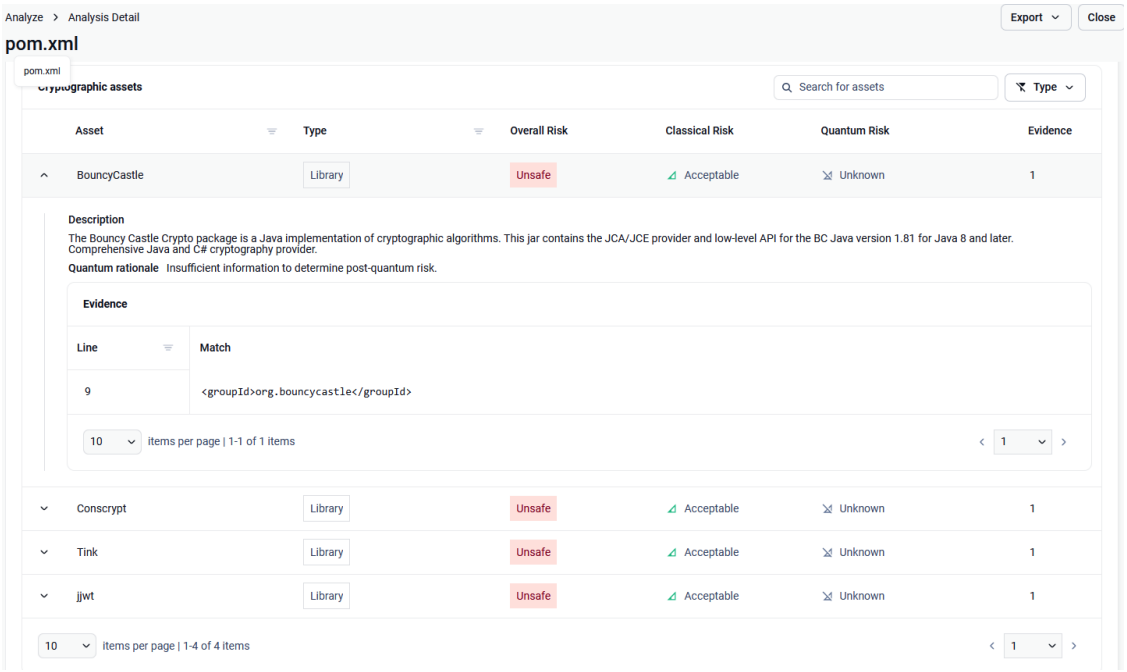
Key files

CBOM parses private and public key files such as PEM-encoded keys. It reports the key itself as related crypto material (for example an RSA key) and the algorithms protecting it: for an encrypted key, the cipher (such as AES-256-CBC) and the key-derivation function (such as PBKDF2).



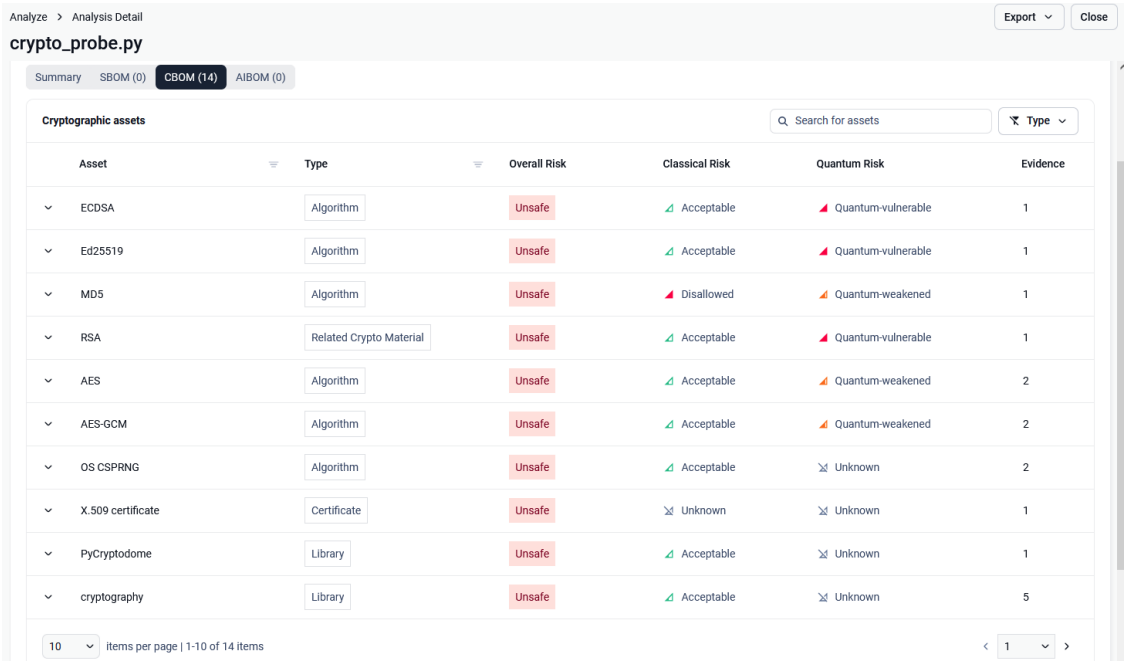
Cryptographic libraries in package manifests

CBOM reads package management files such as `pom.xml` and reports the cryptographic libraries they declare, for example BouncyCastle, Conscrypt, Tink and jjwt. The evidence shows the line where each dependency is declared.



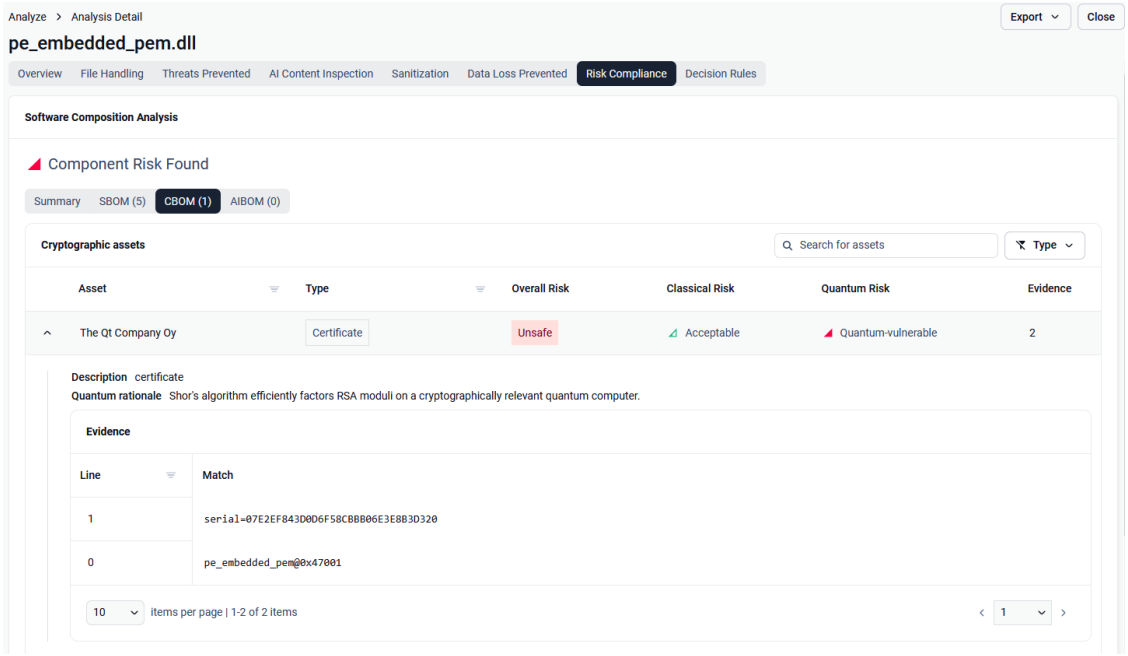
Cryptography used in source code

CBOM scans source code for cryptographic API calls and imports. It reports the algorithms in use (such as ECDSA, Ed25519, MD5, AES-GCM), the libraries that provide them (such as `cryptography` and PyCryptodome), and any keys or certificates the code handles, each with its own classical and quantum risk.



Certificates inside binaries

CBOM extracts certificates embedded in **binary files**: from the Windows PE attribute certificate table (Authenticode signatures) and from PEM blocks embedded in the file. The evidence shows the byte offset where each certificate was found, for example `pe_embedded_pem@0x47001`.



Crypto risk classification

Each asset is graded on two independent axes, so you can see both what is weak today and what needs to migrate to post-quantum cryptography.

Axis	Grades	Meaning
Classical risk	Acceptable · Deprecated · Disallowed · Unknown	Strength against today's (non-quantum) attacks
Quantum risk	Quantum-safe · Quantum-weakened · Quantum-vulnerable · Unknown	Resistance to an attacker with a cryptographically relevant quantum computer

An asset's **Overall Risk** is **Safe** only when it is both classically Acceptable **and** Quantum-safe; Overall Risk is Safe, Unknown or Unsafe. Assets without enough grading data are Unknown, not Unsafe.

For assets that are not quantum-safe, CBOM recommends a post-quantum replacement. Some assets need more than one: RSA, for example, is replaced by ML-KEM (FIPS 203) for key establishment and ML-DSA (FIPS 204) for signatures.

CBOM verdict

CBOM gives each scan one of three verdicts: **Cryptography Found**, **Risk Cryptography Found**, **No Cryptography Found**. The file is blocked when its risk meets the *Block on Cryptographic Risk* setting (default: None, which never blocks).

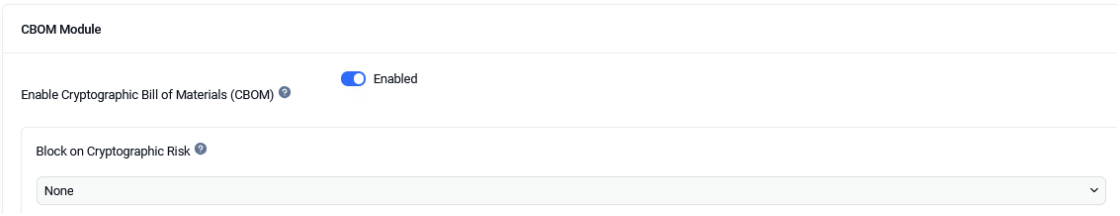
For the full result structure, see CBOM JSON response.

Export

CBOM results can be exported on their own or together with the other BOM types, in JSON or PDF. This lets you route a cryptography report to the CISO and post-quantum migration team without the surrounding software inventory.

Enabling CBOM

CBOM is enabled by default. To configure it, go to **Workflow** > **Workflow name** > **CBOM Module**. Requires MetaDefender Core 5.23 or later.



CBOM JSON response

JSON structure

Inside a component, fields that have no value are **omitted entirely**; they are never returned as `null`, `""` or `[]`. Treat an absent key as "not known", not as "empty". The top-level `components` array is always present and is `[]` when no cryptographic assets were found.

Code · json

```
"cbom_info": {
  "final_verdict": {                    //scan-level summary; advisory only
    "verdict": "<string>",              //"No Cryptography Found" | "Risk Cryptography Found"
| "Cryptography Found"
    "verdict_explanation": ["<string>"],  //one entry, equal to "verdict"
    "blocked": <bool>,                  //true when "Block on Cryptographic Risk" is met;
always false with the default (None)
    "severity": "INFO",                 //always "INFO"
    "total_asset_count": <int>,         //number of entries in "components"
    "asset_counts": {                   //assets per type; all keys always present, zeros
included
      "algorithm": <int>,
      "certificate": <int>,
      "protocol": <int>,
      "related_crypto_material": <int>,
      "library": <int>
    },
    "risk_summary": {
      "overall":   { "safe": <int>, "unsafe": <int>, "unknown": <int> },
                  //safe + unsafe + unknown = total_asset_count
      "classical": { "counts": { "acceptable": <int>, "deprecated": <int>, "disallowed": <int>,
```

```
"unknown": <int> } },
      "quantum":   { "counts": { "quantum-safe": <int>, "quantum-weakened": <int>, "quantum-
vulnerable": <int>, "unknown": <int> } }
    }
  },
  "components": [
    {
      "uid": "<string>",                    //unique id of the asset: crypto/<type>/<name>@<hash>
      "bom_ref": "<string>",                //same value as "uid"
      "name": "<string>",
      "type": "cryptographic-asset",
      "asset_type": "<string>",             //same value as crypto_properties.asset_type
      "crypto_properties": {
        "asset_type": "<string>",
            //"algorithm" | "certificate" | "protocol" | "related_crypto_material" | "library"
        "description": "<string>",
        "<asset_type>_properties": { },     //see "Type-specific properties"; not present for
libraries
        "risk": {
          "overall": "<string>",            //"safe" | "unsafe" | "unknown"
          "classical": {
            "level": "<string>",            //"acceptable" | "deprecated" | "disallowed" |
"unknown"
            "strength": "<string>",
            "safe": <boolean>
          },
          "quantum": {
            "level": "<string>",            //"quantum-safe" | "quantum-weakened" | "quantum-
vulnerable" | "unknown"
            "is_pqc": <boolean>,
            "safe": <boolean>,
            "rationale": "<string>",
            "recommended_replacement": [    //present only when a replacement is recommended
              { "algorithm": "<string>", "nist_standard": "<string>" }
            ]
          }
        },
        "evidences": [                      //where the asset was found
          { "line": <int>, "match": "<string>" }
        ]
      }
    }
  ]
}
```

Type-specific properties

Each component carries one `*_properties` object inside `crypto_properties`, selected by `asset_type`:

asset_type	Object	Fields
algorithm	algorithm_properties	primitive, algorithm_family, parameter_set_identifier, elliptic_curve, mode, padding, execution_environment, implementation_platform, certification_level[], crypto_functions[], classical_security_level, nist_quantum_security_level, reference_primitive, category, key_lengths[], min_key_length, output_size, usage, performance, conditions[], known_weaknesses[], disallowed_for[]
certificate	certificate_properties	subject_name, issuer_name, serial_number, certificate_format, certificate_extension, description, not_valid_before, not_valid_after, public_key_length, signature_algorithm_ref, public_key_ref, public_key_algorithm_ref, signature_algorithm_details, public_key_algorithm_details, fingerprint, certificate_state[], certificate_extensions[], related_cryptographic_assets[] (plus lifecycle dates when imported)
protocol	protocol_properties	type, version, cipher_suites[] {name, algorithms[], identifiers[], tls_groups[], tls_signature_schemes[]}, crypto_ref_array[], ikev2_transform_types, related_cryptographic_assets[], usage, performance, conditions[], known_weaknesses[]
related_crypto_material	related_crypto_material_properties	type, id, state, algorithm_ref, size, format, value, secured_by {mechanism, algorithm_ref}, fingerprint, related_cryptographic_assets[], key_lengths[], m

asset_type	Object	Fields
		in_key_length (plus lifecycle dates when imported)
library	*(none)*	Library details are at the crypto_properties level: vendor, library_type, library_supports[], notes[], detection_kind

For related crypto material, algorithm_ref holds the name of the algorithm the key or secret belongs to (for example Ed25519).

For certificates, signature_algorithm_details and public_key_algorithm_details are full algorithm objects, each with its own risk block. A certificate takes the weaker of its two algorithms' risk on each axis, so these objects explain why it was graded as it was. public_key_algorithm_details.detected_key_length is the key length measured from the certificate itself.

For libraries, detection_kind tells how the library was found: declared-dependency (named in a package manifest) or link-flag (only linked in a build script, so no version or package identity is available).

Field notes

Risk

- crypto_properties.risk.overall is what the UI shows as Overall Risk: safe, unknown or unsafe. Unknown assets are not counted as unsafe. An asset is unknown when there is not enough information to grade it on one of the two axes (classical, quantum).
- When a level is unknown, the matching safe value is false. Read it as "not confirmed safe", not as "found unsafe".
- quantum.is_pqc names a mechanism and is distinct from the quantum-safe level: AES-256 is quantum-safe without being a post-quantum algorithm.
- quantum.recommended_replacement is an **array**, because there can be more than one successor. RSA, for example, needs ML-KEM (FIPS 203) for key establishment *and* ML-DSA (FIPS 204) for signatures; applies_to says which use each entry covers. If you need a single answer, read the first element.

Evidence

evidences[].line is the line where the asset was found and match is the matching text.

Certificates extracted from binary files are byte-offset findings, not line-based ones. Their entries carry line: 0 by design, and the location is given in the match string:

- pe_attribute_certificate_table@0x<offset>: a certificate from the Windows PE attribute certificate table.
- pe_embedded_pem@0x<offset>: a PEM block embedded in the file.

Verdict

`verdict` is evaluated in this order:

1. `No Cryptography Found`: `total_asset_count` is 0.

2. `Risk Cryptography Found`: risk_summary.overall.unsafe is greater than 0 and the workflow's CBOM Module > Block on Cryptographic Risk is set to Unsafe. blocked is then true.

3. `Cryptography Found`: all other cases, including unsafe assets under the default threshold None, which neither reports a risk nor blocks.

> ***Note:*** *`No Cryptography Found` is returned both when an analyzed file yields no cryptographic assets and when the file type is not covered by CBOM analysis. The verdict does not distinguish between these two cases. "Verification Failed" is no longer returned. If CBOM cannot read its input, cbom_info carries err_code and err_details. If CBOM is disabled in the workflow, cbom_info is {}.*

Examples

A private key file (default threshold None)

Code · json


```
"cbom_info": {
  "final_verdict": {
    "verdict": "Cryptography Found",
    "verdict_explanation": ["Cryptography Found"],
    "blocked": false,
    "severity": "INFO",
    "total_asset_count": 1,
    "asset_counts": { "algorithm": 0, "certificate": 0, "protocol": 0,
"related_crypto_material": 1, "library": 0 },
    "risk_summary": {
      "overall":   { "safe": 0, "unsafe": 1, "unknown": 0 },
      "classical": { "counts": { "acceptable": 1, "deprecated": 0, "disallowed": 0, "unknown":
0 } },
      "quantum":   { "counts": { "quantum-safe": 0, "quantum-weakened": 0, "quantum-vulnerable":
1, "unknown": 0 } }
    }
  },
  "components": [
    {
      "uid": "crypto/key/key_ed25519_private.key@sha256:<hash>",
      "bom_ref": "crypto/key/key_ed25519_private.key@sha256:<hash>",
      "name": "key_ed25519_private.key",
      "type": "cryptographic-asset",
      "asset_type": "related_crypto_material",
      "crypto_properties": {
        "asset_type": "related_crypto_material",
        "description": "Modern signature scheme over Curve25519.",
```

```
        "related_crypto_material_properties": {
          "type": "private-key",
          "algorithm_ref": "Ed25519",
          "key_lengths": [256]
        },
        "risk": {
          "overall": "unsafe",
          "classical": { "level": "acceptable", "strength": "strong", "safe": true },
          "quantum": {
            "level": "quantum-vulnerable",
            "is_pqc": false,
            "safe": false,
            "rationale": "Edwards-curve discrete logarithm broken by Shor's algorithm.",
            "recommended_replacement": [ { "algorithm": "ML-DSA", "nist_standard": "FIPS
204" } ]
          }
        },
        "evidences": [ { "line": 1, "match": "Parsed from file: key_ed25519_private.key" } ]
      }
    }
  ]
}
```


With the threshold set to Unsafe, the same file returns "Risk Cryptography Found" and "blocked": true.

A library declared in a Maven `pom.xml` (one component shown)

Code · json


```
{
  "uid": "crypto/library/Tink@<hash>",
  "bom_ref": "crypto/library/Tink@<hash>",
  "name": "Tink",
  "type": "cryptographic-asset",
  "asset_type": "library",
  "crypto_properties": {
    "asset_type": "library",
    "description": "Misuse-resistant cryptography library with opinionated key management.",
    "vendor": "Google",
    "library_type": "crypto-toolkit",
    "detection_kind": "declared-dependency",
    "library_supports": ["AES-GCM", "AES-SIV", "ChaCha20-Poly1305", "ECDSA", "Ed25519", "HKDF"],
    "notes": ["Exposes intents (AEAD, signing) rather than raw primitives, which removes most
parameter-choice mistakes"],
    "risk": {
      "overall": "unknown",
      "classical": { "level": "acceptable", "strength": "strong", "safe": true },
      "quantum": {
```

```
          "level": "unknown",
          "is_pqc": false,
          "safe": false,
          "rationale": "Insufficient information to determine post-quantum risk."
        }
      },
      "evidences": [ { "line": 14, "match": "<groupId>com.google.crypto.tink</groupId>" } ]
    }
  }
```


AI Bill of Materials (AIBOM)

What an AIBOM is

An AI Bill of Materials (AIBOM) is an inventory of the AI and machine-learning model files in a scan: what each model file is, what its own bytes declare about it, and which public repositories are known to publish exactly those bytes.

An SBOM answers *which third-party packages are in this artifact*. An AIBOM answers the provenance question an SBOM cannot: *this model file arrived in our pipeline — what is it, where does it come from, and under what license?*

How it works

AIBOM reads two things: the bytes of the scanned file, and an **offline model database** that ships with the engine. It identifies a model by the SHA-256 of its bytes and looks that digest up in the database.

- **No network access.** The lookup runs entirely against the local database, so an air-gapped deployment gets the same result as a connected one.
- **No risk rating.** AIBOM reports what it read and what it looked up. It makes no claim about whether a model is safe to load.
- **Works without a match.** When a file matches no known repository, AIBOM still reports what the file's own header declares, such as format, parameter count and architecture.

Supported model formats

AIBOM scans a file only when its extension **and** the file type MetaDefender Core detects from its content both match a row below. A file that matches no row reports **No AI Components Found**.

Format	Files
GGUF	`.gguf`, `.gguf-split-a`, `.gguf.part1of<N>` (first part only), `.lora`, `.mmproj`, `.oasr`
SafeTensors	`.safetensors`, `.sft`, `.st`, `.q4nx`
ONNX	`.onnx`, `.axmodel`, `.pt` (ONNX content only)

Format	Files
Keras	`.keras`
PyTorch (TorchScript)	`.jit`, `.ptl`, `.torchscript`
PyTorch 2 export	`.pt2`
NVIDIA NeMo	`.nemo`
llamafile	`.llamafile`
CycloneDX AIBOM	`.json` (CycloneDX 1.5 or later)

AIBOM also reads CycloneDX JSON reports with `specVersion` 1.5 or later (the version that introduced the `machine-learning-model` component type). Each `machine-learning-model` component, with its `modelCard`, becomes an AIBOM component. The AIBOM database is not needed for this path. The Block Base Models and Block Model Types lists, and the AI license list, still apply. A report with no `machine-learning-model` component returns **No AI Components Found**. Reports older than 1.5 are not read by AIBOM.

What gets reported

Each scan reports one **component group** for the scanned file and a list of **components**, one per public repository that publishes that file byte for byte.

- **The component group is the file.** It carries the file's SHA-256, the name it was submitted under, and what its header declares. These facts come from the bytes on disk, so they are always current.
- **Components are repositories, not files.** A model file is often published by several repositories: mirrors, re-uploads and forks. Each becomes its own component, with that repository's name, revision, license and model details. Several components mean several candidate identities for **one** file.
- **Disagreements stay visible.** Repositories that publish identical bytes often declare different licenses or tasks. AIBOM reports all of them rather than picking one, so a license conflict is visible to your policy instead of being resolved arbitrarily.

A file that matches no known repository is reported as a single component of type `machine-learning-model`, named as it was submitted, with the uid `ai/model/<file name>@sha256:<digest>`. A component matched to a repository has the uid `pkg:huggingface/<owner>/<model>@<commit>`, without `@<commit>` when no commit still holds the file.

License classification

Each license a repository declares is checked against the **Block Model Licenses** list in the AIBOM Module. This list is separate from the SBOM license policy, so blocking a license for packages does not block it for models. Block Model Licenses is off by default. Its list comes pre-filled with the non-commercial and no-derivatives Creative Commons

licenses (CC-BY-NC-2.0/3.0/4.0, CC-BY-NC-ND-3.0/4.0, CC-BY-NC-SA-2.0/3.0/4.0, CC-BY-ND-4.0) and Proprietary, and these take effect as soon as you turn the option on. The option is available only when license risk detection is enabled in the SBOM database.

Result	Meaning
Allowed	The repository declared a license that is not selected in Block Model Licenses, or the option is off.
Blocked	The repository declared a license selected in Block Model Licenses, and the option is on
Unknown	The repository declared no license, or the file matched no repository

> ***Note:*** *Only a recognized license can be blocked. A license string that AIBOM does not recognize, such as* `openrail`*, is always reported as allowed, whatever the policy says.*

AIBOM verdict

AIBOM gives each scan one of five verdicts:

- **AI Components Found**: model files were found and none is blocked.
- **License Risk Found**: at least one component declares a license selected in **Block Model Licenses**.
- **Base Model Risk Found**: at least one component matches a **Block Base Models** pattern.
- **Model Type Risk Found**: at least one component's model type matches a **Block Model Types** pattern.
- **No AI Components Found**: the file holds no AI or machine-learning content.

When more than one risk applies, the verdict is the first in the order License Risk Found, Base Model Risk Found, Model Type Risk Found, and `verdict_explanation` lists each one that applies, in that order. Each of the three risk verdicts blocks the file. AIBOM never raises the severity of a scan: its severity is always `INFO`.

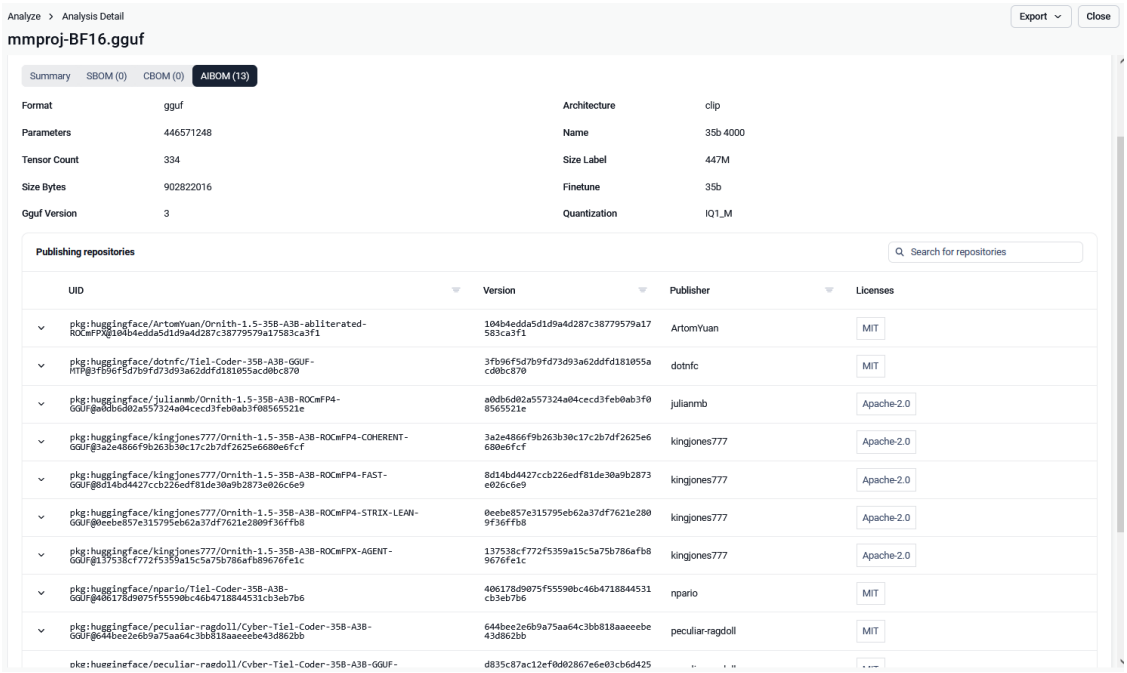
Blocking by base model and model type

Two options in the AIBOM Module block models by name. Both are off by default and start with an empty list.

- **Block Base Models** blocks a model when its name, its base model, or any model it derives from matches one of the patterns. Names read `owner/model`, as on Hugging Face. A model the database does not know is named after the uploaded file.
- **Block Model Types** blocks a model when its model type matches one of the patterns. The model type is the `model_type` in the model's `config.json`, such as `bert`, `llama` or `qwen3`.

Write one regular expression per row. Matching ignores case and finds the pattern anywhere in the value unless you anchor it with `^` or `$`. Look-around and backreferences are not supported. Examples: `^deepseek-ai/` blocks every deepseek-ai model and every model derived from one; `^qwen` blocks `qwen2`, `qwen3` and `qwen3_moe`.

A blocked component carries `model_card.blocked_by.base_model` or `model_card.blocked_by.model_type`, each holding `matched_pattern` (the first pattern that matched) and `matched_value` (the name or type it matched).



For the full result structure, see AIBOM JSON response.

Enabling AIBOM

AIBOM requires MetaDefender Core 5.23.0 or later and is **enabled by default**. Its settings are in the **AIBOM Module** section of Workflow > [workflow name] > SBOM. **Enable AI Bill of Materials (AIBOM)** turns the module on or off. Its three blocking options (**Block Model Licenses**, **Block Base Models**, **Block Model Types**) are off by default. Scanning model files requires the AIBOM offline database, which is delivered with the SBOM database package; CycloneDX documents are read without it. On Core versions earlier than 5.23.0 the AIBOM section is not shown and AIBOM does not run.



AIBOM JSON response

This page describes the `aibom_info` section of the scan result, for integrations that parse it. For what AIBOM detects and how to read it, see AI Bill of Materials (AIBOM) (Beta).

JSON structure

Code · json

```
"aibom_info": {
  "final_verdict": {
    "verdict": "<string>",
        //"No AI Components Found" | "License Risk Found" | "AI Components Found" | "Base Model
Risk Found" | "Model Type Risk Found"
    "verdict_explanation": ["<string>"],   //every statement that applies, in precedence order;
the first equals "verdict"
    "blocked": <boolean>,                  //true when any component is blocked by license, base
model or model type, field blocked_component_count, blocked_base_model_count,
blocked_model_type_count
    "severity": "INFO",                    //always "INFO"
    "total_component_count": <int>,        //number of entries in "components" (repositories,
not files)
    "licensing_component_counts": {        //absent when there are no components; zeros are
always emitted
      "allowed": <int>,
      "blocked": <int>,
      "unknown": <int>                     //components with no declared license
    }
  },
  "component_group": {                     //the scanned file; {} when nothing was found
    "group_id": "sha256:<hex>",            //SHA-256 of the scanned file, with the "sha256:"
prefix
    "target": "<string>",                  //the file name you submitted
    "metadata": [                          //facts read from the file's bytes
      { "name": "<string>", "value": <string | int> }
    ],
    "refs": ["<string>"]                   //uid of every component that contains the scanned
file
  },
  "components": [                          //one entry per repository; [] when nothing was found
    {
      "uid": "<string>",                   //package URL, e.g. pkg:huggingface/<owner>/
<repo>@<commit>
      "type": "machine-learning-model",
      "name": "<string>",                  //<owner>/<repo>
      "version": "<string>",               //repository commit
      "url": "<string>",                   //repository page
      "authors": "<string>",               //repository owner
      "description": "<string>",
      "published_at": "<string>",          //RFC 3339 timestamp
      "last_modified_at": "<string>",      //RFC 3339 timestamp
      "scans_done": <boolean>,             //whether the registry has scanned the repository
      "library": "<string>",               //framework declared by the registry
      "hashes": [
        { "file_name": "<string>", "sha256": "<hex>" }   //no "sha256:" prefix
      ],
      "licenses": {                        //absent when the repository declares no license
        "allowed": [ <license> ],          //or "blocked": [ <license> ]
      },
      "model_card": {                      //facts declared by the publishing registry
```

```
        "base_model": "<string>",
        "base_model_relation": "<string>", //e.g. "quantized"
        "hyperparameters": [ { "name": "<string>", "value": <int> } ]
      }
    }
  ]
}
```

A `<license>` entry has this shape:

Code · json

```
{
  "id": "<string>",                        //SPDX identifier, e.g. "Apache-2.0"
  "name": "<string>",                      //license name as the repository declares it
  "category": "<string>",                  //e.g. "permissive"
  "url": "<string>",
  "usage_restriction": {
    "permissions": ["<string>"],
    "conditions": ["<string>"],
    "limitations": ["<string>"]
  }
}
```

Block states

The shape of `aibom_info` (inside `module_info`, MetaDefender Core 5.23.0 and later) tells you whether AIBOM ran:

Shape	Meaning
`aibom_info` is `{}`	AIBOM did not run: it is disabled in the workflow.
`err_code` + `err_details` only	AIBOM ran and could not finish. `err_details` is `"AIBOM cache unavailable."` when the AIBOM model database is unavailable, and `"The module did not complete this scan."` otherwise.
`final_verdict`, `component_group`, `components`	AIBOM ran. When it found nothing, the verdict is `No AI Components Found`, `component_group` is `{}`, `components` is `[]`, and `licensing_component_counts` and the `blocked_*_count` fields are absent.

Field notes

- **Absent keys mean "not known".** Inside a component, values are never `null`, `""` or `[]` in place of an unknown, so `"version" in component` is a reliable test. Exceptions: the `format` entry of `metadata` is always present when `component_group` is not empty, the counts in `licensing_component_counts` always include zeros, Empty and null values are removed at every depth, including inside `model_card.metrics`. Exceptions: the `format` entry of `metadata` is always present when `component_group` is not empty; the counts in `licensing_component_counts` always include zeros; and when `components` is not empty, `blocked_component_count`, `blocked_base_model_count` and `blocked_model_type_count` are always present, zeros included.
- **Verdict.** `verdict` is evaluated in this order: `No AI Components Found` (`total_component_count` is 0), `License Risk Found` (`licensing_component_counts.blocked` > 0), `Base Model Risk Found` (`blocked_base_model_count` > 0), `Model Type Risk Found` (`blocked_model_type_count` > 0), `AI Components Found`. `verdict_explanation` lists every risk statement that applies, in the same order. Components with an unknown license do not change the verdict.
- **Finding the scanned file in a repository.** In a component's `hashes[]`, the entry whose `sha256` equals `component_group.group_id` **without the `sha256:` prefix** is the file you scanned, under the name that repository gives it. That name often differs from the one you submitted.
- **A model file found in no public repository** is still reported, as one component with:
  - `uid`: `ai/model/<file name>@sha256:<digest>`
  - `type`: `machine-learning-model`
  - `name`: the file name as submitted

  It has no repository fields, and its license block is empty (`{}`), so it counts as unknown in `licensing_component_counts`. By contrast, a component matched to a repository has `uid` `pkg:huggingface/<owner>/<name>@<commit>`, without `@<commit>` when no commit still holds the file.
- **Header facts versus registry facts.** `component_group.metadata` comes from the file's bytes; everything on a component comes from the publishing registry. If they disagree, for example `parameters` in `metadata` against `model_card.hyperparameters`, that is worth reviewing, not an error in the report. A registry often reports figures for the whole repository rather than for the one file you scanned.
- **Metadata names are the scanner's own.** Entries in `metadata` use the scanner's names, not the keys inside the file. A GGUF header's `general.license` is reported as `license`. Its value is free text the file declares; license buckets come from the repository's declaration, not from it.
- **Component order.** Blocked components come first: those blocked by license, then those blocked by a Block Base Models or Block Model Types pattern. The rest follow the model database's ranking, which by default puts first the repository that the other matches name as their base model, then the repository that published the file earliest. Ties are ordered by `published_at`, oldest first, with undated repositories last. `published_at` is the repository's creation date.
- A component blocked by a list carries `model_card.blocked_by.base_model` or `model_card.blocked_by.model_type`, each `{ "matched_pattern": "<pattern>", "matched_value": "<value>" }`. Blocked components are listed first.

Example

A model file found in a public repository

A safetensors file submitted as `model.safetensors`, matched to one repository that keeps it as `1_Dense/model.safetensors`:

Code · json


```
"aibom_info": {
  "final_verdict": {
    "verdict": "AI Components Found",
    "verdict_explanation": ["AI Components Found"],
    "blocked": false,
    "severity": "INFO",
    "total_component_count": 1,
    "licensing_component_counts": { "allowed": 1, "blocked": 0, "unknown": 0 }
  },
  "component_group": {
    "group_id": "sha256:3b5d855b26767da1c16a086c90aef497114c7c0c4451c6fe9df7a0afc2401c86",
    "target": "model.safetensors",
    "metadata": [
      { "name": "format", "value": "safetensors" },
      { "name": "parameters", "value": 16384 },
      { "name": "tensor_count", "value": 1 },
      { "name": "size_bytes", "value": 65624 }
    ],
    "refs": ["pkg:huggingface/NeuML/pylate-bert-tiny@9cb92954e00ee9b127df35354e4714ed5ff70bee"]
  },
  "components": [
    {
      "uid": "pkg:huggingface/NeuML/pylate-bert-tiny@9cb92954e00ee9b127df35354e4714ed5ff70bee",
      "type": "machine-learning-model",
      "name": "NeuML/pylate-bert-tiny",
      "version": "9cb92954e00ee9b127df35354e4714ed5ff70bee",
      "url": "https://huggingface.co/NeuML/pylate-bert-tiny",
      "authors": "NeuML",
      "published_at": "<RFC 3339 timestamp>",
      "last_modified_at": "<RFC 3339 timestamp>",
      "scans_done": <boolean>,
      "hashes": [
        { "file_name": "1_Dense/model.safetensors", "sha256":
"3b5d855b26767da1c16a086c90aef497114c7c0c4451c6fe9df7a0afc2401c86" }
      ],
      "licenses": {
        "allowed": [
          {
            "id": "<SPDX id>",
            "name": "<declared name>",
            "category": "<category>",
            "url": "<license URL>",
            "usage_restriction": { "permissions": ["<...>"], "conditions": ["<...>"],
"limitations": ["<...>"] }
```

```
          }
        ]
      },
      "model_card": {
        "hyperparameters": [ { "name": "parameters", "value": <int> } ]
      }
    }
  ]
}
```

No AI components found

Code · json

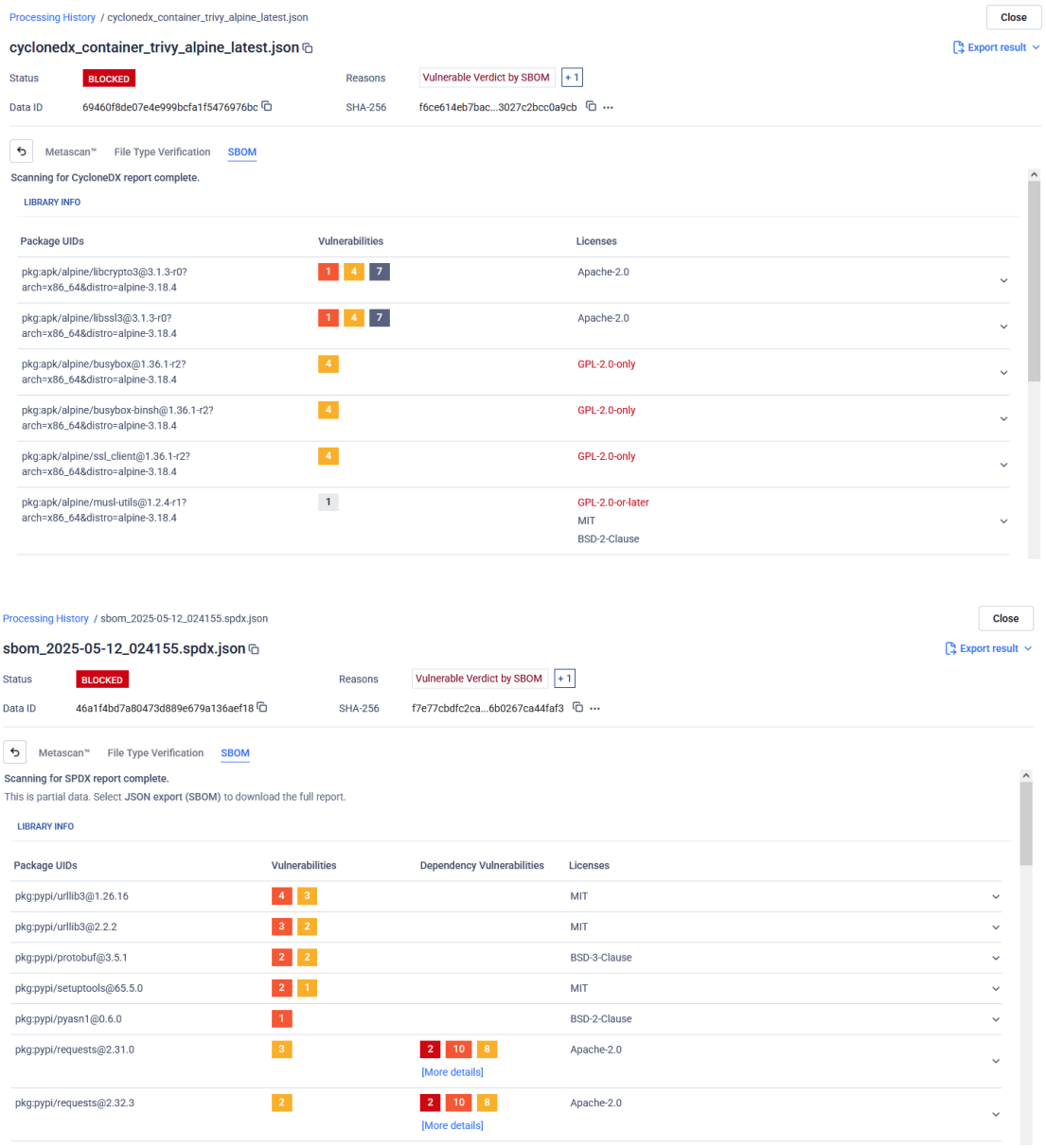
```
"aibom_info": {
  "final_verdict": {
    "verdict": "No AI Components Found",
    "verdict_explanation": ["No AI Components Found"],
    "blocked": false,
    "severity": "INFO",
    "total_component_count": 0
  },
  "component_group": {},
  "components": []
}
```

Analyze CycloneDX/SPDX report file

In the SBOM ecosystem, CycloneDX is commonly associated with vulnerability tracking, while SPDX has historically focused on software license information. The SBOM engine can scan reports in either format and enrich them with any missing data, such as CVE details, license information, and library metadata. The result is a fully enriched SBOM that combines security and compliance insights for more accurate analysis and reporting.

Supported format:

- CycloneDX JSON 1.4, 1.5, 1.6, 1.7. CycloneDX 1.6+ reports also feed CBOM, and 1.5+ reports also feed AIBOM.
- SPDX JSON v2.3, v3.0.1. On MetaDefender Core 5.23 or later, a CycloneDX 1.6 or 1.7 report also feeds the CBOM, and a CycloneDX 1.5–1.7 report also feeds the AIBOM. SPDX reports feed the SBOM only. A report in an unsupported version, or one that fails schema validation, returns no data; to block such files instead, enable **Block on Invalid Input Report**.



CycloneDX CBOM documents

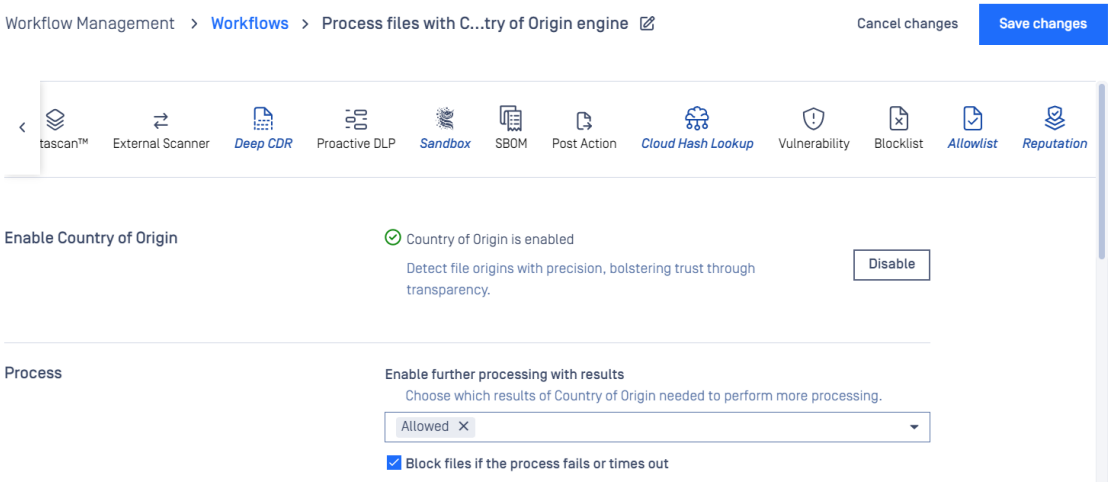
The SBOM engine also reads CycloneDX documents that describe cryptography (a Cryptographic Bill of Materials, supported in CycloneDX v1.6 and later). Their cryptographic assets are added to the CBOM results and graded for classical and quantum risk in the same way as a file the engine analyzes directly. See CBOM – Cryptographic Bill of Materials.

- **References are normalized.** References between components, such as a certificate's signature algorithm or a key's algorithm, are resolved to the target component's `uid` rather than the source document's `bom-ref`, so you never need to understand the producing tool's scheme.
- **Dangling references stay visible.** A reference that cannot be resolved is replaced with a placeholder of the form `crypto/unresolved/@sha256:<hash>` instead of being dropped.
- **Invalid documents are flagged.** A CycloneDX CBOM document that fails validation returns the CBOM verdict `Verification Failed`.

For the result fields, see CBOM JSON response.

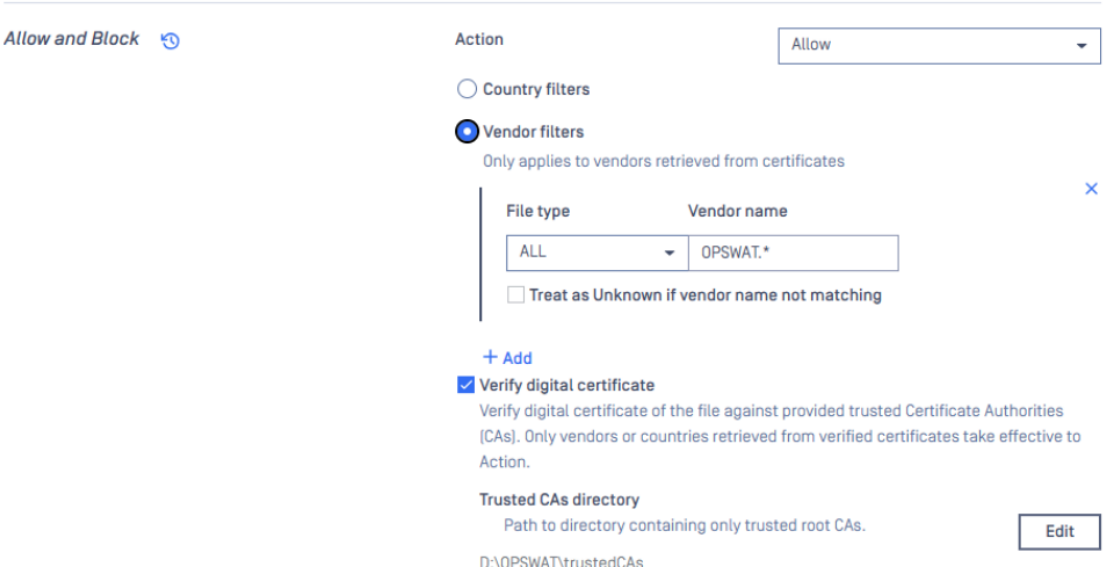
Country of Origin

OPSWAT Country of Origin (COO) engine enables instant detection of a file's geographic source. It analyzes digital fingerprints, metadata to identify restricted locations and vendors automatically. This allows precise filtering that blocks unauthorized data access based on origin while ensuring cross-region regulatory compliance.



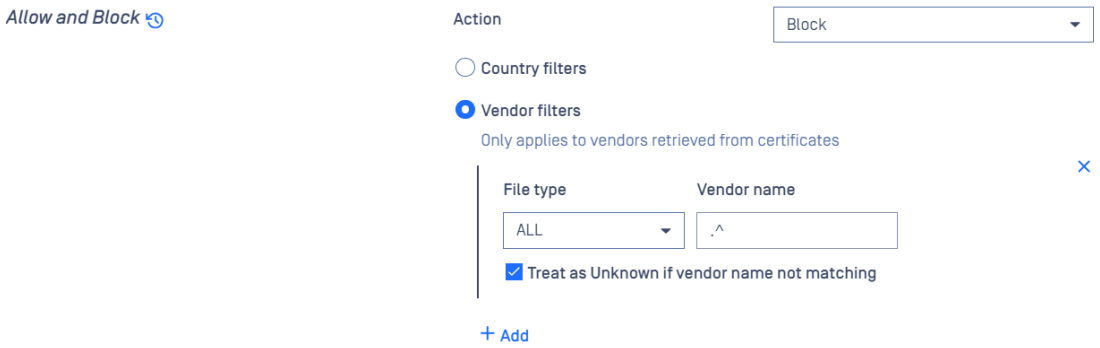
Detect and block by country

You can configure to block or allow from specific countries for each file types



Detect and block by vendor

Similar to country, you can configure to block or allow from specific vendors.

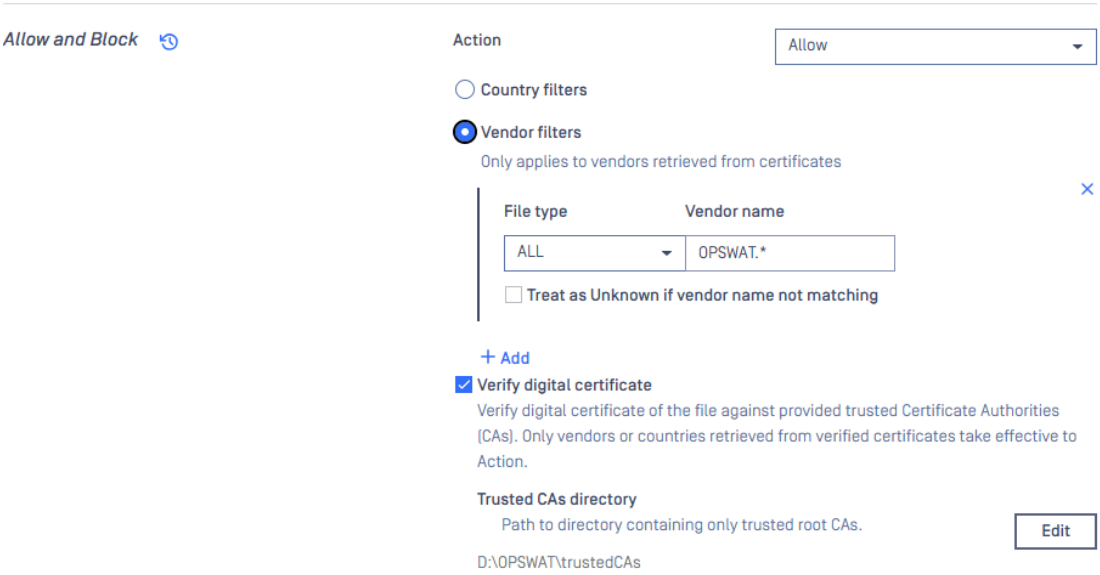


Support file types: DLL, EXE, EXE_SFX, EFI, MSI, and CAB

Digital Certificate Verification

This feature verifies the digital signature of a file to ensure it originates from a trusted vendor. By validating the file's authenticity, it helps prevent the use of tampered or unauthorized software, enhancing overall security and trust in the file source. It is a required component for implementing an Allow List policy, where only files signed by approved vendors are permitted to bypass standard security scans.

Configuration: Workflow > Workflow name > Country of Origin. Enable "**Verify digital certificate**" and update the "**Trusted CAs directory**" to point to a folder that contains trusted CA files. See the "**Best Practices**" section below for instructions on how to obtain these files.



Verification flow

COO engine will try to look for the user's trusted certificate inside "**Trusted CAs directory**" then try to build a trusted chain bottom up from the digital certificate used to sign the input file, up to any trusted certificate collected from "**Trusted CAs directory**".

Only successfully verified certificates are trusted and take effect on `Action` setting. The Action setting (Block/Allow) can only be applied if the Country/Vendor filter match the certificate information and verification passed.

Supported File Type: Digital certificate verification in COO engine supports DLL, EXE, EXE_SFX, EFI, MSI, and CAB file

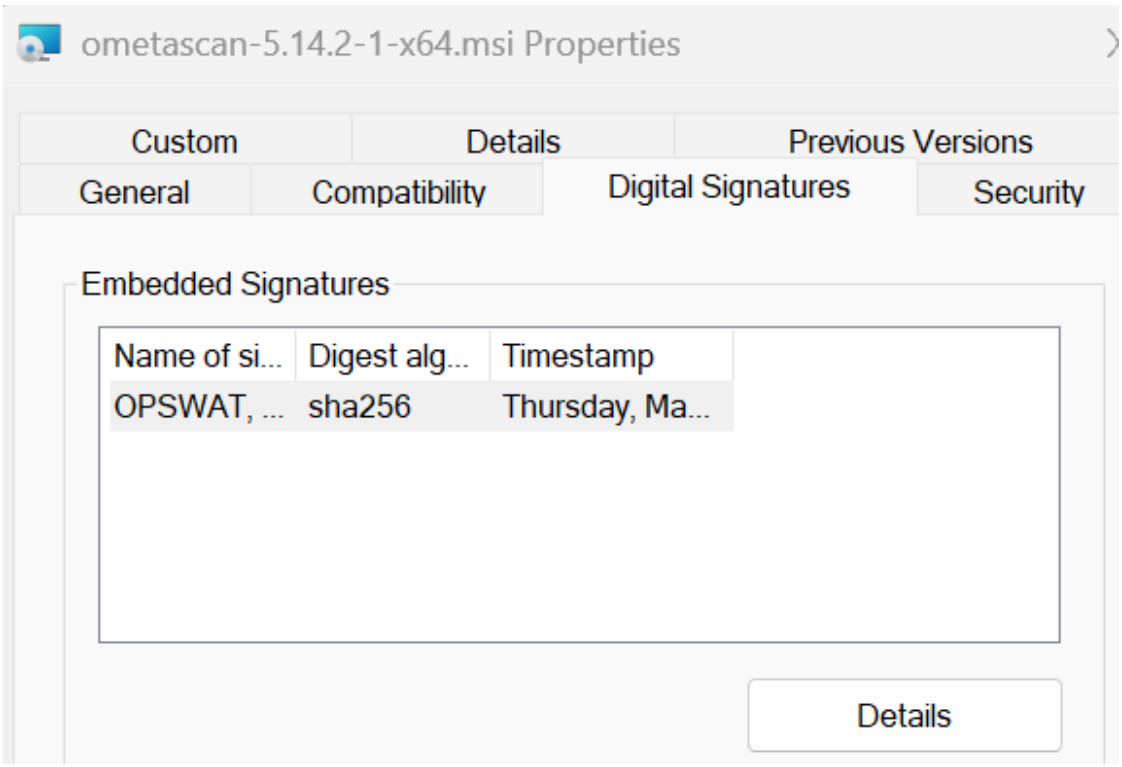
types.

Example: Allowing Files Signed by a Specific Vendor

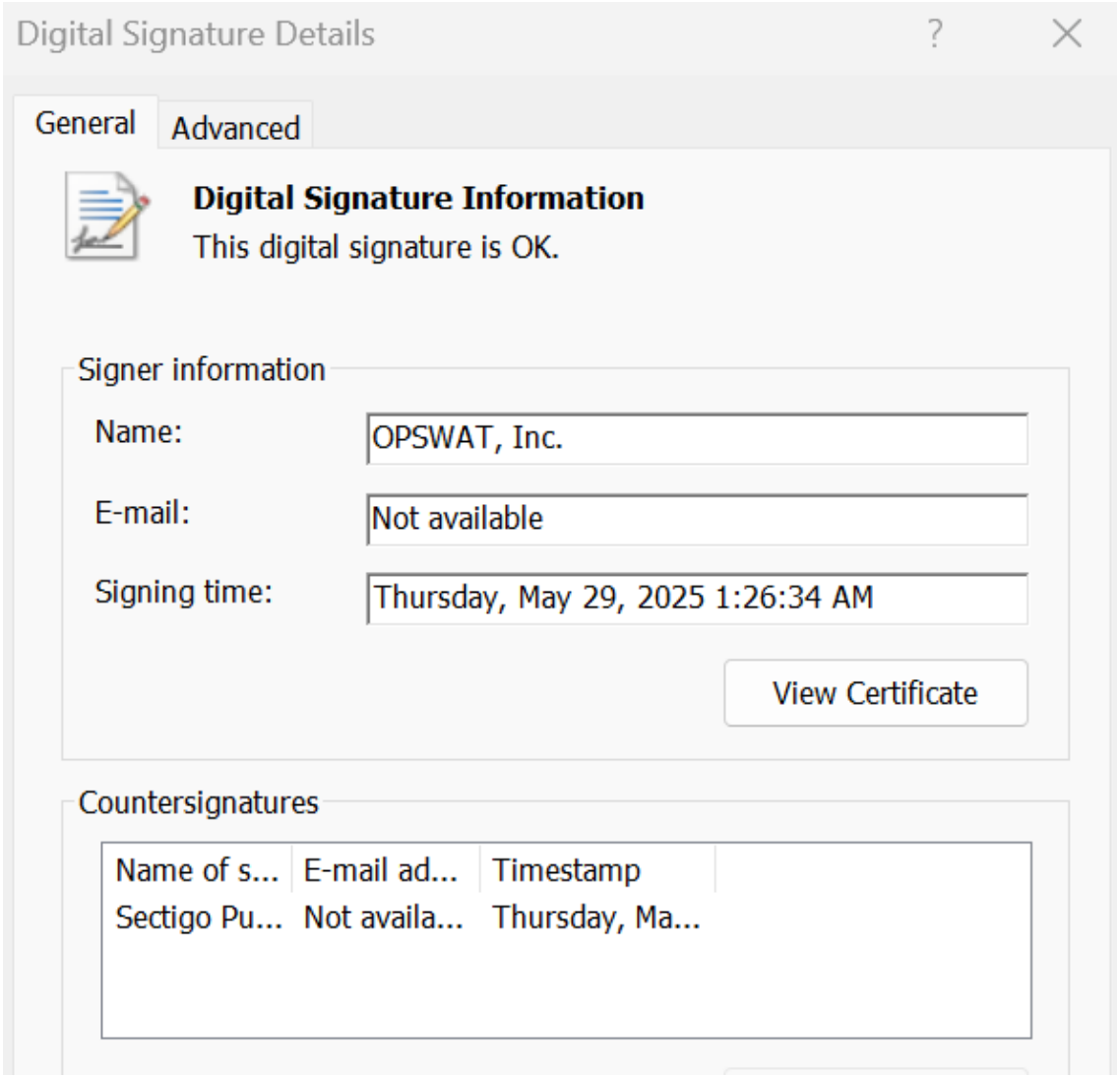
In this example, we will configure the system to allow all files signed by OPSWAT. Any other files will go through the regular scanning process.

1. Collect Trusted CA files:

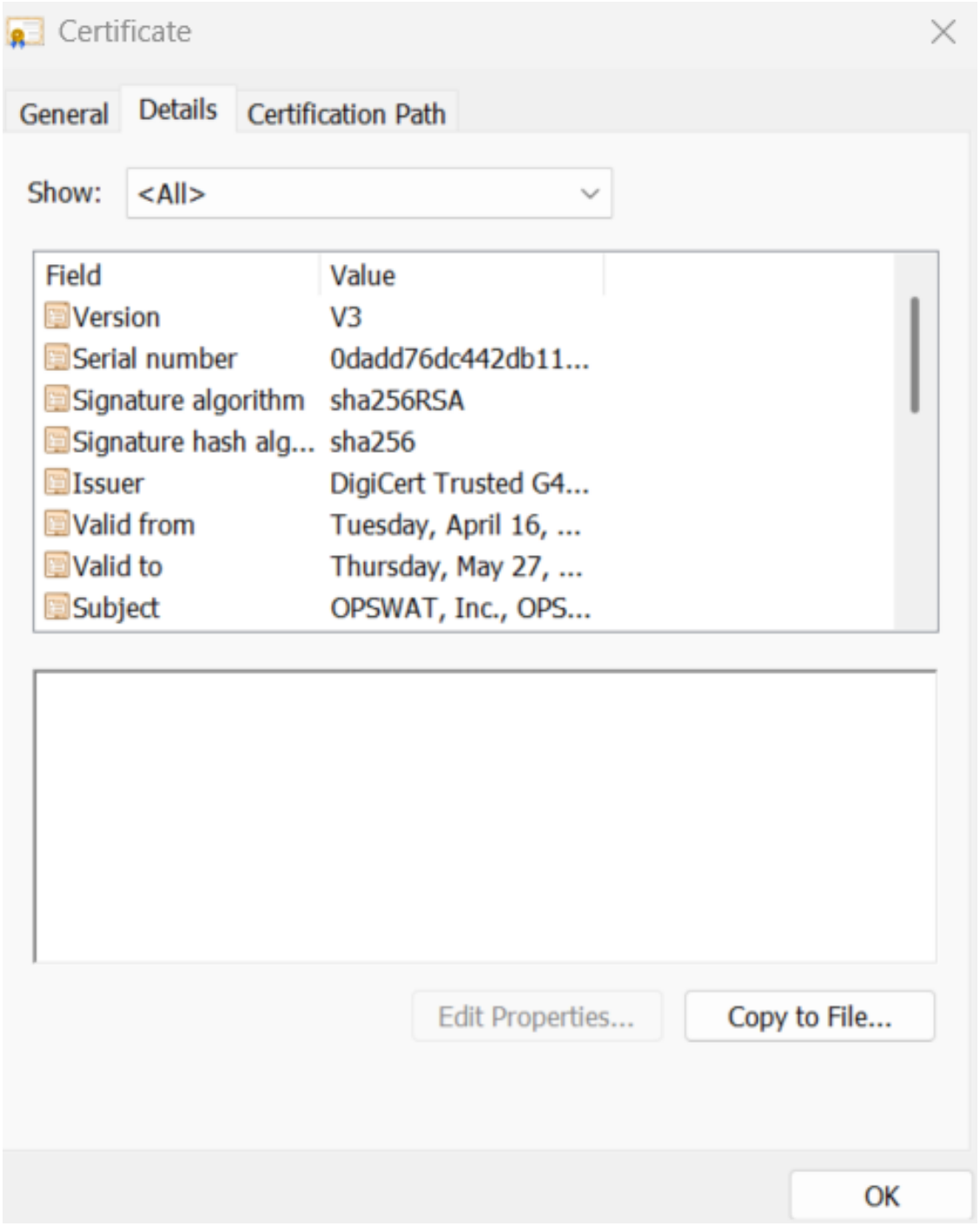
- Created a folder to store Trusted CA files, e.g.: C:\OPSWAT\TrustedCA
- Choose a file that is digitally signed by OPSWAT and that you want to allow.
- Right-click the file and select **Properties**. Go to the **Digital Signature** section, click **Details**



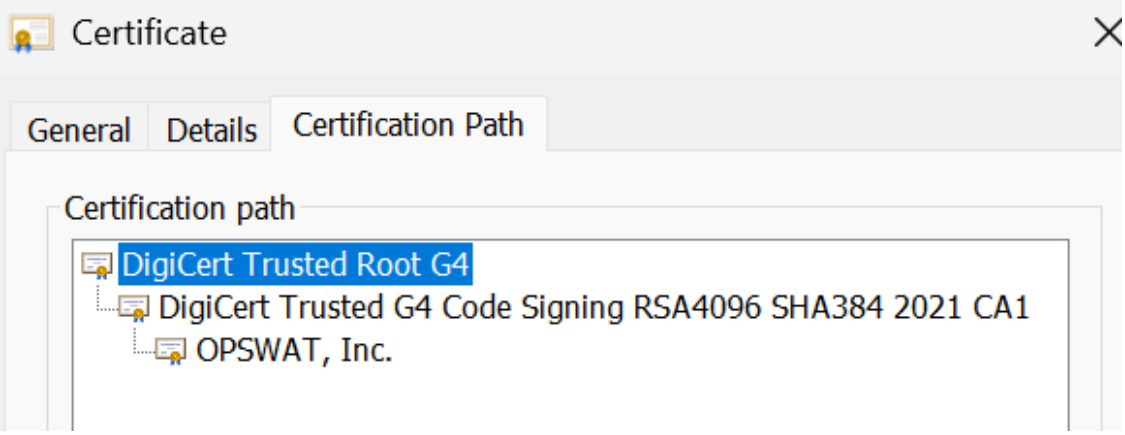
- In the **Signer Information** section, click **View Certificate** to open the certificate details window.



- In the **Details** tab, click on the **OPSWAT, Inc.**, then click **View Certificate** to open its details. Follow the Wizard to save the .CER file to the **C:\OPSWAT\TrustedCA** folder



Notes: If a customer exports a root certificate (e.g., "**DigiCert Trusted Root G4**"), the COO engine will validate the entire certificate chain — including intermediate certificates like "**DigiCert Trusted G4 Code ...**" and the leaf certificate (e.g., "**OPSWAT, Inc.**") — to ensure both integrity and trust.

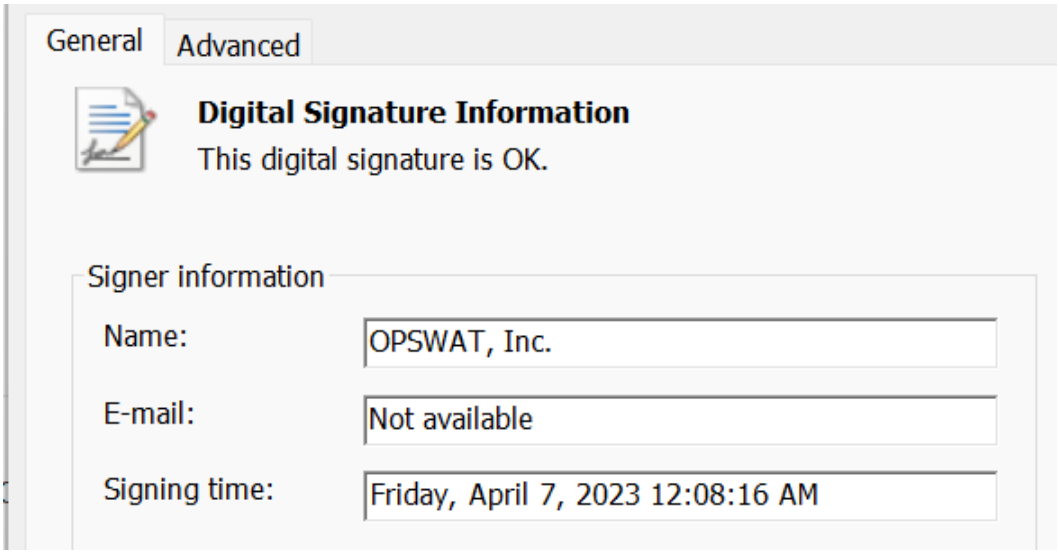


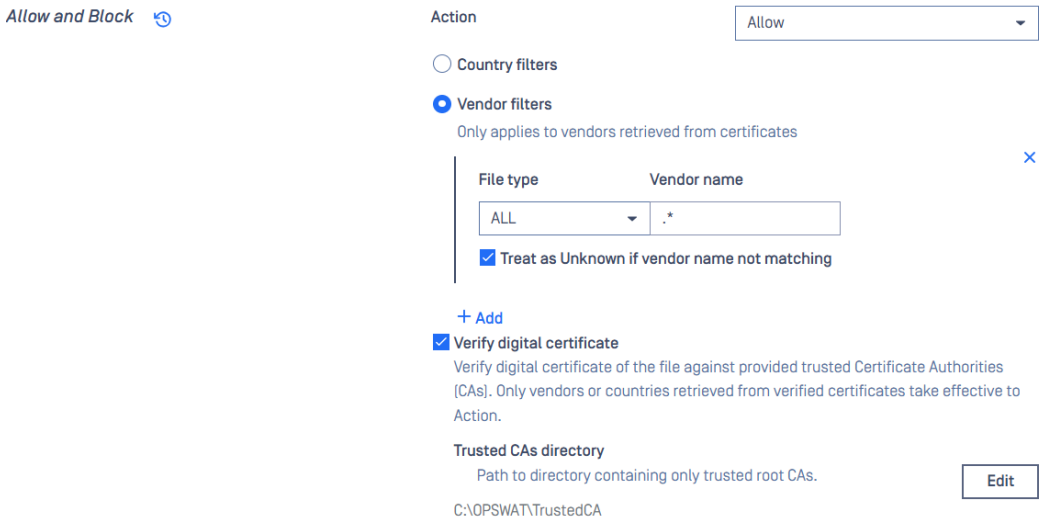
2. Configurations:

- In the **Action** section, select **Allow**.
- Under **Vendor Filters**, configure the rule to allow "**OPSWAT, Inc.**" **(can use ".*" because the trusted certificate already indicates the file from OPSWAT)**.

Info

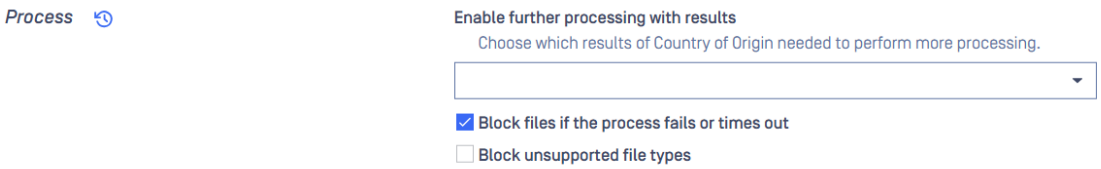
Note: This value is taken from the Signer Name in the digital certificate.





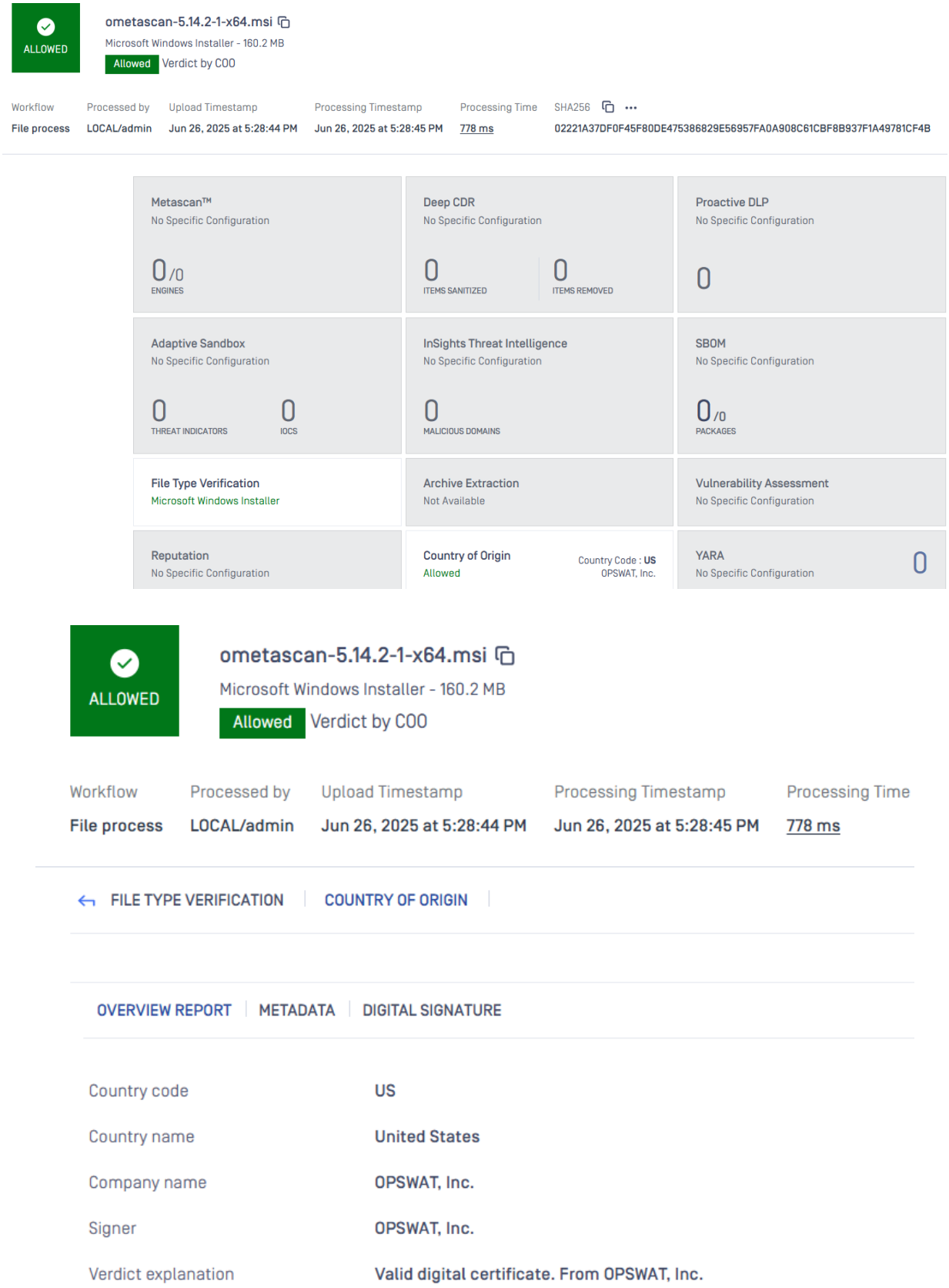
Note:

- Enable the option "**Treat as Unknown if vendor name does not match**". This ensures that any file failing signature validation will be scanned instead of being blocked.
- Do not select any option under "**Enable further processing with results**" to avoid unintended scans.

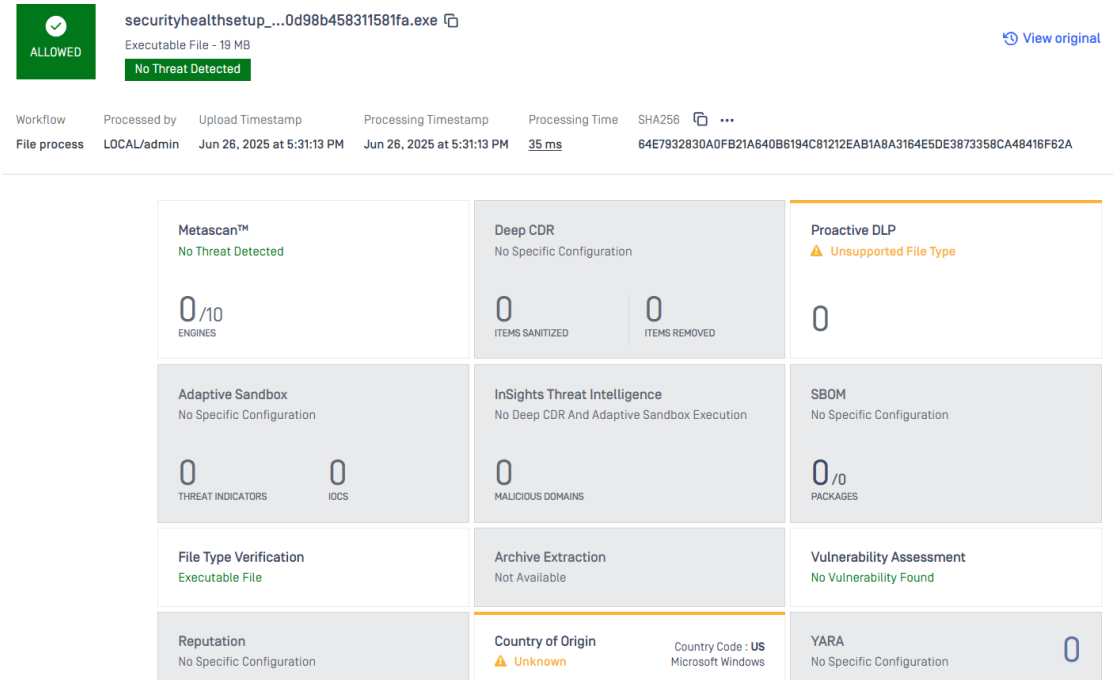


When processing files from OPSWAT with a valid digital signature:

- The file will match the trusted vendor filter ("OPSWAT, Inc.").
- The configured action **Allow** will be applied.
- The file will bypass the regular scanning process and be allowed without further inspection.



If the vendor information is manipulated in the file or not matched the configured vendor name, it will be processed further by other engines for full analysis.


ALLOWED
securityhealthsetup_...0d98b458311581fa.exe
Executable File - 19 MB
No Threat Detected
View original
Workflow
File process
Processed by
LOCAL/admin
Upload Timestamp
Jun 26, 2025 at 5:31:13 PM
Processing Timestamp
Jun 26, 2025 at 5:31:13 PM
Processing Time
35 ms
SHA256
64E7932830A0FB21A640B6194C81212EAB1A8A3164E5DE3873358CA48416F62A
Metascan™
No Threat Detected
0/10
ENGINES
Deep CDR
No Specific Configuration
0
ITEMS SANITIZED
0
ITEMS REMOVED
Proactive DLP
Unsupported File Type
0
Adaptive Sandbox
No Specific Configuration
0
THREAT INDICATORS
0
IOCS
InSights Threat Intelligence
No Deep CDR And Adaptive Sandbox Execution
0
MALICIOUS DOMAINS
SBOM
No Specific Configuration
0/0
PACKAGES
File Type Verification
Executable File
Archive Extraction
Not Available
Vulnerability Assessment
No Vulnerability Found
Reputation
No Specific Configuration
Country of Origin
Unknown
Country Code : US
Microsoft Windows
YARA
No Specific Configuration
0

Overview

Threat Intelligence Engine

The primary purpose of Threat Intelligence is to **detect known threats** by correlating against established intelligence indicators, including known malicious **file hashes, IP addresses, domains, and URLs**. In addition to detection, the engine enriches analysis results by providing **contextual intelligence and metadata**, enabling better-informed security decisions and faster incident response.

Threat Pattern Correlator, is designed to identify and assess **unknown and emerging threats** by correlating them with **known malicious activity**. Instead of relying solely on direct signatures or known indicators, the engine analyzes behavioral and structural characteristics to uncover similarities with established **malware families, attack tactics, infrastructure, and threat campaigns**.

By leveraging **machine learning techniques**, Threat Pattern Correlator enables the detection of **zero-day and previously unseen files** by matching them to patterns derived from already known threats. This correlator-based approach extends protection beyond traditional signature-based detection

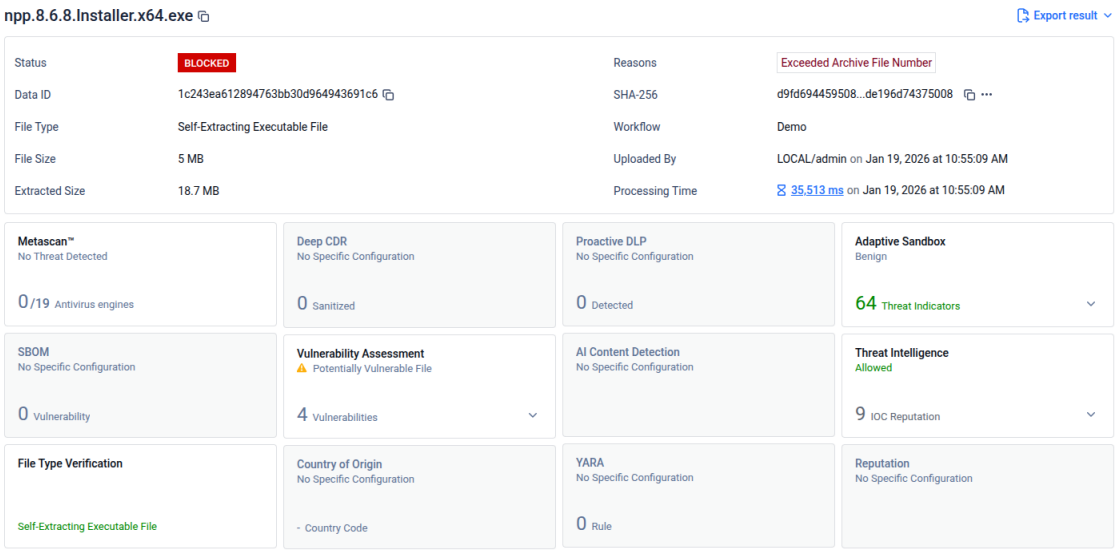
Adaptive Sandbox Required

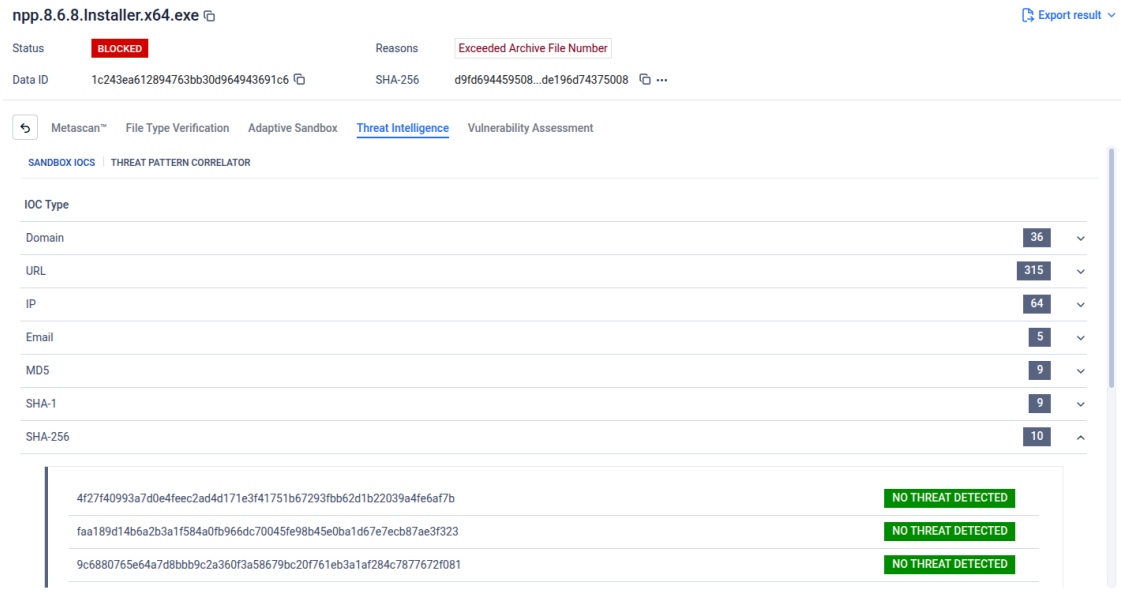
Most features require **Aether** license to function. Learn more: MetaDefender Aether

Adaptive Sandbox must be installed and enabled for the Threat Intelligence Engine to operate correctly.

Displaying engine results

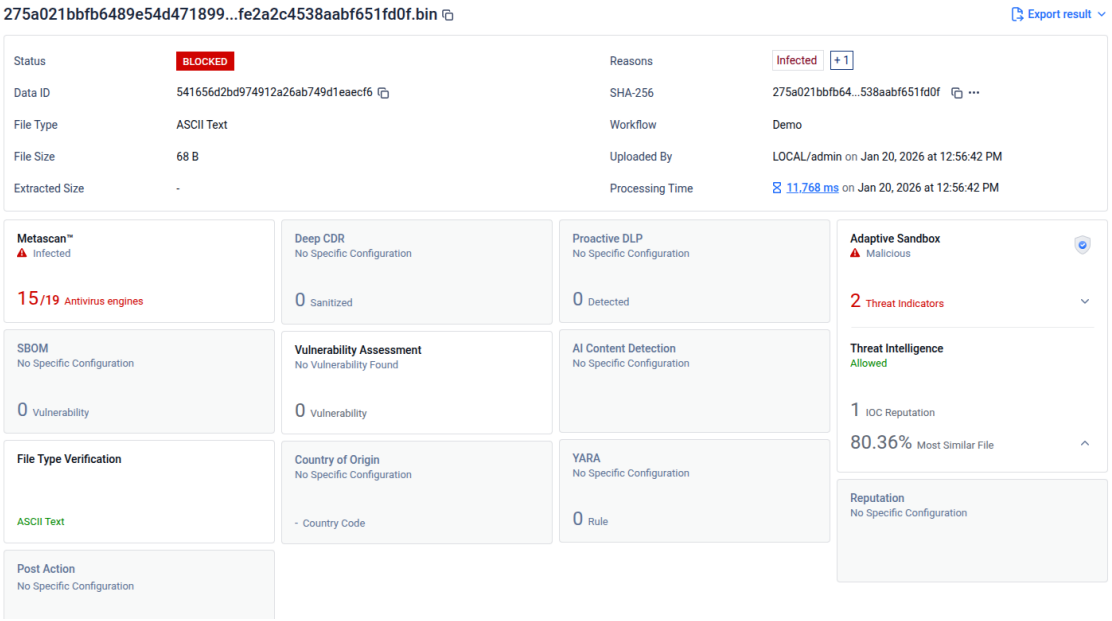
IoC (Indicators of Compromise) Reputation: When Threat Intelligence is enabled, the engine correlates sandbox-generated IoCs with reputation sources and shows how many of those IoCs have reputation data.

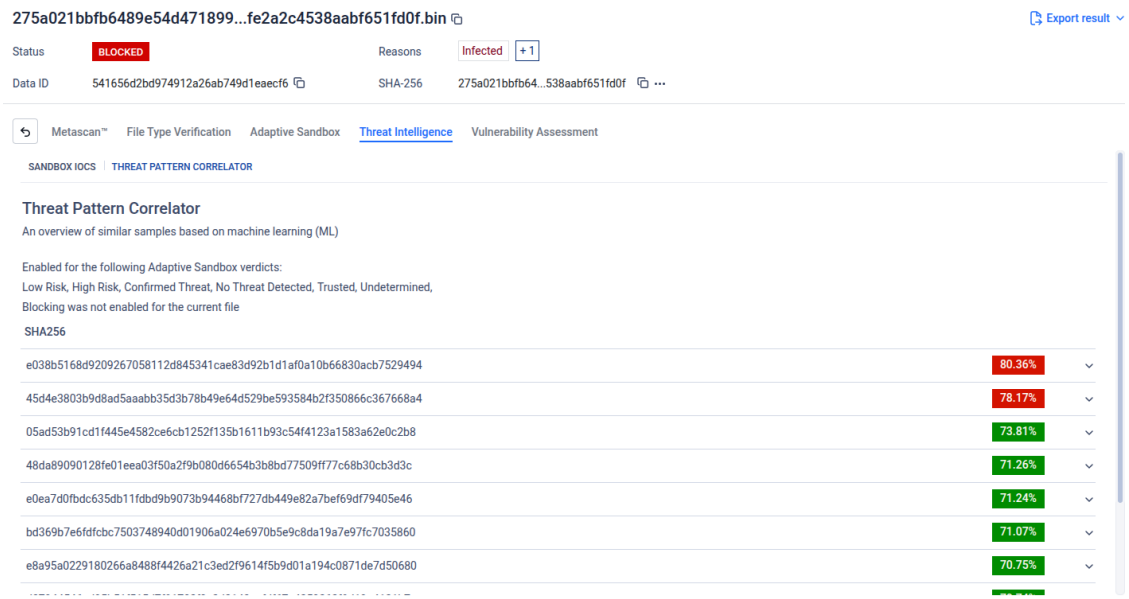




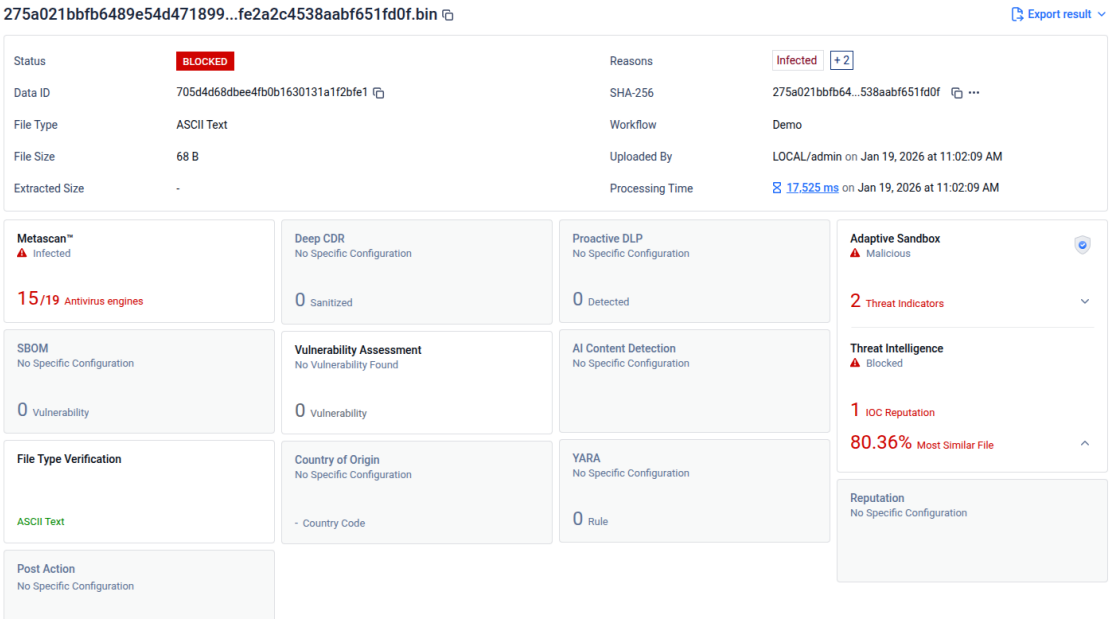
Info

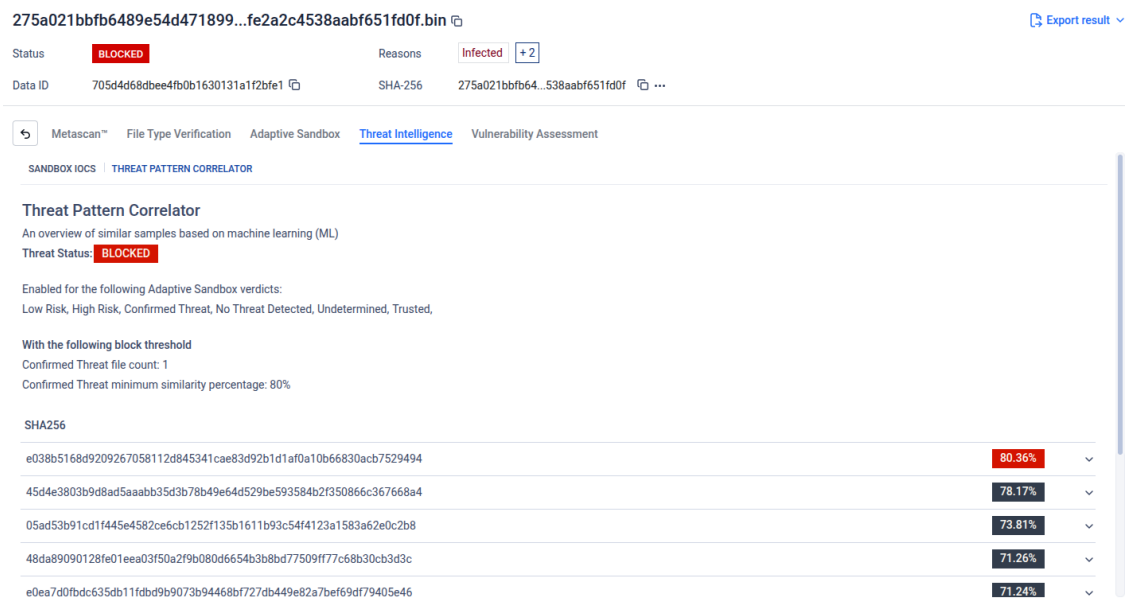
Threat Pattern Correlator (Blocking Disabled): Enabling Threat Pattern Correlator allows the Threat Intelligence Engine to analyze file fingerprints and identify similar files, while operating in a non-blocking mode.





Threat Pattern Correlator (Blocking Enabled): Enabling Threat Pattern Detection allows the Threat Intelligence Engine to analyze file fingerprints, identify similar files, and automatically block those that match known threat patterns.





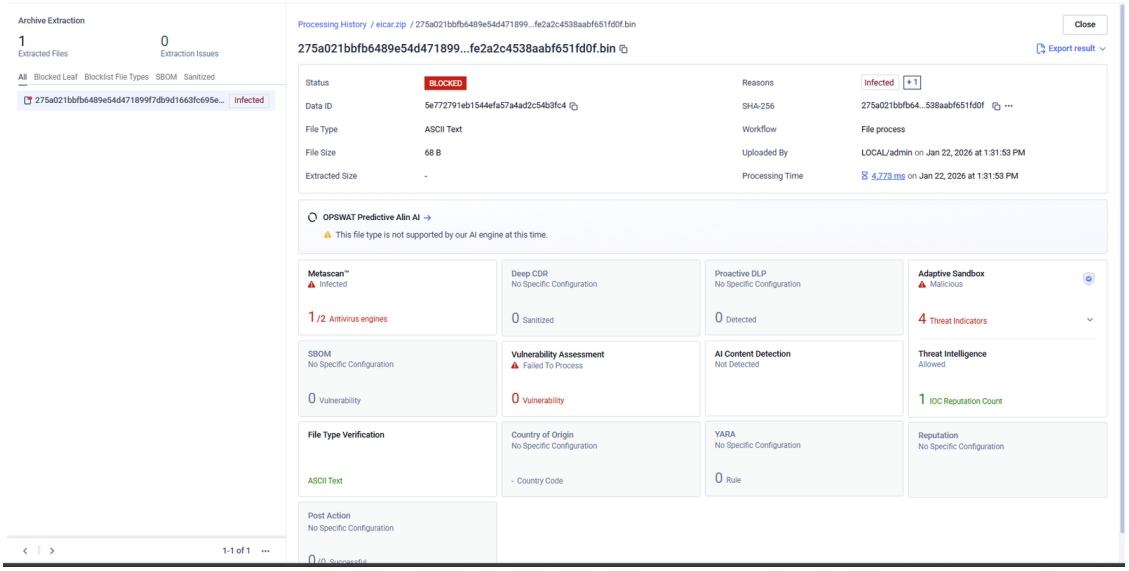
Info

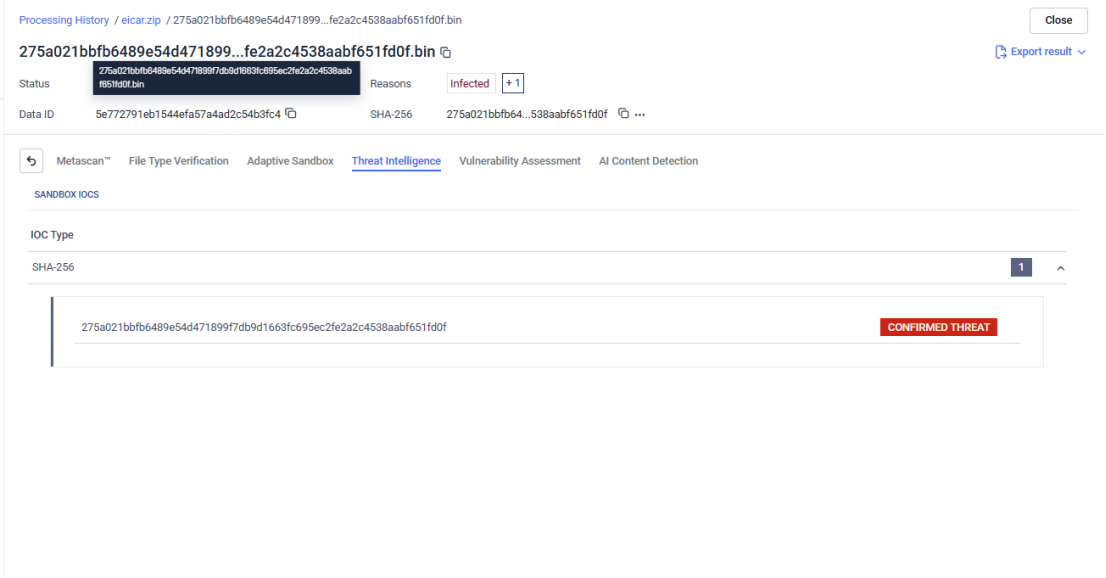
In some cases, a Threat Intelligence result may include an Indicator of Compromise (IoC) classified as **"Confirmed Threat,"** while the overall verdict is shown as **Allowed**.

This behavior is expected. IoCs are **not evaluated in isolation**; instead, they are factored into the final verdict generated by **Adaptive Sandbox** as part of a broader contextual analysis. The sandbox assesses execution behavior, runtime characteristics, and other environmental indicators to determine the overall risk level of the analyzed file.

As a result, the presence of a **"Confirmed Threat"** IoC alone may not be sufficient to classify the sample as malicious from a sandbox perspective, and the verdict may remain **Allowed**.

It is important to note that the **Threat Intelligence engine blocks files only when the Threat Pattern Correlator is enabled**. Without this feature, Threat Intelligence findings— including *Confirmed Threat* IoCs—do not directly enforce blocking decisions and are used for correlation and analysis purposes only.





Licensing

Availability of Reputation Engine for its users:

Free of charge

All MetaDefender Core users will automatically receive **free access** to the Threat Intelligence Engine

Attention

Most features require **Aether** license to function. Learn more: MetaDefender Aether

Adaptive Sandbox must be installed and enabled for the Threat Intelligence Engine to operate correctly.

System Requirements

OS Support

The Threat Intelligence Engine is compatible with the same operating systems as MetaDefender Core.
Recommended System Configuration

Adaptive Sandbox Required

> Most features require **Aether** license to function. Learn more: MetaDefender Aether > > Adaptive Sandbox must be installed and enabled for the Threat Intelligence Engine to operate correctly.

Installation

Info

Threat Intelligence engine is officially supported starting from Core 5.17.1

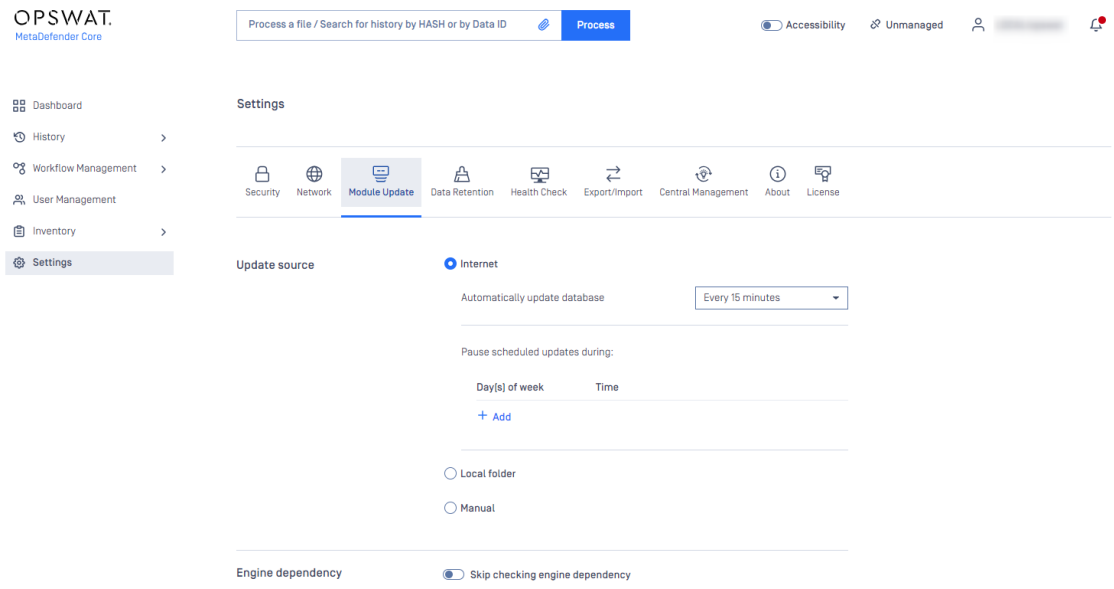
Pro-Tip

Threat Intelligence Engine supports engine updates in either offline or online enviroments

Engine updates

Go to **Settings > Module Update**

MetaDefender Core will automatically download all the Engines available in your license from our online update infrastructure, and download new engine versions automatically based on your **Module Update** configuration.



Offline Engine Updates

Please see the link below for more details.

Inventory Management

Configuration

Info

Threat Intelligence Engine processing is disabled by default in the Workflow Management.

Users will need to enable it first during the Workflow Configuration.

Adaptive Sandbox Required

Most features require **Aether** license to function, Learn More: MetaDefender Aether

Adaptive Sandbox must be installed and enabled for the Threat Intelligence Engine to operate correctly.

Integration mode of Threat Intelligence Engine:

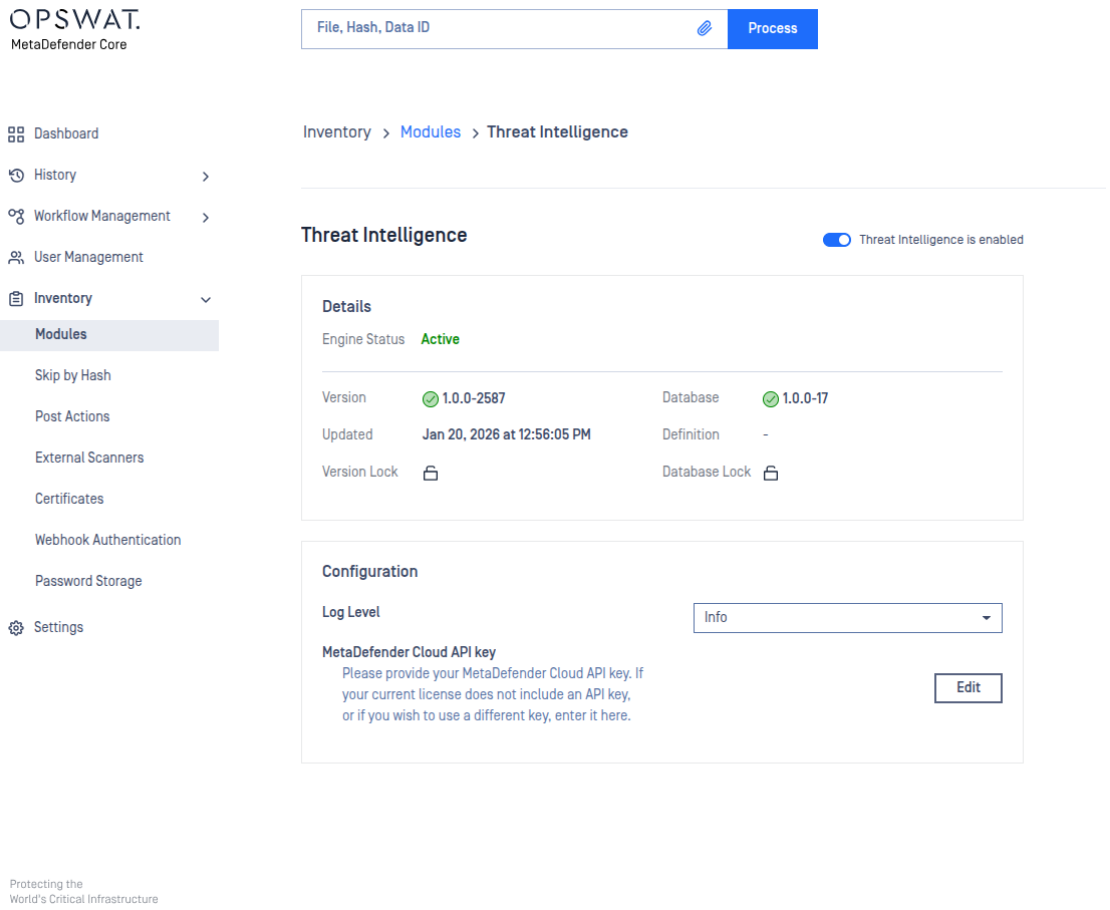
Inline: working as a part of MetaDefender Core processing workflow (real-time processing). Allowing users to block files based on the outcome and decision of the Threat Pattern Detection.

Global configuration

Go to **Inventory > Modules** and select the **Reputation** module

Enabling the engine

See details like engine version and status.



Workflow Configuration

Go to **Workflow Management > Workflows** and select your workflow

In the carousal below, select **Threat Intelligence**.

The Threat Intelligence Engine can be configured in the MetaDefender platform under Workflow Management section, as shown in the image below:

Configuration steps

- First, enable Threat Intelligence Engine in your workflow
- Optionally, modify the default workflow configuration, as described below

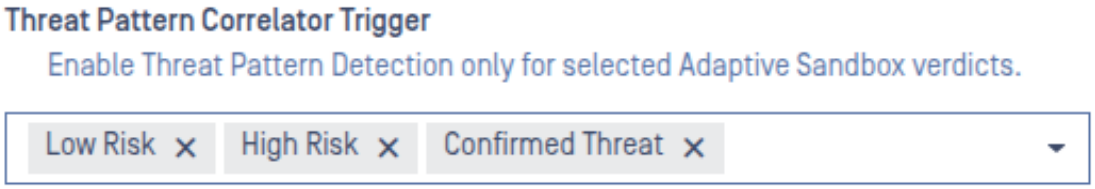
> **Warning**
>
> **These settings might impact performance**

Enable Threat Pattern Correlator



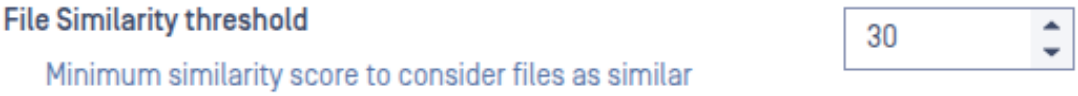
Enabling Threat Pattern Correlator allows the Threat Intelligence Engine to analyze file fingerprints and identify similar files, while operating in a non-blocking mode.

Threat Patter Correlator Trigger



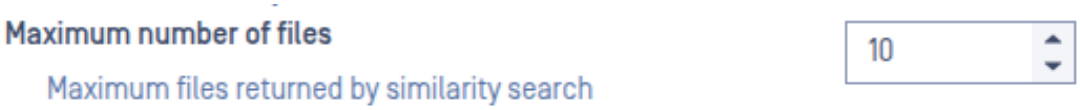
Configure the Threat Pattern Correlator to run conditionally based on **Sandbox** engine result.

File Similarity Threshold



Configure the **overall similarity threshold** that determines which files are displayed on the **Threat Intelligence** tile's results page.

Maximum number of files



Configure the **maximum number of files** displayed on the **Threat Intelligence** tile's results page.

Auto-Block Similar Threats



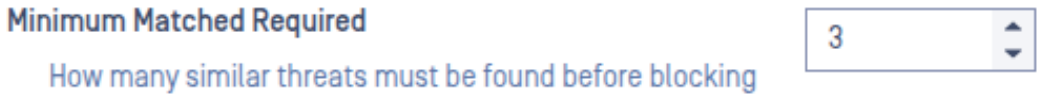
Enable blocking of files based on similar files identified by **Threat Pattern Correlator** that have a **Confirmed Threat** final verdict.

Blocking if Similarity above



Configure the minimum overall similarity required for **Confirmed Threat** files to contribute to the final blocking verdict.

Minimum Matched Required



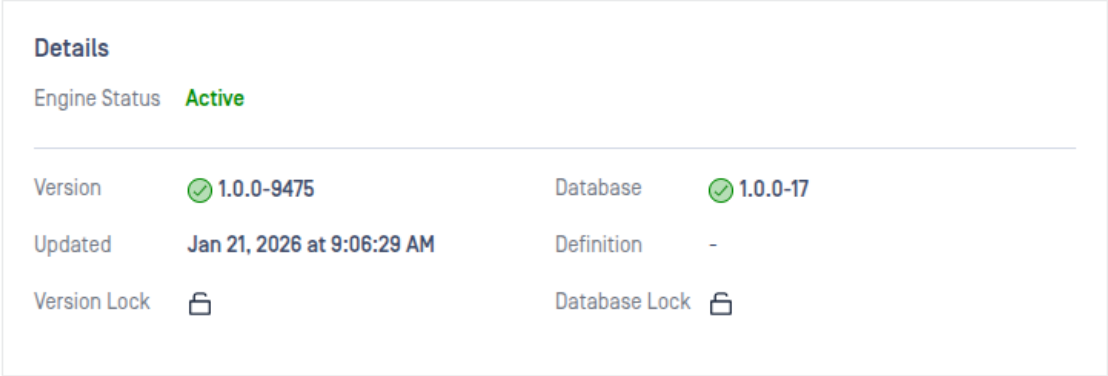
Configure the **number of Confirmed Threat files** that must meet the **required overall similarity threshold** to trigger file blocking.

Global Configuration

Go to **Inventory > Modules** and select the **Threat Intelligence** module

Engine details

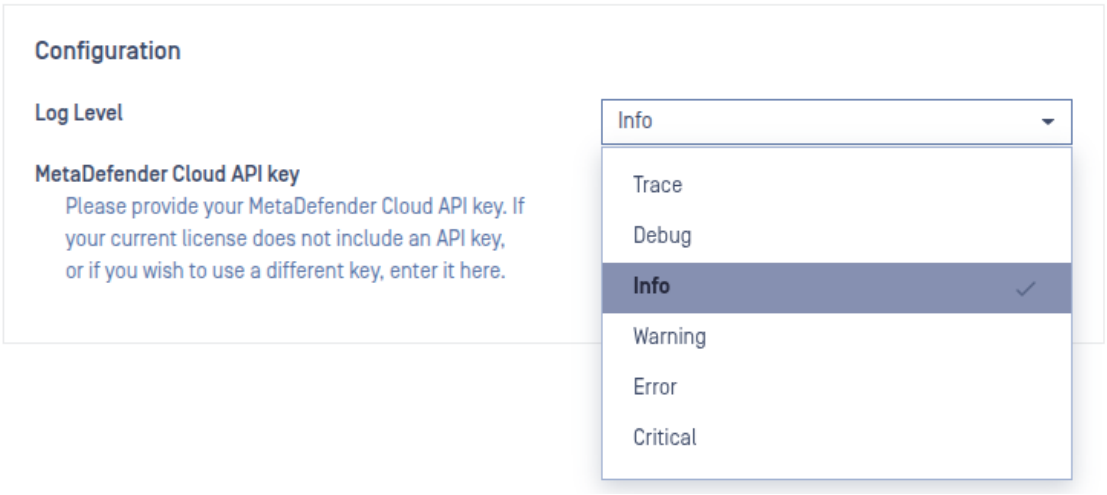
See details like engine version and status.



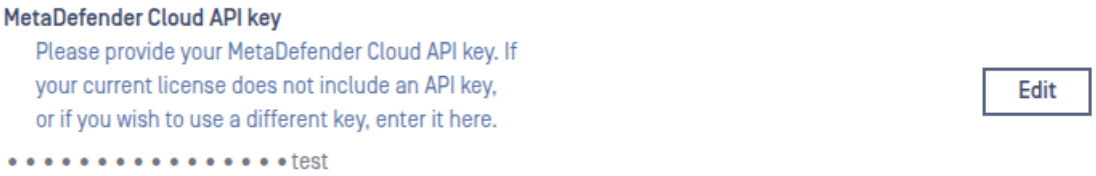
Configuration

Log level

Configure the log level of the Threat Intelligence Engine. In case of issues a lower log level (**Debug**) might help the support team.



MetaDefender Cloud API key



Configure your **MetaDefender Cloud API key** to enable the **Threat Pattern Correlator** task.

Only update this setting if:

- Your license type is **not** an **Aether License**, or
- You want to use a **different MetaDefender Cloud API key** than the default one provided with your license.

Knowledge Center

You will find here answers for FAQs about Threat Intelligence Engine.

What is the minimum version of MetaDefender Core required for compatibility with the engine?

MetaDefender Core version: **5.17.1**

Release Notes

Our release notes have been moved to here.

What is Predictive Alin AI?

Overview

Predictive Alin AI is OPSWAT's proprietary malware detection engine for pre-execution file analysis. It runs in parallel with the Metascan multiscanning package to help identify malicious files before detonation or endpoint execution.

At a glance

- Best for: high-volume enterprise file flows
- Implemented: before execution (static analysis)
- Helps with: unseen, low-prevalence, and structurally modified malware

Why it matters

Predictive Alin AI helps close the gap between malware emergence and signature availability by making a prediction before execution.

What it's designed for

Predictive Alin AI is built for high-volume enterprise environments and focuses on threats that are often underrepresented in signature-first models, including:

- Previously unseen variants
- Low-prevalence malware
- Structurally modified or obfuscated samples

Primary use cases

Organizations typically use Predictive Alin AI to strengthen pre-execution controls in workflows that process large volumes of inbound and outbound files, for example, to:

- Reduce exposure to emerging malware variants
- Increase consistency of **allow / block / quarantine** policy enforcement
- Maintain detection performance at operational scale

How it works (high level)

Predictive Alin AI evaluates intrinsic file characteristics using machine learning models, rather than relying exclusively on signature or hash matches. This improves detection coverage for patterns that may not yet be represented in traditional signature catalogs.

The analysis is **static and pre-execution**, which supports high-throughput environments where detonation is not always operationally practical.

Supported scope

- Platforms: Windows, Linux
- File types: PE, ELF, MACHO, PDF

Availability

Predictive Alin AI is available **only through MetaDefender products and solutions**.

> **Licensing**
>
> **Predictive Alin AI** requires separate licensing for **MetaDefender** to enable this module.

AI-Powered Threat Detection

Predictive Alin AI is a premium engine that operates in parallel with enterprise malware defenses to deliver **advanced pre-execution threat detection**. It helps organizations identify and stop malicious files before execution, reducing exposure across high-volume file flows.

Limitations of a Traditional Antivirus

Conventional antiviruses remain highly effective for known malware and commodity threats, using signature databases, hash matching, heuristics, and reputation systems. However, signature-first detection is **inherently reactive**: threats must be discovered, analyzed, and cataloged before broad protection can be updated.

This creates meaningful exposure in scenarios such as:

- Zero-day malware and exploits
- Polymorphic or obfuscated malware
- Structurally modified binaries designed to evade signatures
- Targeted or low-prevalence attacks

The Predictive Alin AI Approach

Predictive Alin AI applies machine learning to identify malicious intent from intrinsic file characteristics, rather than relying only on historical prevalence. The engine is optimized for fast, pre-execution decisioning and is well-suited for enterprise production workflows where speed and scale matter.

Core capabilities:

- Static structural analysis of supported executables and documents
- Detection of polymorphic, repacked, and structurally modified malware

- Generalization to previously unseen attack patterns
- High-confidence classification to support lower false-positive rates

Continuous Threat Intelligence Pipeline

Predictive Alin AI is powered by a production-grade Machine Learning pipeline engineered for one outcome: **stronger threat detection at enterprise scale**. The pipeline continuously advances model quality while preserving strict standards for data quality, label confidence, and release readiness.

Pipeline goals

- Improve detection of new, modified, and low-prevalence threats
- Maintain high-confidence classifications suitable for production workflows
- Reduce noisy signals that can increase false positives
- Continuously adapt to malware evolution with disciplined model lifecycle controls

> **Info**
> The pipeline performs pre-execution threat detection for supported file formats across the MetaDefender Platform and product lines.

Data Lifecycle

1. Ingestion

The pipeline ingests curated malicious and benign samples from trusted intelligence, research, and controlled repositories. This creates the breadth required for resilient real-world detection.

2. Integrity control

Every sample is normalized through integrity controls, including hashing, deduplication, and validation. The result is cleaner data, lower noise, and more reliable model behavior.

3. Intelligence enrichment through MetaDefender Aether

Labeling is performed through MetaDefender Aether.

Labels are built from multi-source evidence, not a single signal. Evidence channels include:

- Static file characteristics and structural indicators
- Dynamic analysis outcomes from controlled execution environments
- Consensus-oriented malware verdict context
- Threat intelligence enrichment and historical signal correlation

This approach delivers higher-confidence ground truth, reduces label noise, and materially improves generalization against unknown and evasive threats.

4. Feature preparation and model training

Prepared datasets are used to train on representative benign and malicious populations, enabling stable and scalable performance across enterprise file flows.

5. Quality gating

Each model candidate is validated against holdout datasets and false-positive benchmarks. Only models that meet release thresholds are promoted.

6. Continuous refresh and monitoring

The pipeline continuously refreshes datasets and models to address active threat evolution by monitoring in place to quickly identify drift and drive targeted improvement.

Why this matters

- Stronger resilience against novel and modified malware
- More consistent detection behavior in changing threat environments
- Faster adaptation cycles for emerging attack patterns
- Higher trust in pre-execution decisioning for production pipelines

Pre-execution File Analysis

Predictive Alin AI performs file analysis before execution to estimate malicious intent early in the inspection workflow. This supports earlier policy decisions (allow, block, quarantine, or escalate) and reduces dependency on post-execution detection stages.

The analysis is static and structure-driven. Instead of waiting for runtime behavior, the engine evaluates representative file characteristics that are correlated with malicious or evasive patterns.

What this stage does

Pre-execution analysis evaluates file structure and metadata to estimate intent before any code runs.

Executable analysis

For executable formats, analysis focuses on structural and metadata signals that indicate code quality, packing behavior, and anomaly patterns. Representative features include:

- Section layout consistency, section count, and section naming patterns
- Header and metadata integrity, including malformed or unusual field combinations
- Import and export table characteristics, including suspicious API concentration patterns

- Entropy distribution across sections, which can indicate packing or obfuscation behavior
- Entry point and control-flow related structural indicators
- Resource and embedded object composition patterns
- String and token-level structural distributions at a high level

PDF analysis

For PDF files, analysis focuses on document structure, object relationships, and indicators commonly associated with malicious document delivery. Representative features include:

- Object graph complexity and unusual object relationship patterns
- Cross-reference and trailer consistency checks
- Embedded stream and object characteristics (including compression and encoding usage patterns)
- JavaScript-related object presence and structural context
- Action and annotation structures associated with redirection or execution workflows
- Embedded file and launch-related object indicators
- Metadata and document structure anomalies

Why this matters

By evaluating these structural signals pre-execution, Predictive Alin AI improves detection coverage for previously unseen and structurally modified threats while preserving throughput for enterprise file pipelines.

File Types

Executable Format Support

Executable file formats account for the vast majority of malicious binaries circulating in enterprise and internet-facing environments. By providing deep structural analysis across PE, ELF, and Mach-O files, Predictive Alin AI delivers broad coverage of executable-based threats, including zero-day and low-prevalence malware.

The engine supports the three primary executable formats used across modern platforms:

- **PE (Portable Executable)** – Windows operating systems
- **ELF (Executable and Linkable Format)** – Linux and Unix-based systems
- **Mach-O (Mach Object)** – macOS and Apple platforms

Productivity File Formats

Beyond native executables, Predictive Alin AI extends analysis capabilities to commonly used productivity documents. These file types are widely exchanged across email systems, collaboration platforms, and external content-sharing channels, and are frequently leveraged in phishing campaigns and document-based attack chains.

By including productivity formats within its inspection scope, the engine provides broader visibility into enterprise file flows and strengthens protection across business-critical communication channels. Additional coverage within this category is planned to further expand document-level threat detection capabilities.

Currently supported productivity formats include:

- **PDF (Portable Document Format)**

Supported File Types

	File Type	Header
1	.exe	Windows executable application file
2	.dll	Dynamic Link Library containing reusable Windows code and functions
3	.com	Legacy DOS executable format supported by Windows
4	.cpl	Windows Control Panel extension module
5	.ocx	ActiveX control used by Windows applications
6	.drv	Legacy Windows driver file
7	.sys	Windows system or kernel-mode driver file
8	.efi	UEFI executable used during system boot or firmware operations
9	.msi	Windows Installer package file for software installation
10	.msp	Windows Installer patch file used for application updates
11	.elf	Standard binary file format used

	File Type	Header
		primarily on Linux and Unix-like systems
12	.so	Shared object library used in Linux/Unix systems
13	.o	Compiled object file not yet linked into a final executable
14	.ko	Linux kernel module file
15	.out	Default Unix/Linux compiled executable output (legacy naming)
16	.dylib	Dynamic library (macOS equivalent of `.so` on Linux)
17	.bundle	Loadable module/plugin bundle
18	.a	Static library archive
19	.framework	macOS framework bundle (contains dynamic library + resources)
20	.kext	Kernel extension (driver/plugin for macOS kernel)
21	.app	macOS application bundle (directory containing Mach-O executable + resources)
22	.plugin	macOS application plugin bundle
23	mach-o (no extension)	macOS Executable binary
24	.pdf	Adobe Portable Document Format

Maximum File Size

	File Type	Maximum File Size
1	Portable Executable	2GB
2	Executable and Linkable Format	2GB
3	Mach Objects	2GB
4	Portable Document Format	100MB

Engine Response Structure

Engine Verdicts

Predictive Alin AI returns standardized verdict codes upon file analysis. These verdicts are consistent across all integrated OPSWAT products and indicate the outcome of the scanning process.

The following verdict values may be returned:

- **0: No Threat Detected** -*The file was successfully analyzed and no malicious characteristics were identified.*
- **1: Infected** -*The file was analyzed and determined to be malicious.*
- **10: Not Scanned** -*The file was not analyzed. This may occur if the file exceeds the supported size limit or if an internal processing error prevents completion of the scan.*
- **23: Unsupported File Format** -*The submitted file type is not currently supported by the engine.*

Threat Naming Convention

When a file is classified as malicious, **Predictive Alin AI** returns a standardized detection name in the following format: **<platform>/malicious<threat_confidence>**

platform:

- indicates the execution ecosystem associated with the file type. Windows is used for PE-based executables, Linux for ELF-based binaries, and Darwin for Mach-O-based binaries. Generic is used for non-platform-specific formats such as PDF.

threat**_confidence :**

- represents the model's threat confidence level for the analyzed file. Higher values indicate higher malicious confidence. For example, **malicious_99** indicates very high confidence that the file is malicious.

Examples:

- PE file: `Windows/malicious_99`
- ELF file: `Linux/malicious_99`

- Mach-O file: `Darwin/malicious_99`
- PDF file: `Generic/malicious_99`

Supported Operating Systems

Windows

Predictive Alin AI supports **64-bit** Windows only.

Supported versions

- **Windows 11** (23H2) and **Windows 11 IoT** (22H2) - **end-user systems only**
- **Windows Server** 2019, 2022, 2025

> **Windows 11 usage note**
>
> Based on the Microsoft License Terms, Windows desktop editions are not licensed for server use. As a result, Predictive Alin AI should not be deployed on Windows 11 for server use cases.

Linux

Predictive Alin AI supports **64-bit** Linux only.

Supported versions

- Red Hat Enterprise Linux 8, 9
- Rocky Linux 9
- Oracle Linux 9 (9.5 or later)
- Debian 11, 12
- Ubuntu 22.04, 24.04
- Amazon Linux 2023 (supported via Amazon Machine Image (AMI))

> **Kernel requirement for RHEL and Rocky**
>
> Predictive Alin AI may not function correctly on Red Hat Enterprise Linux or Rocky Linux systems running older kernels (for example, 372.9 or 372.13).
>
> Upgrade to **kernel 372.26 or later** to help ensure proper functionality.

Hardware dependencies

- CPU that supports AVX2 (BMI2) instruction set

What is File Structure Validation engine?

File Structure Validation (FSV) is a module that inspects a file against its own format specification, enumerates every object the file contains, and blocks the file when its structure violate policy. FSV does not modify files. It returns a verdict and a detailed report; the original bytes are passed through unchanged or rejected outright.

FSV is built on the file-parsing technology behind Deep CDR, which gives it deep structural awareness across a wide range of formats — but it applies that awareness to a different decision: *is this file well-formed and permitted?* rather than *what must be removed from this file?*

Supported formats (Total 166)

Family	File types
Adobe	pdf, ai, ait
Microsoft Office Word	doc, docm, docx, dot, dotm, dotx, xml-docx
Microsoft Office Excel	xls, xlsb, xlsm, xlsx, xlt, xltm, xltx, xlam, xml-xls
Microsoft Office PowerPoint	ppt, pptm, pptx, pot, potm, potx, pps, ppsm, ppsx, ppam, sldm, sldx, xml-pptx
Visio	vdx, vsdm, vsdx, vssm, vssx, vstm, vstx, vsx, vtx
Microsoft Project	mpp
Microsoft PowerBI	pbix
OneNote	one
Open Document	odg, odp, ods, odt, otp, ots, ott
Hancom	cell, hwp, hwpx, hwt, show
Ichitaro	jhd, jtd, jtdc
DocuWorks (Windows only)	xdw, xbd, xct
RTF / Text	rtf, txt, log
Markup and data	bml, csv, har, hta, html, ics, jnlp, json, lyrx, mpd, rdf, tds, tsv, ttml, twb, vcf, vcs, xml, zei

Family	File types
Image	apng, bmp, cur, dcm, emf, emz, gif, heic, ico, jpg, jpx, mng, nef, png, svg, tga, tiff, tiff64, wdp, webp, wmf, wsq
Audio	aac, acm, aiff, bw64, bwf, mp3, ogg, opus, rf64, vtxt, w64, wav
Video	avi, flv, m2ts, m4a, mov, mp4, mpeg, mts, mxf, webm, wma, wmv
CAD and 3D	dwg, dwt, dws, dxf, dwf, dwfx, easm, jww, jwc, sfc, stp, p21, ifc, catpart, cgr, bfo, pxt, 3mf
Geospatial (ESRI / Shapefile)	shp, shx, prj, sbn, sbx, gdbtable, lpkx, mpkx
ML model	onnx, gguf, sts, parquet
Font	eot, otf, woff
Database	dbf
Email and contact	vcf, vcs, ics
Other	aem, dpa, lnk, url, nrbf, ribcd, crl

Formats are added continuously, check the release notes of each version for newly supported types

Features

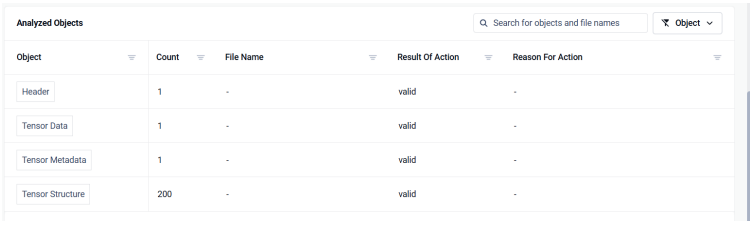
File Structure Validation performs the following functions in a MetaDefender Core™ workflow. Each one is configured on the File Structure Validation tab of the workflow, and each can be scoped per file type.

- Structure validation: Checks the file against its format specification and marks the structure valid or invalid
- Object extraction and routing: Extracts the objects in the file and sends them to downstream engines in a workflow
- Rule-based blocking: Blocks on a built-in object class, or on a custom JSON condition matched against object attributes
- Polyglot detection: Reports files that carry the signatures of more than one format

- Format-specific validation: XSD schema for XML.

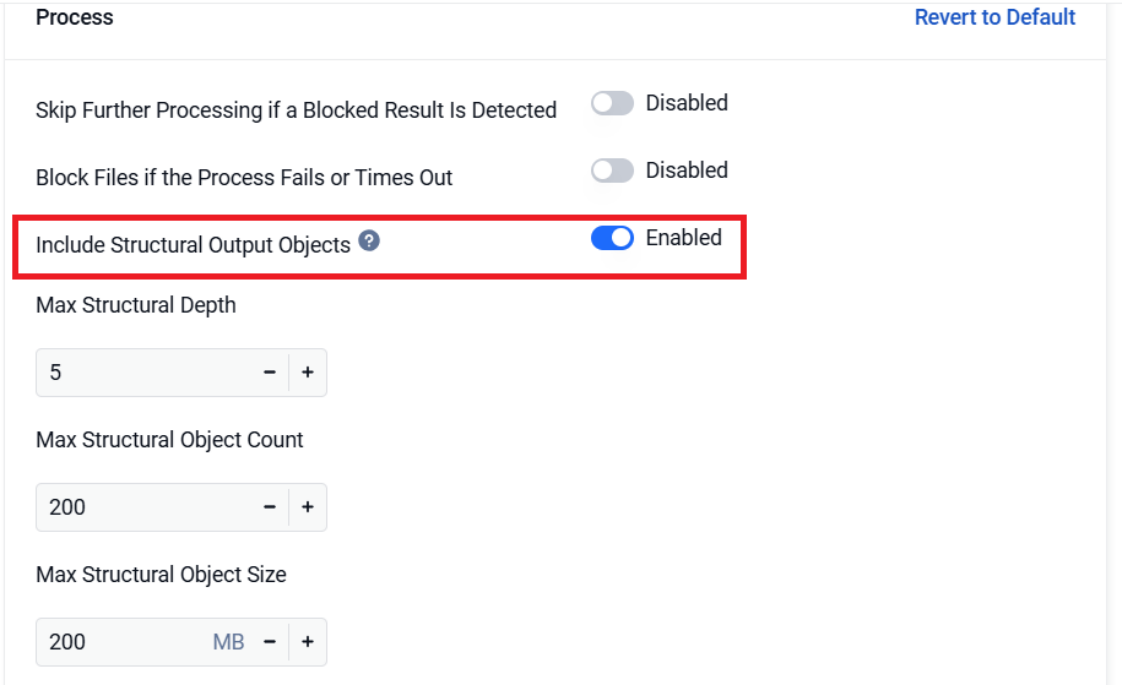
Extracting objects and recursive processing

Breaks files into discrete components (macros, hyperlinks, metadata, images, and embedded executables) for comprehensive visibility inside complex and nested file formats.

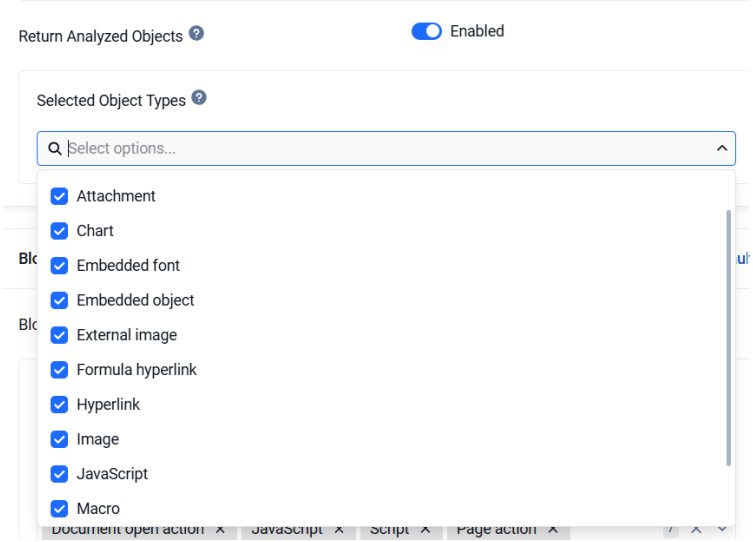


Routing extracted objects to other engines: You can choose which object classes are returned as output files.

Enable "*Include Structural Output Objects*"

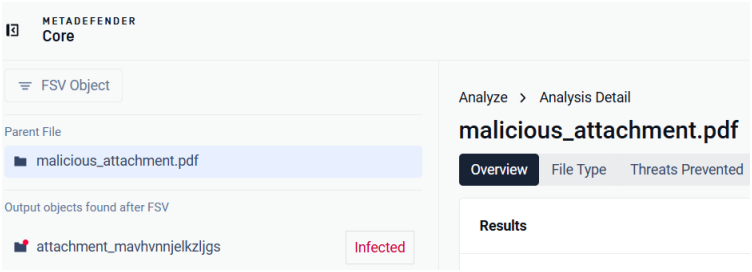


And enable "Return Analyzed Objects", choose which objects should be processed recursively.

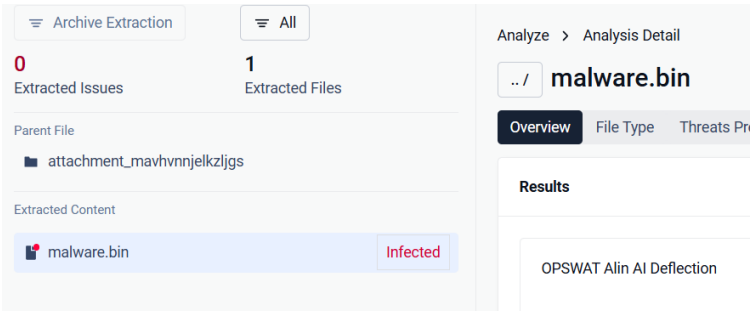


Objects that FSV extracts are handed back to the analysis workflow as regular files, so Multiscanning, Proactive DLP, Adaptive Sandbox, and other configured engines act on the child object rather than only on the outer container. Extracted objects are processed by every engine in the workflow (except Deep CDR).

An example an PDF file contains a zip attachment



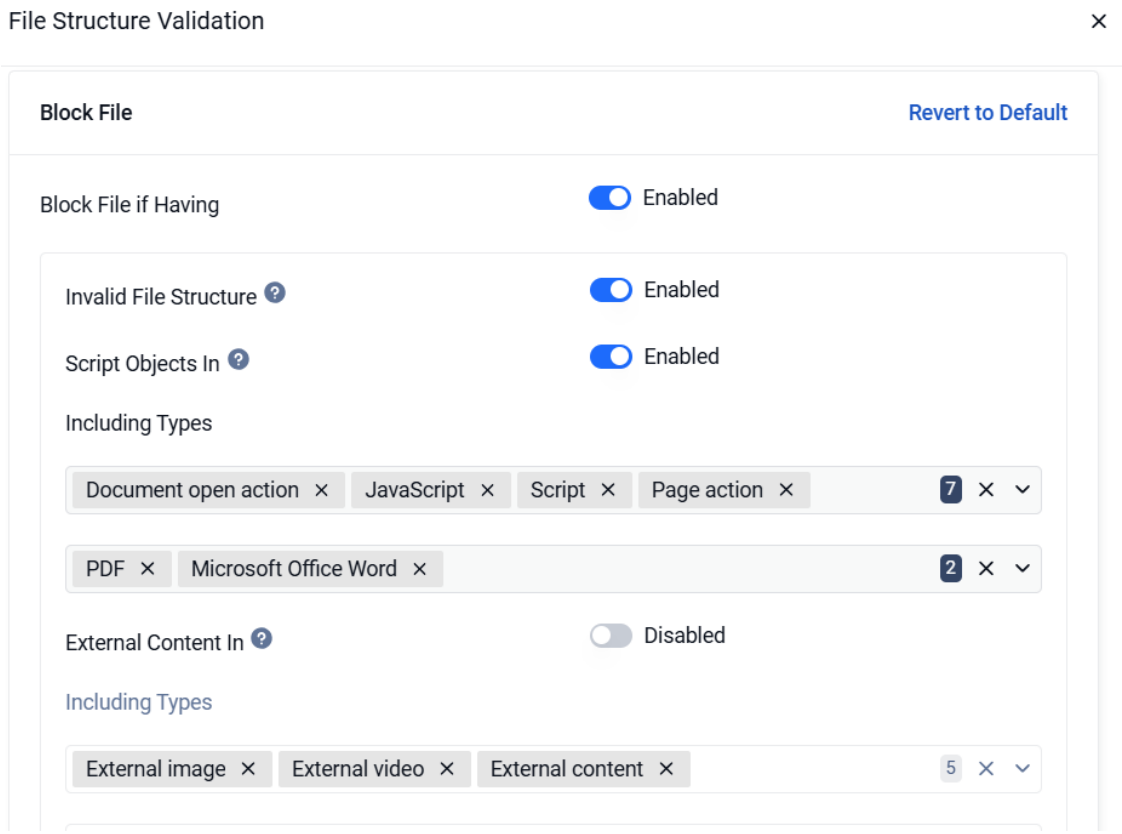
The attachment is extracted by the Archive module and its child file is processed as well.



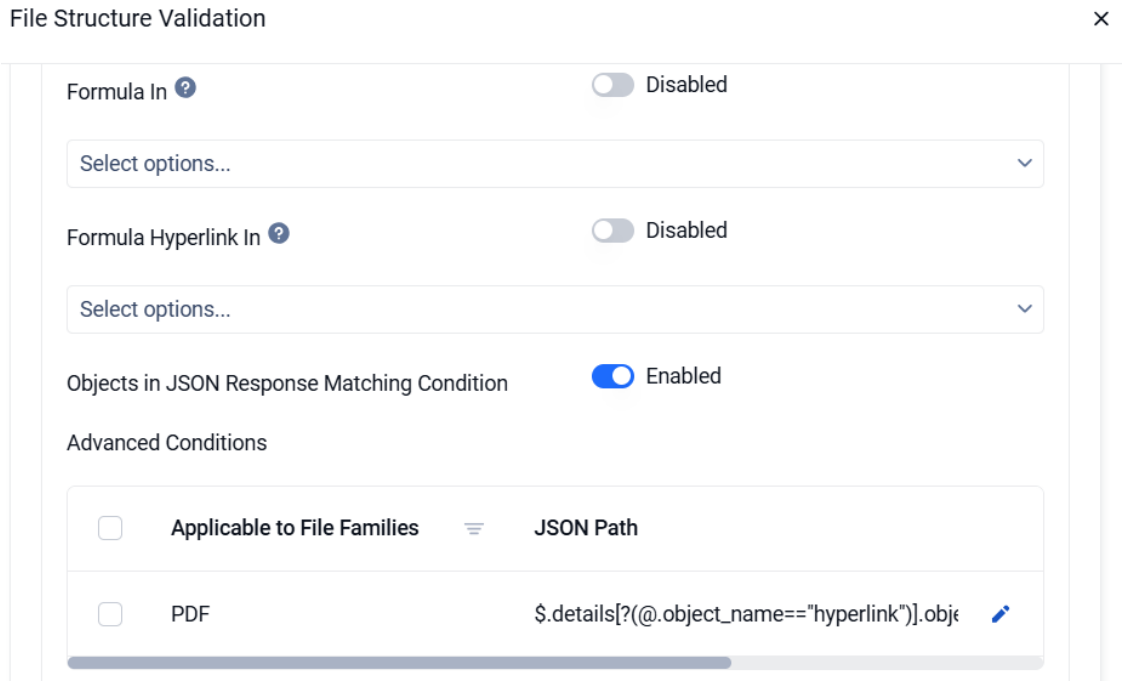
Blocking files based on rules

All blocking rules are off by default, with no rules enabled, FSV only reports what it finds. FSV supports two rule types,

- **Built-in rules**: block on invalid file structure, or on the presence of an object class such as macros, scripts, or embedded objects. These are toggled per workflow.



- **Custom rules**: block on a condition matched against an extracted object's attributes.



For example, block any PDF file whose embedded hyperlink points to `http://malicious.example`.

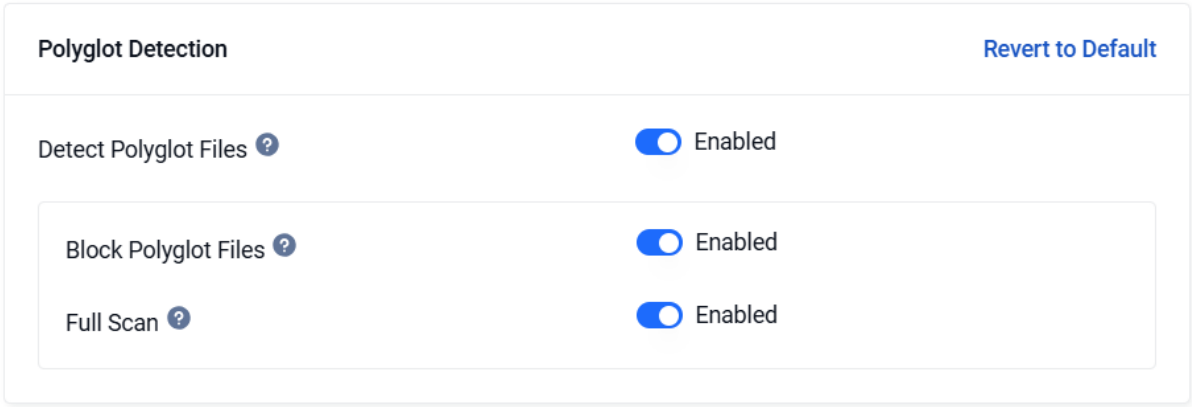
JSON Path	Value Pattern
$.details[?	http://malicious.example

JSON Path	Value Pattern
(@.object_name=="hyperlink")].object_details[*]	

Polyglot detection

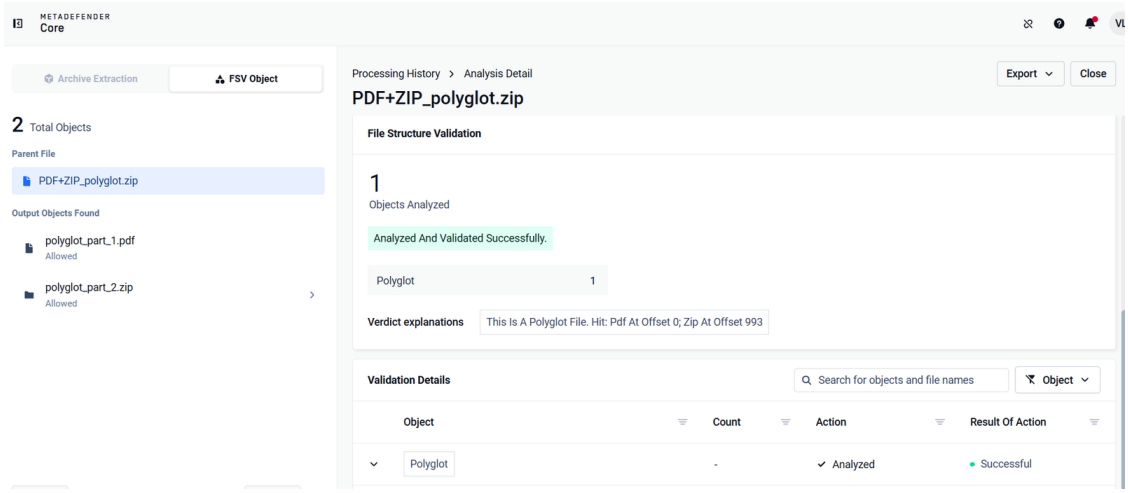
A polyglot file is a single file that is validly interpreted as two or more different formats, for example an image that is also a Java archive, or a PDF carrying a complete ZIP after its logical end. Each face is standards-compliant on its own, so scanners that assign a file exactly one type analyze only that face and the other format rides through untouched.

Settings:



Full scan: Scan the entire file. When off, files larger than 16 MB are scanned only at their first and last 8 MB, trading depth for speed.

Example:

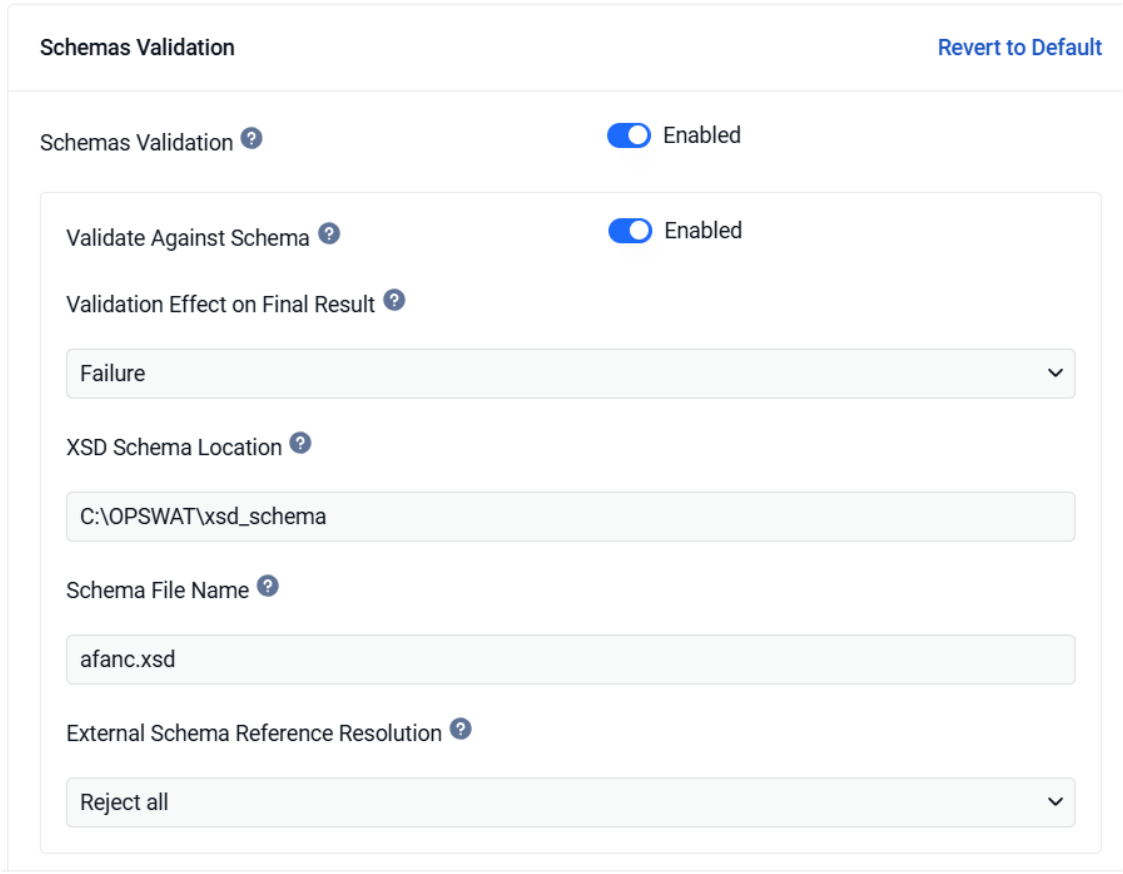


When `polyglot` is among the selected object types, each face is also carved out as its own file — `polyglot_part_1.pdf`, `polyglot_part_2.zip`, … and handed back to the workflow for full processing as what it really is.

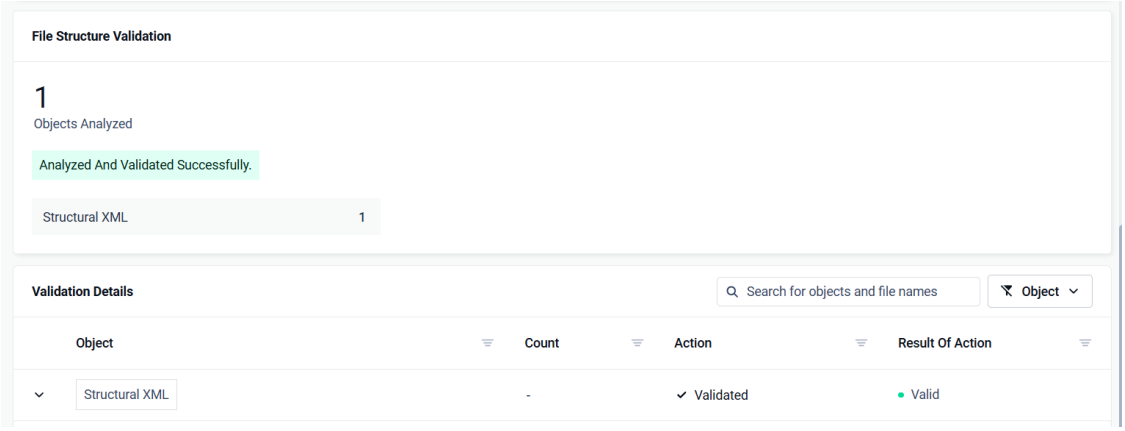
Note: Only stacked-file polyglots are supported for now; other types of polyglots are coming soon.

XML schema validation

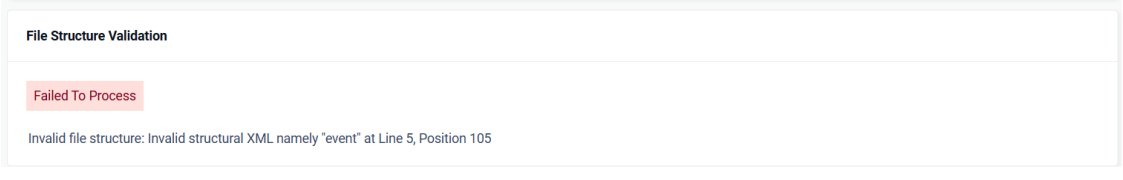
File Structure Validation lets you enforce XML Schema (XSD) validation on XML files. If a file does not conform to the specified schema, it fails validation, and the configured Validation Effect on Final Result determines the outcome of the scan.



A valid XML file that conforms to the schema



An invalid XML file that fails schema validation



Specify the schema through the REST API

When you submit a file through the REST API (`POST /file`), you can choose which XML schema File Structure Validation validates the file against. Pass the schema file name in the `engines-metadata` request header:

Code · json

```
{
  "fsv": {
    "schema_validation.validate_against_schema.schema_name": "<schema file name>"
  }
}
```

Parameter	Description
`schema_validation.validate_against_schema.schema_name`	File name of the XSD to validate against, for example `cargo-exchange.xsd`.

JSON schema validation

Overview

FSV can check JSON files against a JSON Schema that you provide. Before the file is sanitized, the engine validates it against the schema and reports whether it conforms. You choose whether a non-conforming file is only reported or is blocked.

Applies to: JSON, HAR, LYRX.

Schema dialects: Draft 4, 6, 7, 2019-09 and 2020-12, picked from the schema's `$schema` keyword. A schema without `$schema` is accepted; an unrecognised `$schema` value fails with *Schema error*. `format` keywords (`date-time`, `email`, `uri`, ...) are enforced, not just annotated.

Settings

In the Core web console: **Workflows** → open the workflow → **File Structure Validation** → **Schemas Validation** → **Schema-based JSON Validation**.



Quick start

1. Copy your schema files to a folder on the Core server, e.g. `C:\schemas\orders\`:

```
C:\schemas\orders\  order.schema.json      <- root schema  address.schema.json    <- referenced by order.schema.json
```

2. In the workflow rule, enable **Validate against Schema** under Schema-Based JSON Validation.
3. Set **JSON Schema Location** = `C:\schemas\orders` and **Schema File Name** = `order.schema.json`.
4. Pick the **Validation Effect**: start with `Success` to see results without blocking, switch to `Failure` once the schema is trusted.
5. Scan a JSON file and check the result (see below).

Example schema `order.schema.json`:

Code · json

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "object",
  "required": ["id", "amount"],
  "properties": {
    "id":     { "type": "string" },
    "amount": { "type": "number", "exclusiveMinimum": 0 },
    "shipTo": { "$ref": "address.schema.json" }
  }
}
```

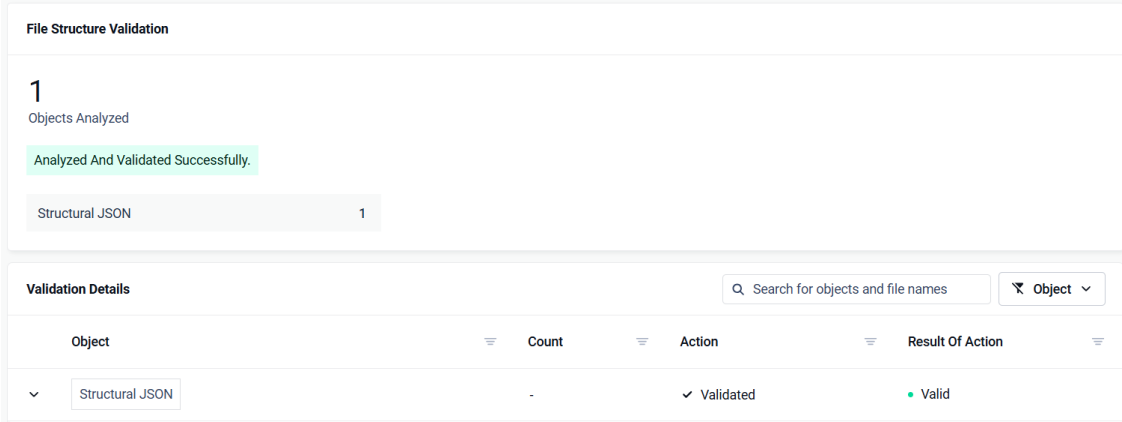
Referenced schemas (`$ref`)

- A `$ref` to another file is resolved **by file name, in the same directory as the root schema**. `address.schema.json`, `./address.schema.json` and `https://example.com/schemas/address.schema.json` all load `address.schema.json` from that directory.
- Nothing is downloaded. To use a schema that references a remote URL, save the referenced file beside the root schema under the same file name.
- A referenced file that is missing or not valid JSON fails the file with *Schema error*.

Results

Results appear in the Core scan result under `filestructurevalidation_info.process_details`, as the processed object `structural JSON` in `details`. A blocked file has the message in `details` and a `failure_category` instead.

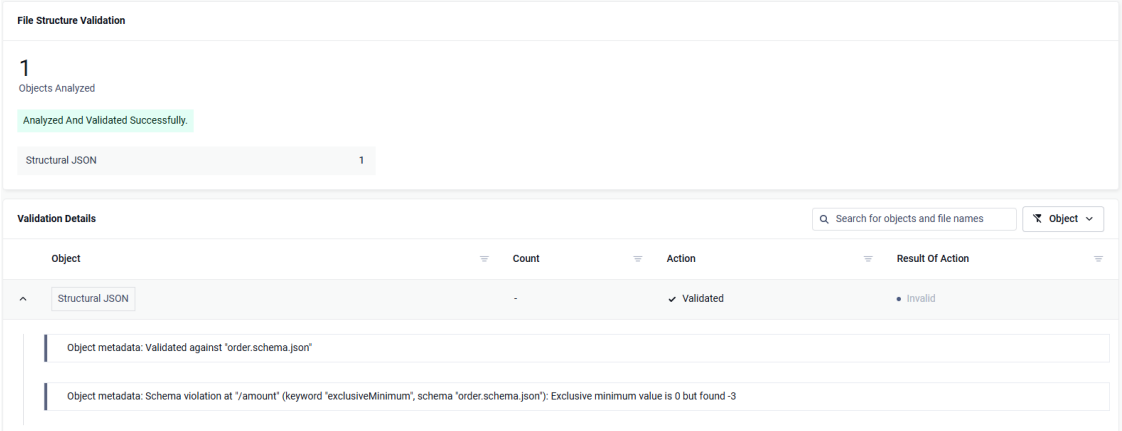
Valid file



Code · json

```
{
  "object_name": "structural JSON",
  "result_of_action": "valid",
  "object_metadata": [ "Validated against \"order.schema.json\"" ]
}
```

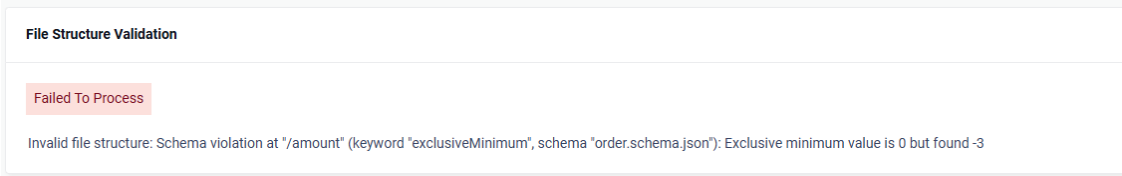
Invalid file, effect = `success` — the file is still allowed; the violation is reported:



Code · json

```
{
  "object_name": "structural JSON",
  "result_of_action": "invalid",
  "object_metadata": [
    "Schema violation at \"/amount\" (keyword \"exclusiveMinimum\", schema \"order.schema.json\"):
Exclusive minimum value is 0 but found -3",
    "Validated against \"order.schema.json\""
  ]
}
```

Invalid file, effect = `failure` — the file is blocked, category *invalid file structure*, with the same message as the failure details.



Message reference

Situation	Category	Details
Instance violates the schema	invalid file structure (effect `failure`) / reported as `invalid` (effect `success`)	`Schema violation at "<JSON pointer>" (keyword "<kw>", schema "<file>"): <reason>` — only the first violation is reported
File is detected as JSON but cannot be parsed (e.g. NDJSON)	invalid file structure (both effects)	Effect `failure`: `Malformed JSON at line L, column C: <reason>`. Effect `success`: `Invalid JSON file structure`
Location empty / missing	failed to sanitize	`Schema unavailable: the schema location is empty` / `... does not exist`
Schema file name empty	failed to sanitize	`Schema unavailable: the schema file name is empty`
Schema file not found	failed to sanitize	`Schema unavailable: the JSON schema file "<name>" does not exist at the schema location`

Situation	Category	Details
Schema is broken or a `$ref` cannot be resolved	failed to sanitize	`Schema error in "<file>"[ at line L, column C]: <reason>`

Schema configuration problems (*Schema unavailable*, *Schema error*) always fail the file, whatever the validation effect, so a misconfiguration is never silently passed.

Specify the schema through the REST API

When you submit a file through the REST API (`POST /file`), you can choose which JSON schema File Structure Validation validates the file against. Pass the schema file name in the `engines-metadata` request header:

Code · json

```
{
  "fsv": {
    "json_schema_validation.validate_against_schema.schema_name": "<schema file name>"
  }
}
```

Parameter	Description
`json_schema_validation.validate_against_schema.schema_name`	File name of the XSD to validate against, for example, for example `purchase-order.schema.json`.

Known limitations

- **NDJSON** (newline-delimited JSON) is not validated per line. With effect `failure` the file is blocked as `Malformed JSON`; with effect `success` it is allowed and reported as `invalid` with the `Malformed JSON` message.
- **UTF-16** JSON files are not validated: the file is allowed and no `structural JSON` entry is reported.

Understanding the Validation Report

FSV results appear in the scan result JSON under `filestructurevalidation_info`, with extracted objects listed under the sibling block `fsv_output_files`.

Code

```
"filestructurevalidation_info": {
  "process_details": {
    "description": "Analyzed and validated successfully.",
    "details": [
```

```
      {
        "object_name": "hyperlink",
        "action": "validated",
        "count": 3,
        "result_of_action": "valid",
        "object_details": ["https://example.com", "..."]
      }
    ],
    "verdict_explanations": []
  },
  "final_verdict": { "verdict": "allowed" },
  "version": "2.0.0"
},
"fsv_output_files": {
  "fsv_output_files_count": 2,
  "files_in_fsv_output": [
    {
      "object_type": "macro",
      "display_name": "vbaProject.bin",
      "file_size": 15360,
      "file_type_id": "..."
    }
  ]
}
```


Verdict

`final_verdict.verdict` is `allowed` or `blocked`. When blocked, `verdict_explanations` lists the reasons in plain language, for example:

- `invalid file structure`
- `contain JavaScript`, `contain DDE`, `contain document open action`, … (one per blocked object class)
- `This is a polyglot file. Hit: PDF at offset 0; ZIP at offset 204849`

Object details

Each entry in `details[]` describes one object class found in the file:

Field	Meaning
`object_name`	The object class, for example `macro`, `hyperlink`, `embedded object`, `metadata`
`action`	What FSV did: `validated`, `extracted`, `detected`, or `skipped`

Field	Meaning
`count`	Number of instances
`result_of_action`	`valid` or `invalid`
`object_details`	Instance data, for example the hyperlink URLs found
`object_metadata`	Spec-level explanation, for example `Graph valid against specification`
`object_sha256`	Per-instance content hashes
`skipped_reason`	Why an object was not processed: `signed_file`, `limit_exceeded`, `marked_as_skipped`, …

With the **deep dive** setting enabled (default), entries also carry structural data — object relationships and offsets inside the file.

Custom blocking rules match against these fields with a JSON path — see Blocking files based on rules.

Extracted output files

`fsv_output_files.files_in_fsv_output[]` lists every object returned to the workflow, with `object_type`, `display_name`, `file_size`, and its detected file type. Which classes are returned is controlled by the selected object types — see Extracting objects and recursive processing.

Failures

A file FSV could not process carries `process_details.failure_category` — most commonly `invalid file structure`, meaning the file violates its format specification badly enough that parsing cannot continue.

Release notes

Version	**5.24.1**
Release date	18 November 2026
Scope	Lets Adaptive Sandbox analyze emails based on the Deep CDR details of their attachments. Aligns the engine update source shown in Central Management with MetaDefender Core and refreshes the Explore section of the Resources page. Upgrades SQLite and fixes two stability issues under heavy load.

Before you upgrade MetaDefender Core:

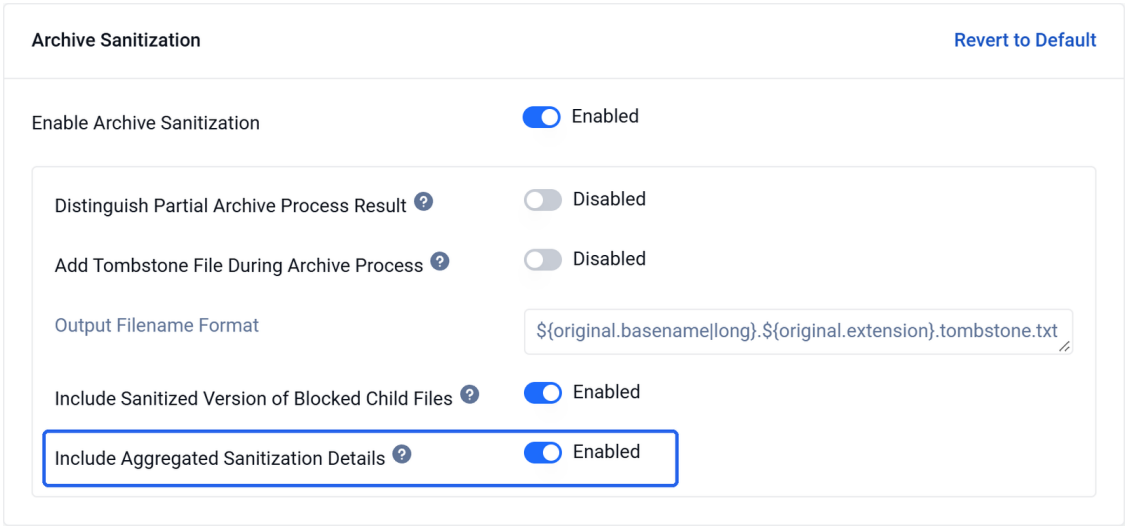
Making sure to check out the Known Limitations.

New Features, Improvements and Enhancements

Adaptive Sandbox for emails whose attachments contain active content

Adaptive Sandbox can now analyze a container file, such as an MSG email, based on its attachments. A new workflow option, **Include Aggregated Sanitization Details**, under Archive Sanitization on the rule's Compression tab, adds the Deep CDR details of each attachment to the result of the container file. With **Attachment** selected in *Enable for Active Content(s) found by Deep CDR* on the Adaptive Sandbox tab, an MSG file that has an attachment is sent to Adaptive Sandbox.

Default: off.



Requirements and limitation:

- Archive Compression and Archive Extraction engines 8.1.3 or later are required.
- The current Adaptive Sandbox release (3.3.1) is supported. No Adaptive Sandbox upgrade is needed.
- Triggering on active content nested inside an attachment or the email body, such as a hyperlink or a macro, is not supported yet and requires a later Adaptive Sandbox version.
- Password-protected Office and password-protected PDF attachments are not yet included in the aggregated sanitization details of an MSG or EML file, so active content inside them does not trigger Adaptive Sandbox for the email; other attachments are processed as usual, and this will be enhanced in an upcoming release.

Further Enhancements

1) Engine update source shown consistently in Central Management

For instances managed by My OPSWAT Central Management, the engine update source in a Central Management policy now reads *My OPSWAT Central Management* instead of *Internet*, matching what the MetaDefender Core console shows.

2) Updated Explore section on the Resources page

The Explore section of the Resources page now includes File Structure Validation, and OPSWAT SBOM has been renamed OPSWAT BOM.

Security Enhancements

Upgraded 3rd-party libraries:

- SQLite v3.54.1
- PostgreSQL v18.7

Bug Fixes

- Fixed an issue where MetaDefender Core could become unresponsive until restarted when many scan or result requests arrived simultaneously shortly after service startup.
- Fixed an issue where MetaDefender Core could consume excessive memory and crash when scanning large archives with Proactive DLP enabled and a high scan queue size (for example, 10,000).

New Prerequisites for Core Modules

Microsoft Visual C++ Redistributable version 14.44 or later is required for version 8.0 of the following modules: File Type, Archive Extraction, Archive Compression, Country of Origin, and Deep CDR.

OS support discontinuation

OPSWAT will discontinue support for the following OS in MetaDefender Core and its associated engines:

- Ubuntu 22.04 since **Apr 2027**.

Known Limitations

Bundled PostgreSQL v18 problem with non-ASCII locale name	PostgreSQL v18 itself cannot initialize its database cluster when the Windows system locale name contains non-ASCII characters. For example, Turkish Windows uses the locale `Turkish_Türkiye.1254`, which contains `ü`. Instruction for safe upgrade at Upgrade to MetaDefender Core 5.21.0 or later on Windows systems with a non-ASCII locale.
Bundled PostgreSQL v18 problem with non-ASCII password	If your bundle PostgreSQL password contains non-ASCII characters (for example `£`, `€`, or accented letters), the upgrade may fail and prevent console login. ASCII special characters such as `&`, `~`, and `\` are not affected. Before upgrading, change the PostgreSQL password to ASCII-only: Database Management **MetaDefender Core v5.22.0** and above will resolve this issue
Displaying issue of Processing History	In MetaDefender Core v5.21.0, PostgreSQL has problem with some special characters in results returned by engines. As a consequence, Processing History on web console becomes empty in those cases. This is a displaying issue; your data in database is retained as is, and REST API integrations work properly. **Upgrading to MetaDefender Core v5.22.0** will resolve this issue.
Some Settings Not Editable When Managed by Central Management (CM7) in version v5.21.0	When MetaDefender Core v5.21.0 is managed by Central Management v7, the following configurations

	are read-only and cannot be modified: • Post Actions • External Scanners • Webhook Authentication To modify these settings, temporarily disconnect the MetaDefender Core instance from CM7, apply the required configuration locally, then reconnect the instance.
Kubernetes v1.35 or containerd v2.2.0 could not deploy MetaDefender Core images	This issue is a bug of `containerd` - containerd issue reported on GitHub. Until the vendor provides a fix, use one of the following mitigations: • Downgrade containerd to a supported version (e.g., 1.7.x) • Use a Kubernetes node image that does not include containerd 2.2.x • Pin node runtime version in cluster provisioning More details at Unable to deploy MetaDefender Core in Kubernetes with containerd engine 2.2.x
Slow or Inaccessible Management Console	**This issue has been resolved in version 5.13.2** In version 5.12.0, an issue was identified that caused some APIs to load more slowly than expected. As a result, the Web Management Console might experience slower performance or become unresponsive Please read more details on this page: Slow or Inaccessible Management Console.
The 'Proxy server requires password' setting cannot be disabled once it has been enabled	**This issue has been resolved in version 5.14.2.** In version 5.14.1, there was an issue that prevented disabling the `Proxy server requires password` setting once it has been enabled. As a workaround, you can export the current settings, locate and remove the username and password fields under the relevant proxy configuration, and then import the modified configuration.

Database connection failure occurred in a specific circumstance after upgrading to version 5.11.0	**This issue has been resolved in version 5.11.1.** This issue does not affect all cases when upgrading to version 5.11.0. After applying the authentication method `scram-sha-256` to enhance security for the bundled PostgreSQL, a database connection issue started occurring after the upgrade, in a specific circumstance. • If the application was previously upgraded from version 5.5.1 or older to version 5.6.0 or newer, this issue will occur when users upgrade to version 5.11.0. We prepare a Knowledge Base (KB) for troubleshooting the issue and bringing the system back online: How to troubleshoot an error related to connection to database failing after an upgrade to v5.11.0? The issue will not occur in the following scenarios: • Upgrading directly from version 5.5.1 or older to version 5.11.0. • Upgrading from a fresh installation of version 5.6.0 or newer to version 5.11.0.
Archive compression may fail with very large archive files that contain a large number of subfiles	**This issue has been addressed in version 5.14.0.** MetaDefender Core has a limitation when compressing very large archive files that contain a high number of subfiles. In our test scenario, it failed when processing an archive with 300,000 or more subfiles.
Reuse processing result by hash might be slow in high-load situations	**This issue has been resolved in version 5.10.1.** Since its introduction in version 5.8.0, this feature has helped improve overall performance and reduce significant load when processing similar files. However, we have realized this feature might run slowly in high-load scenarios against large database sizes.
Temporary files in the resource folder may not be	Starting from MetaDefender Core version 5.10.1, if

properly cleaned up if the Archive Extraction engine crashes	the Archive Extraction engine crashes, temporary files from specific extraction transactions may not be properly cleaned up. However, this issue is relatively rare.
Reject importing non-empty required_engines setting in containerized environments	This issue occurs only in containerized environments. If the config zip file includes non-empty **required_engines** setting, MetaDefender Core will reject the import. **Workaround:** 1. Extract the config zip file. 2. Open the "`export_settings.json`" and set "`required_engines`" to an empty array. 3. Recompress the files into a new zip. 4. When executing the docker run command, set the following environment variables: `MDCORE_HEALTH_CHECK`, `MDCORE_REQUIRED_ENGINES`. For more details, please refer to Health Check settings on docker
The Engine Update feature may not work as expected in certain environments	We have observed that the Engine Update feature may not work properly in an environment protected by a Palo Alto firewall. In the log file, you might find the error message '`SslHandshakeFailedError`'. If upgrading to the latest version of MetaDefender Core does not solve the issue, please consider setting up MetaDefender Update Downloader product. This product is responsible for downloading engines, and MetaDefender Core will retrieve and update its engines from there.
Stability issues on Red Hat / CentOS systems with kernel version 372.13	MetaDefender Core version 5.2.1 or later may not function correctly with Red Hat or CentOS operating systems that use kernel 372.13. Red Hat is addressing the kernel issues. Please try upgrading to kernel version ***372.26***.
PostgreSQL and MetaDefender Core services cannot initialize in certain containerized environments	**This issue was addressed in version 5.11.1.** In a containerized environment, MetaDefender Core

	version 5.2.0 or newer may work properly when: • The Linux kernel version of the host machine is newer than 4.18.0 including 5.x.y and 6.x.y. • The Docker base image is CentOS 7. • The bundled PostgreSQL database is used (`DB_TYPE=local`). **Workarounds for older versions:** 1. Switch to using a Docker base image RHEL 8 or Debian. 2. Switch to using a remote PostgreSQL database.
MetaDefender Core's NGINX web server will not start if weak cipher suites are used for HTTPS	On MetaDefender Core version 5.2.0 and later, OpenSSL 1.x has been replaced by OpenSSL 3.x within the product and its dependencies, including PostgreSQL and NGINX, to enhance security and address known vulnerabilities in OpenSSL 1.x. However, NGINX's implementation of OpenSSL 3.x in MetaDefender Core enforces strong encryption by rejecting all weak cipher suites. It only accepts "HIGH" encryption cipher suites as defined by OpenSSL https://www.openssl.org/docs/man1.1.1/man1/ciphers.html. This means ciphers based on MD5 and SHA1 hashing are no longer supported. Consequently, if you previously configured MetaDefender Core for HTTPS connections using a weak SSL cipher with your certificate, the service will not start due to NGINX's OpenSSL 3.x security enforcement. To prevent and remediate the issue before upgrading MetaDefender Core, please refer to the following resources: HTTPS Failure on MetaDefender Core 5.2.0 (or newer).
TCP socket port exhaustion may cause the service trouble, preventing from restarting, and Workflow configuration corrupted	This issue affected MetaDefender Core (MD Core) version 5.15.0 and earlier and is enhanced starting from version 5.15.1. TCP socket port exhaustion might be triggered by other applications; for example, MetaDefender KIOSK v4.7.6.3514 (fixed in later releases).

	Consequently, MD Core may behave abnormally, corrupt its Workflow Configuration, and fail to restart.
Workflow configuration fails to synchronize from OPSWAT Central Management to MetaDefender Core after creating a new Workflow template	This issue affects MetaDefender Core versions 5.17.0 and 5.17.1. Workflow configuration from OPSWAT Central Management will fail to synchronize to MetaDefender Core (MD Core) once a new Workflow template is created. To restore normal synchronization, the newly created Workflow template must be deleted. As a workaround for creating new templates on these affected MD Core versions, the Clone Workflow Template feature can be used as an alternative.
Temporary files may persist if an OPSWAT AI Content Inspector scan is canceled	If a scan is canceled while the OPSWAT AI Content Inspector engine is actively processing, intermediate files may remain in the temporary directory. These files do not affect the current or any subsequent scan results. **Resolution:** This issue has been fixed in **MD Core 5.20.0**. **Workaround (if you remain on MD Core 5.19.0):** The temporary files are automatically removed when the OPSWAT AI Content Inspector engine restarts.

Archived release notes

Version v5.24.0

Keep instance-specific settings under a Central Management group policy

A My OPSWAT Central Management (MOCM) group policy can now leave selected settings to each instance, while every other setting continues to sync from the policy. A new exclusions tab in the policy covers:

- **HTTPS certificate:** each instance keeps its own host-specific certificate instead of receiving one certificate for the whole group.
- **Adaptive Sandbox:** each instance keeps its own Sandbox connection, such as the server address and API key, so instances in the same group can use different Sandbox servers.

Turning an exclusion off applies the policy values again on the next sync, with no need to re-enroll the instance.



Splunk configuration from Central Management

Splunk integration can now be configured from My OPSWAT Central Management for instances it manages. In the setup wizard, the Splunk step now comes before Central Management enrollment and is skipped once the instance is enrolled, since Central Management then owns that setting.

Engine lifecycle notifications

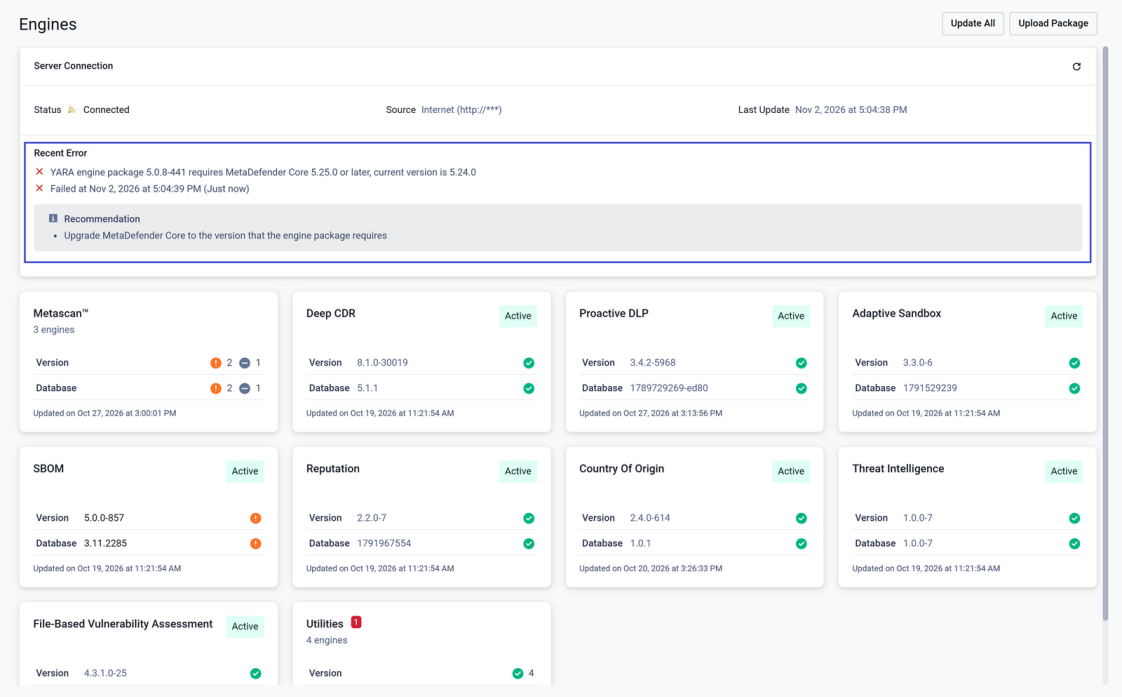
Administrators are now told when an engine needs attention:

- **End of support:** a console notification appears when an engine is near or past its end of support.

image



- **Newer MetaDefender Core required:** an engine package that requires a newer version of MetaDefender Core is no longer installed, whether it arrives by download, manual upload, or pickup folder, and the current package stays in service. The rejection is shown in the update status on the Engines page, including when updates are applied manually, with a recommendation to upgrade instead of retrying. An installed engine that needs a newer version shows a warning on its tile.



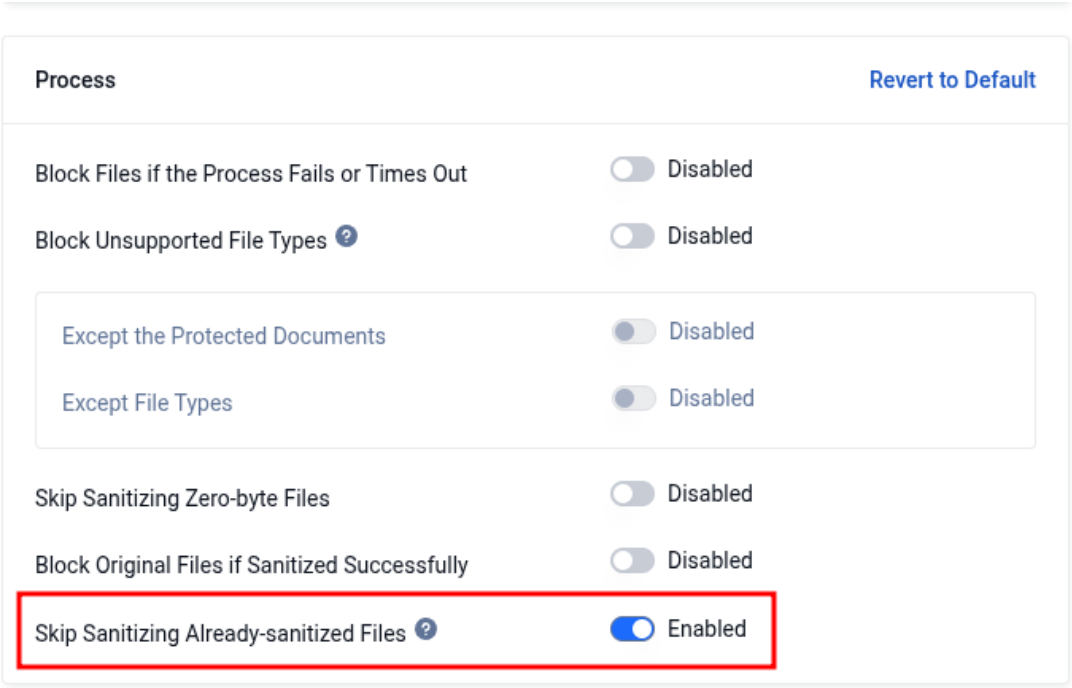
Percentile processing-time metrics in the Executive Report

The Executive Report and related dashboard views add P50, P90, P95, and P99 processing-time metrics. A few slow objects no longer hide typical processing speed behind average and maximum alone



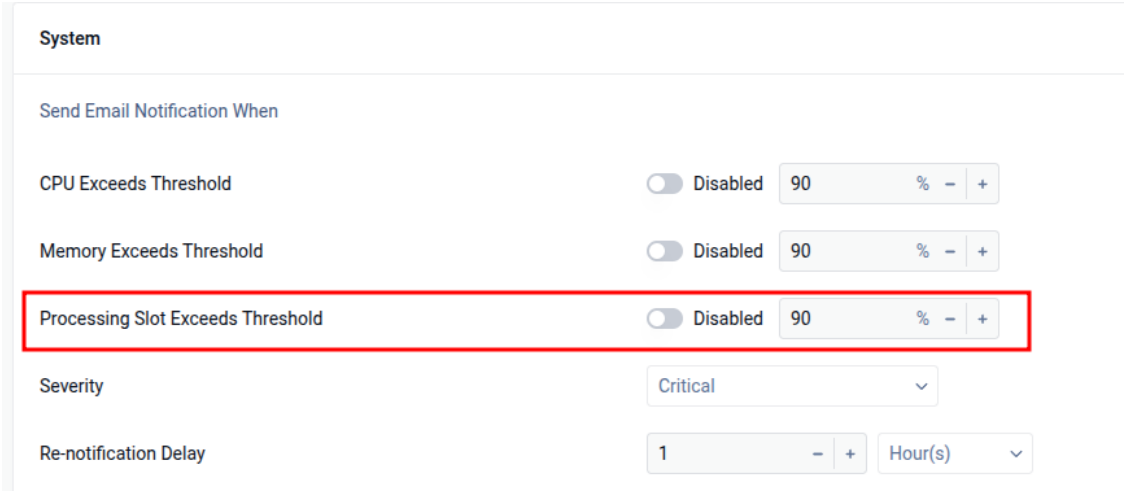
Optional skip for already-sanitized Deep CDR output

Administrators can enable an option so Deep CDR does not sanitize a file again when Core already produced a sanitized result. The option is off by default. It helps MetaDefender Kiosk and USB repeat-scan flows avoid growing piles of sanitized copies. Details: Skip Sanitizing Already-sanitized Files



Email notification when processing slots exceed a threshold

Administrators can configure alerts when the scan queue or processing-slot usage crosses a defined threshold. Alerts can go by email and surface in the dashboard



Further Enhancements

1) Reliable reporting to Central Management over slow connections

For deployments managed by My OPSWAT Central Management across slow or long-distance connections, reports are now sent reliably during continuous, high-volume scanning. The local report buffer uses far less disk space and memory, and reporting no longer stalls for long periods under load.

2) Centrally managed certificates in Central Management policies

The certificate fields of a My OPSWAT Central Management policy, for Secure Connection and for batch signing, now list the certificates assigned to that policy. Each instance in the policy uses its own copy of the selected certificate,

even when the instance stores it under an adjusted name.

3) Skip by Hash expiration from Central Management

The Skip by Hash list now shows the expiration date that My OPSWAT Central Management sets on a centrally managed entry, so administrators can see when a temporary approval or block ends. After the upgrade, existing centrally managed entries are refreshed once so they show their expiration.

4) File Structure Validation works with Adaptive Sandbox

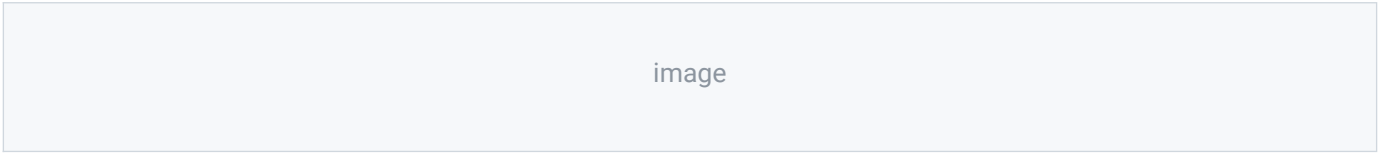
File Structure Validation findings, such as macros and hyperlinks, can now decide whether a file is sent to Adaptive Sandbox, the same way Deep CDR findings already can. Organizations that rely on File Structure Validation instead of Deep CDR for this decision keep the same Sandbox filtering behavior. Files that File Structure Validation extracts are also scanned as soon as the engine returns them, for engine versions that return results during processing.

5) Add a certificate and its private key from a single file

Adding a certificate by file upload now accepts a single file that contains both the certificate and its private key, so a combined certificate file can be used as it is delivered.

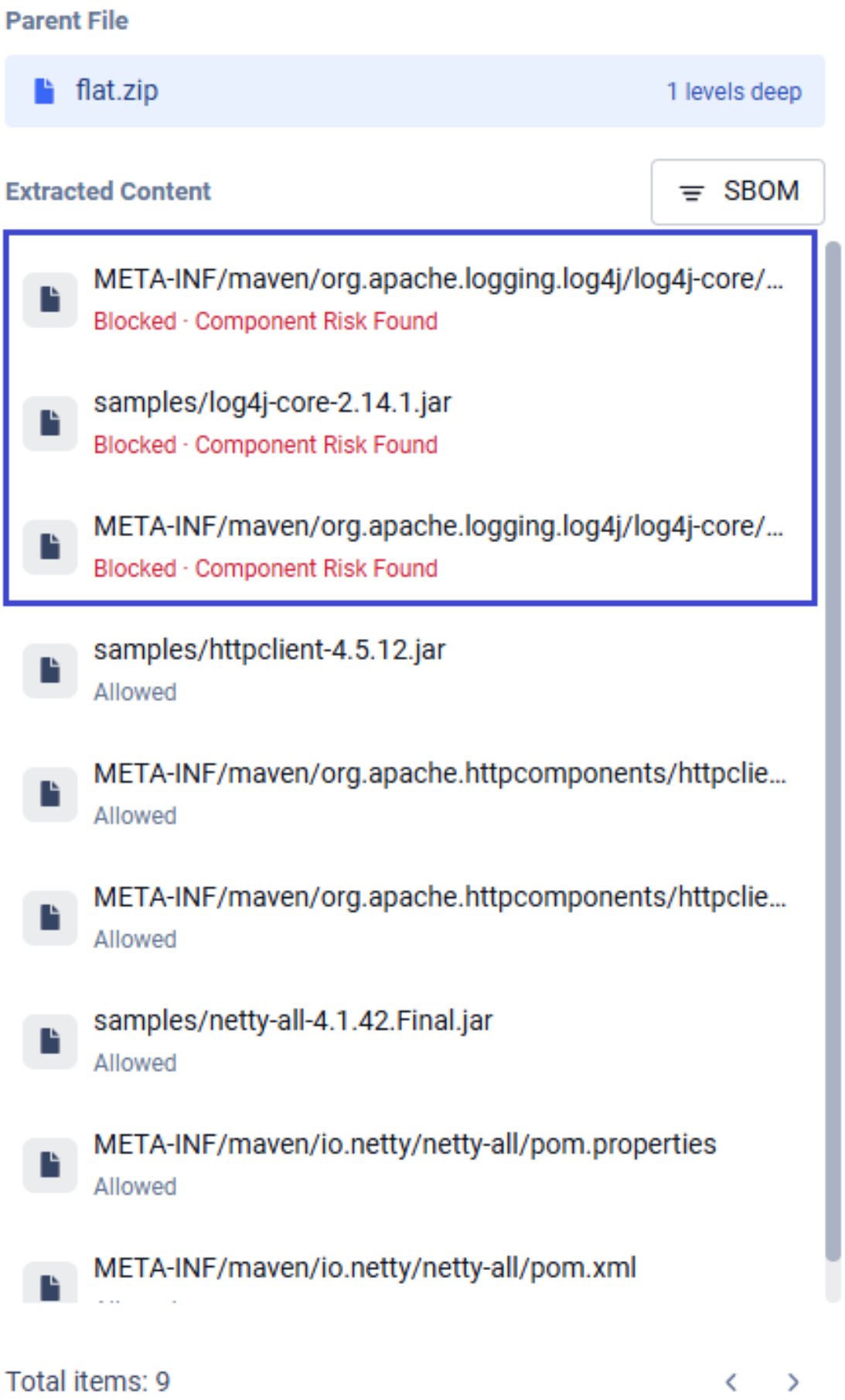
6) Clearer Deep CDR summary on the processing result

The Deep CDR card on the processing result now leads with the total number of objects processed, instead of the sanitized count alone, which often read as zero because Deep CDR mostly removes objects.



7) Vulnerable packages listed first in the SBOM filter

When the SBOM filter is selected in an archive's extracted file list, files with vulnerable packages are now listed first, so they no longer have to be found among thousands of clean entries.

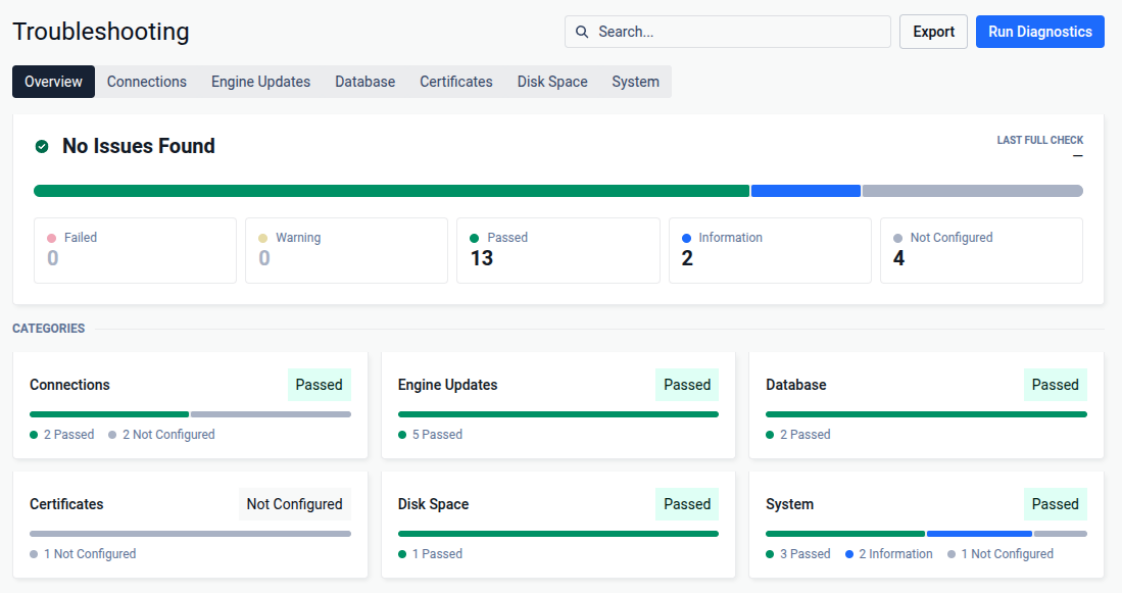


8) Full engine name in the File Processing Time chart

The File Processing Time chart on the dashboard now labels File Structure Validation by its full engine name in the legend and tooltips, consistent with the other engines in the chart.

9) Clearer Troubleshooting page layout

Expanded Troubleshooting content is easier to read and navigate in the Management Console.



10) File name redaction in logs (data ID instead of file name)

A setting can replace scanned file names in logs with the scan data ID. This reduces exposure of sensitive names in SIEM and log exports while keeping correlation to the scan record

11) Usage Report settings included in configuration export

Usage Report settings are included when you export Core configuration, so container and automated deployments can preserve those choices across environments.

Security Enhancements

- Several issues found in an internal security review have been addressed, tightening access control on administrative and configuration operations and strengthening input validation.

- Upgraded 3rd-party libraries:
  - Protobuf: upgrade to v33.5
  - gRPC: upgrade to v1.84.0

Bug Fixes

- Fixed an issue where a split archive submitted with a workflow rule other than the default was processed with the default rule's settings, although the result showed the selected rule.
- Fixed an issue where a request refused through a My OPSWAT Central Management connection returned the same generic error for every request, instead of naming the refused request.

- Core can start the bundled PostgreSQL in immutable, non-root containers with a read-only root file system when FIPS-related OpenSSL config must be written to a writable location
- Upgrading to 5.23.0-era builds could add a MetaDefender Managed File Transfer rule bound to a disabled security zone, which blocked workflow and zone saves with “Rule refers to invalid zone”. That upgrade path is corrected
- Importing Skip by Hash CSV lists larger than about 10,000 entries completes reliably. Comments with unpaired UTF-16 surrogates no longer reject the whole batch with HTTP 500
- A crash path in `ometascan` while scanning and viewing Processing History results was fixed.
- Post Action webhooks use the network manager from its owner thread, which removes related Qt warnings from syslog

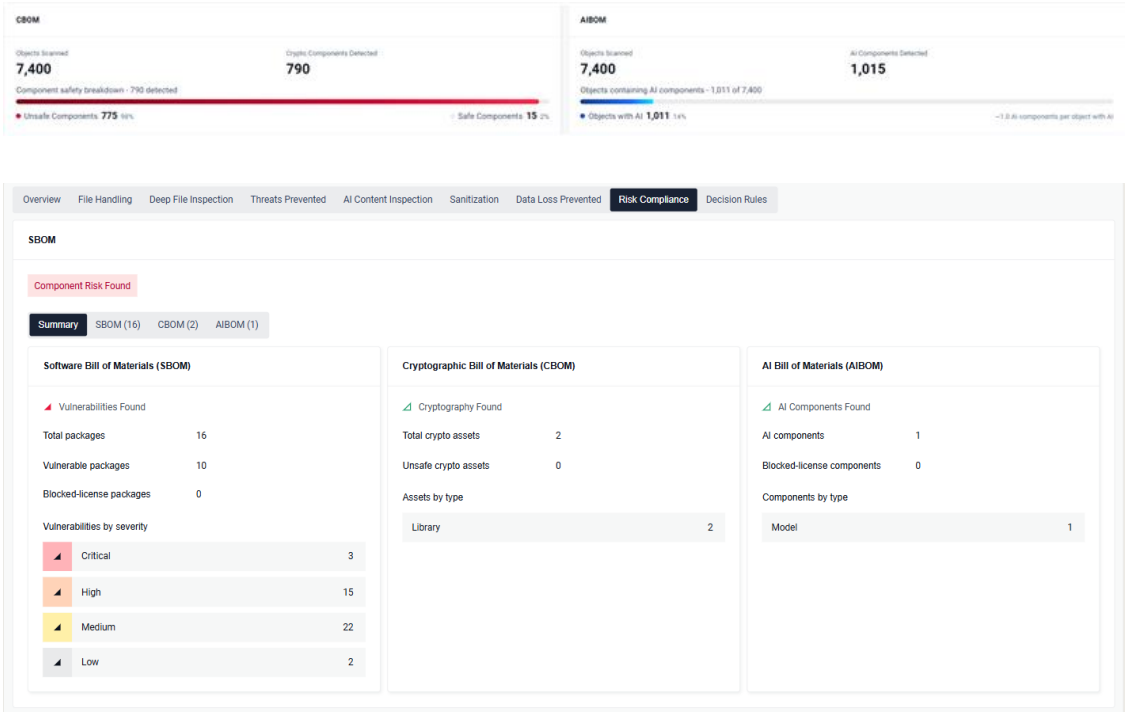
Version v5.23.0

Release Date: 05 October 2026

Beyond SBOM: CBOM and AIBOM detection

The Bill of Materials engine now supports CBOM and AIBOM detection. CBOM inventories cryptographic assets and evaluates quantum resistance, while AIBOM identifies AI models and associated licenses.

CBOM and AIBOM findings appear in processing results and on the Security dashboard. Results for individual files, including files in archives and batches, can be exported as JSON, PDF, or HTML.



Retrieve and manage quarantined files from Central Management

Administrators can now download quarantined files from My OPSWAT Central Management (MOCM) without accessing the instance console or opening an inbound port. This supports restricted environments where instances permit only outbound connections.

Files are delivered on demand through the existing Central Management connection as password-protected archives. Administrators can also protect, unprotect, and delete quarantined items using the same permissions

required by the instance console.

Password Storage synchronized with Central Management

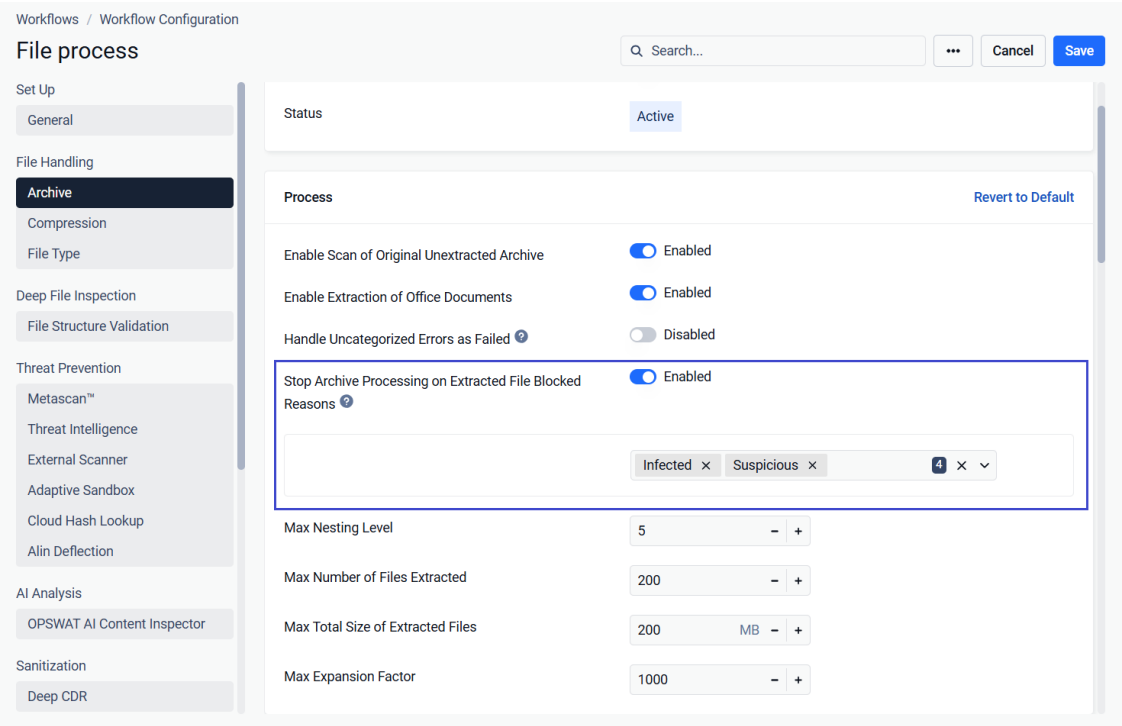
Password Storage is now synchronized from MOCM to enrolled instances, similar to post actions and external scanners. Workflow policies that reference a password storage can therefore be applied consistently across an instance group.

Centrally managed storages are labeled by origin, remain read-only on the instance, and are preserved when a configuration is imported.

Stop processing an archive as soon as a blocked file is found

A workflow can now stop further archive extraction and scanning as soon as an extracted file is blocked for a configured reason, reducing unnecessary processing.

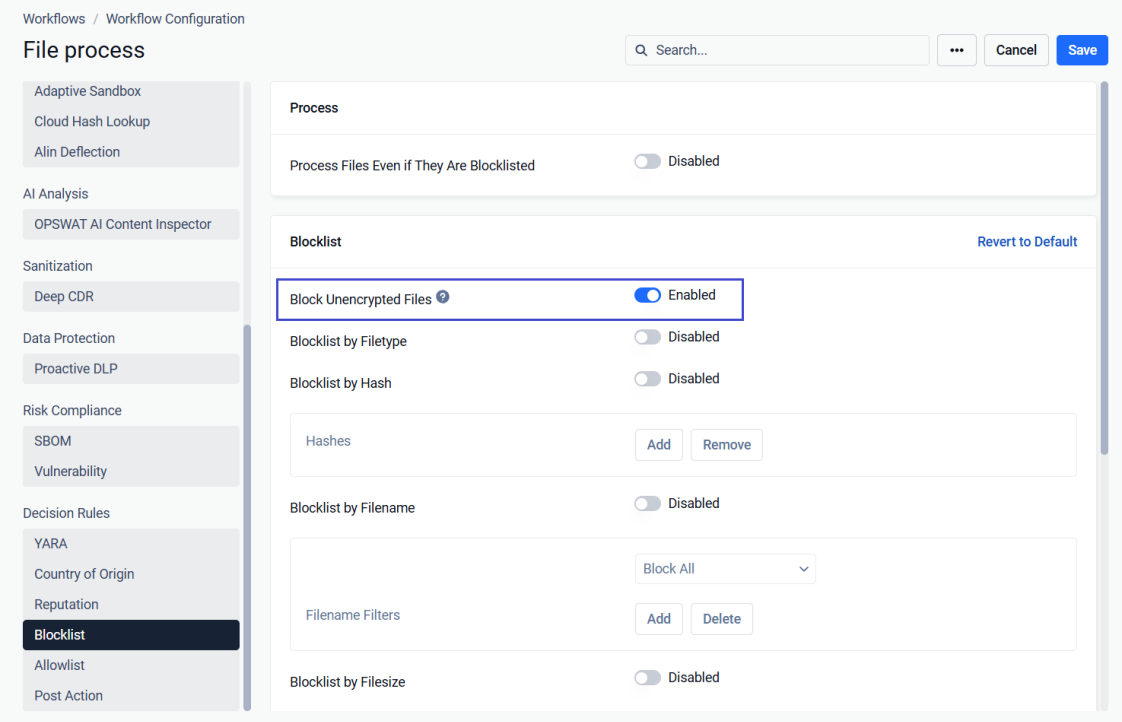
Files already being scanned complete their current stage. The option applies independently to each archive.



Accept only password-protected files

The new **Block unencrypted files** workflow option blocks files that are not password protected with the reason *Password Protection Required*. Password-protected files are processed when the correct password is provided and blocked when no password is supplied.

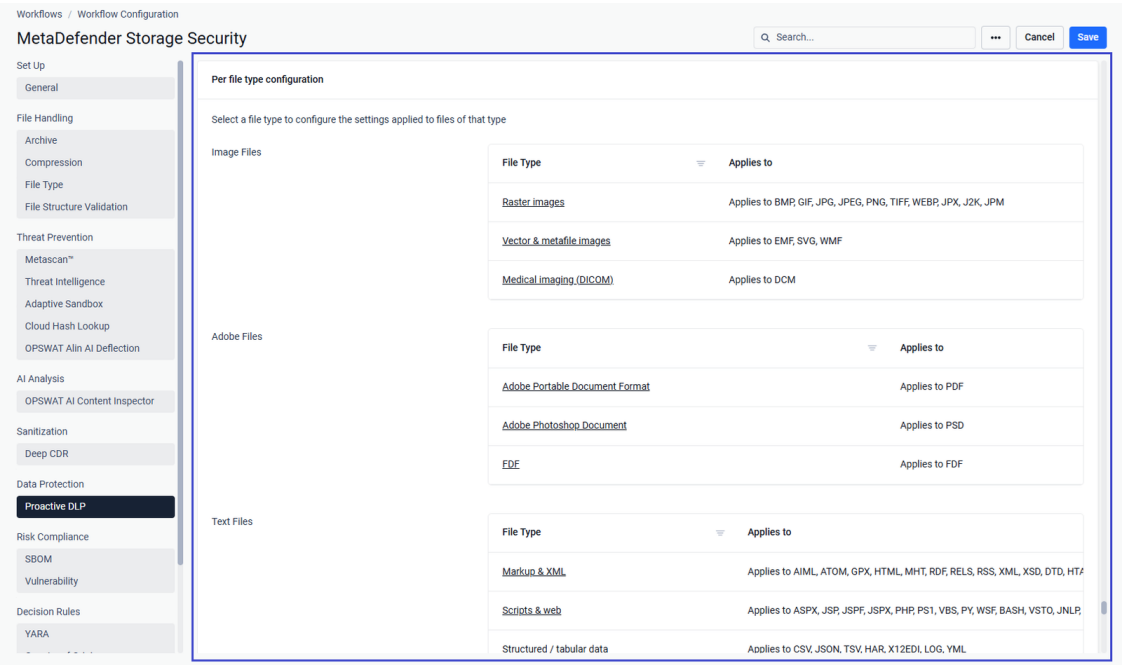
This option is available for MetaDefender Core standalone deployments.



Per-file-type configuration for Proactive DLP

Proactive DLP detection and redaction settings can now be configured by file type, allowing different policies for formats such as PDF and Office documents.

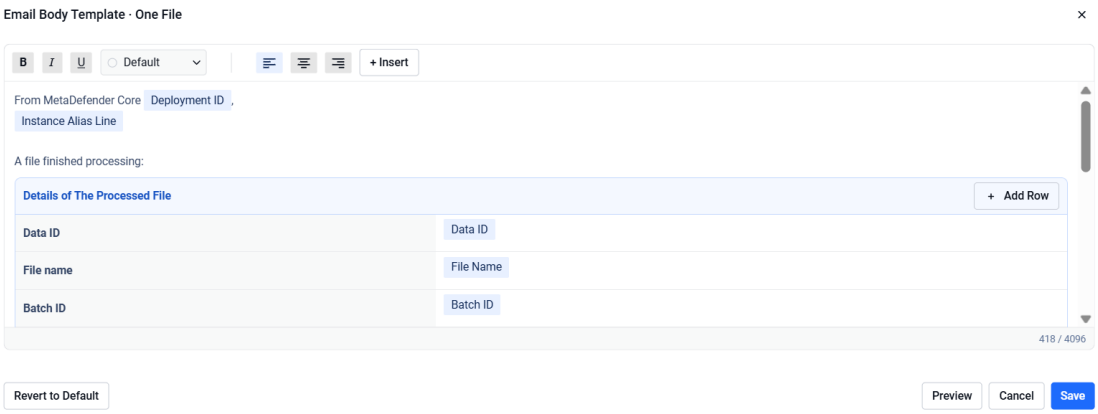
Per-file-type settings apply when Proactive DLP is enabled for that file type and require a supported Proactive DLP engine version.



Customize Core email notification templates

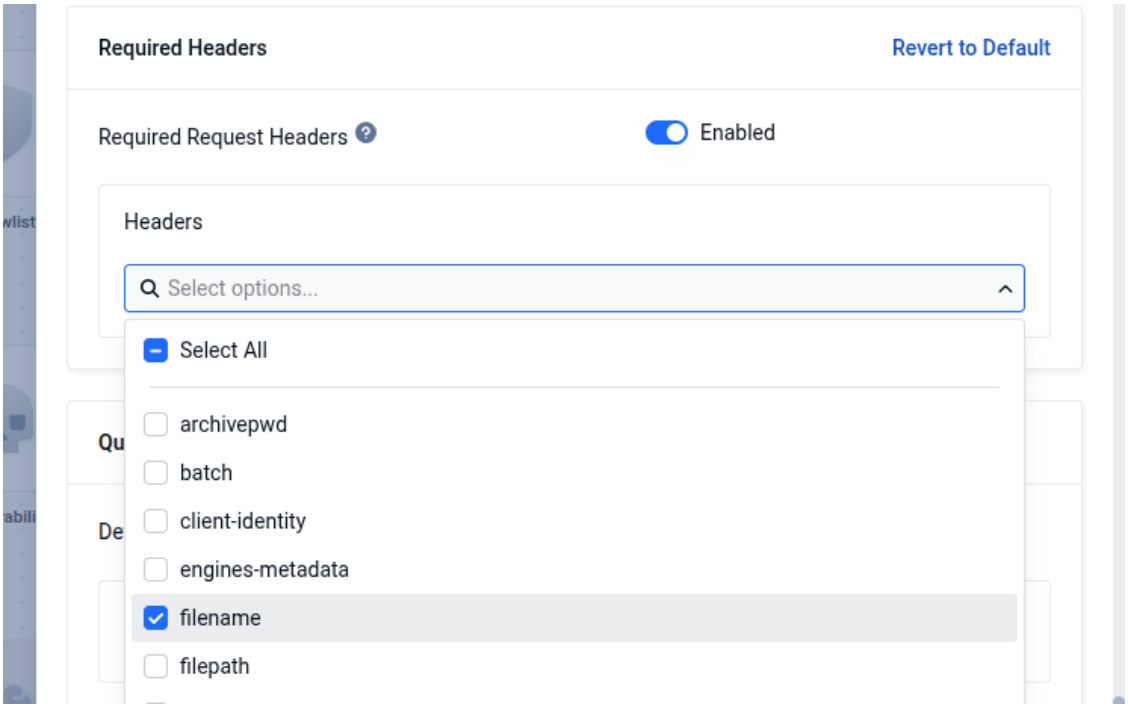
Core email notifications can use customized message templates instead of only the built-in wording.Administrators can adapt notification content to their organization's language, branding, or operational guidance.This covers the various notification types Core already sends, so outbound mail matches local process needs.

Please check for more details: Email Template



Enforce required headers on file analysis API

Administrators can configure which HTTP headers must be present on file analysis requests (for example `filename`).If a required header is missing, MetaDefender Core rejects the submission with a clear API error instead of accepting an incomplete request. This keeps Processing History and audit trails identifiable when business units submit files in bulk.The list of required headers is configurable so each deployment can enforce only what it needs.



Further Enhancements

1) Large SBOM and vulnerability reports without the wait

SBOM and vulnerability reports can now be generated in the background as PDF or HTML. Completed reports are stored and reused instead of being generated again.

A new reports tray shows progress and provides download or retry actions from any console page. The export timeout, previously fixed at ten minutes, is now configurable in Settings.

2) Clearer overall results when no AV engine is licensed

For deployments without licensed antivirus engines, a new workflow option allows the overall result to reflect engines that analyzed the file, such as Adaptive Sandbox, instead of returning *Not Scanned* solely because no antivirus engine participated. Existing behavior remains the default.

3) More accurate File Structure Validation results

Internally generated files, empty directories, and symbolic links that are intentionally not scanned are no longer included when File Structure Validation results are rolled up to the original file. The result view also opens the tab containing the issue first.

4) Faster processing history at scale

Per-engine summaries are now calculated when scan results are stored rather than on each request, improving Processing History performance for large deployments and bulk integrations such as MOCM.

5) Processing History stays readable whatever an engine returns

Engine results are sanitized before storage so unusual characters no longer cause blank threat details in Processing History.

6) Deep File Inspection

File Structure Validation is now named **Deep File Inspection** throughout the console. It has a dedicated workflow section, processing-result tab, and Processing History column.

9) Clearer Adaptive Sandbox license notice

Adaptive Sandbox now shows a license notice only when the daily limit is reached, including the limit and remaining scans. The previous generic partial-data notice has been removed.

10) Active performance statistics by source system

Integrators can include a reserved source identifier in scan metadata to filter active performance statistics by submitting system and detect when a source stops submitting files.

11) Email notification when processing time exceeds a threshold

A new email notification can alert administrators when a file's processing time crosses a configured threshold. The threshold is set in Core email-notification settings (for example 60 minutes). The scan itself continues; the alert is informational so operators can investigate slow or stuck work.Useful for monitoring long-running or queue-backed workloads without watching the UI continuously.

12) Timeout filters on History Scheduled Reports

History Scheduled Reports can filter on timeout-related verdicts. Administrators can schedule reports (and related alerts) that surface files which timed out or were not completed in time.

13) Redact license key in logs

An option can redact the license key wherever it appears in log output. When enabled, only the last four characters remain visible for identification; the rest is masked.

Security Enhancements

- Archive passwords kept out of debug logs: passwords retrieved from Password Storage to open protected

archives are no longer written to debug logs.

- Log file settings from Central Management are validated: log file name validation now also applies to settings pushed from My OPSWAT Central Management.
- Upgraded 3rd-party libraries:
  - OpenSSL v3.5.9
  - NGINX v1.30.5
  - 7-Zip v26.03

Bug Fixes

- Fixed an issue that caused a server error after uploading a split archive part larger than 2 GB.

Version v5.22.1

Release Date: 15 September 2026

Upgraded 3rd party libraries:

- OpenSSL v3.5.8
- Libxml2 v2.15.4

Bug Fixes

Fixed a set of displaying issues introduced with the all-new MetaDefender Core interface in version 5.21.0, where the processing details for several engines showed less information than the previous interface. The following items have been restored and corrected:

- Deep CDR: the *Verdict Explanations* that state why a file received its verdict are shown again on the Sanitization tab, and rows in the sanitization details table can no longer be clicked when they have no further details to show.
- Sandbox: the scan settings tab is available again, the SHA-256 hash and file type are populated in extended details, the severity filter is matched by a visible column, long YARA rule tags are no longer cut off, the severity heading now reads *Verdict*, and the list of detected formats matches the previous interface.
- Proactive DLP: data types with long names are displayed in full, text no longer overlaps, tags are spaced correctly from the results table, the results list respects the selected number of items per page, and the message shown for an unsupported file type links to the list of supported file types again.
- Post actions: the post actions list in the processing result respects the selected number of items per page instead of showing every entry at once.

Version v5.22.0

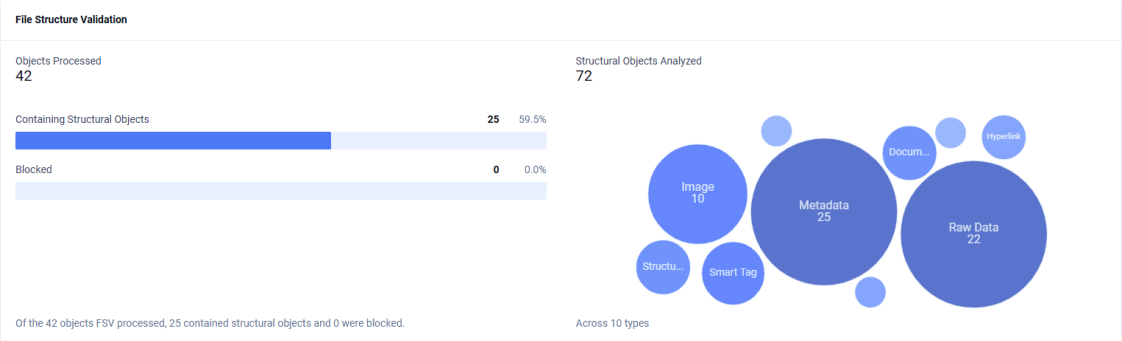
Release Date: 26 August 2026

New File Structure Validation Engine

Malicious content increasingly hides within document structures where traditional signatures cannot reach. **File**

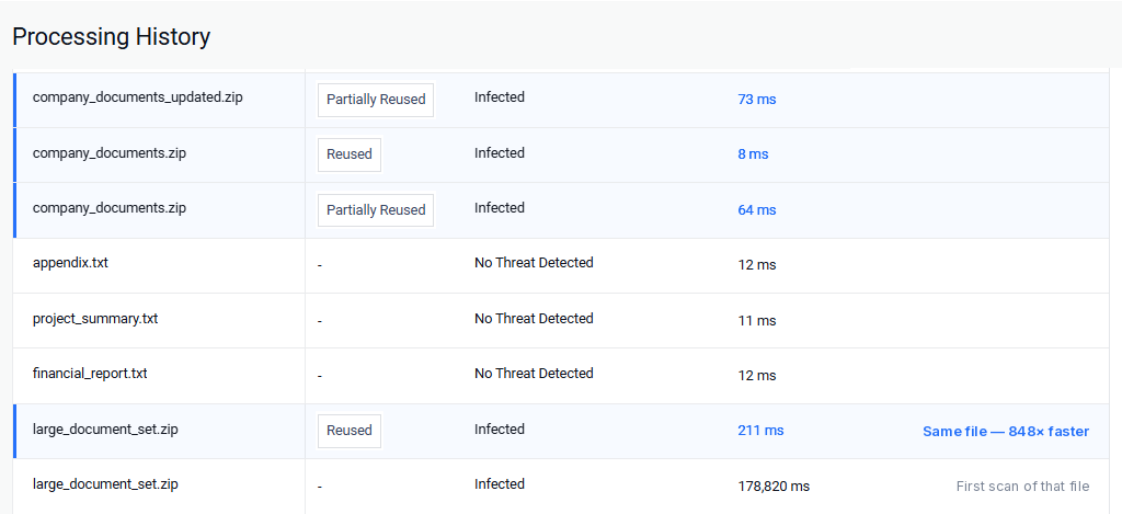
Structure Validation (FSV) inspects each file against its format specification and identifies embedded files, images, fonts, hyperlinks, metadata, and other elements. It enforces structural security policies without modifying the original file, while extractable content is scanned separately so content is not judged by its wrapper alone.

FSV is integrated across MD Core: per-file-type control in workflow configuration, structural findings in the processing result, activity on the dashboard, and a dedicated section in both PDF and CSV executive reports.



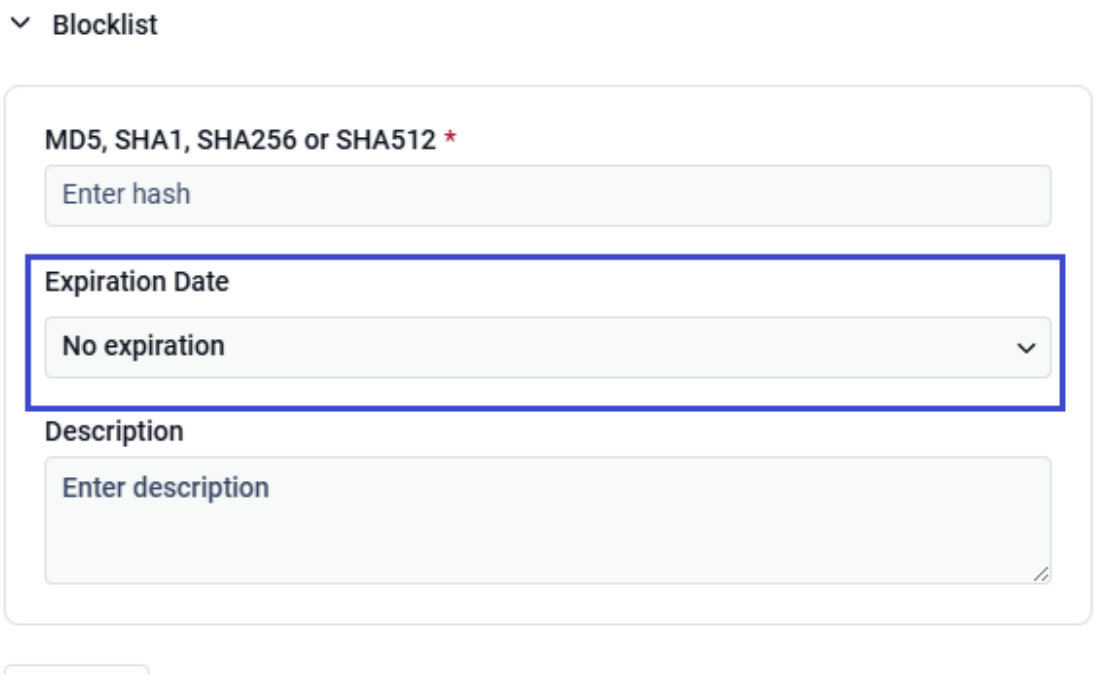
Reuse processing result for archives

Hash-based reuse of processing results now covers archive files, not only non-archive submissions. Repeated scans of the same archive can skip a full re-extract and re-scan when reuse is enabled.



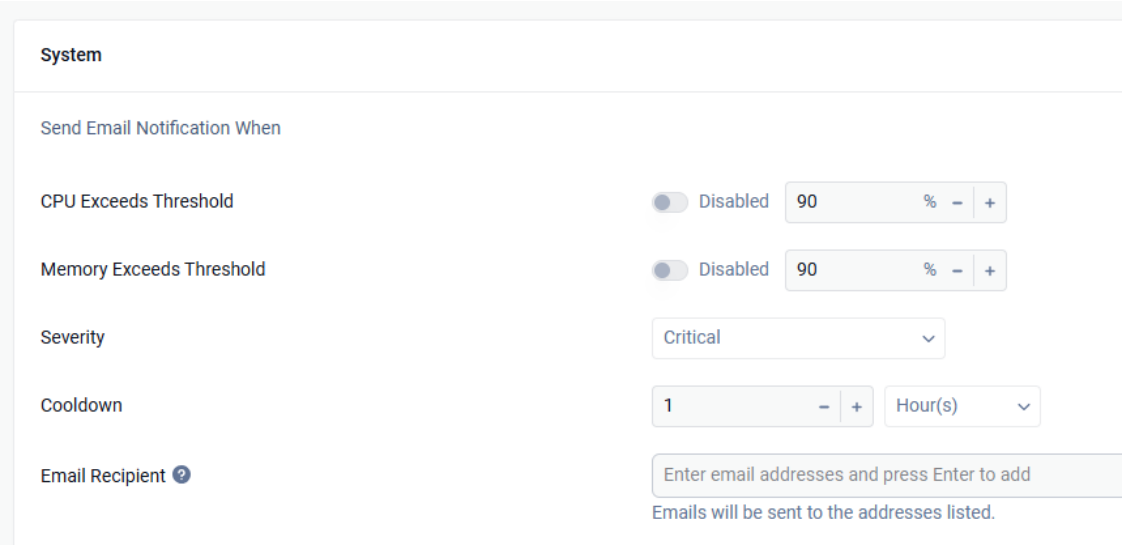
Skip by Hash approval expiry and auto-removal

Skip-by-hash allowlist and blocklist entries can carry an expiry date. When the time is reached, the hash is removed automatically so temporary approvals do not stay forever.



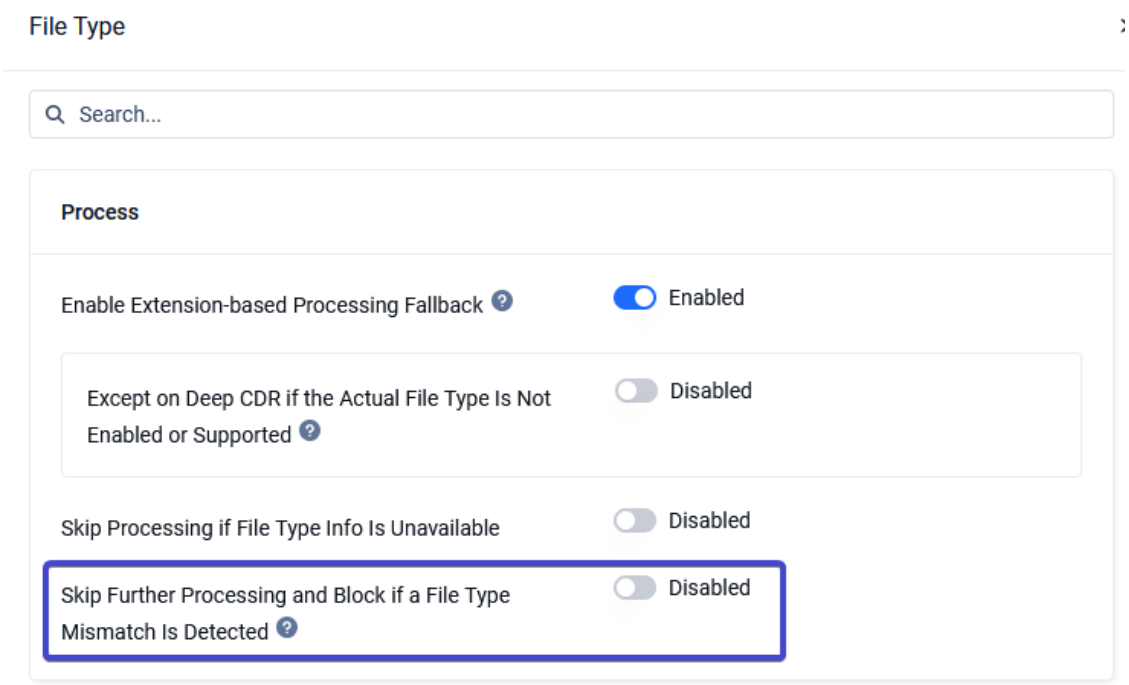
Email alerts for RAM and CPU utilization

Email notifications can fire when RAM or CPU crosses a configured threshold, in the same style as existing disk-space alerts.



Stop processing files with a mismatched file type

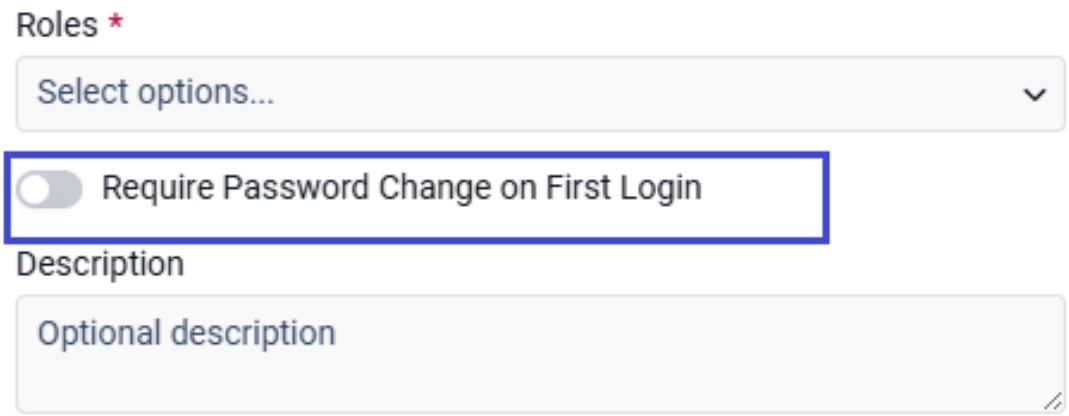
A workflow can now be configured to stop processing a file as soon as a mismatch between its detected type and its extension is identified, rather than continuing through the remaining engines. For customers who reject mismatched files as a matter of policy, this shortens scan time and spares resources on traffic that would be blocked anyway.



Further Enhancements

1) Force password change on first local login

A configurable policy can require local users to change their password on first login. LOCAL accounts only



2) Create user API can mint an API key The Create user API can create an API key as part of user creation, so automation does not need a second step after the account exists. Please check details: POST - Create user

3) Webhook callback API key

An custom header can be passed through to a webhook server when a file is submitted to the callback API, so the webhook endpoint can authenticate the request. POST - Analyze File (Asynchronous mode)

`callbackheader-name`	`string`	Header name Core attaches to the webhook POST (with `callbackurl`). Pair with `callbackheader-value`. Extra pairs: `callbackheader-name1` … `callbackheader-name10` with matching `callbackheader-valueN`. At most **10** pairs. Requires a valid `apikey`. Name: RFC 7230 token, max **128** characters. Duplicate names: highest index wins. Suffixes outside `1` – `10` (for example `01`, `11`) are ignored. Reserved (case-insensitive): `host`, `content-length`, `content-type`, `transfer-encoding`, `connection`, `user-agent`, `accept-charset`, `accept-language`, `digital_signature`, `x-core-id`. maxLength: 128
`callbackheader-value`	`string`	Value for `callbackheader-name` (or `callbackheader-valueN` for numbered pairs). Max **4096** characters. Empty is allowed. `CR`, `LF`, and `NUL` are rejected. maxLength: 4096

4) In-product notice when a newer Core version is available

Administrators can be notified in the console when a newer MetaDefender Core version is available.

5) API to list `data_id` values

A new API returns a list of `data_id` values so integrations can page or batch-query scan objects without walking history one file at a time. MetaDefender Core API

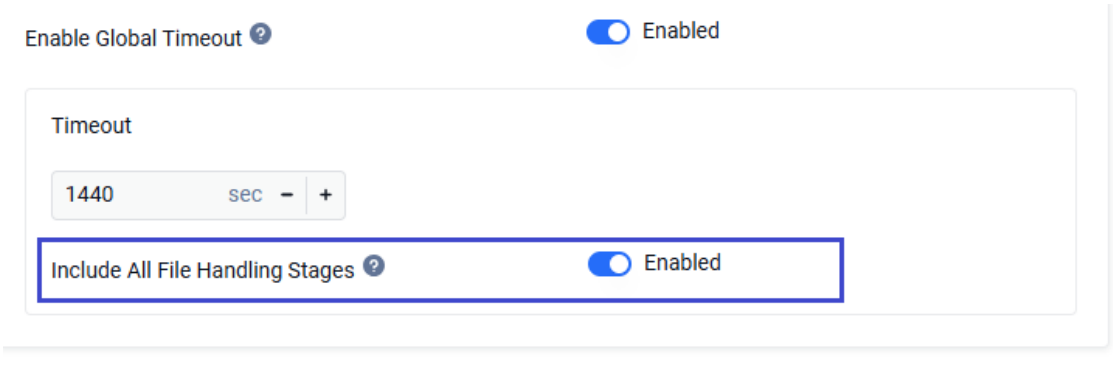
`GET` - `/file/ids`

Support filtering:

1. Time windows
2. Size and limit
3. Original file (submitted file) vs extracted file vs batch

6) Global timeout now includes end-to-end file handling time, not only engine scan time

Under Enable Global Timeout, a new child option Include all file handling stages in timeout calculation can count the full file lifecycle, not only time after the file enters the scan queue. That includes upload or streaming, write to storage, hash calculation, queue wait, and scan processing.



7) Dependable central management, even when an instance is struggling

Health status reporting to My OPSWAT Central Management (MOCM) is now paced to at most one report per configured reporting interval. A struggling instance no longer sends many reports per second, loading Central Management at the moment it is already under strain.

When MOCM offers a configuration that MetaDefender Core (MD Core) cannot apply, the rejection is now reported in full only for the first few consecutive attempts and summarized after that. One invalid setting no longer writes the complete configuration into the audit log on every sync interval.

8) Health reporting to central management recovers automatically

An enrolled instance that stopped reporting its health to My OPSWAT Central Management, because the reply to an outstanding report never arrived, now starts reporting again by itself. Recovery happens immediately on re-enrollment, and otherwise within the hour. Previously the instance stayed silent indefinitely and appeared stale in the Central Management console, with an unenroll or a full service restart as the only way back.

9) More reliable engine updates from Central Management

A damaged update package no longer stalls the engine and signature database updates queued behind it. Packages received through My OPSWAT Central Management (MOCM) that cannot be processed now fail cleanly and stay available to retry, while the remaining updates continue to install.

10) Engine deployment that fails fast instead of hanging

An engine or database deployment that cannot start no longer leaves the module stuck in Staging. The wait is now bounded: if the start does not complete, the deployment fails with a clear warning, cleans up after itself, and stays available to retry. Recovering no longer means removing the engine or restarting the service. Additional log detail across the staging phase also gives Support a clearer picture when a deployment does not behave as expected.

11) A Failed count you can act on

The *Failed* figure on the dashboard and in processing history now counts genuine processing failures only: *Failed*, *Extraction Failed* and *Scan Failed*. Objects that an operator cancelled, and objects that reached a configured processing limit, are no longer counted among them, so the number means what it says and can be trusted as an alerting signal.

12) One consistent vocabulary across every result

The six analysis categories in Processing History, covering advanced threats, vulnerabilities, threat signatures, origin, data loss and sanitization, now describe their outcomes in the same language. The same wording is used across the Processing History list, its CSV export and the executive report, so a result no longer reads differently depending on which column or which export it appears in.

13) Clearer feedback when adding a certificate

Selecting an empty certificate file in the *Add certificate by content* dialog now explains why the file cannot be used, instead of leaving the Add button disabled with no indication of the cause.

Security Enhancements

Upgraded 3rd party libraries:

- PostgreSQL v18.6
- SQLite v3.53.4

Bug Fixes

- Fixed an issue that Processing History could not show data after a scan of special cases. This is a displaying issue; your data in database is retained as is, and REST API integrations work properly.
- Fixed an issue where a cancelled scan was not reported as cancelled in JSON exports, log files, and syslog.
- Resolved an issue where importing a settings package could fail intermittently when a configuration value exceeded its length limit, silently keeping the previous configuration.

Version v5.21.0

Release Date: 30 July 2026

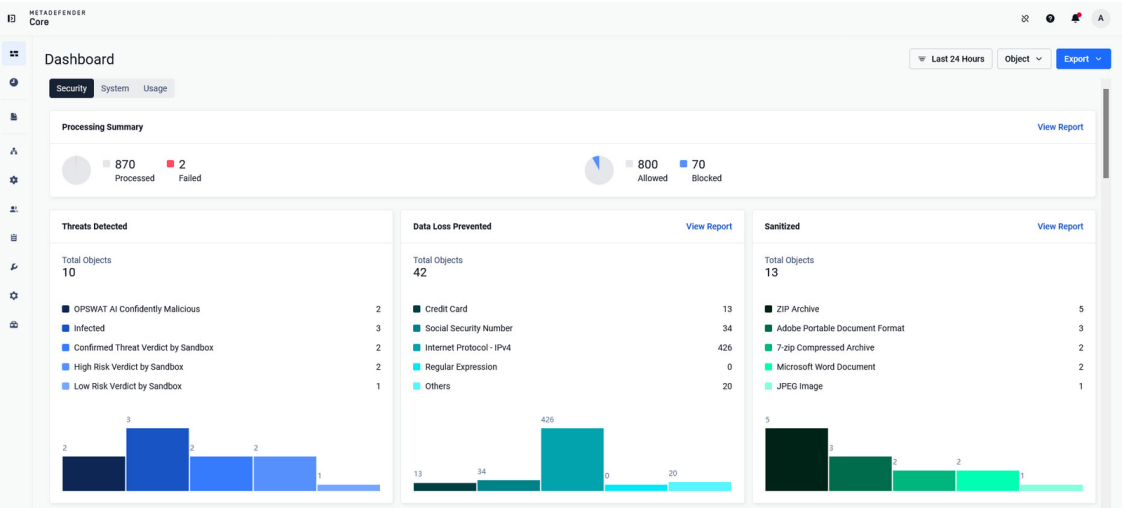
All-new MetaDefender Core interface

Meet the all-new interface—designed for speed and comfort. With built-in dark mode, enhanced accessibility support, and seamless interactions, every workflow feels more natural and efficient.

Below are some of the key highlights:

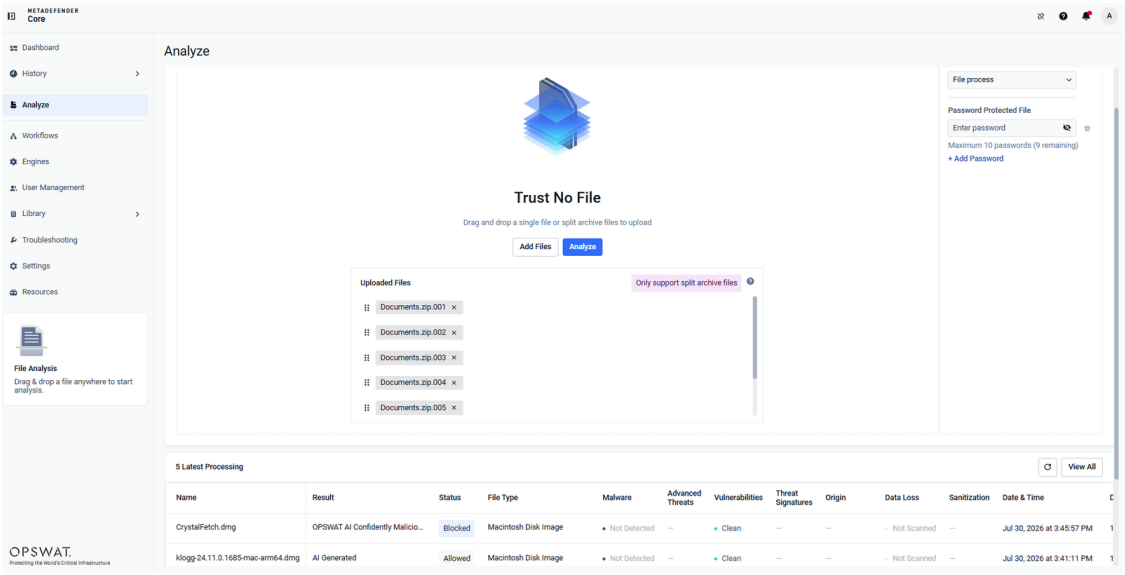
A brand-new, comprehensive dashboard

A completely redesigned dashboard featuring refreshed versions of existing metrics, making it easier to monitor your system, along with brand-new insights such as the CVE Detection Trend.

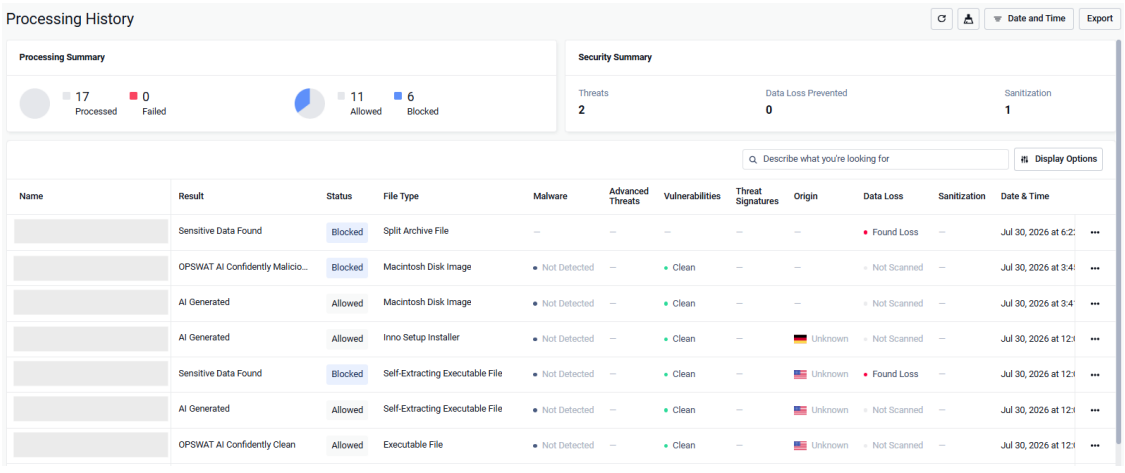


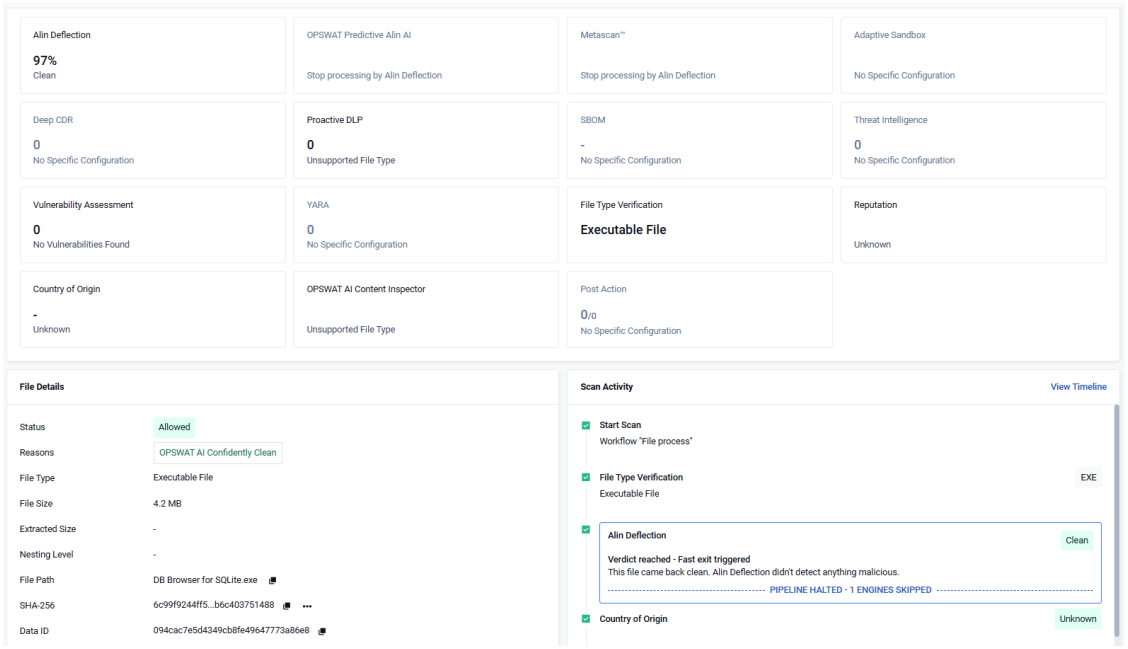


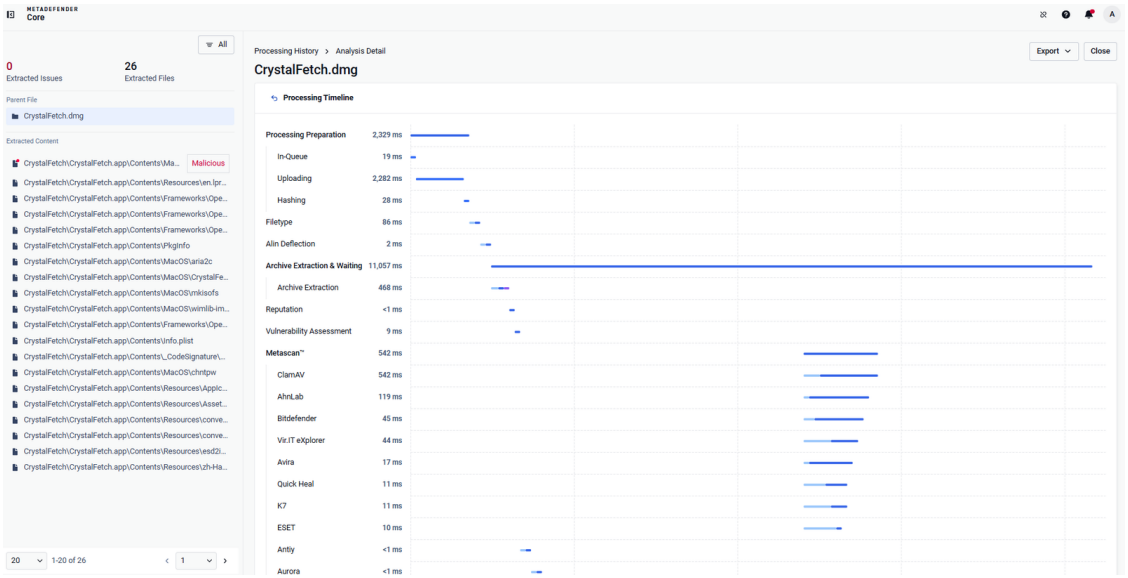
Easily monitor and submit scans, including split archive submissions



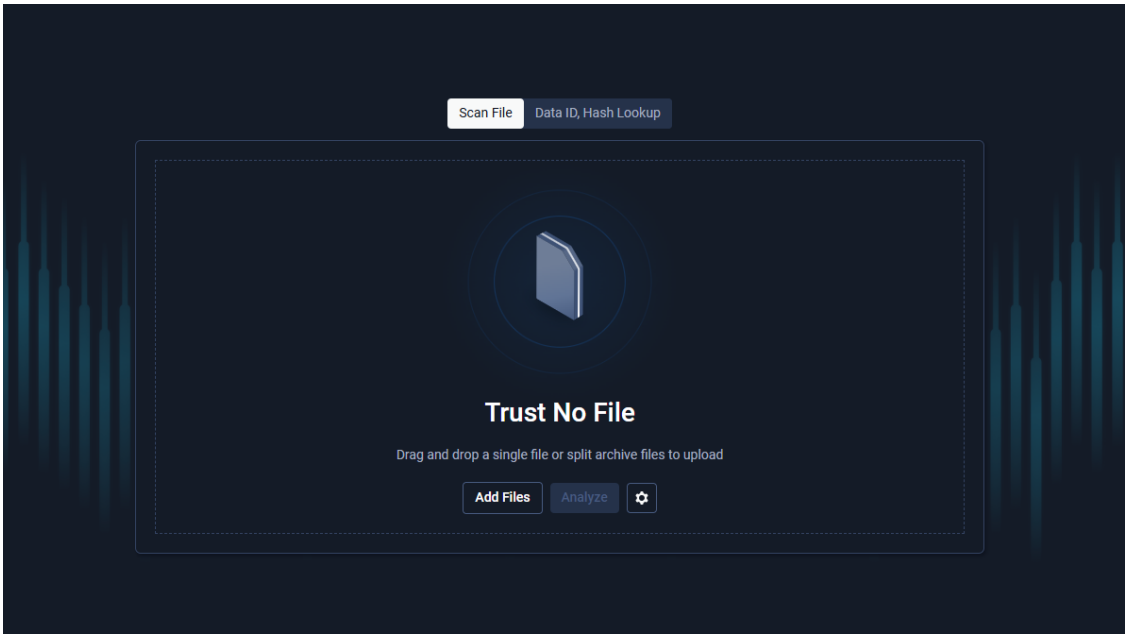
New look and feel for the Processing History and Processing Results pages







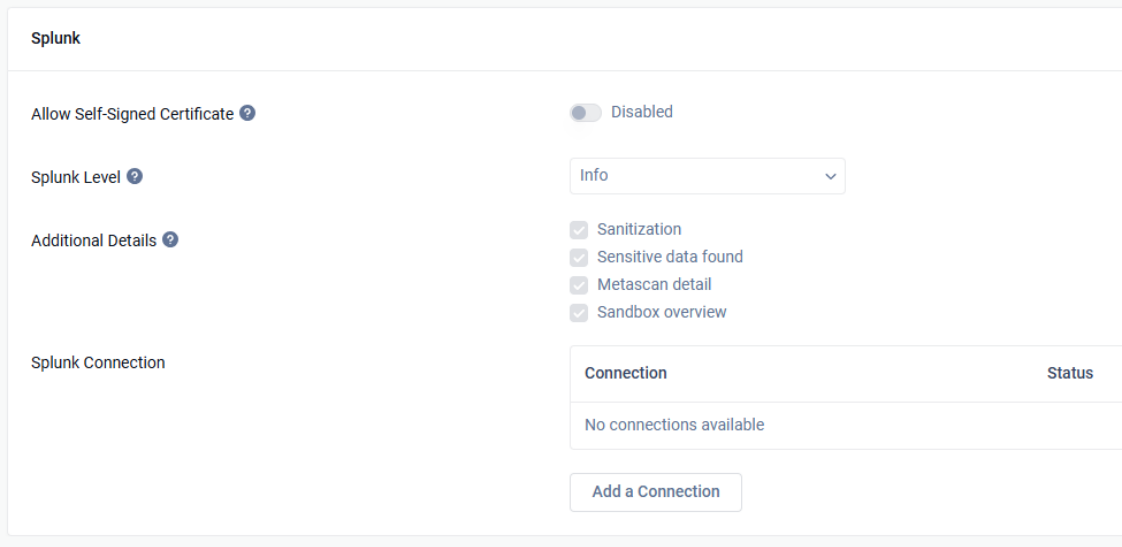
Work comfortably in low light with built-in dark mode



Splunk integration settings in MetaDefender Core

A dedicated **Splunk** section in MetaDefender Core makes it easier to connect Core to your Splunk environment. Add and manage Splunk connections, set the log level, allow self-signed certificates when needed.

Users can also integrate with Splunk during installation via Wizard steps.



Support PostgreSQL v18

MetaDefender Core now starts supporting PostgreSQL v18 (verified with 18.4).

Bundled PostgreSQL server comes with MetaDefender Core standalone deployment also comes with this new version.

Note:

- PostgreSQL v18 itself cannot initialize its database cluster when the Windows system locale name contains non-

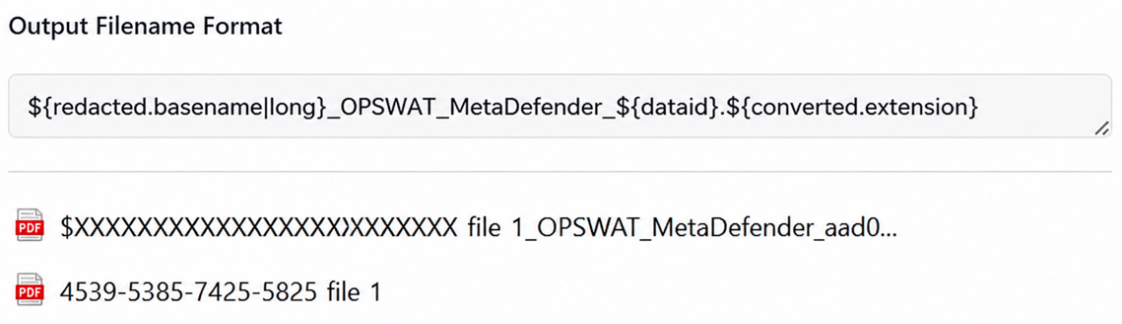
ASCII characters. For example, Turkish Windows uses the locale `Turkish_Türkiye.1254`, which contains `ü`. Instruction for safe upgrade at Upgrade to MetaDefender Core 5.21.0 or later on Windows systems with a non-ASCII locale.

- If your bundle PostgreSQL password contains non-ASCII characters (for example `£`, `€`, or accented letters), the upgrade may fail and prevent console login. ASCII special characters such as `&`, `~`, and `\` are not affected. Before upgrading, change the PostgreSQL password to ASCII-only - please check Database Management for change password

---

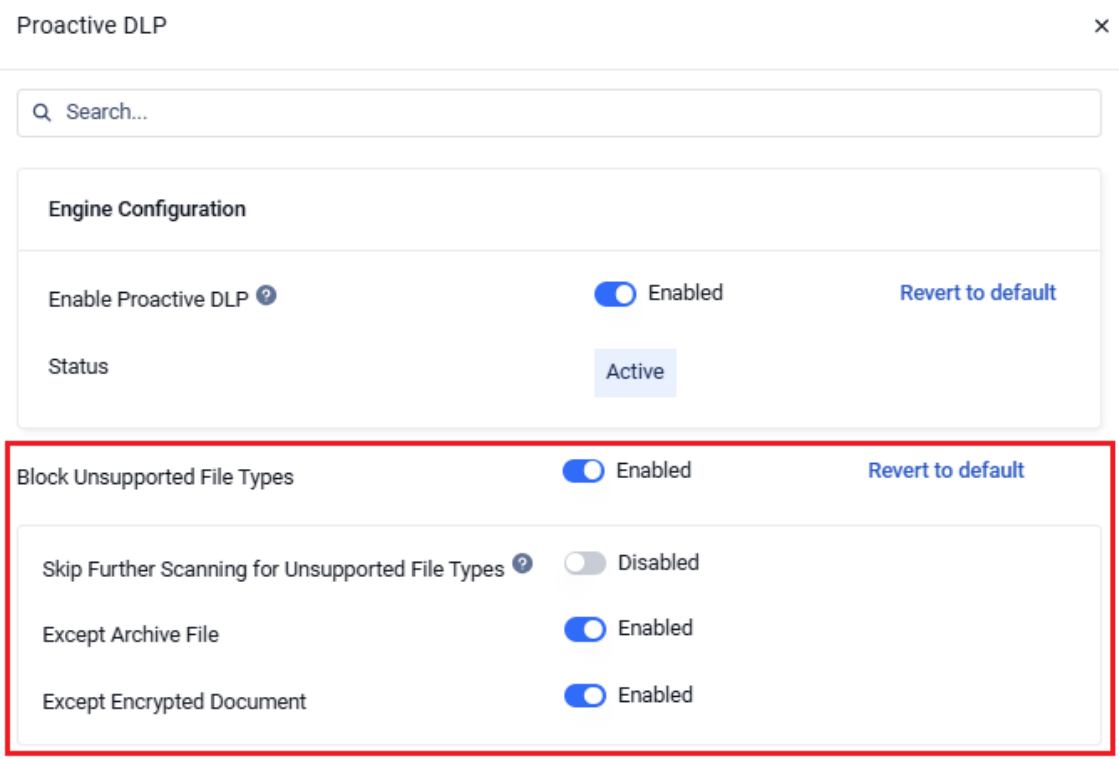
Redacted file name in output filename configuration

Output filename formatting now supports Proactive DLP redaction results through two new tokens: `${redacted.basename}` and `${redacted.extension}` (available in `short` and `long` formats). These tokens allow for the automatic use of masked filenames across DLP output, Deep CDR sanitized output, and archive tombstone names, ensuring sensitive information is removed from file names. If no redaction occurs, the tokens automatically fall back to the original filename and extension.



Block unsupported file types in Proactive DLP

Proactive DLP can now be configured to block files it identifies as an "Unsupported File Type", rather than releasing them as "No Threat Detected". This gives security teams stricter control over content that Proactive DLP cannot fully inspect, aligning DLP behavior with fail-closed policies.



Centralized "Skip by Hash" management via My OPSWAT Central Management

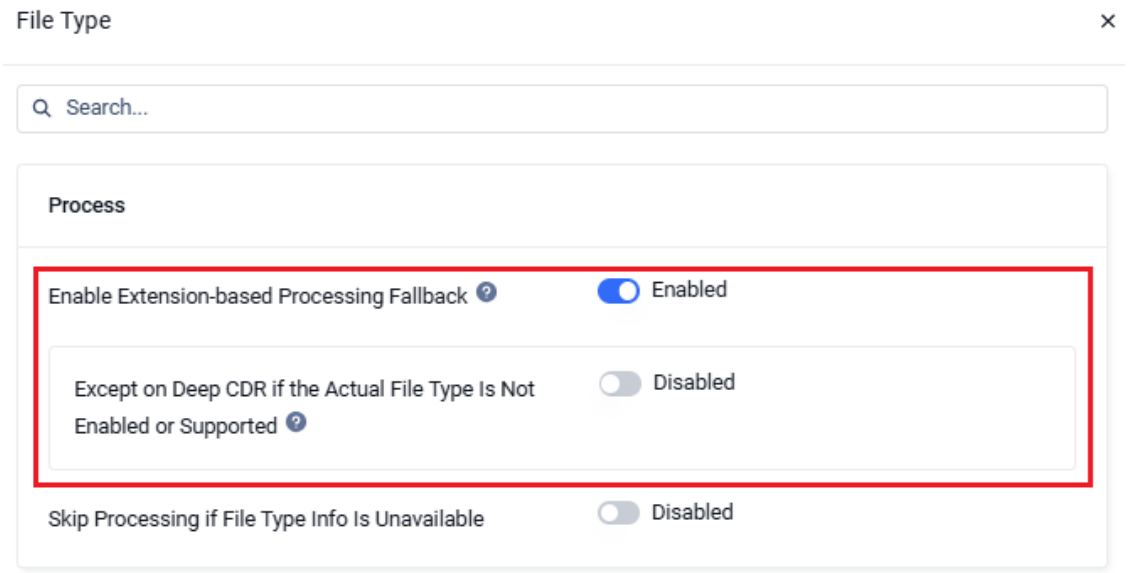
MetaDefender Core now supports centrally managing the "Skip by Hash" configuration (Skip Engines, Allowlist, and Blocklist) from My OPSWAT Central Management (MOCM). Administrators managing multiple Core instances can maintain hash lists once in Central Management and have them synchronized to all enrolled Cores over the existing heartbeat channel, instead of maintaining lists on each instance individually. Centrally managed hashes are enforced as read-only on the local Core while enrolled, a new Origin column and awareness banner indicate which entries are managed by Central Management, and configuration backups from older Core versions remain importable.

Dependency and availability: This feature requires updates on **both MetaDefender Core and My OPSWAT Central Management (MOCM)** — the Core-side changes ship in this release, but end-to-end centralized management is only available once the corresponding MOCM support is released. MOCM plans to release this feature in **mid-August 2026**.

Further Enhancements

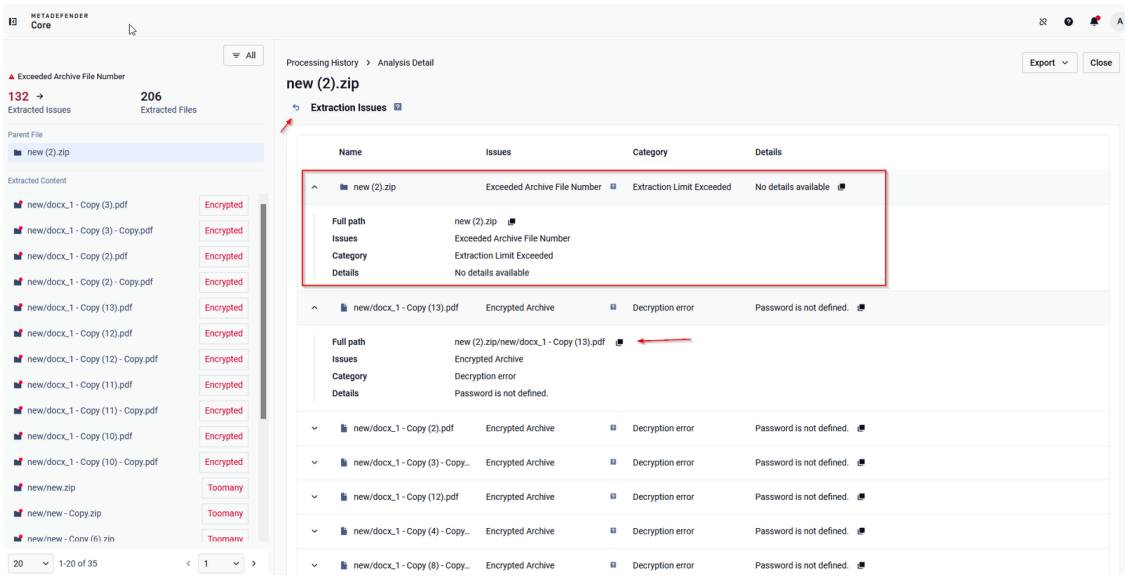
1) Clearer "Extension-Based Processing Fallback" configuration

The "Fallback file type detection to current extension if needed" option has been renamed to "Enable Extension-Based Processing Fallback" and its description has been updated to better reflect its expanded functionality. The option now uses the file extension as a secondary file type for processing when needed, reducing confusion for administrators when configuring workflows.



2) Improved error visibility for archive extraction failures

The scan-result UI now makes it easier to see the full error path when extraction fails deep inside large, multi-level archives, so operators can identify the problematic nested file without clicking through the archive one level at a time.



3) More meaningful extraction error messages

Extraction error messages have been improved to provide clearer, more detailed information, replacing generic or ambiguous wording so customers can better understand why extraction did not complete.

4) Classify Global Timeout Exceeded as Allowed when configured in workflow general settings.

Global Timeout Exceeded is now available under Workflow Management General Processing results to be classified as Allowed, so you can choose whether scans that hit the global timeout are treated as Allowed like other configurable outcomes.

5) Post Actions support sanitized files and optional filtering by sanitized file type.

Post Actions can now run for **sanitized** files, not only threat-related verdicts. You can also limit actions to **specific file types** (for example PDF, XLS, or DOCX) so downstream steps apply only to the formats you choose.

6) In-app alerts when disk space is low

When available disk space drops below configured thresholds, administrators now get an **in-app notification** in the MetaDefender Core web UI—in addition to existing email alerts and dashboard status—so low-disk conditions are hard to miss no matter which page you're on.

Security Enhancements

- Hardened license activation connectivity test: the license activation server connectivity test endpoint has been hardened to restrict the destinations it will connect to, mitigating a blind server-side request forgery (SSRF) issue that could otherwise be used to probe internal hosts and ports
- Upgraded 3rd party libraries:
  - c-ares (bundled within the gRPC) v1.34.8
  - yamlcpp v0.9.0
  - nghttp2 v1.69.0
  - NGINX v1.30.4
  - sqlite v3.53.3
  - libxml2 v2.15.3
  - PostgreSQL v18.4

Bug Fixes

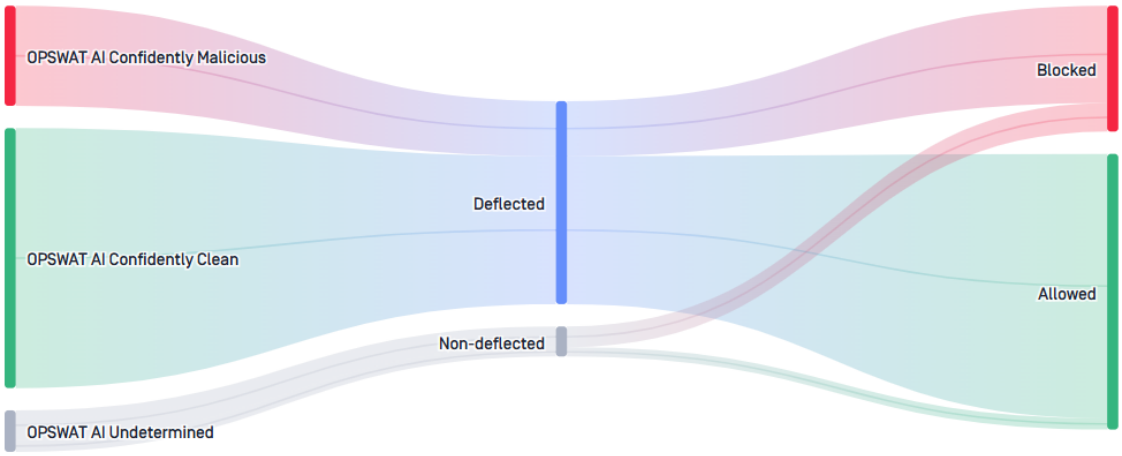
- Improved startup when several MetaDefender Core nodes share one database (shared-db), so concurrent restarts are less likely to hit database privilege deadlocks and fail to come online.
- Improved NGINX log rotation on Windows by storing runtime config outside temp and recovering when the config file is missing.

Version v5.20.0

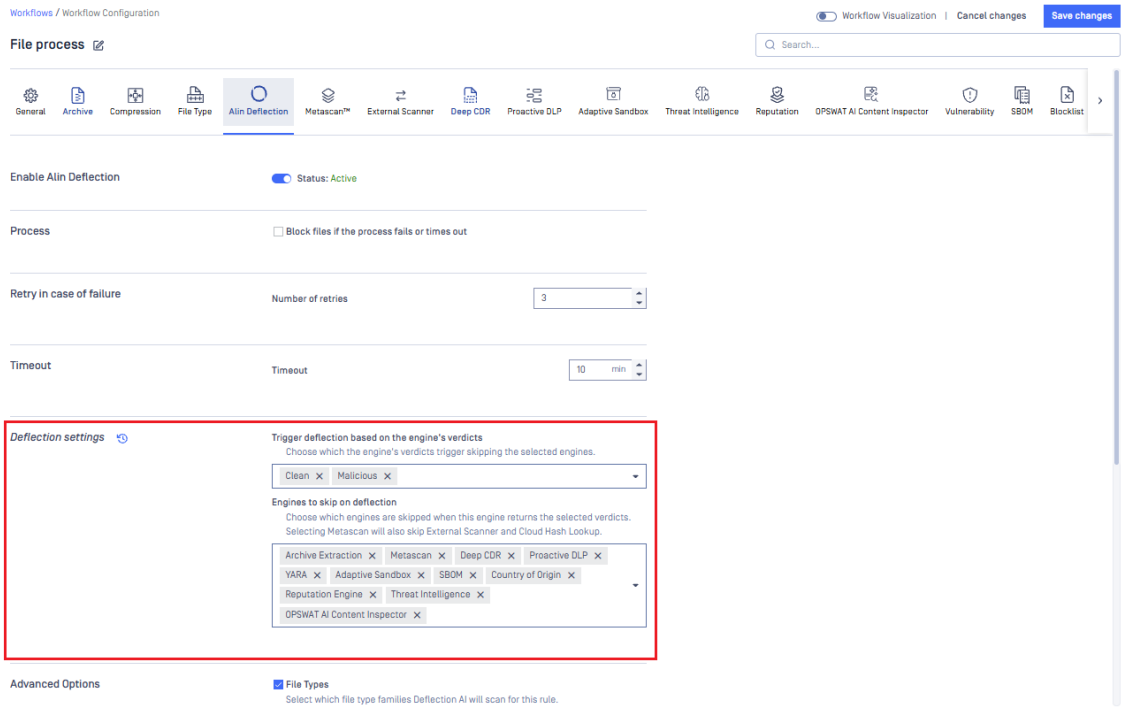
Release Date: 30 June 2026

New Technology: OPSWAT Alin AI Deflection

OPSWAT Alin AI Deflection is an AI-driven, ML-based malware deflection engine. It analyzes the structural and byte-level signals of executable, document, and office files to determine whether a file can be confidently skipped from heavier scanning engines — reducing scan overhead without compromising detection.



Administrators can configure deflection in the **Workflow Configuration** settings for **OPSWAT Alin AI Deflection** to define which verdicts trigger deflection and which processing engines are skipped when a file is deflected.



Remote product upgrade through My OPSWAT Central Management

Administrators can now remotely upgrade MetaDefender Core (MD Core) instances directly from My OPSWAT Central Management (MOCM), eliminating the need to log in to each instance individually. This feature is supported on both Windows and Linux.

Note: Remote upgrades require MD Core **5.20.0** or later. You must first manually upgrade each MD Core instance to version 5.20.0. Once an instance is running 5.20.0, all subsequent upgrades can be performed remotely through MOCM.

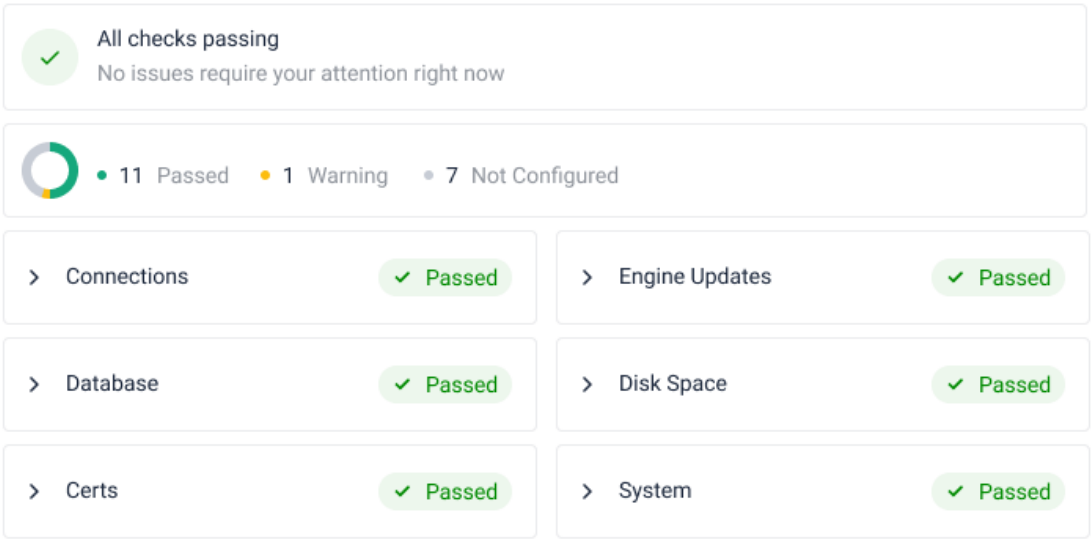
Auto Troubleshooting — know your MetaDefender Core deployment is healthy at a glance

No more digging through logs or running manual tests to confirm everything is running smoothly. The new Troubleshooting dashboard gives administrators a clear, real-time view of system health in one place.

Issues are surfaced automatically and explained in plain language, so you know exactly what needs attention — and what's just fine as-is. When you want a closer look, a single click runs a deeper analysis on demand.

Spend less time investigating and more time being confident that your deployment is protected. Spot problems early, resolve them faster, and share results with support in seconds.

Check more details at Auto Troubleshooting.

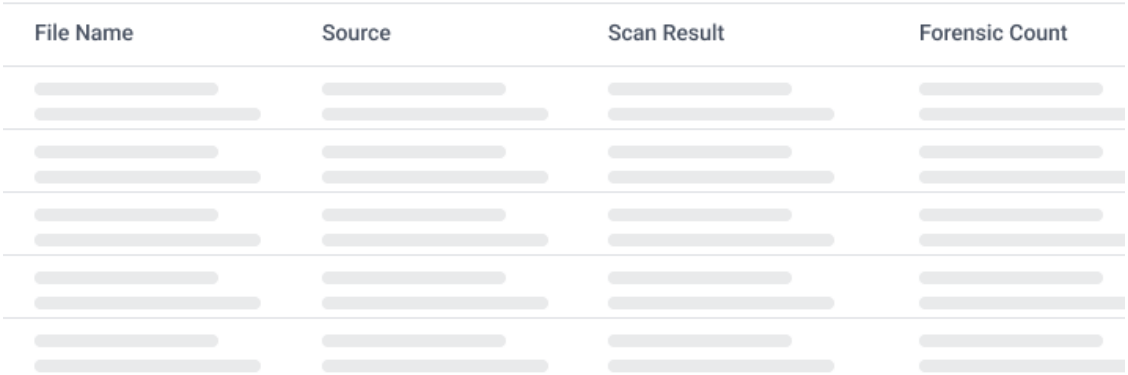


Archive Forensic

Finding a single problematic file buried inside an archive just got easier. Archive Forensic automatically captures flagged sub-files during archive processing and stores them in a dedicated, secure location — so you can retrieve exactly what you need without sifting through the entire archive.

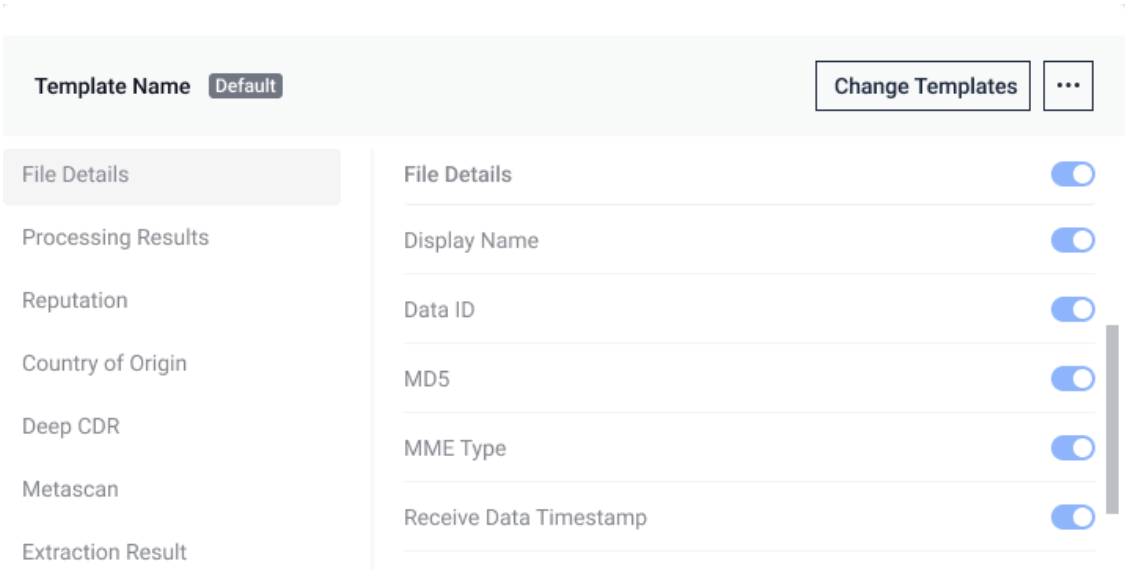
When enabled per scanning rule, MetaDefender Core preserves child files matching your configured trigger verdicts, encrypts them at rest with AES-256-GCM, and makes them available as password-protected ZIPs. Office documents are smartly skipped to keep your forensic store clean.

View, search, download, and manage your captures anytime: Archive Forensic.



Export Templates for Core Scan Results

Allow user to save export scan result options selection as named, reusable templates for ease of use when exporting scan results. Templates are tied to each user, maximum of 10 templates per user.



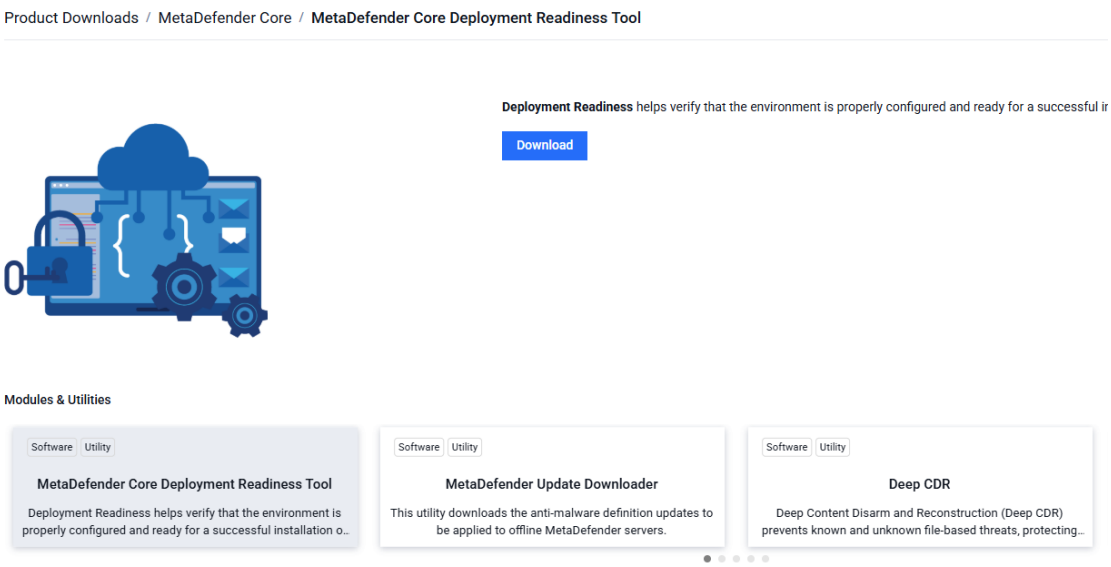
MetaDefender Core Deployment Readiness Tool

Wondering if your system is ready to install or upgrade MetaDefender Core? The new Deployment Readiness tool gives you an instant answer — no more manually cross-checking documentation. Just download the single, lightweight binary (no installation needed) and run it.

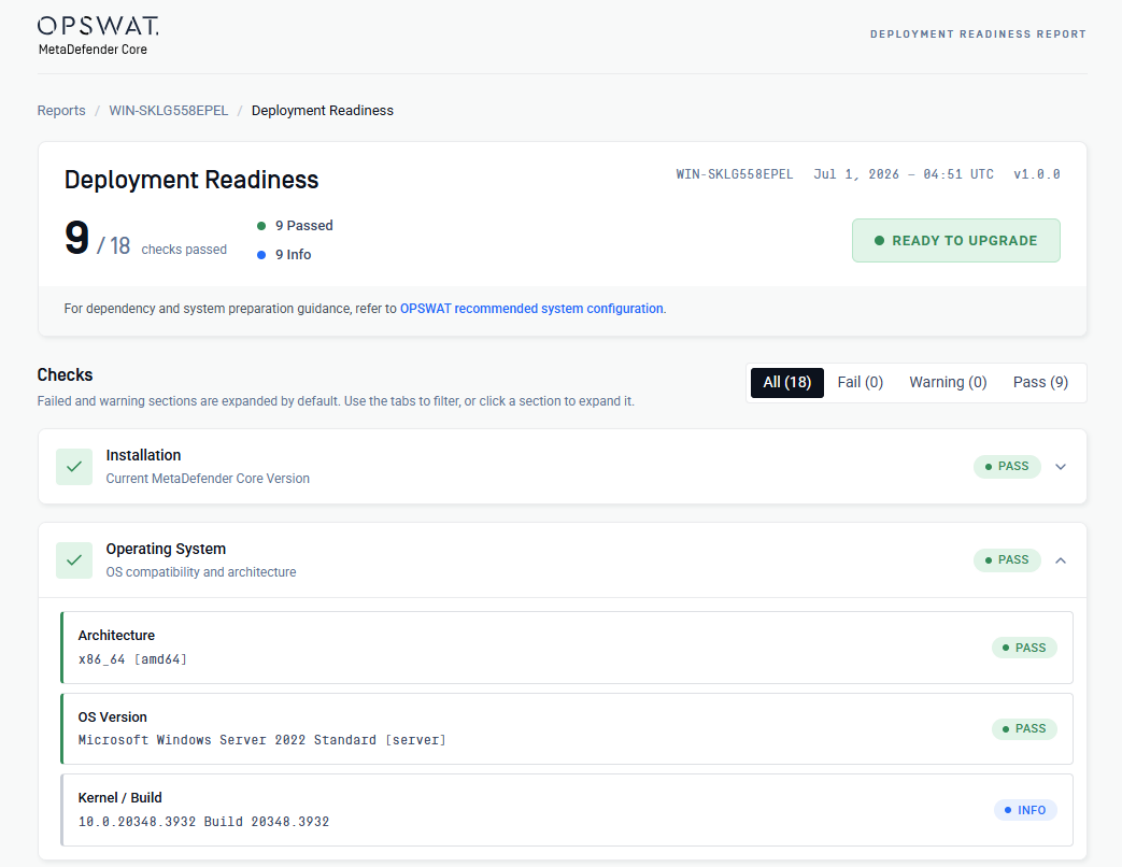
In one pass, it checks your OS, hardware, dependencies, database, ports, permissions, and potential software conflicts — then produces a clear console summary and a self-contained HTML report you can save or share. It even auto-detects your engine count and product configuration, and keeps its requirements up to date automatically from OPSWAT servers.

Available for both Windows and Linux: Download

Under MetaDefender Core / MetaDefender Core Deployment Readiness Tool



Example of HTML report



Further Enhancements

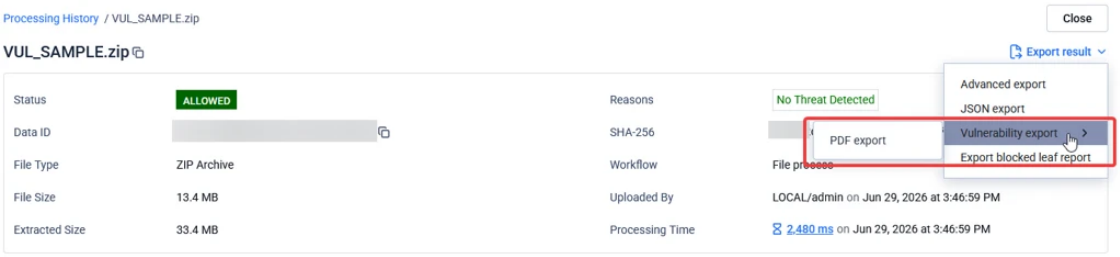
1) SSO Support Behind SSL-Terminating Reverse Proxies

Added support for the `X-Forwarded-Proto` header to improve SSO compatibility with deployments where SSL/TLS is terminated by a reverse proxy. MD Core now correctly detects the original request protocol when generating SSO redirects.

2) SBOM and Vulnerability Assessment reporting enhancements

Reporting coverage has been extended for software bill-of-materials (SBOM) and Vulnerability Assessment data.

- SBOM export (PDF) is now supported for batch scans.
- Vulnerability Assessment details are now included in exported reports (PDF) for subfiles within an archive scan.



OPSWAT.
MetaDefender Core

VULNERABILITY REPORT

VUL_SAMPLE.zip

Generated time	June 29, 2026 at 03:47:05 PM
Data ID	
Total files with vulnerabilities	1

Total CVEs
16

Affected Files
1

Vulnerability Severity	Count
CRITICAL	0
IMPORTANT	11
MODERATE	5
LOW	0
UNKNOWN	0

3) Sanitized file size added to logs

The size of a file after sanitization is now recorded in the logs, improving traceability, auditing, and operational visibility when sanitization occurs.

4) AI Content Inspector output folder included in storage monitoring

The dedicated folder that the AI Content Inspector uses to store output images is now included in the storage-folder monitoring feature, so its disk usage is visible on the dashboard alongside other monitored storage.

5) My OPSWAT Central Management integration enhancements

- Newly enrolled MetaDefender Core (MD Core) instances now synchronize to their assigned set and pull engine and definition updates immediately upon registration with My OPSWAT Central Management (MOCM).
- The last MOCM sync time is now written to the MD Core logs instead of the configuration history.

6) Configure Log Rotation from the Web Console

Managing log retention no longer means editing config files. You can now control how MetaDefender Core handles logging directly from the web console — set the **maximum log file size**, choose your **rotation interval**, and define how many **rotated files** to keep. Tune your logging to fit your storage and compliance needs in just a few clicks.

7) Test Your Email Server Configuration

No more guessing whether your SMTP settings are correct. Under **Settings > Network > Email**, you can now send a test email directly from MetaDefender Core to verify your server configuration in seconds.

8) Configurable Syslog Facility

You can now tag MetaDefender Core syslog messages with a specific facility, making it easier for your SIEM, Splunk, or rsyslog server to route, filter, and retain Core logs separately from other applications. Set it under **Settings** → **Logging** → **General**, under Log format.

Most users can leave this alone — the default (`daemon`) keeps the same behavior as before. Change it only if your security or operations team asks Core logs to carry a specific category (commonly `local0`–`local7`). Changes take effect immediately, with no service restart required, and a single facility applies to all syslog output across both remote (UDP/TCP/TLS) and local syslog.

9) Multiple Schedules per Report

You're no longer limited to a single schedule per report. MetaDefender Core now lets you configure up to three scheduled times for each report type — so you can, for example, send a weekly History report every Monday morning *and* a monthly summary on the 1st. Set each schedule's timing, time range, and verdicts independently to match the way your team actually consumes reports.

10) Richer Report Columns

Scheduled reports and manual Processing History exports now include six new columns — **Threat Found, Total AVS Engines, Threat Detected By, File Size (KB), Threats, and Metadata** — giving you deeper insight into every scan. Columns are also reordered to group related fields logically (scan timing, threat details, and file metadata together).

11) Usage Reporting

With Usage Reporting enabled, MetaDefender Core shares anonymous usage metrics with OPSWAT — giving us the insight to improve performance, prioritize the features you rely on, and deliver a better overall experience.

We only collect neutral operational data. No sensitive information, no file contents, no personal data — ever. You stay in control and can enable or disable reporting at any time.

Security Enhancements

- Fixed an authentication bypass vulnerability that could allow an attacker to impersonate a user by using a valid SAML response.
- Added a limit on the number of parts accepted in `multipart/form-data` request bodies to improve request handling and security.
- Fixed an XML External Entity (XXE) injection vulnerability in the SAML response XML parser,
- Fixed an operating-system command injection vulnerability caused by unsanitized command-line arguments passed to popen().
- Fixed a postMessage origin-validation bypass in the management console caused by endsWith suffix confusion

during origin checks.

- Upgraded 3rd party libraries:
  - 7z v26.02
  - OpenSSL v3.5.7
  - NGINX v1.30.3

Bug Fixes

- Fixed a heap-corruption crash in the engine process that could occur when engines were repeatedly disabled and enabled over an extended period.
- Fixed an issue where an engine's `maximum_thread_count` and a custom parallel count was not enforced at runtime, so the engine remained capped at the default limit even when configured higher.
- Fixed an issue where password-protected documents failed when "Block unsupported file types" was enabled in Deep CDR during split-archive processing.
- Fixed an issue where scan-result tab names were truncated when the tab bar overflowed.
- Fixed an issue where a temporary file created by the AI Content Inspector was not deleted after a scan request was canceled.

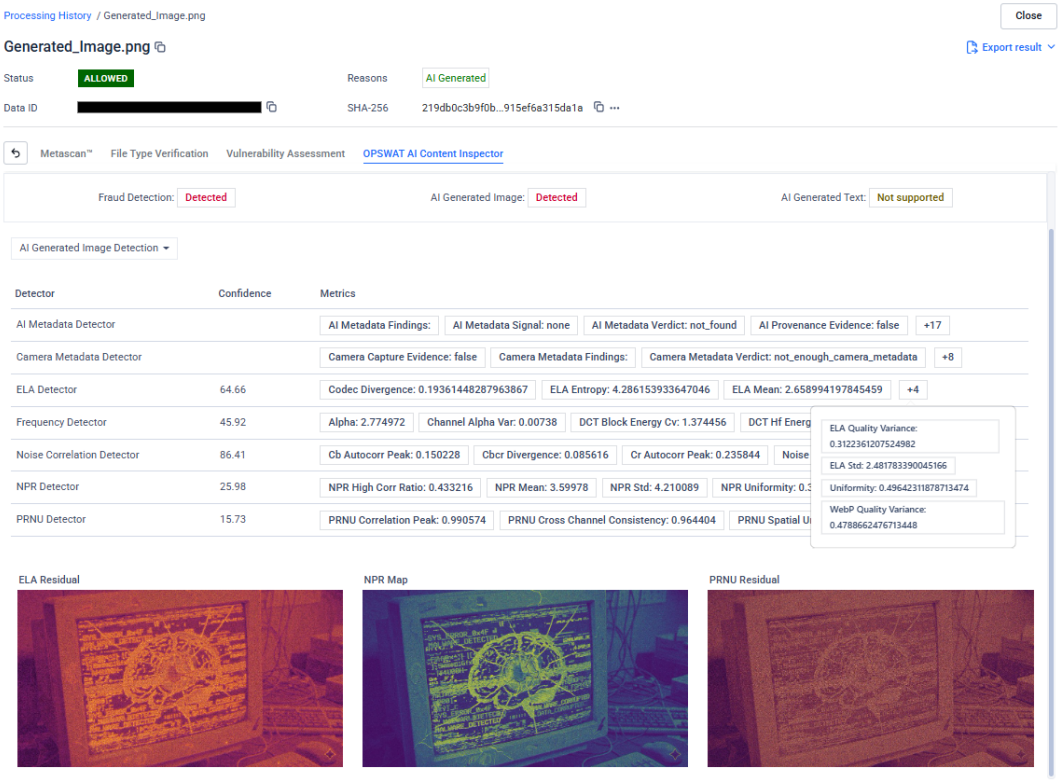
Version v5.19.0

Release Date: 21 May 2026

New Technology: OPSWAT AI Content Inspector

OPSWAT AI Content Inspector is a new technology introduced in MetaDefender Core. It is an AI-driven content authenticity and document fraud detection engine designed to help organizations identify suspicious or manipulated content before review, approval, or payout processes.

It analyzes visual, textual, and structural signals across documents and images to detect potential fraud in invoices, insurance claims, accident photos, and other high-risk content.



Automated support package generation and transfer to My OPSWAT Central Management

MetaDefender Core's support packages can now be generated on demand from MOCM and automatically uploaded to MOCM for centralized retrieval. This is particularly valuable for instances in remote or restricted environments where Support previously had to coordinate with the local administrator to collect logs.

Differentiated verdict for RMS-protected Office documents

Password-protected and RMS-protected (label-protected) Office documents were previously classified under a single verdict, "Password Protected Document." Both file types are now reported separately as *Password Protected Document* and *RMS Protected Document*. The new verdict is recognized end-to-end in verdict filtering on Processing History, the verdict result highlighter, scheduled report emails, and email notifications for processing activity.

> **Info**
>
> To preserve backward compatibility, *RMS Protected Document* is enabled by default in existing settings (such as email notifications and scheduled reports) during upgrade or configuration import, so customers continue to receive coverage equivalent to their previous configuration.

Further Enhancements

1) Faster processing result retrieval during engine updates

Processing result API calls could previously stall behind an in-progress engine update, occasionally exceeding the 150-second NGINX timeout for customers with large workloads. The internal lookup of engine configuration last-modified time is now served from a cache, so result retrieval no longer waits on engine updates to complete and

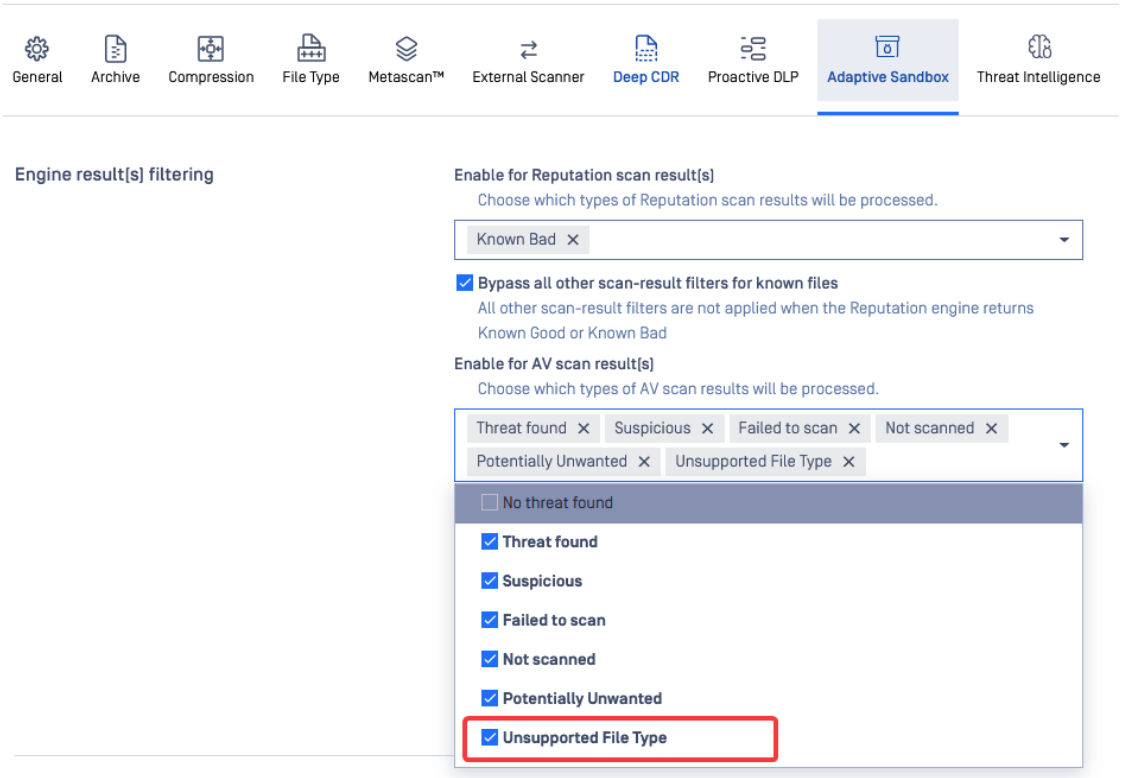
returns results promptly even while engines are mid-update.

2) Verdict origin indicator for scan categories and errors

The management console now indicates whether a scan category or error (for example, "Calculate Hash Error") was determined by MetaDefender Core itself or by the underlying scan engine. A visual indicator clarifies the origin so administrators and Support can more quickly route and triage issues.

3) Adaptive Sandbox skip handling for unsupported file types

Adaptive Sandbox previously sent files with the Metascan verdict `Unsupported File Type` for sandbox analysis because that verdict was missing from the skip-evaluation list. This verdict is now recognized, ensuring unsupported file types are skipped from sandbox processing as intended and reducing unnecessary sandbox resource consumption.

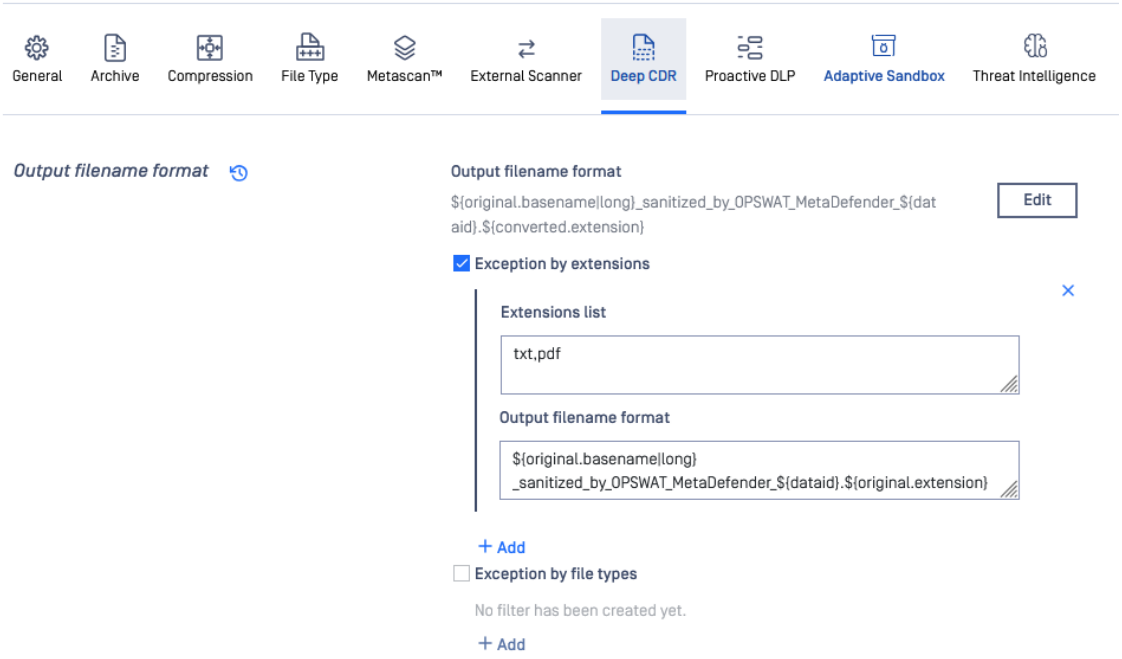


4) A clearer blocked reason for scans interrupted by the MetaDefender Core restart

When the MD Core service restarts mid-scan, in-process scans are now reported with an explicit failed block reason rather than an empty reason. This makes interrupted scans easier to identify in processing history and downstream reporting, improving auditability after unexpected restarts.

5) Deep CDR output filename exception by filename and extension

Deep CDR users can now define output filename exceptions based on filename or extension, in addition to the existing exceptions based on detected true type. This enables, for example, converting one extension to .txt while preserving another extension's original suffix in the sanitized output filename, even when both files share the same detected true type.



6) Engine package compatibility flag

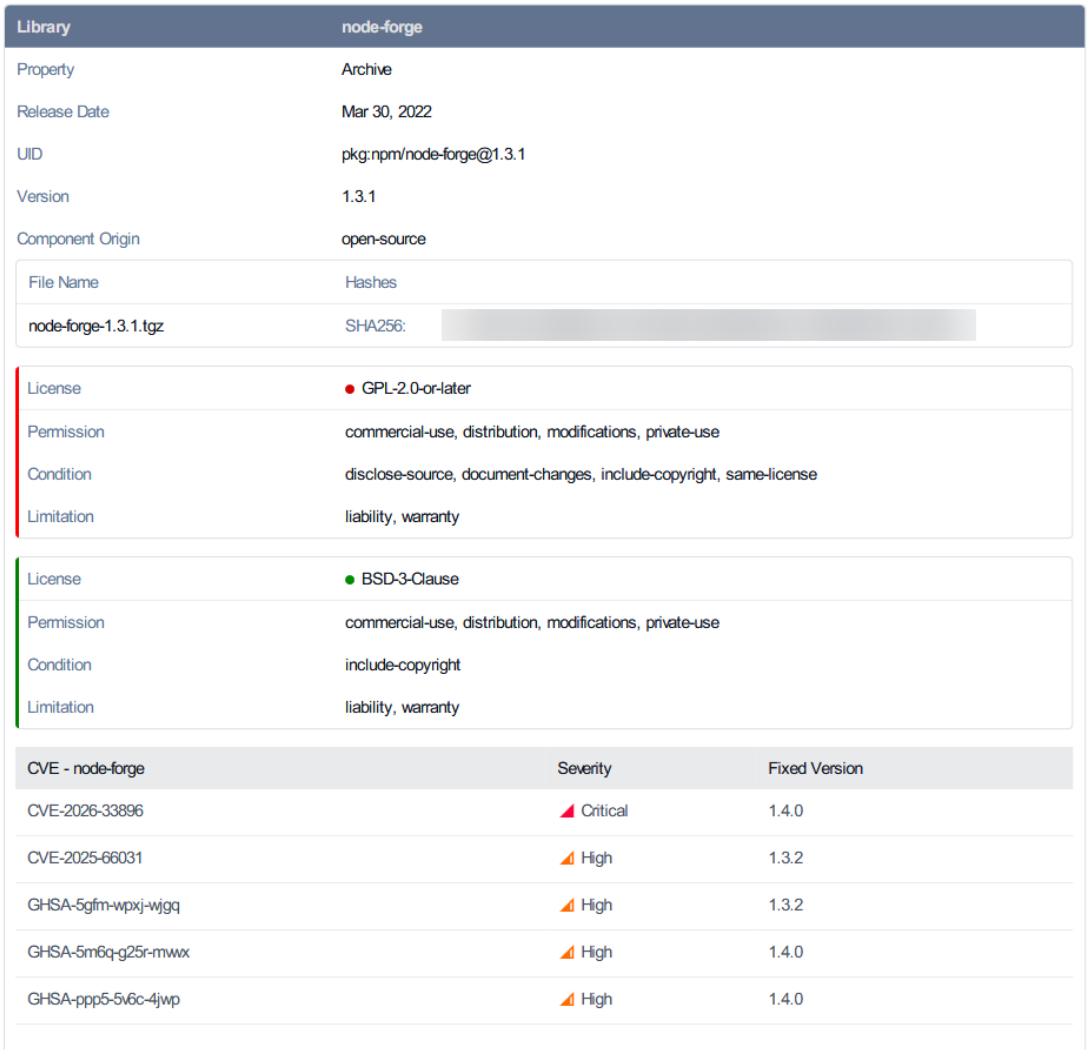
The `stat/packages` API now includes a boolean-supported field on each engine and database package entry, making it clearly visible whether a package is compatible with the currently running MetaDefender Core version. This simplifies inventory and compatibility validation for administrators and integrators.

7) Delayed start for the MetaDefender Core service on Windows

On Windows, the MD Core service is now configured with the "Automatic (Delayed Start)" startup type. This ensures the operating system has finished initializing dependent services — particularly network interfaces — before MD Core starts, preventing a few unexpected behaviors.

8) Additional SBOM fields in the PDF report

The Software Bill of Materials (SBOM) PDF report has been expanded with additional fields requested for compliance frameworks such as CERT-in. Customers benefit from a richer SBOM artifact that more closely aligns with regulatory and industry reporting requirements.



Library	node-forge
Property	Archive
Release Date	Mar 30, 2022
UID	pkg:npm/node-forge@1.3.1
Version	1.3.1
Component Origin	open-source

File Name	Hashes
node-forge-1.3.1.tgz	SHA256:

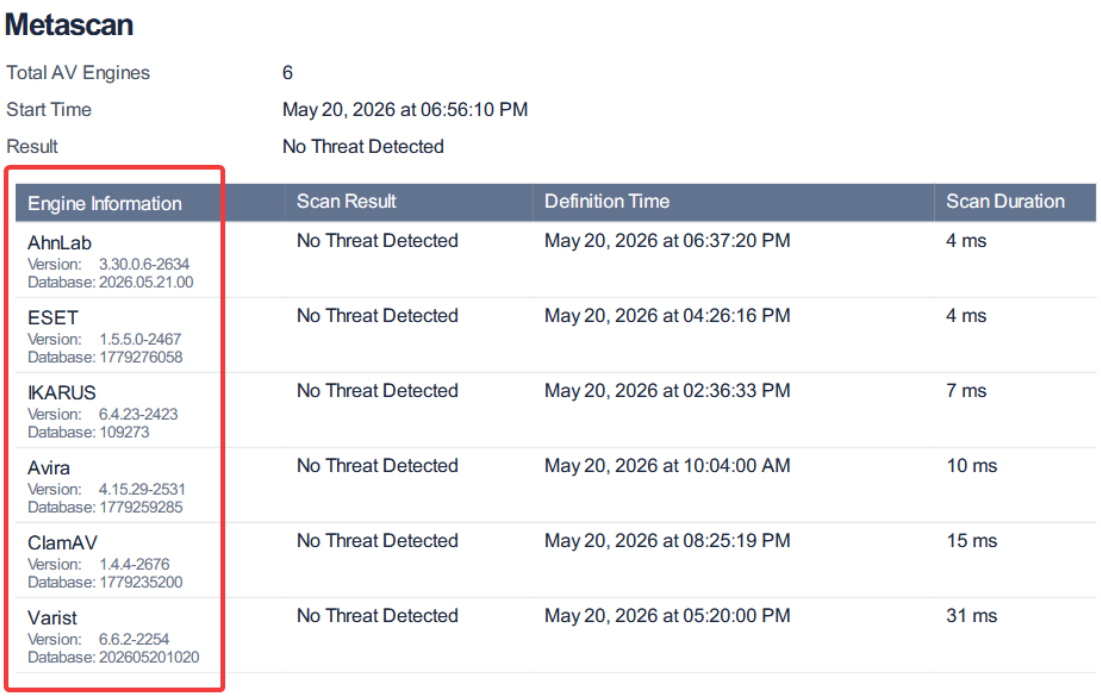
License	GPL-2.0-or-later
Permission	commercial-use, distribution, modifications, private-use
Condition	disclose-source, document-changes, include-copyright, same-license
Limitation	liability, warranty

License	BSD-3-Clause
Permission	commercial-use, distribution, modifications, private-use
Condition	include-copyright
Limitation	liability, warranty

CVE - node-forge	Severity	Fixed Version
CVE-2026-33896	Critical	1.4.0
CVE-2025-66031	High	1.3.2
GHSA-5gfm-wpxj-wjgq	High	1.3.2
GHSA-5m6q-g25r-mwwx	High	1.4.0
GHSA-ppp5-5v6c-4jwp	High	1.4.0

9) Additional Metascan engine information in the Export Result

The Export Result option for processing details now includes per-engine information for Metascan engines. The "AV Engines" column in Advanced Export has been renamed to "Engine Name," and two new fields — "Engine Version" and "Database Version" — are now exported alongside it. These values are also preserved for results that were originally scanned on a previous MetaDefender Core version after the upgrade.



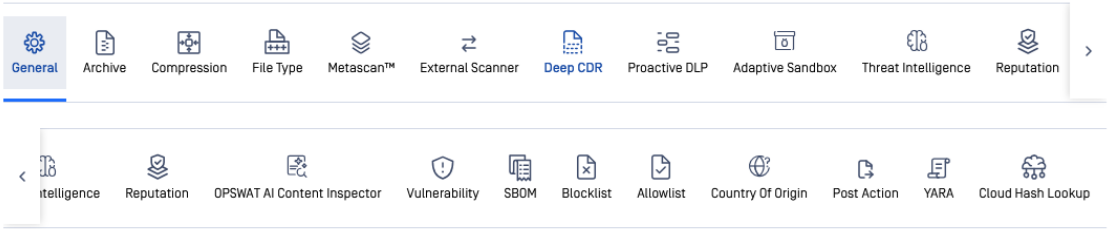
Metascan

Total AV Engines	6
Start Time	May 20, 2026 at 06:56:10 PM
Result	No Threat Detected

Engine Information	Scan Result	Definition Time	Scan Duration
AhnLab Version: 3.30.0.6-2634 Database: 2026.05.21.00	No Threat Detected	May 20, 2026 at 06:37:20 PM	4 ms
ESET Version: 1.5.5.0-2467 Database: 1779276058	No Threat Detected	May 20, 2026 at 04:26:16 PM	4 ms
IKARUS Version: 6.4.23-2423 Database: 109273	No Threat Detected	May 20, 2026 at 02:36:33 PM	7 ms
Avira Version: 4.15.29-2531 Database: 1779259285	No Threat Detected	May 20, 2026 at 10:04:00 AM	10 ms
ClamAV Version: 1.4.4-2676 Database: 1779235200	No Threat Detected	May 20, 2026 at 08:25:19 PM	15 ms
Varist Version: 6.6.2-2254 Database: 202605201020	No Threat Detected	May 20, 2026 at 05:20:00 PM	31 ms

10) Reorganized workflow configuration tabs

The Workflow configuration tabs have been reorganized by functional group, making settings easier to discover and configure.

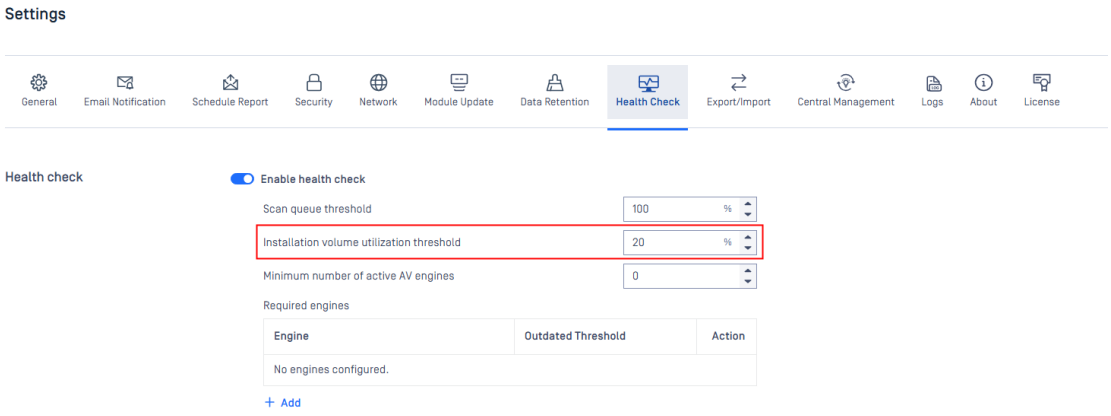


11) Improved tab display for long engine names in Advanced Export

Engine names in the Advanced Export tab are now properly fitted within their containers, eliminating overflow that occurred for engines with long display names.

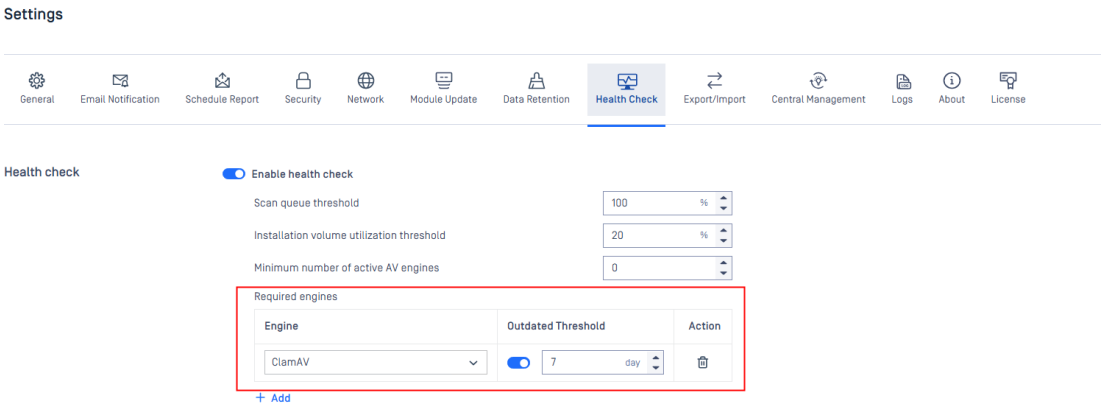
12) Disk space monitoring in health check policies

Health check policies can now monitor the free disk space on the MetaDefender Core installation drive. If available space falls below the configured threshold, the instance is flagged as not ready for file processing.



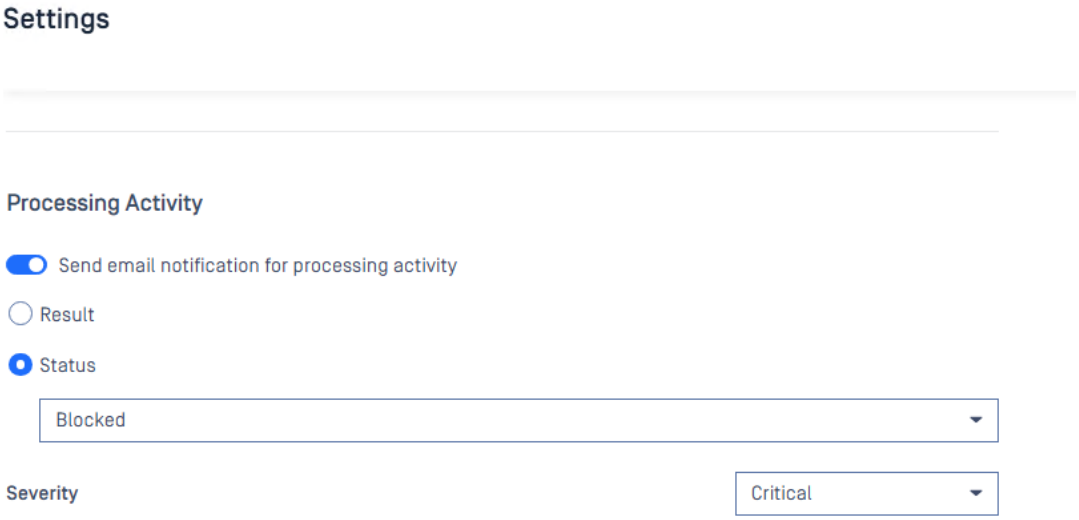
13) Outdated engine detection in health check policies

Health check policies can now flag engines whose definitions have not been updated within a configurable number of days. Administrators can specify which engines are required and how many days old their definitions can be before the health check fails.



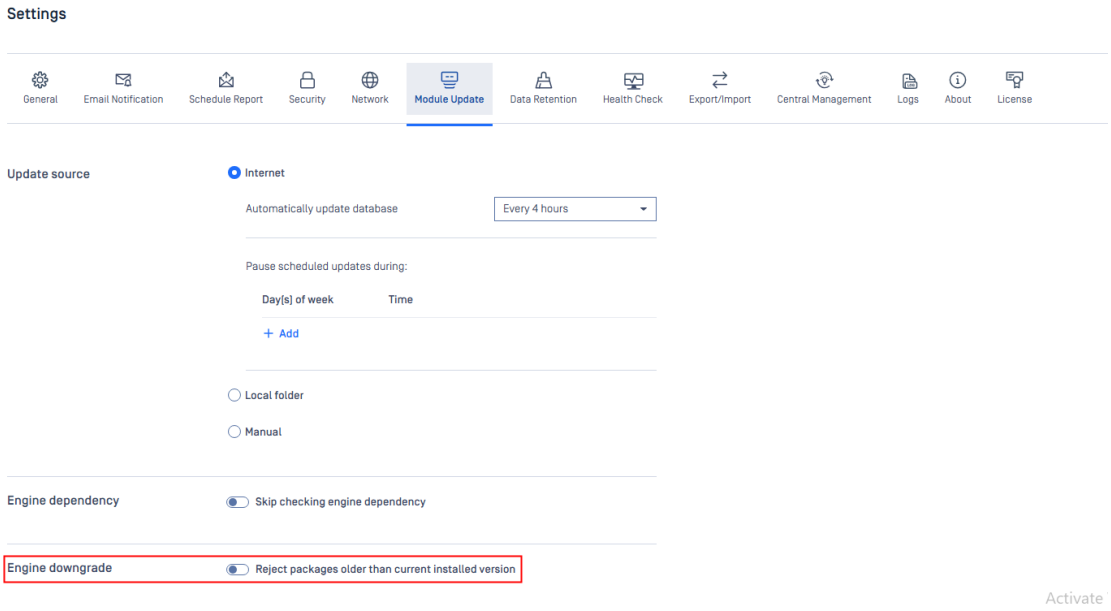
14) Email notifications based on final scan status

Email notifications can now be triggered by a file's final status, such as when a file is blocked, in addition to the existing scan-result triggers (Infected, Suspicious, and so on). This is useful when you care more about enforcement outcomes than the specific findings behind them.



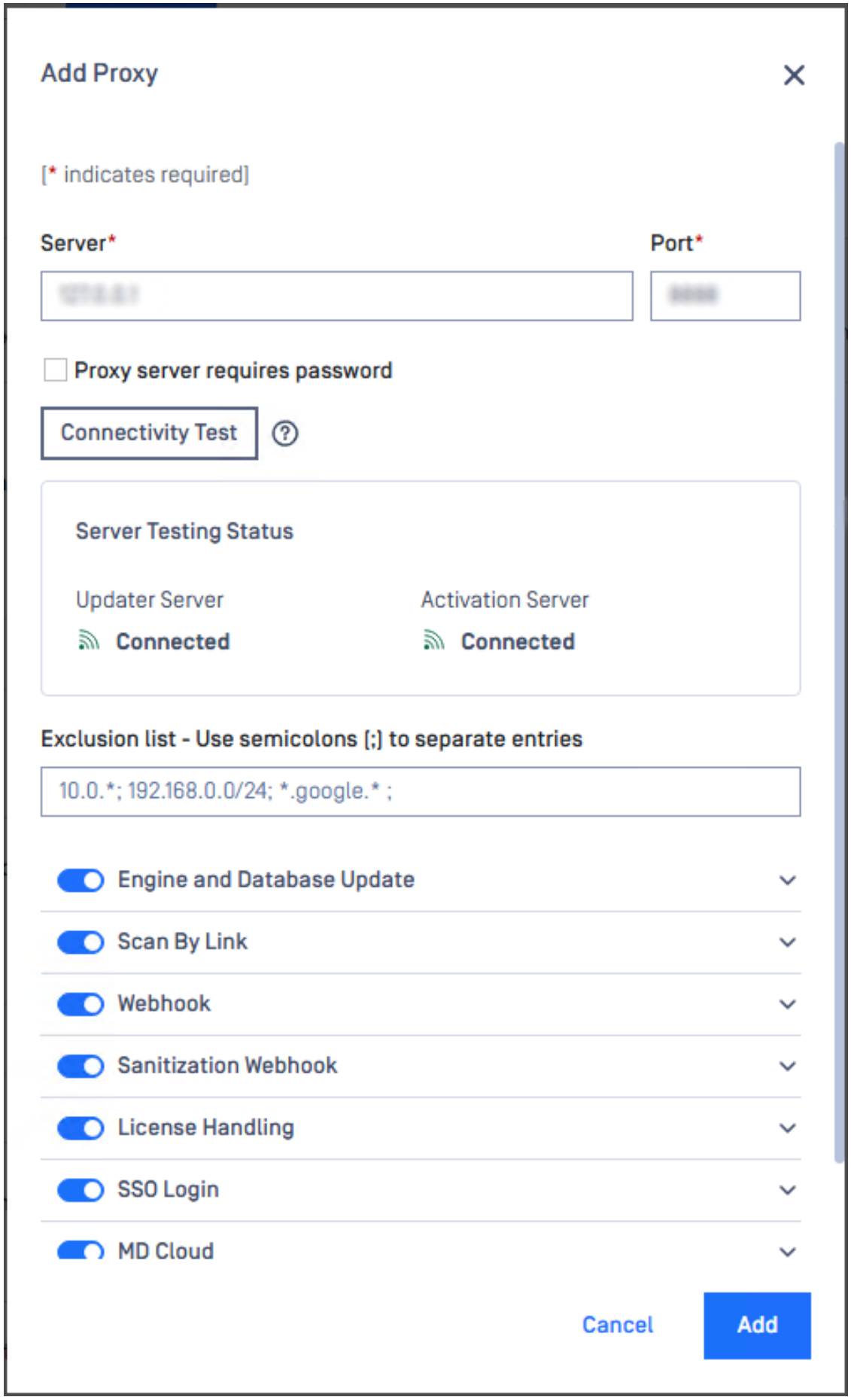
15) Prevent engine downgrade

MetaDefender Core now includes a setting that blocks downgrades of engine and database packages, protecting deployments from accidental rollbacks to outdated threat definitions. This setting is disabled by default.



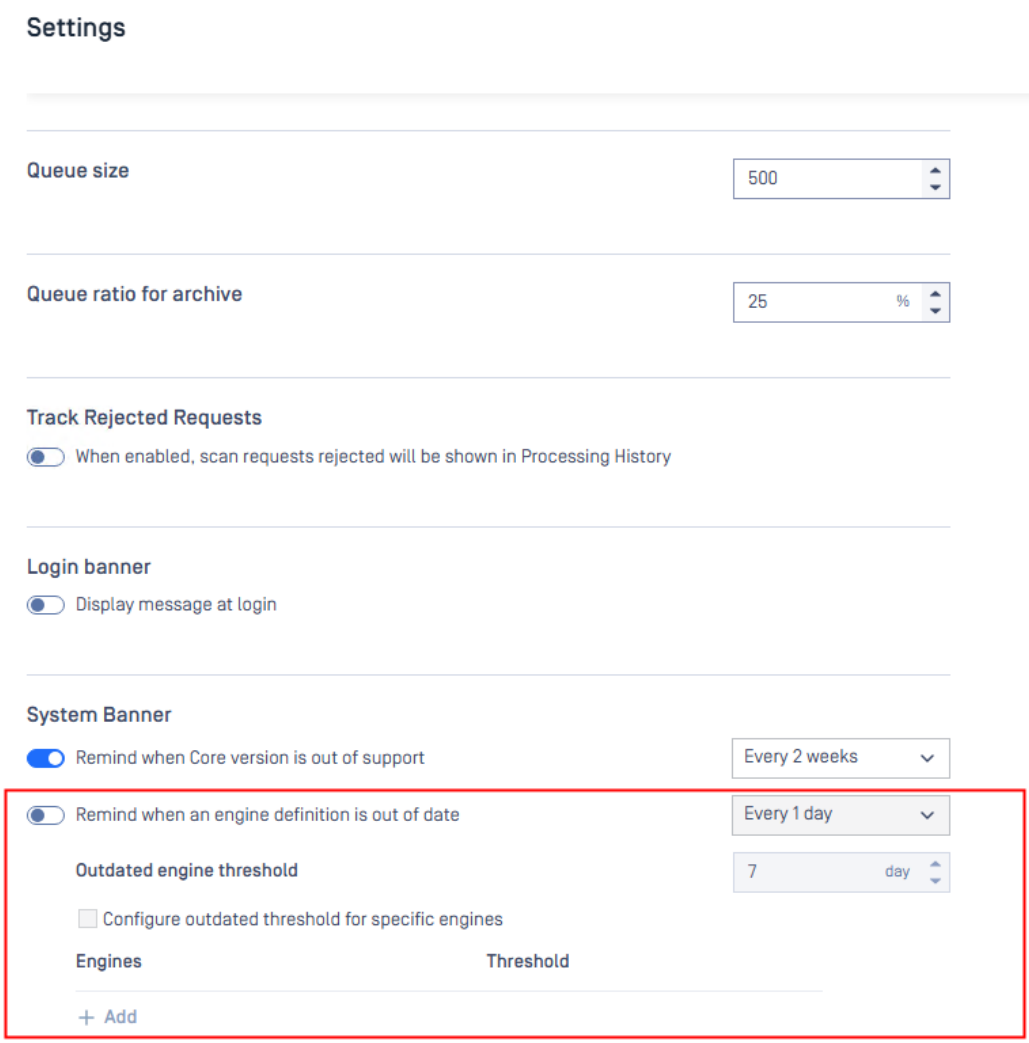
16) Proxy connectivity test

Administrators can now validate proxy settings before saving them. The test verifies connectivity to the updater and activation server, helping avoid misconfigurations that would otherwise only surface when updates silently fail.



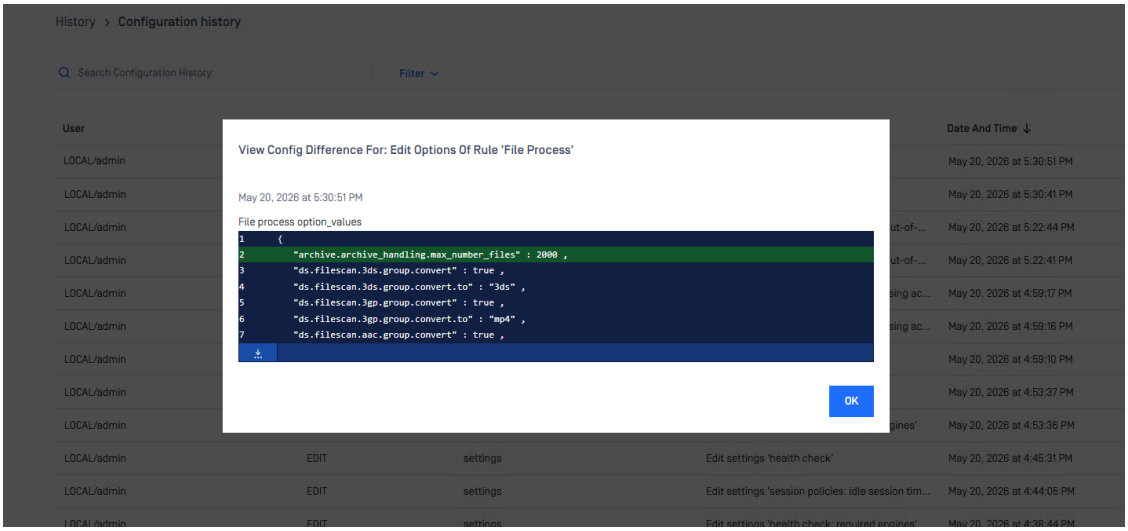
17) Out-of-date engine notification banner

The system banner now includes a setting to notify administrators when scan engines are out of date. A global threshold can be configured in days, with the option to override it per engine.



18) Configuration History — Collapsed Diff View

The Configuration History page now displays only the changed lines in a collapsed diff view, making it easier for administrators to identify what was modified in configuration or workflow policy changes.



Security Enhancements

Upgraded 3rd party libraries:

- NGINX v1.30.1
- PostgreSQL v14.23

Bug Fixes

- Fixed an issue where some My OPSWAT Central Management-related configuration screens reported a successful save while the underlying value failed validation.
- Fixed an issue where the Adaptive Sandbox verdict was reported as "System Error" in the PDF export when the scan was cancelled mid-processing.
- Fixed an issue where archive scan results displayed an "Exceeded Archive File Number" extraction issue without an associated count of affected files. The count is now correctly populated for the root archive.
- Fixed an issue where the workflow rule name filter in Executive Reports did not reflect renamed rules, leaving the old name in the filter. The filter now correctly reflects the current rule name.

Version v5.18.1

Release Date: 15 April 2026

Upgraded 3rd party libraries:

- Libxml2 v2.15.2
- OpenSSL v3.5.6
- Angular libraries

Bug Fixes:

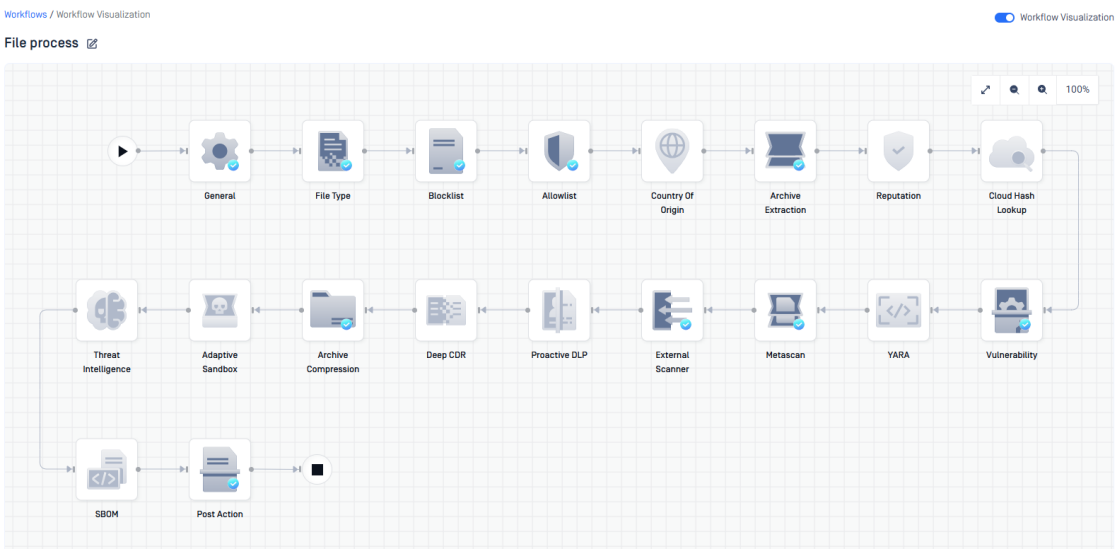
- Fixed a stability issue that, in rare cases under heavy load, could cause MetaDefender Core within MetaDefender Cluster to crash during extraction distribution.

Version v5.18.0

Release Date: 31 March 2026

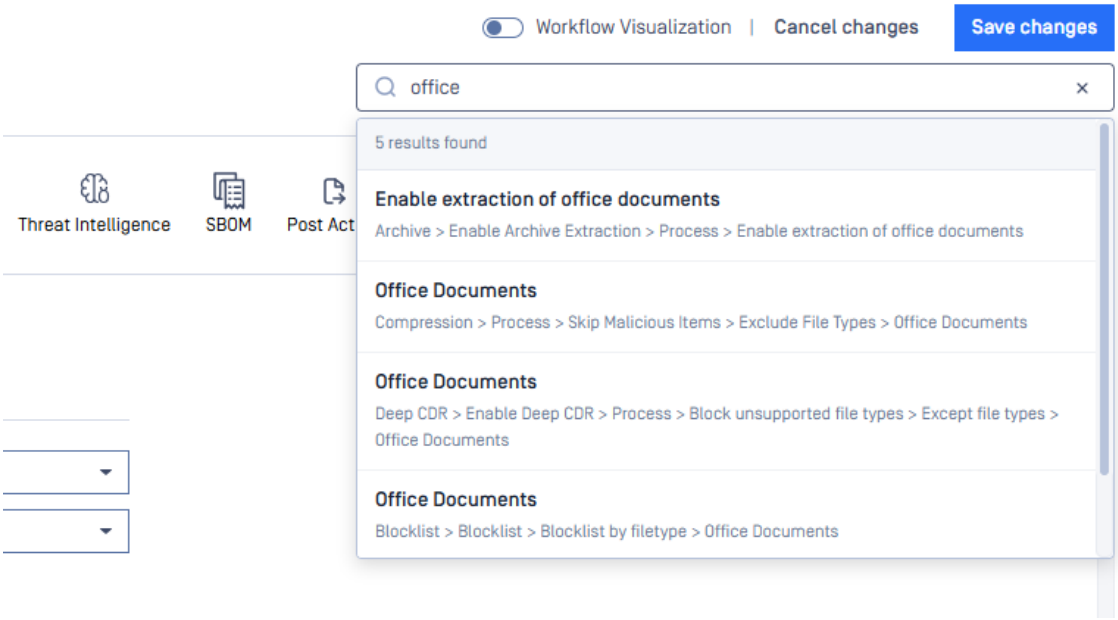
Workflow Visualization

The newly introduced Workflow Visualization provides a high-level overview of scan workflows. By rendering engine execution sequences as intuitive directed graphs, this feature enables users to perform inline configuration edits and search specific settings directly within the visualization interface.



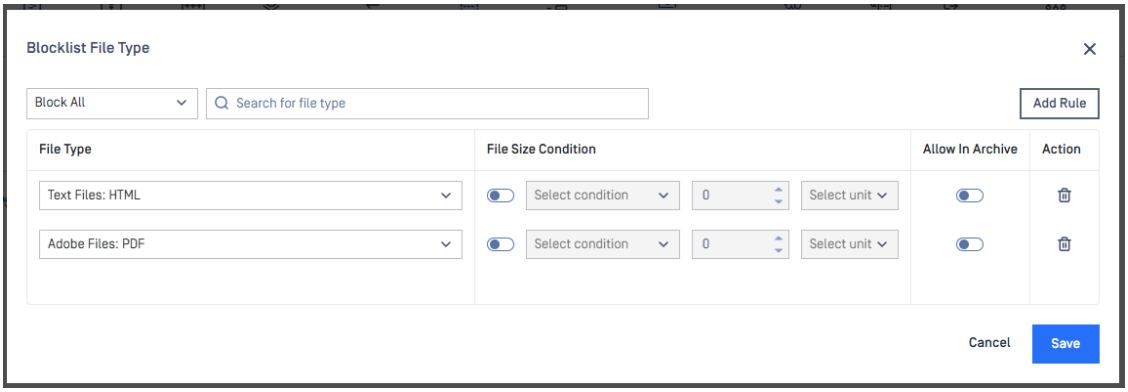
Workflow Setting Instant Lookup

The introduction of global search functionality across workflow rule settings facilitates the instantaneous location of configurations. By providing the full breadcrumb navigation path within search results, the system allows for seamless redirection to any specific setting.



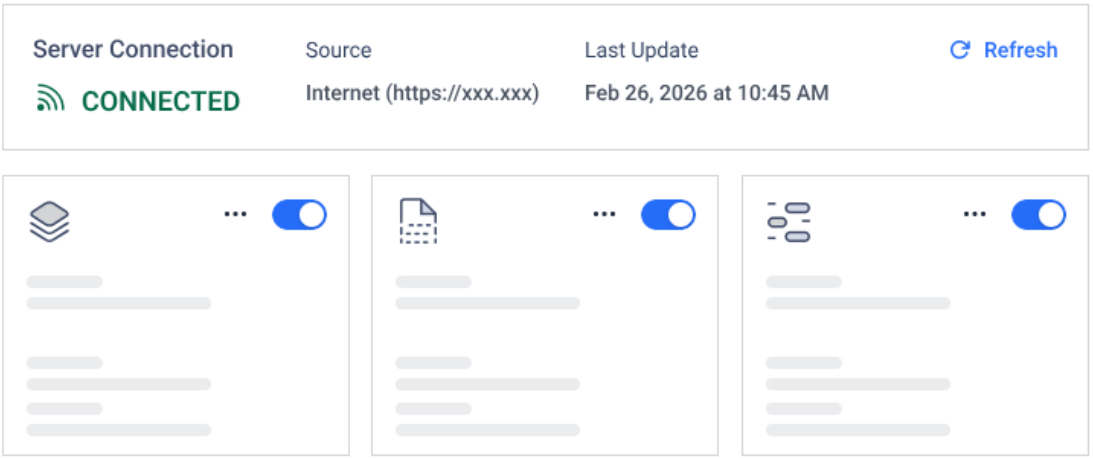
Feature Enhancement: Blocklist and Allowlist File-Type Configuration

Less friction, more control. Users can now configure lists in seconds via a simplified interface, while a dedicated search feature allows for navigation to specific file types. More details at Blocklist by filetype configuration.



Connectivity Monitoring and Troubleshooting

Gain immediate visibility into the connectivity status between MetaDefender Core and its update sources. The system now diagnoses update impediments and provides remediation steps to ensure all engines remain current.



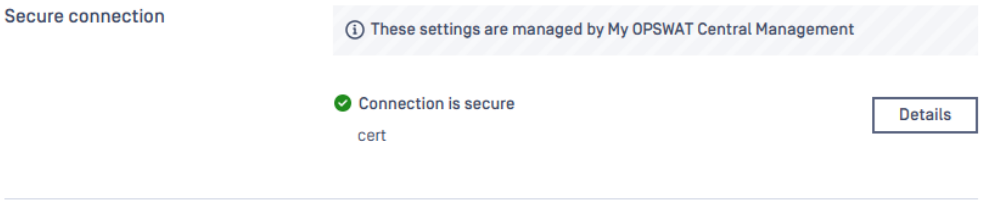
Shared Database Mode supports My OPSWAT Central Management v10

Shared Database Mode is now supported for enrollment in My OPSWAT Central Management v10 (MOCM). This enhancement enables multiple instances sharing a database to be managed through MOCM, though certain limitations apply to certificate management and configuration synchronization in mixed-mode environments.

Please refer to My OPSWAT Central Management for more details.

Centralized SSL/TLS Secure Connection Management

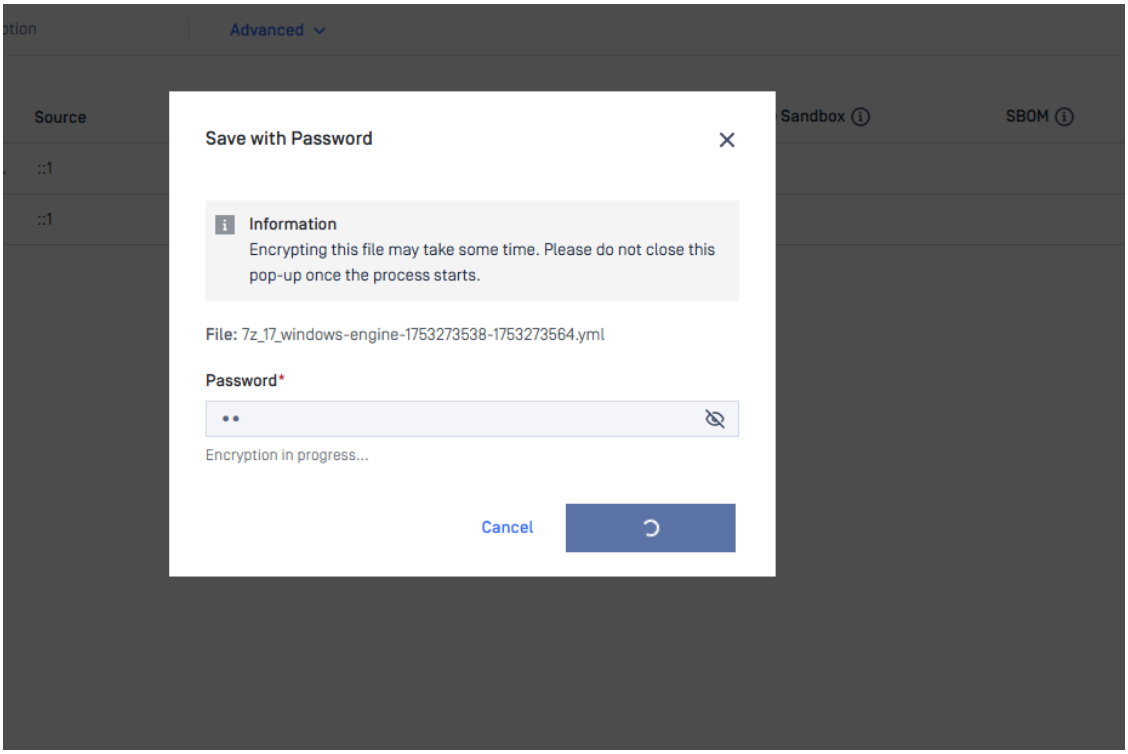
MetaDefender Core has been enhanced to support synchronization of SSL/TLS configurations with MetaDefender Managed Central Management (MOCM). This improvement allows for centralized management of secure connection settings and certificates, ensuring consistent security policies across managed instances.



Encrypted Download for Quarantined Files

Starting from version 5.18.0, users can download quarantined files with password protection. This feature enhances security by allowing files to be encrypted during download, helping prevent accidental execution due to unintended user actions.

More details at Download quarantine files.



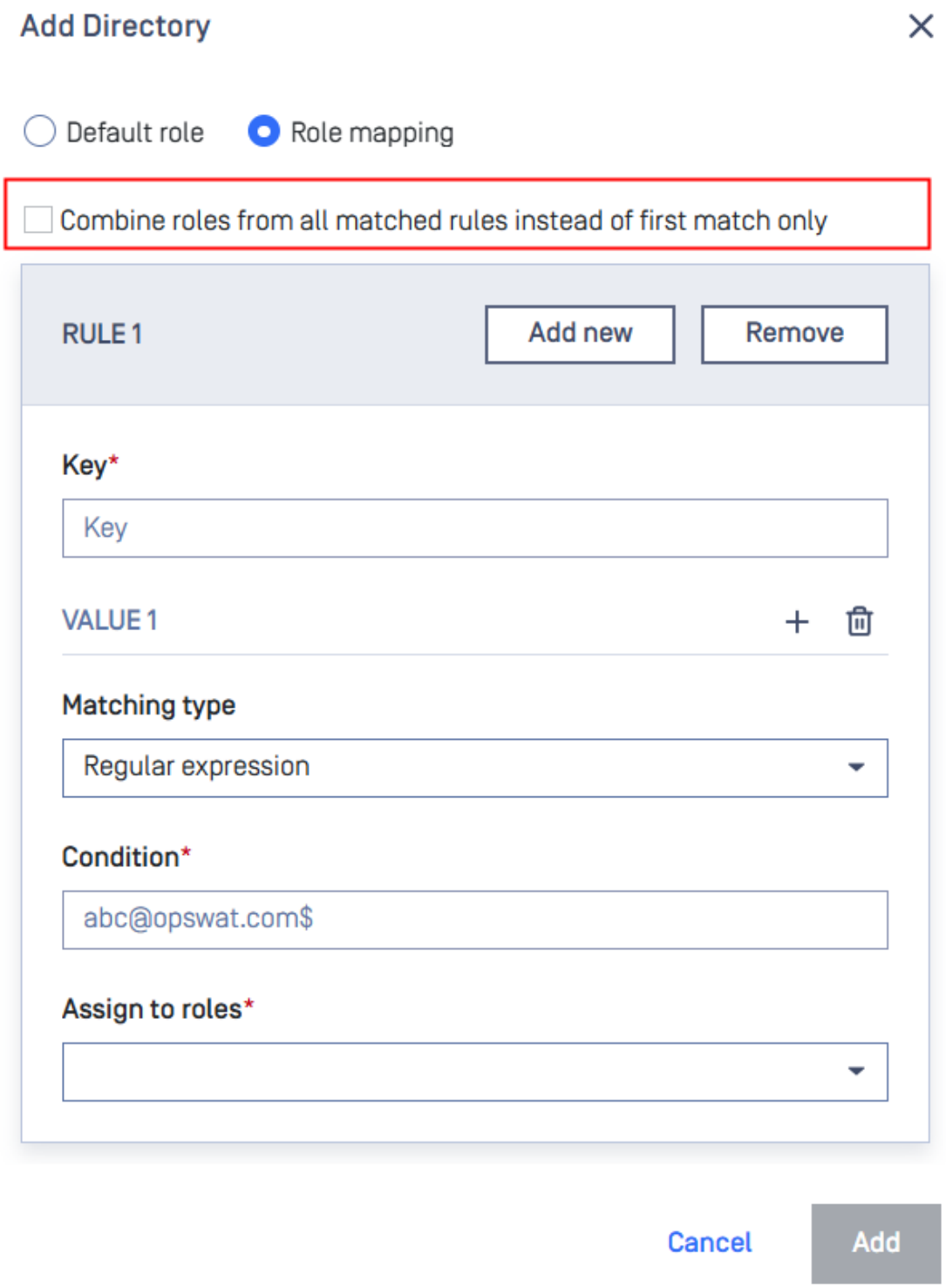
New OS Support

This release of MetaDefender Core introduces official supports for Debian 13, Windows 11 24H2 and Windows 11 25H2.

Note: Windows 11 only for End-User systems (e.g. Kiosk deployments).

New Option to Combine SSO User Roles from All Matching Rules

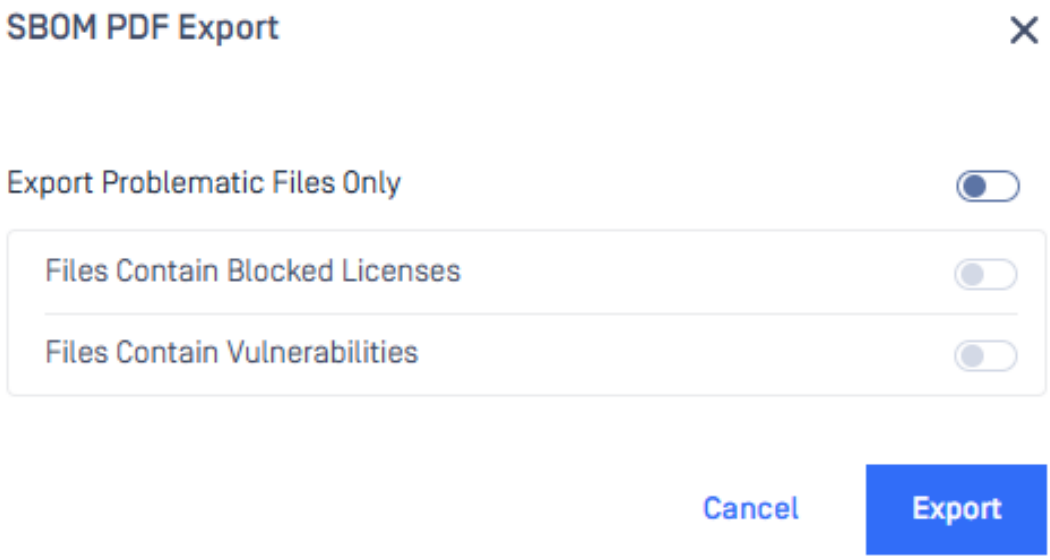
SSO role mapping now supports combining roles from all matched rules, allowing users to receive a broader set of permissions when multiple mappings apply. This option is disabled by default.



Further Enhancements

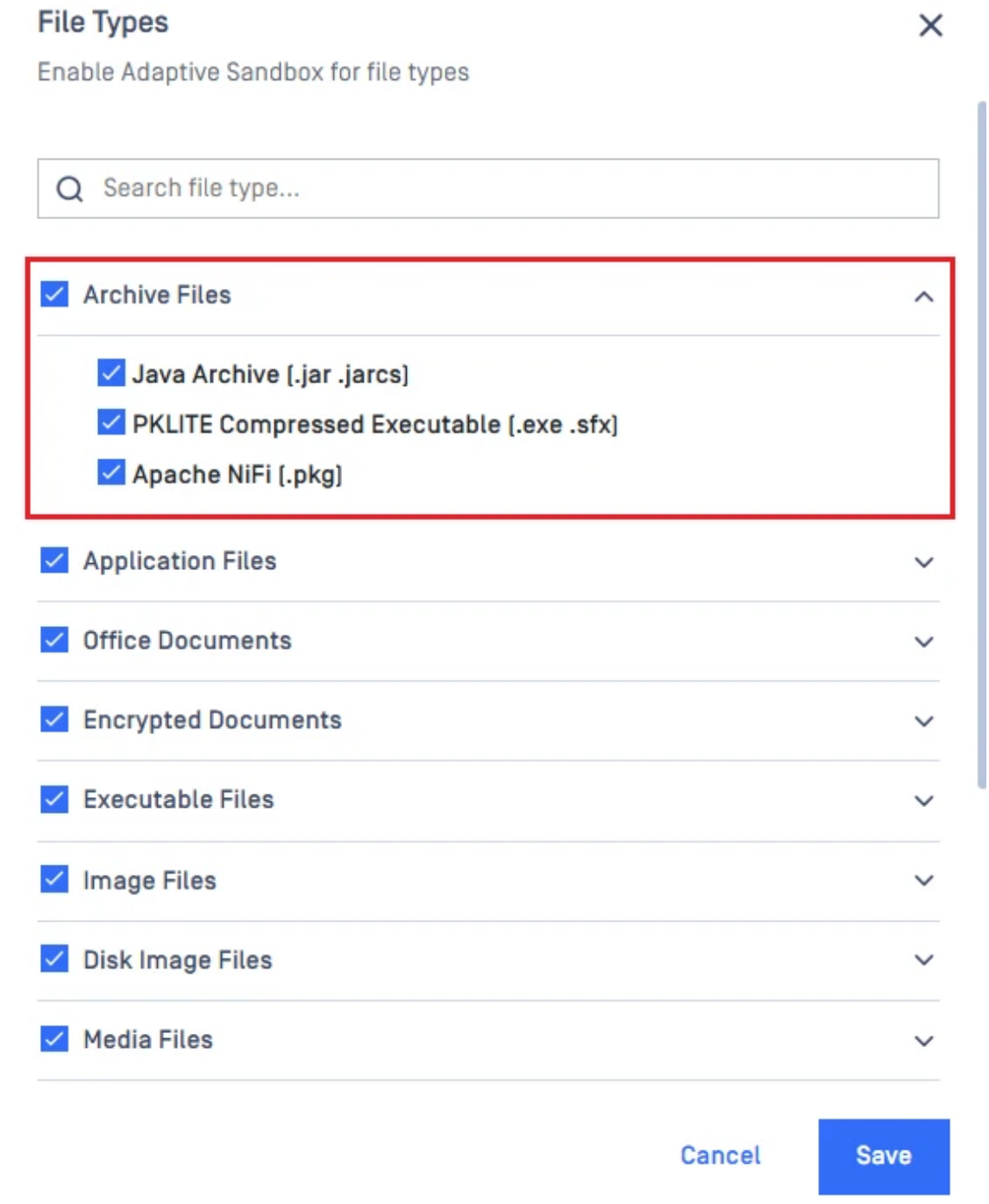
1) Selective SBOM PDF Export Options

The Software Bill of Materials (SBOM) PDF export functionality has been enhanced to allow you to selectively export only vulnerable or problematic files rather than the entire package list. This improvement significantly reduces PDF generation time for large packages and provides a more focused, concise report of security concerns.



2) Default Archive Support for Adaptive Sandbox

Adaptive Sandbox has been enhanced to enable supported archive file types by default, ensuring files such as JAR, SFX, and PKG are automatically processed for security analysis without manual configuration. This feature is available when using MetaDefender Core 5.18.0 or later in combination with Adaptive Sandbox 3.1.1 or later.



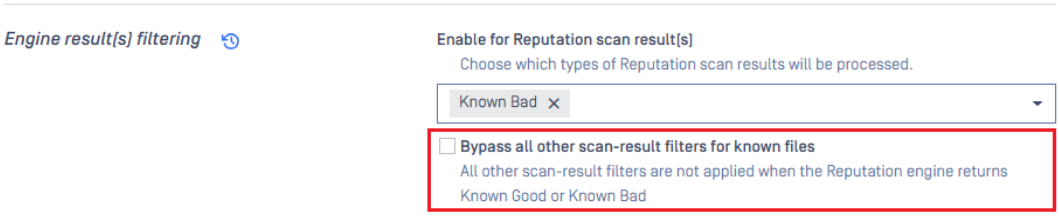
3) Granular Sandbox Trigger Rules for Deep CDR Verdicts

Adaptive Sandbox trigger rules have been enhanced to provide independent configuration options for "Sanitized" and "Sanitized Partially" verdicts. This improvement allows for more granular control, enabling users to trigger sandbox scans specifically for partially sanitized files while excluding fully sanitized ones to optimize resource usage.



4) Enhanced workflow-level filtering of Reputation results for Adaptive Sandbox

MetaDefender Core has been enhanced to allow bypassing all other scan-result filters when a file is identified as Known Good or Known Bad by the Reputation engine. This improvement prevents unnecessary sandbox analysis for files with established reputations, optimizing system performance and resource utilization by default.

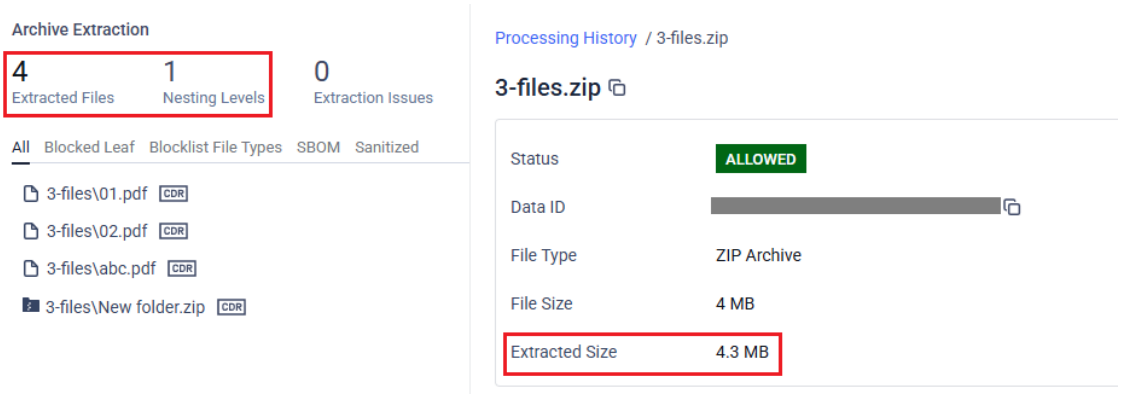


5) Support for Trailing Slashes in OCM Enrollment URLs

MetaDefender Core has been enhanced to automatically handle trailing slashes in My OPSWAT Central Management (MOCM) enrollment URLs.

6) Enhanced Archive Extraction Visibility in Scan Results

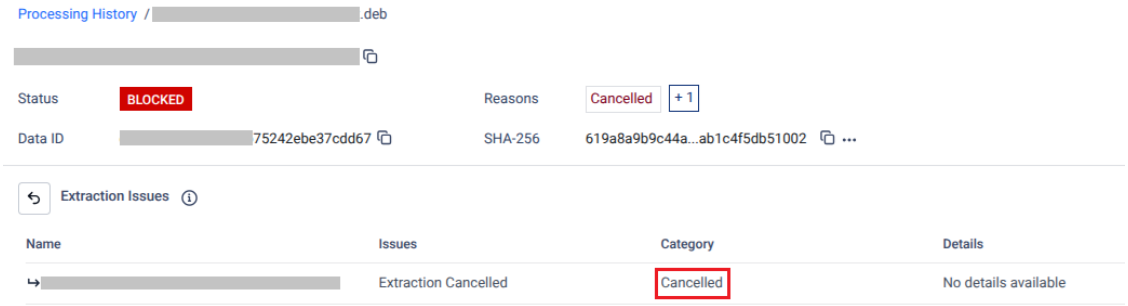
Scan results and executive reports have been improved to provide greater transparency into archive processing by including critical extraction details. These reports now explicitly display the maximum archive nesting level reached, the total number of extracted files across all nested levels, and the total size of extracted data, allowing for better configuration of workflow extraction limits.





7) Clearer Extraction Status for Cancelled Archive Scans

Archive extraction monitoring has been enhanced to accurately report when a scan is manually cancelled. Extraction issues previously categorized as "Extraction Error / Unknown" due to canceled are now specifically identified as "Extraction Cancelled," providing clearer visibility into scan termination and ensuring a more precise audit trail for multi-level archive processing.



8) Simplified Deep CDR Final Verdicts in API Responses

Scan result API responses (GET - Fetch Analysis Result and GET - Fetch Analysis Result By Hash) have been enhanced to include a top-level engine results object. This improvement introduces a `sanitization_result` field that provides a clear, simplified verdict of "Success," "Failed," or "Not Run," streamlining programmatic processing by eliminating the need to parse multiple internal fields across different file types.

9) Layout Refinements and Minor Adjustments for the PDF Processing Result Report

The PDF Processing Result Report has been improved with a more refined layout and formatting.

10) Enhanced Threat Intelligence Verdict Visualization on Dashboard

The dashboard has been enhanced to apply standardized color coding for Threat Intelligence engine verdicts, ensuring visual consistency with other analysis engines. "Allowed" verdicts now correctly display in green and "Blocked" verdicts in red within dashboard filtering and statistics.



11) Improved Formatting for Adaptive Sandbox Verdicts in Executive Reports

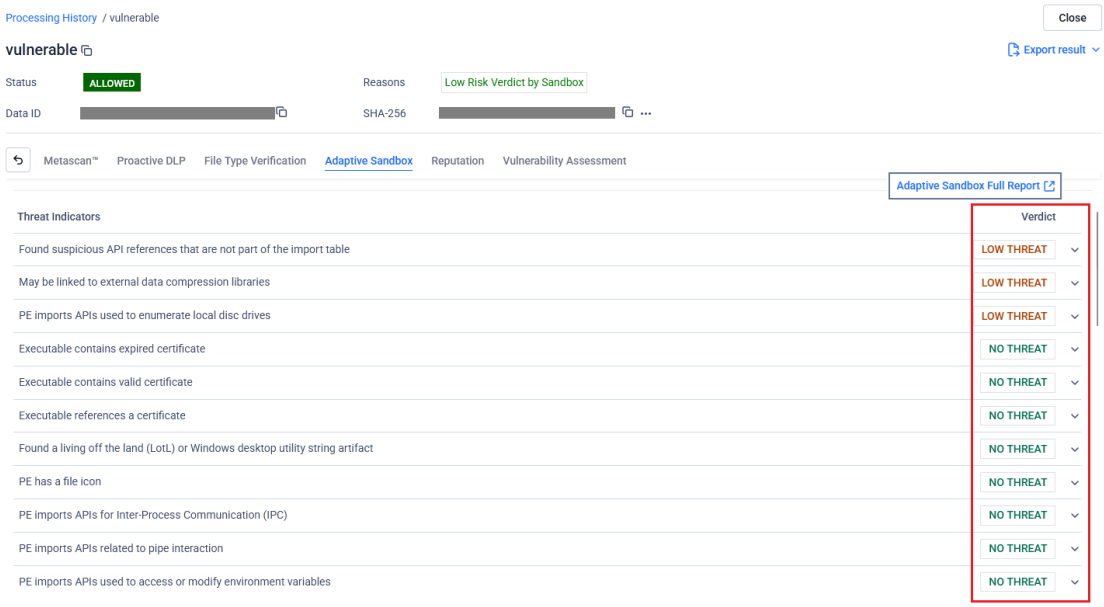
The Adaptive Sandbox verdict display in the Executive Report has been enhanced for better readability. Adaptive

Sandbox results now utilize spaces instead of underscores and feature proper capitalization for each word.

Result	Number of Objects	Percentage	Change
Confirmed Threat	1	25%	-
Low Risk	1	25%	-
No Threat Detected	1	25%	-
Trusted	1	25%	-

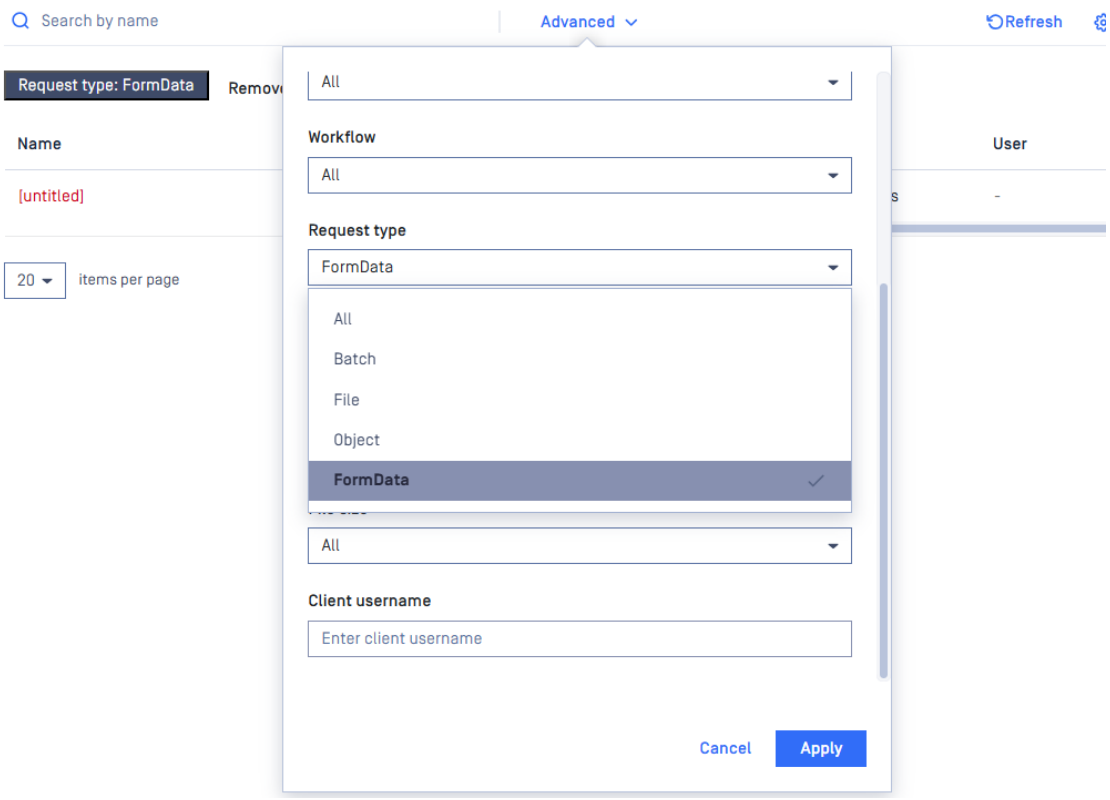
12) Enhanced Processing Result Label Styles

The UI for processing result preference labels has been improved to align with the latest design standards, featuring updated font weights, background transparency, and refined border coloring. These enhancements ensure a consistent and modern visual experience across the management console.



13) Support new filtering for multipart/form-data request

Requests using the `multipart/form-data` content type can now be searched in the Processing History by filtering for **FormData** under the `Request Type`.



14) Webhook optimization

Significant improvements have been made to the webhook response mechanism, enhancing stability, reducing TCP usage, and improving overall response time.

15) Multiple enhancements for generate support package

- Lightweight mode: Optional lightweight support package reduces size and shortens generation time by collecting a smaller diagnostic set.
- Custom time range: Support package generation for MetaDefender Core supports a custom time range so collected logs and data match the period you select.

16) Email notification enhancement

- Out-of-date definitions (per engine): Administrators can control out-of-date definition notifications per engine instead of a single global setting.
- Scheduled reports: quarterly period - Adds Quarterly as an option for report time range and schedule time. The history report covers the full calendar quarter; the email is sent on the first day of the next quarter at the hour the user picks in the web console.

Security Enhancements

1. Enhanced log file security and validation: the log file configuration has been enhanced to include strict path and extension validation, preventing the use of unauthorized locations or system files. This improvement ensures system integrity by restricting log files to approved formats such as .log, .txt, .json, .csv, and .out.
2. Upgraded 3rd party libraries:

- NGINX v1.29.7
- OpenSSL v3.5.5
- PostgreSQL v14.22
- 7z v26.00
- Zlib v1.3.2
- Nghttp2 v1.68.1

Bug Fixes

- Fixed an issue where the system would potentially crash when a package or engine was removed while still being downloaded.
- Fixed an issue where initiating multiple license activation requests concurrently could potentially cause engine reloads to time out or fail.
- Resolved an issue where exported PDF results for Threat Intelligence engines would display incorrect or redundant options when the engine name was changed.
- Fixed an issue where validation icons failed to appear in error messages when the synchronization time was out of range.
- Fixed an issue where pickup folder is empty and cannot monitor after network issue/or folder not found.
- Fix an issue that prevented Proactive DLP engine from being deployed on Windows due to the long file path limitation.
- Fix an issue where 1 MB `multipart/form-data` requests failed when submitted to MetaDefender Core.
- Fix an issue where MetaDefender Core failed to parse `multipart/form-data` requests when the boundary parameter was enclosed in quotes.

Version v5.17.1

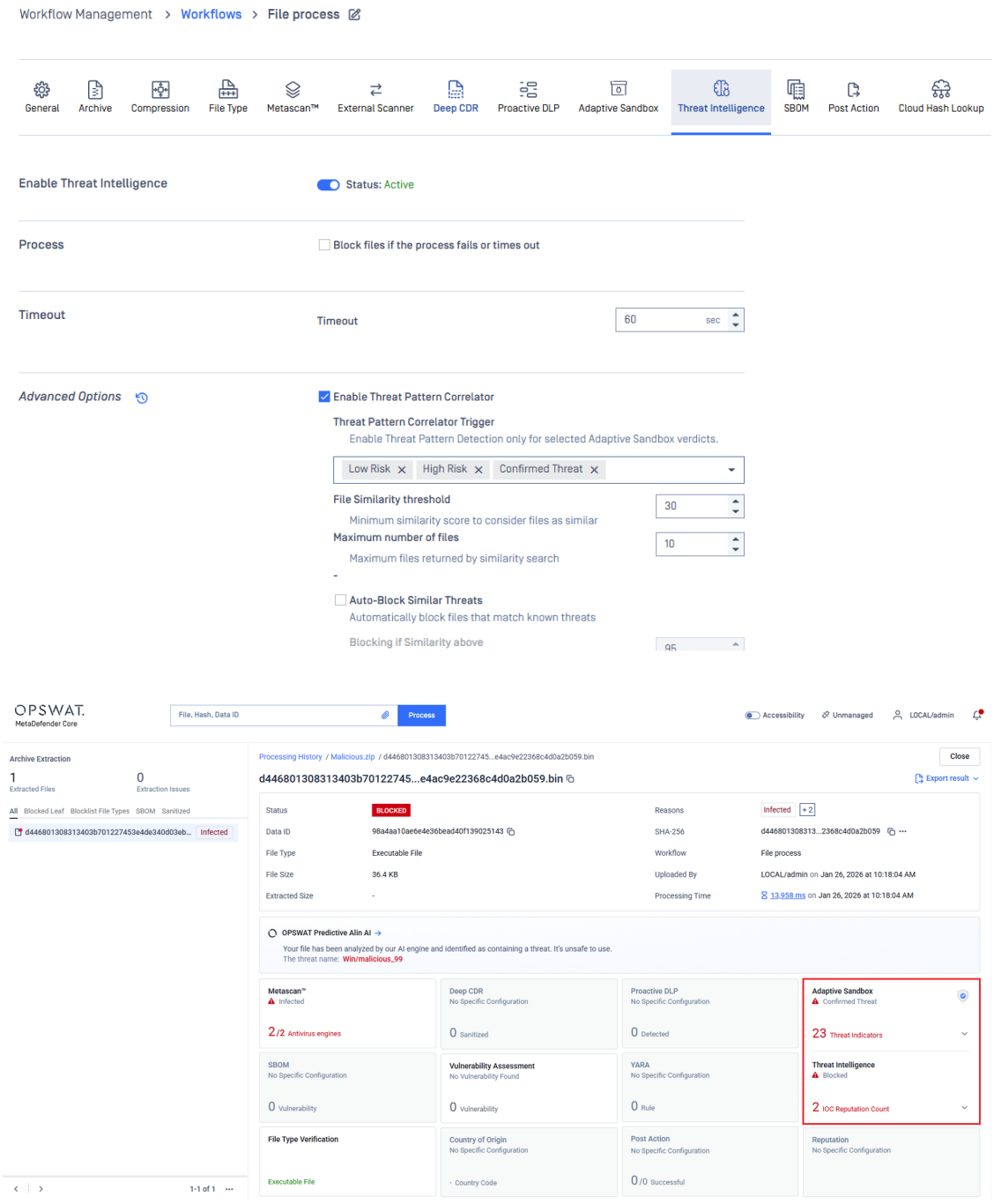
Release Date: 26 January 2026

Next-Gen Threat Intelligence Engine and MetaDefender Aether

MetaDefender Core introduces a branch new technology called **MetaDefender Aether**, a next-generation technology that brings Adaptive Sandbox and Threat Intelligence together into one powerful, unified pipeline. With Aether, security teams can detect, analyze, and respond to advanced threats with unprecedented speed and precision. This upgrade streamlines your threat detection workflow, reduces complexity, and delivers deeper insights, empowering you to stay ahead of even the most sophisticated cyber threats.

For more details about the all-new Threat Intelligence engine, please refer to this documentation: Threat Intelligence.

For details on the Threat Intelligence response JSON format, as exposed in the new `threatintelligence_info` field, see: GET - Fetch Analysis Result

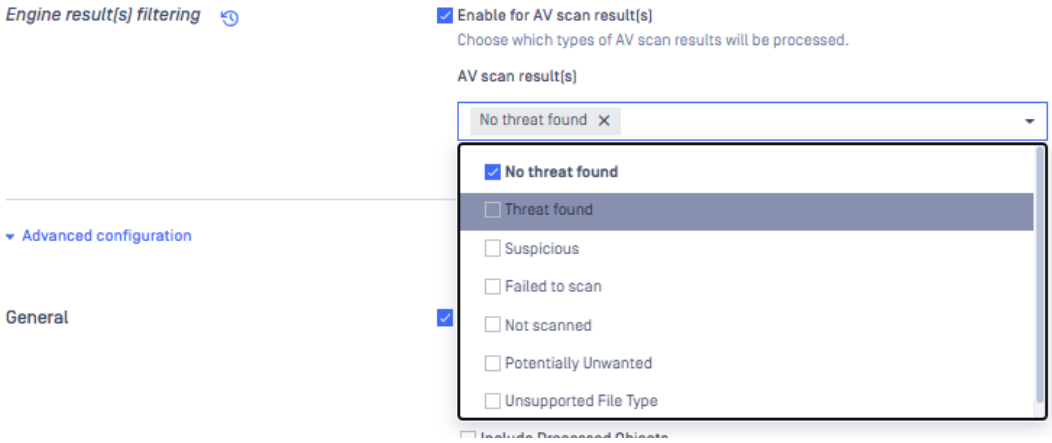


Workflow Option for Running Deep CDR Only on Certain Metascan Verdict

Added the ability to configure Deep CDR in the workflow to trigger only for files that are reported as infected by any of the scanning engines in Metascan. Clean files will not be sent for Deep CDR, and the workflow will stop after the initial Metascan stage with the corresponding scan status.

Default behavior/value: Disabled.

When enabled, only infected files will be processed by Deep CDR, while clean files bypass the sanitization stage.



Enhanced Regex Management for Proactive DLP

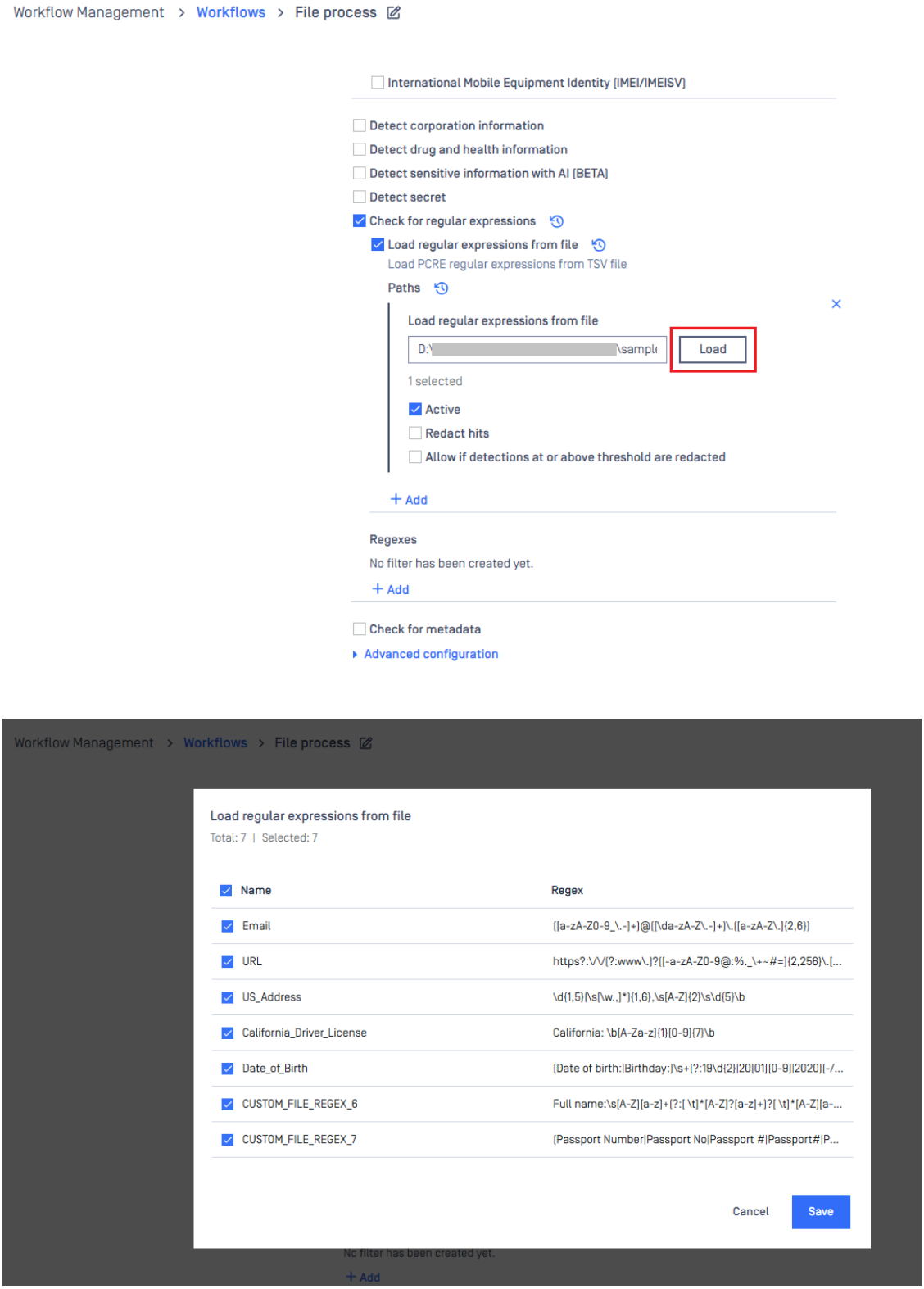
MetaDefender Core made it easier for admins to manage and select regex patterns for Data Loss Prevention (DLP) workflows. With this update, you can now upload multiple TSV files containing regex definitions, view their contents in a slick pop-up modal, and pick exactly which regexes you want to enable from the Workflow console.

Please note the following as of now:

1. The “Pick regexes” pop-up model is not yet supported in My OPSWAT Central Management. Instead, you can provide the path to the TSV file in the MetaDefender Core workflow configuration on the My OPSWAT Central Management console. By default, all regexes in the TSV file are selected, and you can edit the TSV to remove existing regexes or add new ones.
2. The DLP engine currently supports only the pop-up model for Detection configuration, but this may be expanded with additional options in the future.

Here is a sample TSV format:

```
Email	([a-zA-Z0-9_\.-]+)@([\da-zA-Z\.-]+)\.([a-zA-Z\.]{2,6})
URL	https?:\/\/(?:www\.)?([-a-zA-Z0-9:%._\+~#=]{2,256}\.[a-z]{2,6}\b)*(\/[\/\d\w\.-]*)*(?:[\?])*(.+)*
US_Address	\d{1,5}(\s[\w.,]*){1,6},\s[A-Z]{2}\s\d{5}\b
California_Driver_License	California: \b[A-Za-z]{1}[0-9]{7}\b
Date_of_Birth	(Date of birth:|Birthday:)\s+(?:19\d{2}|20[01][0-9]|2020)[-/.](?:0[1-9]|1[012])[-/.](?:0[1-9]|[12][0-9]|3[01])\b
Full name:\s[A-Z][a-z]+(?:[ \t]*[A-Z]?[a-z]+)?[ \t]*[A-Z][a-z]+\b
(Passport Number|Passport No|Passport #|Passport#|PassportID|Passportno|passportnumber)\W*\d{9}\b
```

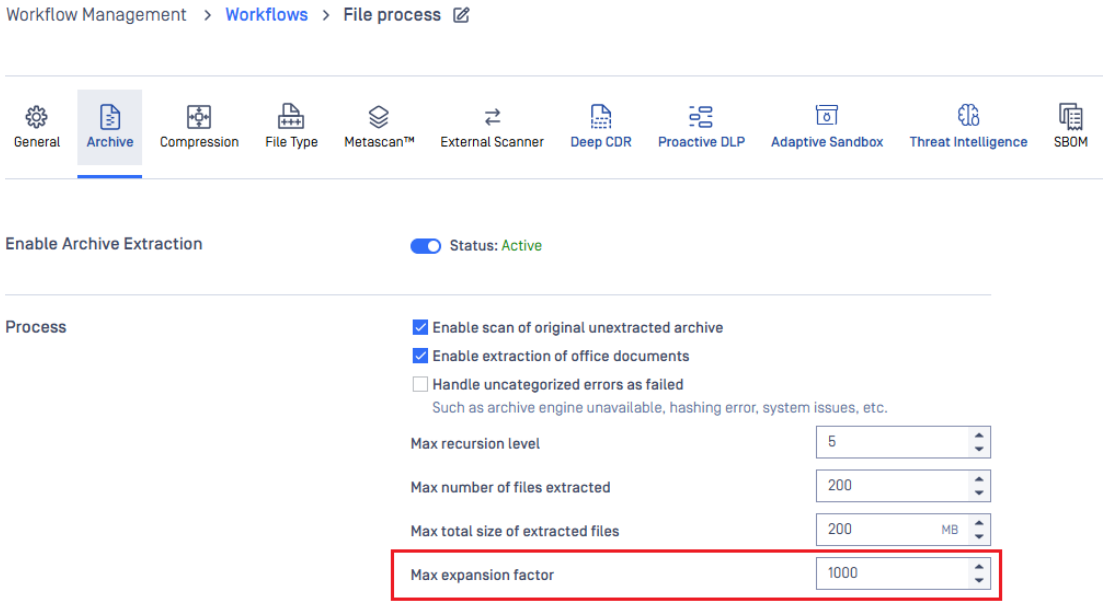


New Threshold to Strengthen Protection Against Archive Bomb

MetaDefender Core introduced a new archive configuration that lets you control the maximum allowed compression expansion factor for extracted archives. This helps prevent “zip bomb” attacks, where a tiny archive expands into a massive file and can overwhelm your system.

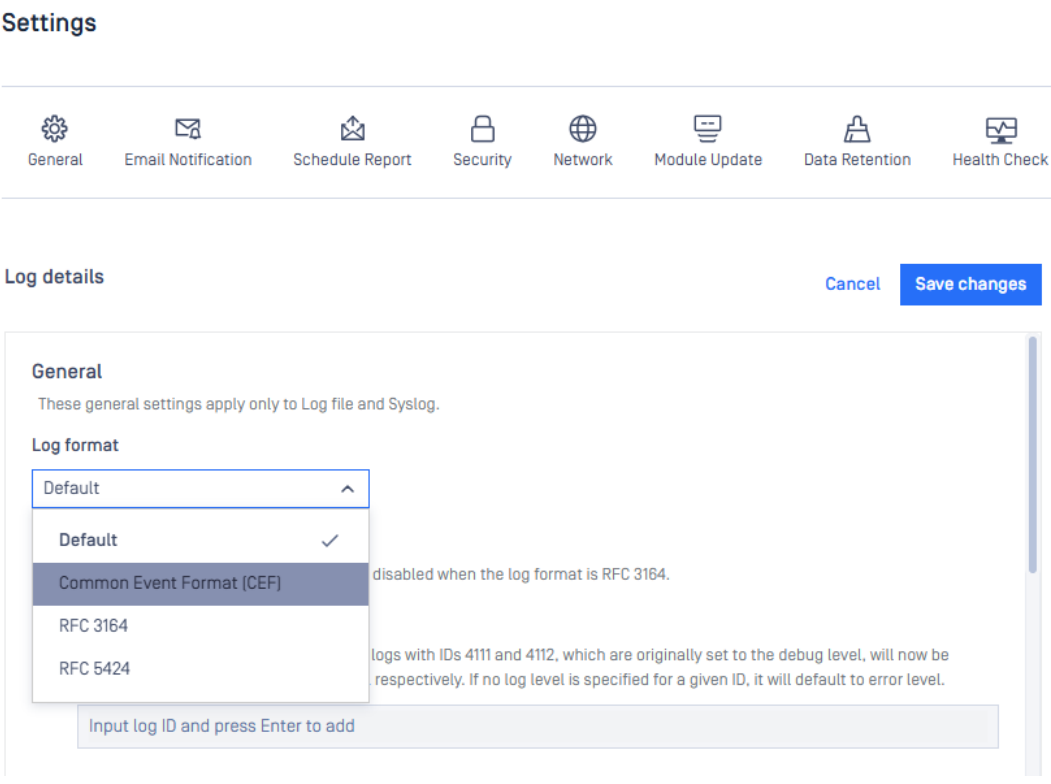
By default, the system uses a high threshold (1000:1), so existing customers won’t be affected unless they choose

to tighten the setting. If an archive exceeds the configured value, it's now blocked with a new verdict: **Exceeded Archive Expansion Factor**.



Support to Configure RFC 5424 and RFC 3164 for Syslog

The application now allows selecting the syslog message format used for CEF output to integrate with different SIEM systems without breaking existing deployments. Supported formats include: Default, CEF, RFC 3164, and RFC 5424.



Support Content-type: multipart/form-data for POST /file API

Before version 5.17.1, MetaDefender Core only supported `Content-Type: application/octet-stream` where clients must submit whole content of one file in HTTP payload.

Now with version 5.17.1, asynchronous scan of the product supports a new content type, `Content-Type: multipart/form-data`. Utilizing this new content type, clients may submit multiple files in one request payload. The product will treat and process each part of the form-data as an individual file scan and associated with the same batch-like scan.

This scan type is not supported for requests that include the following headers: batch, filepath, sanitizedurl, or downloadfrom.

Notes: Before upgrading to v5.17.1 or later,

1. Verify whether your client applications are using a relevant Content-Type header and its corresponding payload for Analyze File (Asynchronous mode) - POST /file.

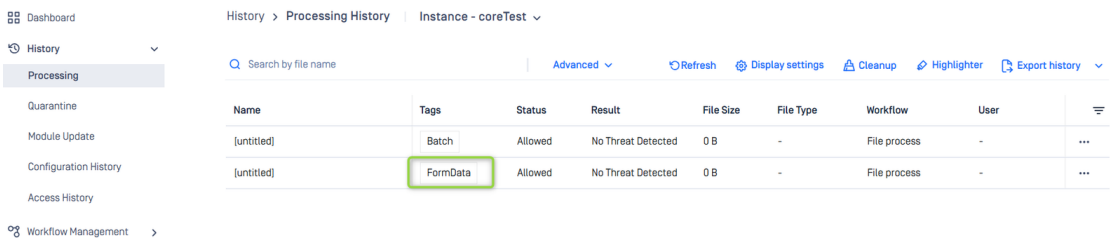
- `application/octet-stream`: The payload must contain exactly one file's content.
- `multipart/form-data`: The payload may contain multiple files' content and must strictly adhere to the standard format (including `Content-Disposition`, `Content-Type`, and the raw data). Non-compliant payload will result in an HTTP 400 error and the message: "Invalid multipart/form-data payload received".

2. Verify that client-side parsing logic for the endpoint Analyze File (Asynchronous mode) - POST /file aligns with the submitted `Content-Type` header:

- `Content-Type: application/octet-stream`: The API returns a `data_id`.
- `Content-Type: multipart/form-data`: The API returns a `batch_id`.

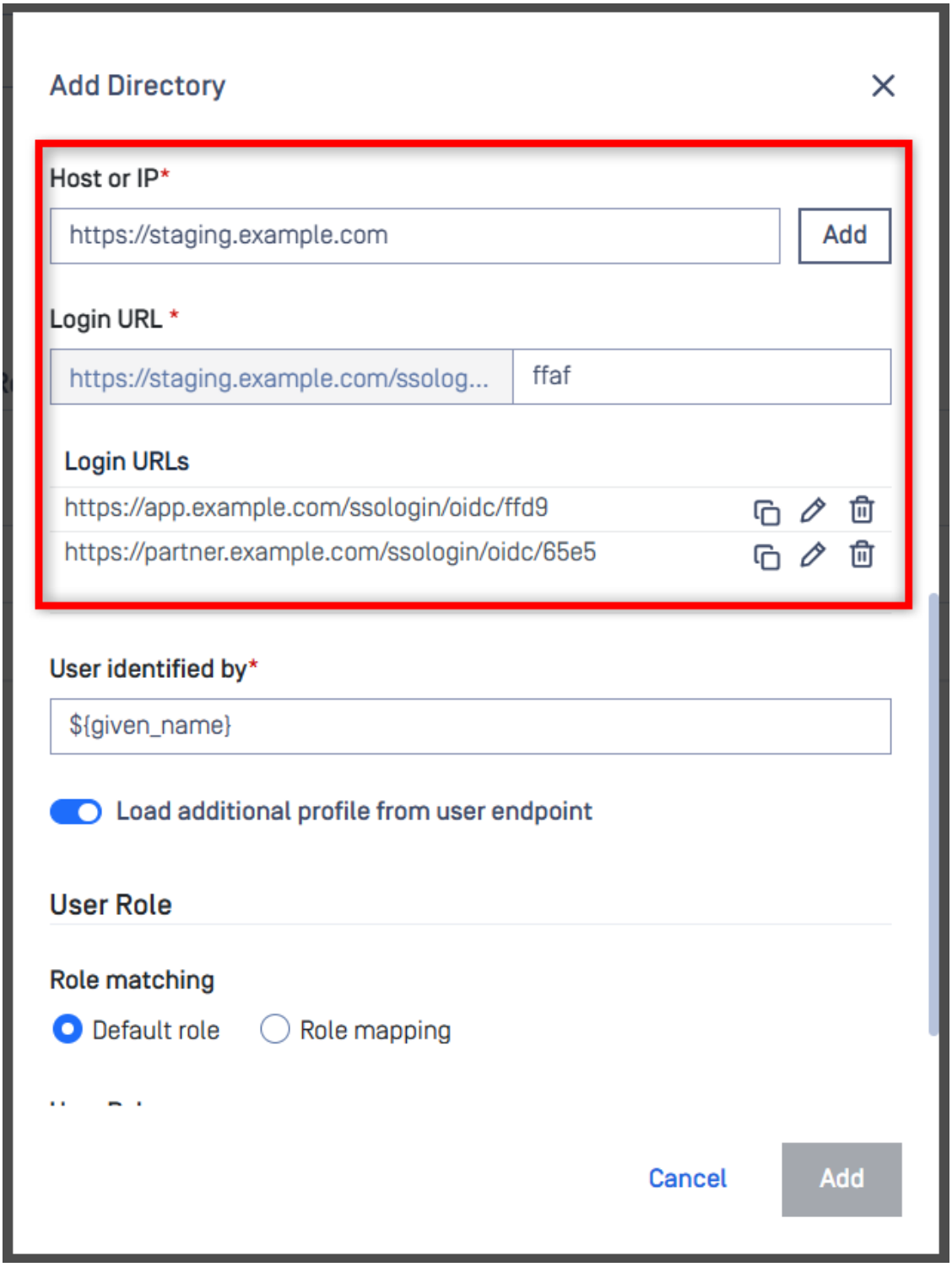
3. Any improper implementations sending non-standard payload or expecting a `data_id` while using multipart/form-data headers must be refactored.

Users can also view this new kind of request in Processing History, having tag "FormData".



Support Multiple Login URLs for SSO Directories

MetaDefender Core now supports **multiple** login/redirect URLs as part of the SSO authentication flow. This enhancement allows deployments spanning multiple domains or client applications to dynamically route users to the correct redirect endpoint during login.



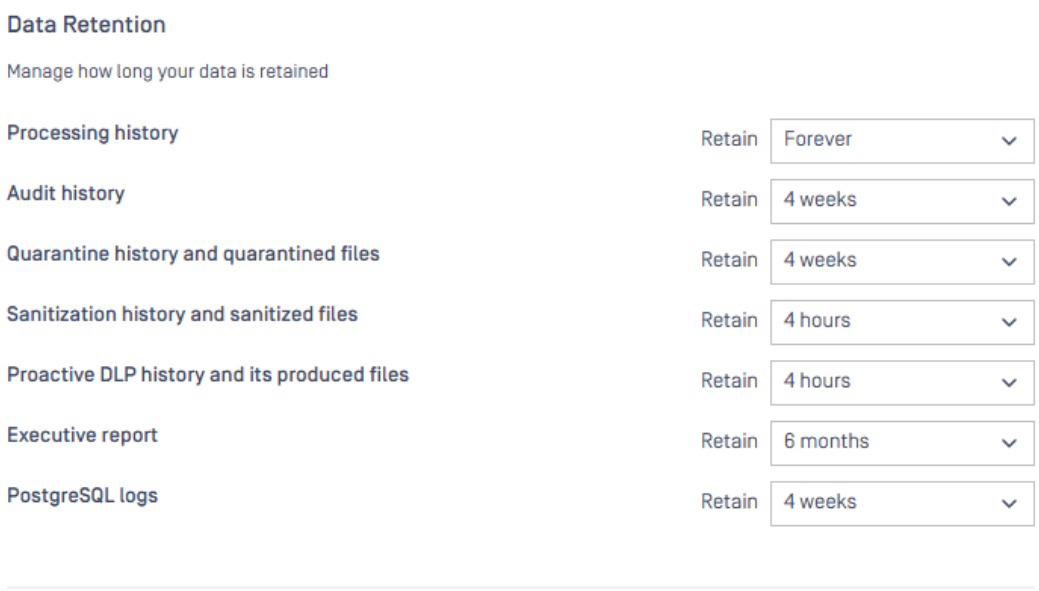
IPv6 Support for MD Core - My OPSWAT Central Management Integration

MetaDefender Core can now communicate seamlessly with My OPSWAT Central Management in IPv6 environments. This ensures that organizations transitioning to IPv6 can confidently deploy and operate My OPSWAT Central Management in modern network domains.

Further Enhancements

1) Data Retention Term Change for "Off" Option

Changed the Data Retention "Off" option label to a clearer term to avoid confusion. The previous "Off" label under the "Data retention" section was misleading; it now displays as "Retain: Forever" to better indicate the actual behavior.



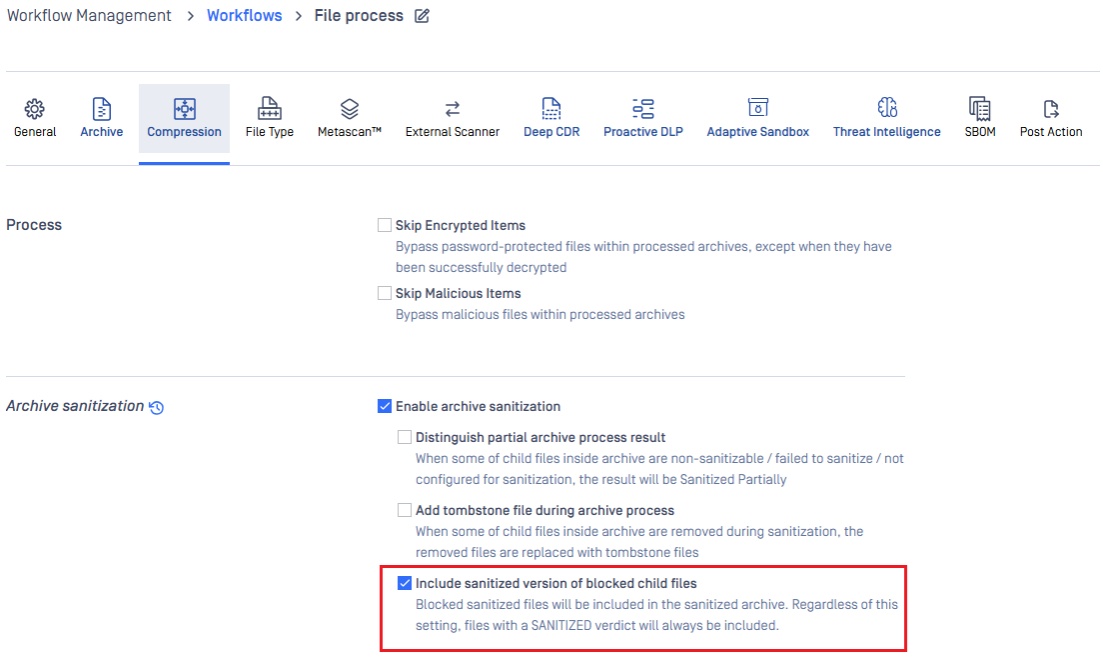
2) Add Synchronous File Scan Timeout to Config Export and Import

The synchronous file scan timeout setting (`/admin/config/file/sync`) is now included in the configuration export and import package.

By default, the `file_sync_timeout_setting` is included in the exported package.

3) UI Streamlining: Option Relocated for Better Usability

To make configuration more intuitive, the “Include sanitized version of blocked child files” option has been moved from the Deep CDR tab to the Compression tab. This update helps users find and manage this setting where it makes the most sense, without changing any underlying functionality.



4) Rename OPSWAT Alin Engine

OPSWAT Alin engine is now renamed to OPSWAT Predictive Alin AI.

5) Enhance Archive Extraction Issues Filtering Performance

In MetaDefender Core v5.17.0, you can filter Extraction Issues in the Processing History. However, the performance of this filtering needed improvement, and it has been optimized in MetaDefender Core v5.17.1.

6) Improved Error Categorization for Archive Extraction Issues

MetaDefender Core made it easier to understand and troubleshoot archive extraction errors.

Previously, when an archive exceeded certain extraction limits (like file count, size, or depth), the error was shown as "Unknown." Now, these are clearly categorized as either Extraction Limit Exceeded or Extraction Timeout Exceeded.

7) Update Adaptive Sandbox Displayed Verdicts on Web Console

Starting from MetaDefender Core version 5.17.1, verdicts from the Adaptive Sandbox will be changed in the product web console and other related reports. Note that only the UI and reports are updated; the JSON response remains unchanged to ensure compatibility with existing integrations. The detailed changes are as follows:

Previously displayed verdict	Newly displayed verdict
Benign	Trusted
No Threat / No Threat Detected	No Threat Detected
Unknown	Undetermined
Suspicious / Suspicious Verdict by Adaptive Sandbox	Low Risk / Low Risk Verdict by Adaptive Sandbox
Likely Malicious / Likely Malicious Verdict by Adaptive Sandbox	High Risk / High Risk Verdict by Adaptive Sandbox
Malicious / Malicious Verdict by Adaptive Sandbox	Confirmed Threat / Confirmed Threat Verdict by Adaptive Sandbox

Security Enhancements

Upgraded libraries for vulnerability fixes:

- NGINX v1.29.4
- c-ares v1.27.0

Bug Fixes

- Addressed the crash issue that could occur when clients push files during MetaDefender Core service startup.
- Addressed an issue where API requests failed due to RAMDISK Temp Directory setup and using the `downloadfro`

`m` header - causing incorrect disk space comparison.

- Other cosmetic fixes.

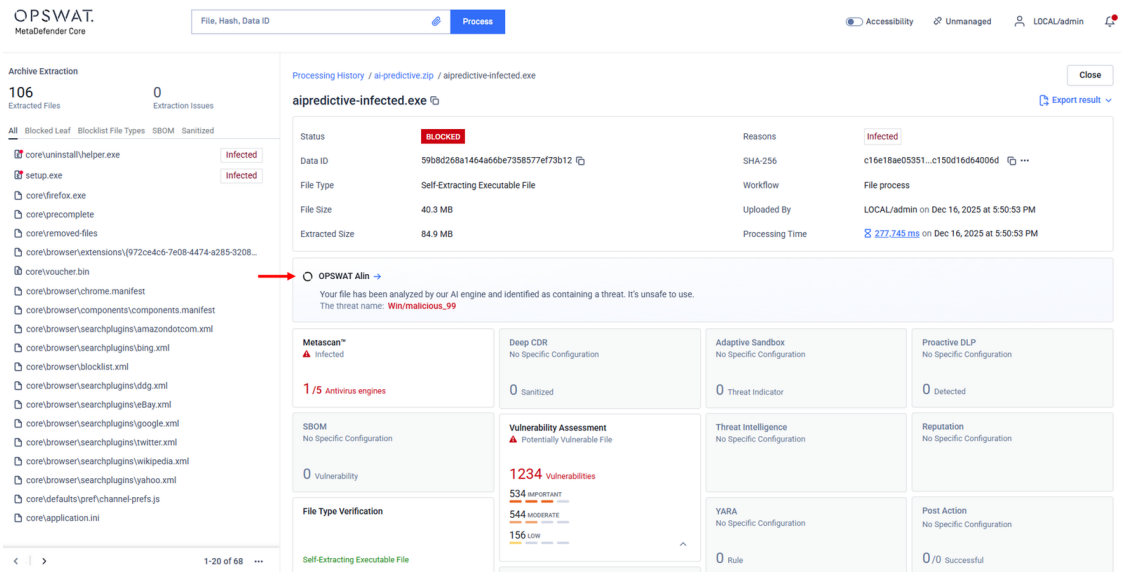
Version v5.17.0

Release Date: 16 December 2025

Brand New Technology: OPSWAT Predictive AIin AI

MetaDefender Core introduces a brand new technology called **OPSWAT Predictive AIin AI** that works alongside Metascan to improve threat detection depth and accuracy.

The new technology analyzes file structures, entropy patterns and semantic relationships using a domain-trained machine learning model to identify potentially harmful files earlier and with greater precision.



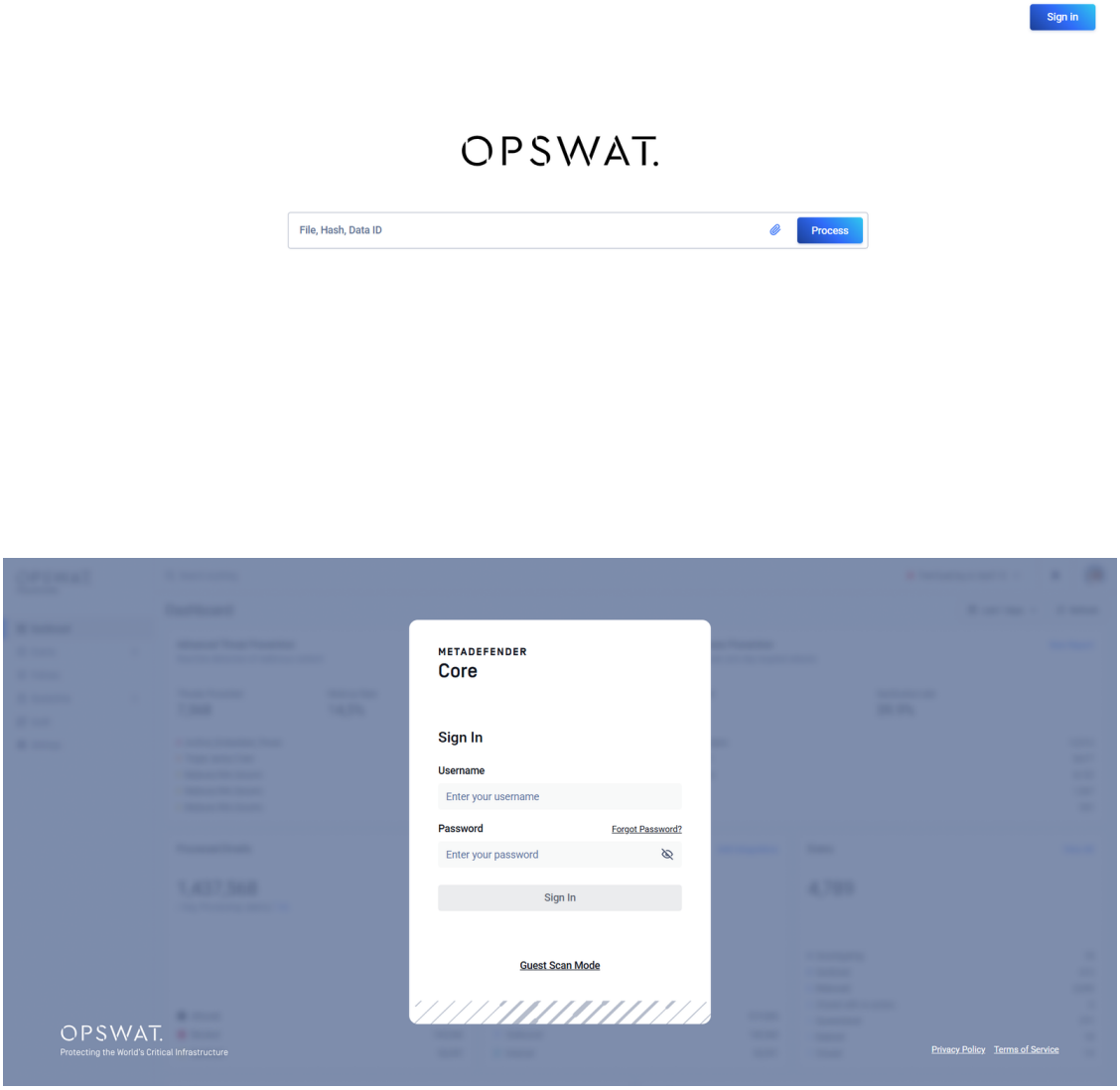
Revamped Processing Result Interface

As you can see above, our processing result interface has been redesigned to deliver clearer, more actionable insights, making it easier for administrators and support teams to interpret scan outcomes and take appropriate action.

We have also made it easier for users to know original file type and Deep CDR's converted destination type directly on the processing result screen.

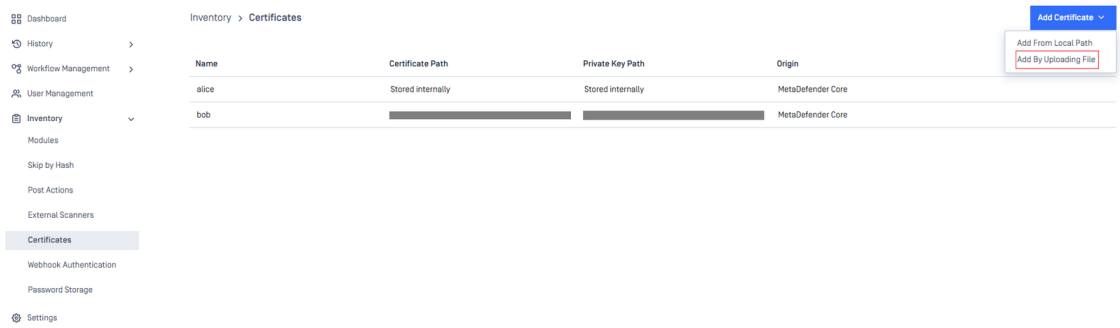
Unified User Interface and Experience of Login and Guest Scan Screens

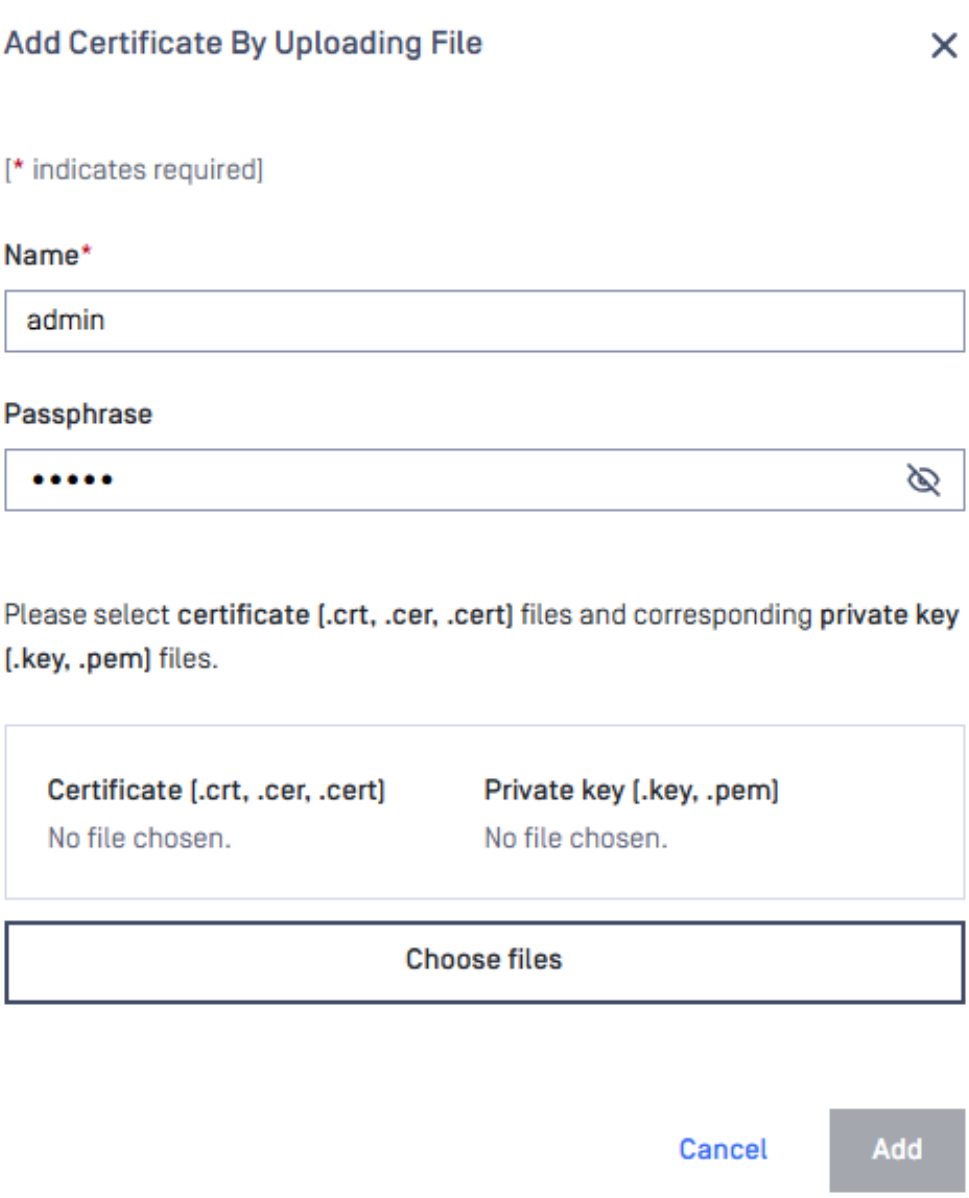
Login and Guest Scan screens have been unified across MetaDefender platform, bringing the similar feeling for customers of the platform.



Certificate Management Enhancements

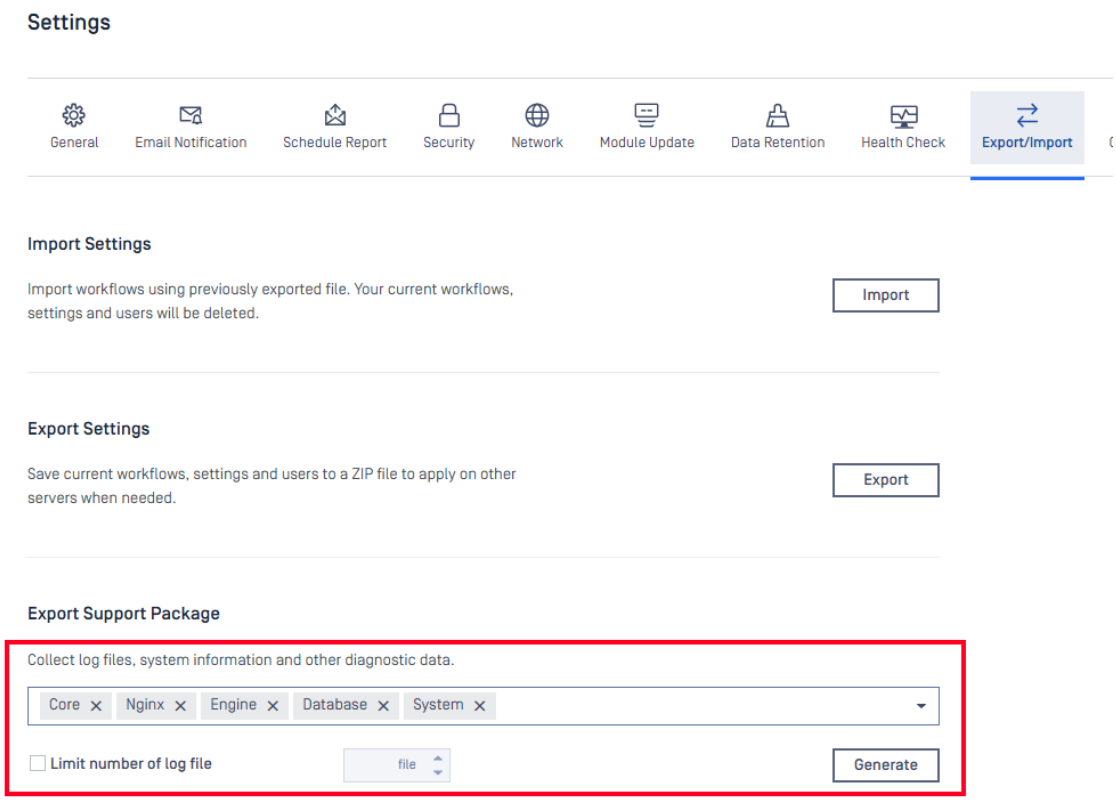
Certificate management now allows administrators to upload certificates directly by content, simplifying secure configuration and compliance. You can now add or update certificates by directly uploading certificate content.





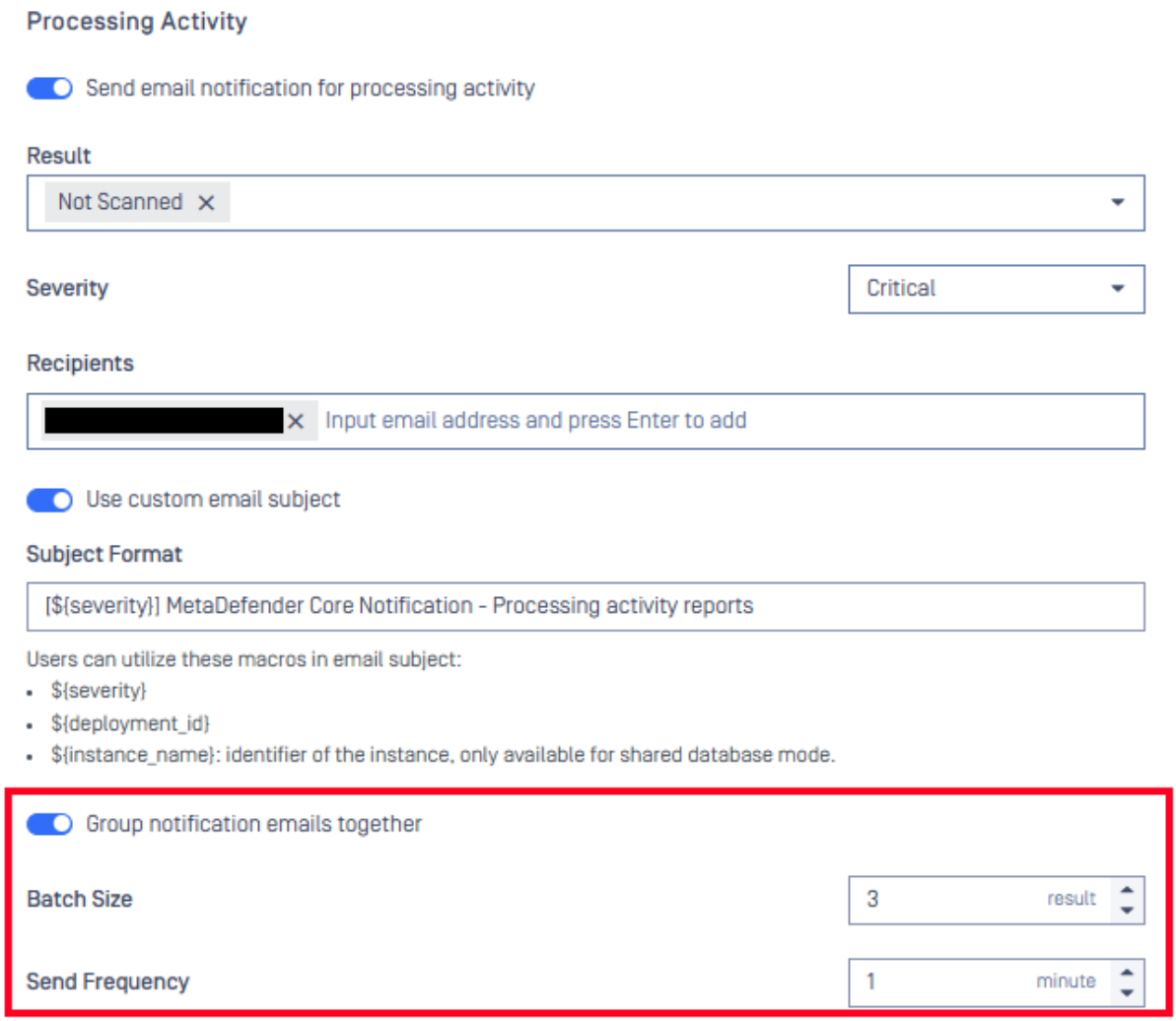
Support Package Enhancements

Support packages now include additional log types and process IDs, enabling users to select specific logs and set a maximum log size prior to generating.



Email Notification Enhancement

Email notifications have been improved to support batch aggregation and clearer messaging. This improves reporting efficiency for large-scale processing.



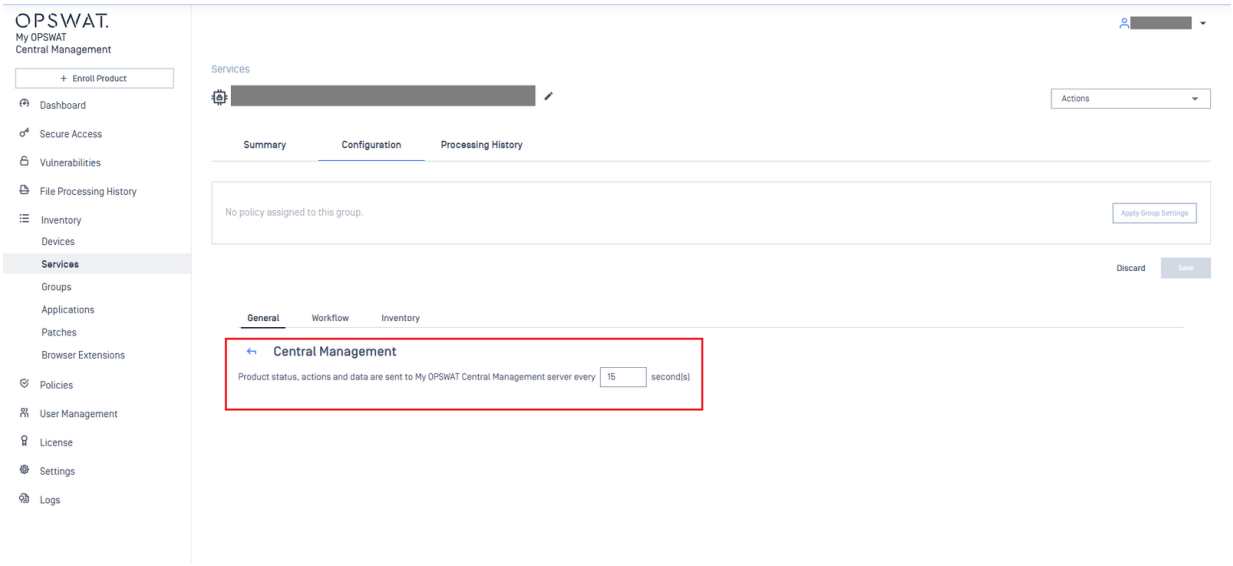
Support for Split Archive Files

MetaDefender Core now supports comprehensive scanning Split Archive file type.

The product accepts individual requests with split-archive parts sent via a new set of API endpoints, reassemble them, then extract and process nested files. Integration instruction at Split archive upload guideline.

Further Enhancements

1) A new synchronization interval setting for My OPSWAT Central Management integration provides greater control over data sync timing.



2) When you create a new workflow rule with Deep CDR enabled, all file types will be activated by default. Note: HTML and TXT, which are reserved for email-specific use cases, will stay unselected.

3) The product now supports teamed NIC (Network Interface Card) configurations, preserving Deployment ID and maintaining stable license activation when network adapters are changed or switched within the team.

4) The response of POST - Login now will return the header X-Core-Id if custom header is configured. It will help the customers identify which instance behind load balancer users login to.

5) Introduce new metrics regarding disk IO percentage in GET /stat/nodes API, providing better visibility into node performance and potential bottlenecks. Adding `disk_stat_info` :

- Disk utilization (%): Min-Max value from 0 to 100
- Read speed (Mb/s): Min 0, Max depend on disk
- Write speed (Mb/s): Min 0, Max depend on disk

6) Filtering and exporting processing history is now enhanced with file size and extraction issue filters, allowing users to quickly locate large or small files processed by the system and troubleshoot archive processing events.

7) Roll out a major enhancement for HotSwap-type MetaDefender Core deployments. Now, both local and My OPSWAT Central Management environments handle backup licenses with accurate engine lists, clear status reporting, and smooth configuration management.

Security Enhancements

1) Upgraded libraries for vulnerability fixes:

- Qt v6.8.1
- PostgreSQL v14.20

2) Implements rate limiting on database connection test and save endpoints to mitigate brute-force and abuse risks.

3) Add more length validations for text components in Workflow/Global Configuration to strengthen security and consistency.

4) Update the PDF export process to ensure hyperlinks are rendered as plain text, preventing accidental clicks.

5) Tightening up security for exporting Sandbox JSON and SBOM reports. Previously, it was possible to export these reports without providing a valid and authorized API key. Now, the system strictly validates user permissions and denies unauthorized access.

6) Enhance the validation on SSL certificates for HTTPS connections by rejecting mismatched cert & key, incorrect format, encrypted private key (if no passphrase provided).

Bug Fixes

- Fixed an issue where value of others_time may be negative when processing encrypted document files.
- Applied consistent name validation rules when updating workflows.
- Resolved issues with resetting global configuration for Adaptive Sandbox.
- Corrected a permissions issue where users without full module rights could modify engine configurations.
- Fixed an issue where AV engines would reappear in scan results after Metascan SDK deletion.
- Fixed an issue where AV engines of Metascan SDK could not be fully deleted and cause orphaned subprocesses after removing its database.
- Fixed an issue preventing configuration changes or AV removal when both Metascan SDK and traditional AVs were installed.
- Fixed an issue where files were retained if a download was aborted.
- Improved the accuracy of warning messages when users input invalid API keys.
- Fixed an issue where files in quarantine could not be scanned by Adaptive Sandbox, caused by missing data in the quarantine process.
- Fixed an issue where the password reset email would truncate passwords containing the '<' character.
- Other cosmetic fixes.

Version v5.16.1

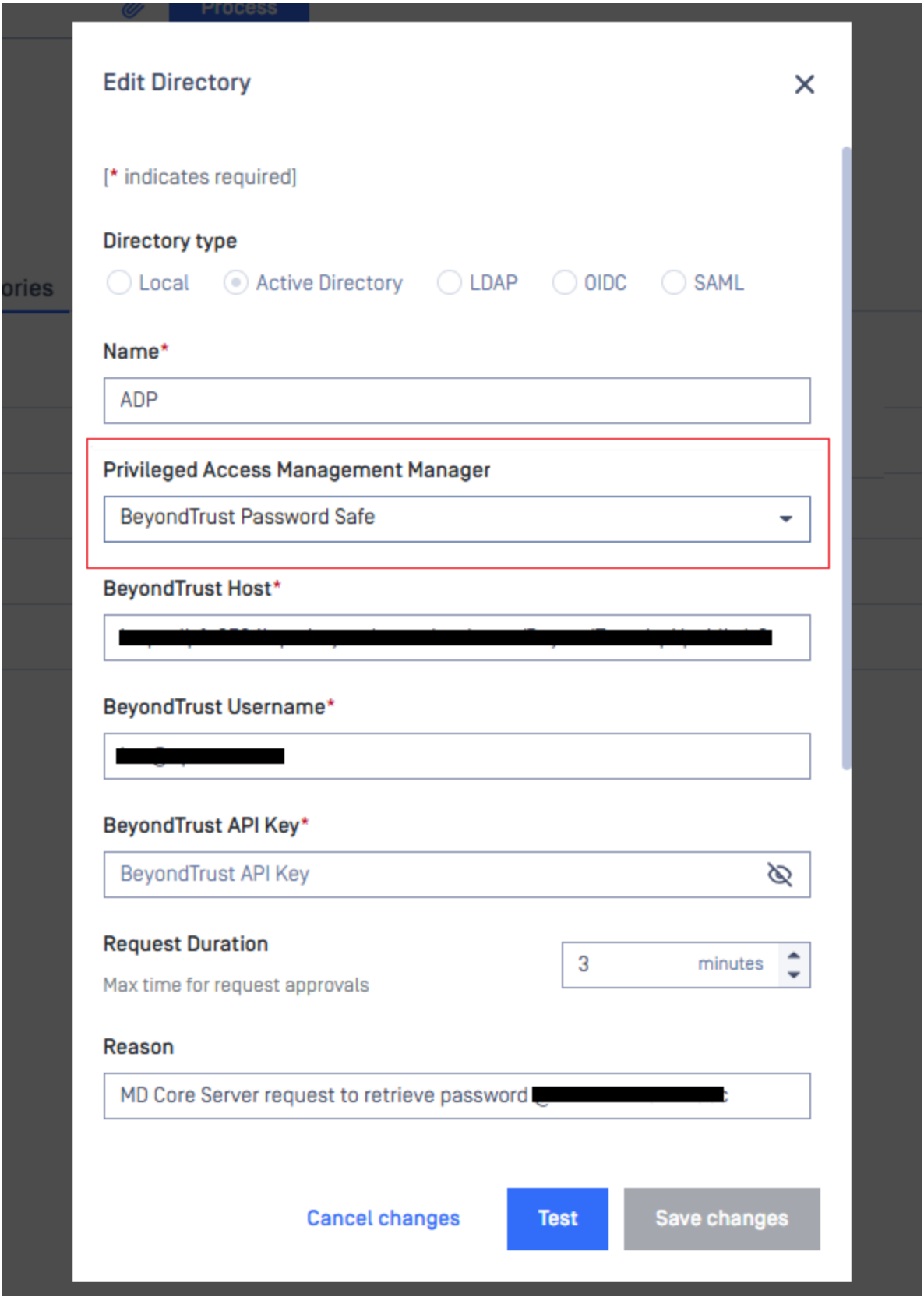
Release Date: 30 October 2025

Integration with BeyondTrust Password Safe for AD Directory

MetaDefender Core now supports integration with BeyondTrust's Password Safe, enabling secure, automated AD password retrieval of rotated Active Directory (AD) service account passwords.

This integration allows organizations to eliminate static AD bind passwords, and it supports configuration, status monitoring, and seamless login with AD users whose credentials are managed by Password Safe.

More details at Integrate Active Directory with BeyondTrust Password Safe.



PDF Report Restructuring & Threat Indicator Integration

Delivering a major upgrade to MetaDefender Core's PDF reporting. The processing result PDF report has been completely restructured for clarity and consistency, featuring a new and improved layout. Threat Indicators and IOCs from Adaptive Sandbox analysis are now included directly in the report for deeper visibility into potential threats.



OPSWAT.
MetaDefender Core

PROCESSING RESULT REPORT

File Details

Display Name	[redacted]
Data ID	cbf0932d5d6e494d9c0e967a57c020ad
Size	65150 bytes
Type	JavaScript Object Notation
Type ID	JSON
MIME Type	application/json
Receive Data Timestamp	October 23, 2025 at 06:35:07 PM
Upload Time	11 ms
MD5	3bd9d86d890772a0984ec0c7ae890e59
SHA1	a0e338b43fcbea65fbf7eb0beff6e85694b4e71d
SHA256	5af6b68517a1a9b8f670f103f2679be1d595e1ab0ca8e94d24a4b1bc9156a559
SHA512	-

Process Result

Result	BLOCKED
Verdict	Vulnerable Verdict by SBOM
Blocked Reasons	Vulnerable Verdict by SBOM | License Risk Verdict by SBOM
Processing Time	20301 ms
Workflow Rule	File process
Hash Time	1

Metascan

Total AV Engines	2
Start Time	October 23, 2025 at 06:35:08 PM
Scan Result	No Threat Detected

Adaptive Sandbox

Verdict	No Threat Detected
Result	Adaptive Sandbox Allowed
Threat Indicators	1

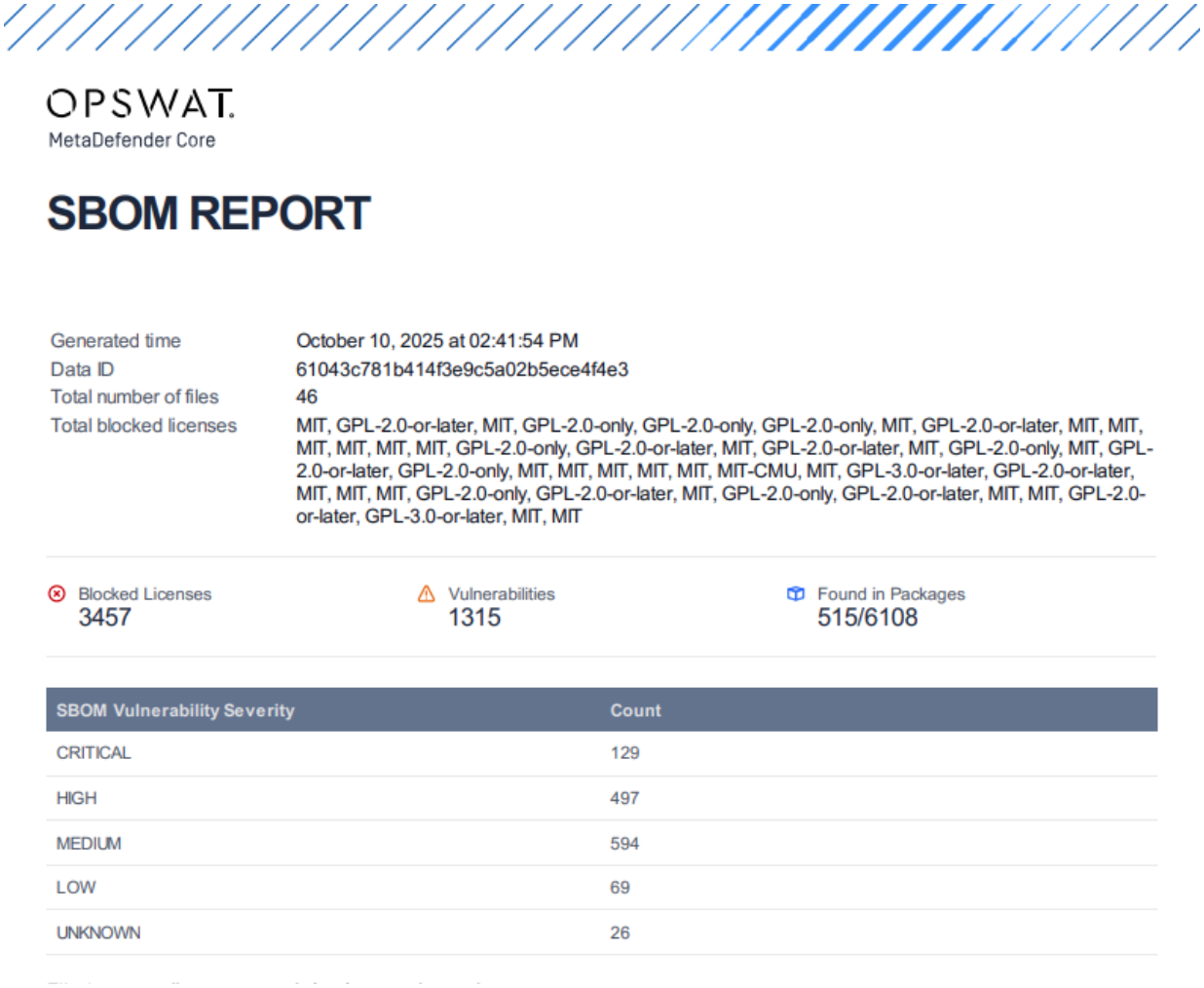
Threat Indicators	Verdict
OSINT source detected benign resource(s)	No Threat Found

IOC - Domain	Verdict
[redacted]org	Unknown
[redacted]org	Unknown
[redacted]org	Unknown
[redacted]org	Unknown

Enhanced SBOM PDF Report for Better Security Insights

The SBOM PDF report has been enhanced to display vulnerability severity statistics, highlight blocked licenses, and

list problematic files and libraries at the top. These improvements help users quickly identify and address the security and compliance issues in their software bill of materials.



OPSWAT.
MetaDefender Core

SBOM REPORT

Generated time	October 10, 2025 at 02:41:54 PM
Data ID	61043c781b414f3e9c5a02b5ece4f4e3
Total number of files	46
Total blocked licenses	MIT, GPL-2.0-or-later, MIT, GPL-2.0-only, GPL-2.0-only, GPL-2.0-only, MIT, GPL-2.0-or-later, MIT, MIT, MIT, MIT, MIT, MIT, GPL-2.0-only, GPL-2.0-or-later, MIT, GPL-2.0-or-later, MIT, GPL-2.0-only, MIT, GPL-2.0-or-later, GPL-2.0-only, MIT, MIT, MIT, MIT, MIT, MIT-CMU, MIT, GPL-3.0-or-later, GPL-2.0-or-later, MIT, MIT, MIT, GPL-2.0-only, GPL-2.0-or-later, MIT, GPL-2.0-only, GPL-2.0-or-later, MIT, MIT, GPL-2.0-or-later, GPL-3.0-or-later, MIT, MIT

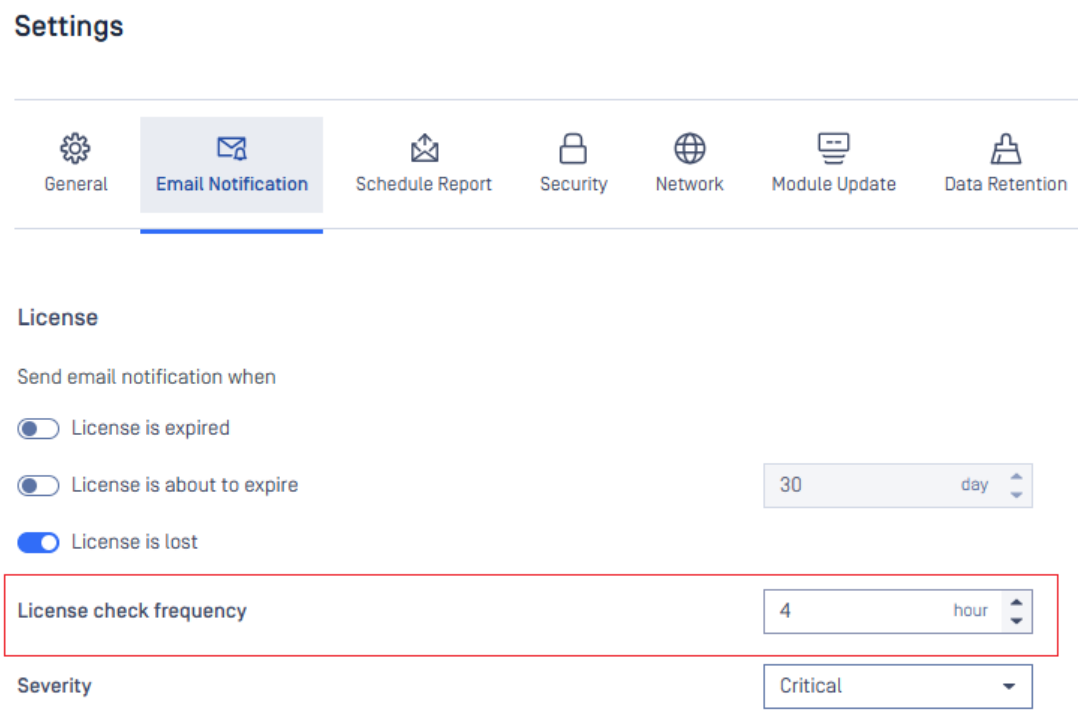
Blocked Licenses 3457 | Vulnerabilities 1315 | Found in Packages 515/6108

SBOM Vulnerability Severity	Count
CRITICAL	129
HIGH	497
MEDIUM	594
LOW	69
UNKNOWN	26

Customizable Frequency for License Expiry Email Notifications

Users can configure how often they receive email notifications about license expiration and related events, helping to reduce alert fatigue and avoid multiple reminders in a single day.

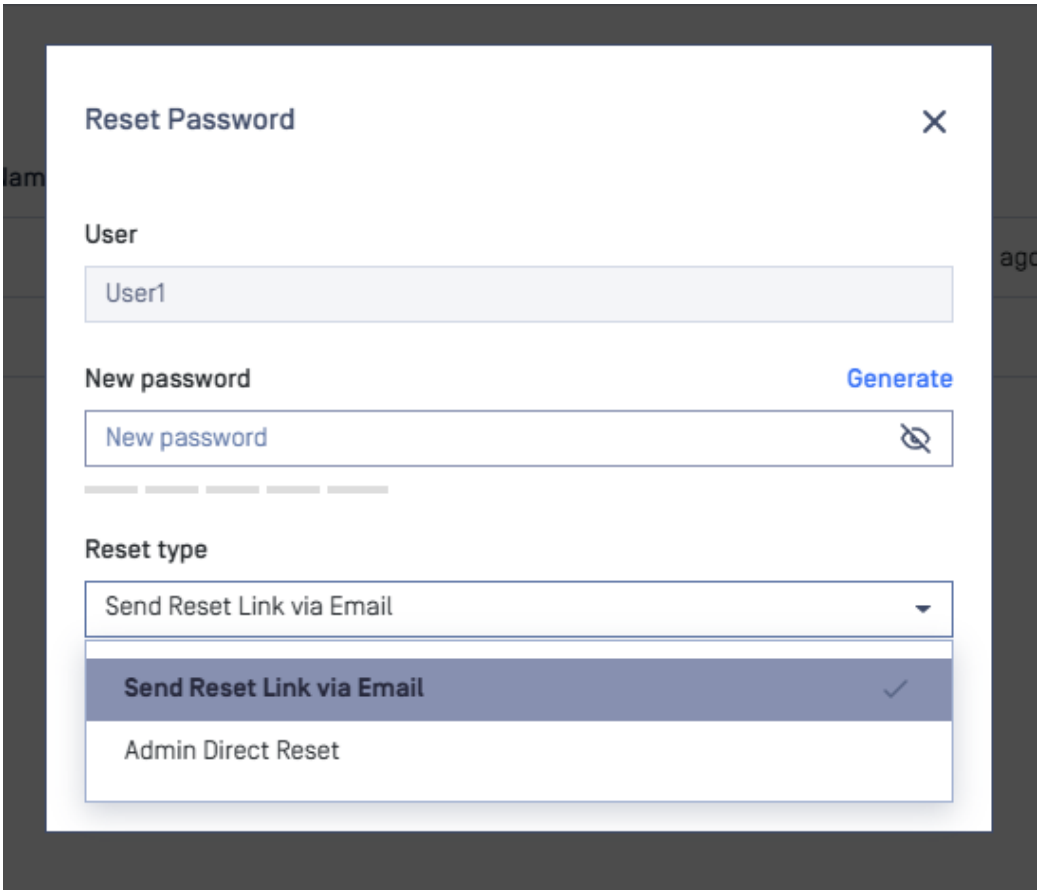
A new setting allows administrators to set the notification frequency from 1 to 48 hours, with a default of every 4 hours.

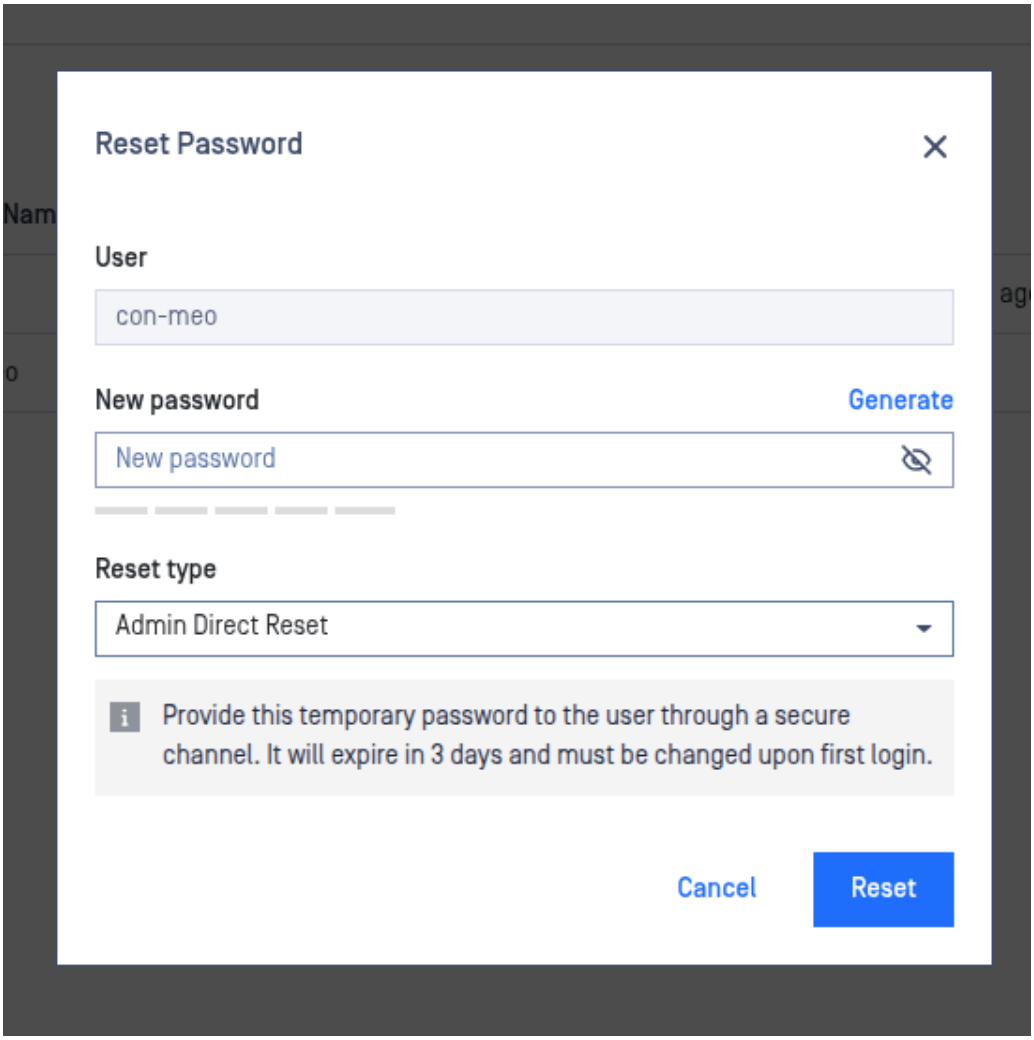


Resetting User Passwords Without SMTP Configuration

Admins can reset user passwords even when no SMTP or email server is configured, addressing a key need for secure environments where email infrastructure is restricted.

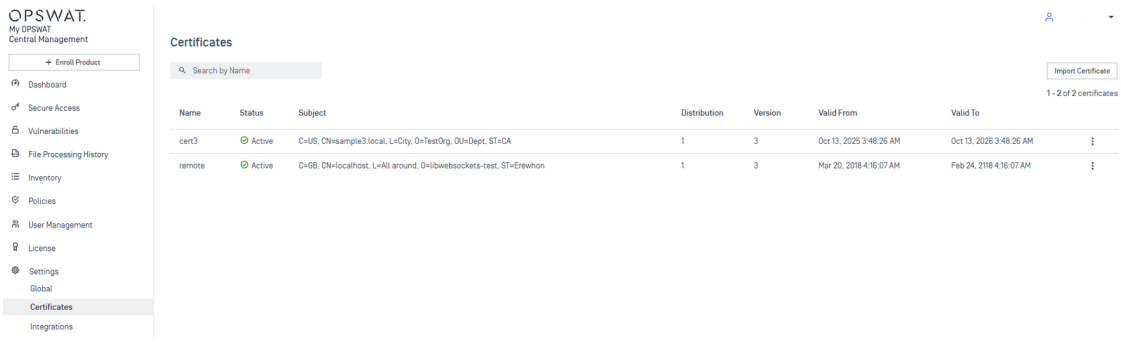
With a new option, MetaDefender Core allows administrators to generate a temporary password and securely share it with users through alternative channels. The feature clearly instructs admins on next steps, and the temporary password will expire in 3 days, requiring a change upon first login.


Reset Password
User
User1
New password
Generate
New password
Reset type
Send Reset Link via Email
Send Reset Link via Email
Admin Direct Reset

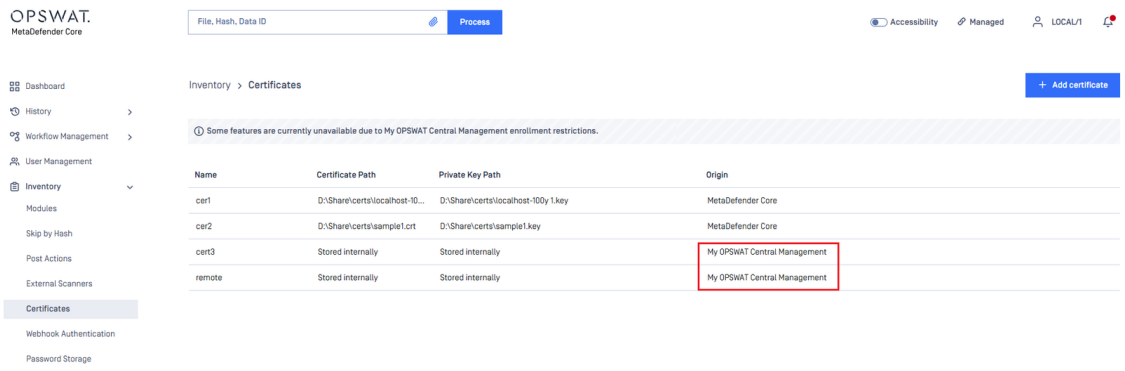


My OPSWAT Central Management Integration: Certificate Management

MetaDefender Core is able to automatically synchronize certificates with My OPSWAT Central Management (MOCM), supporting both Standalone and Shared DB modes, and prevents accidental deletion or modification of synchronized certificates.



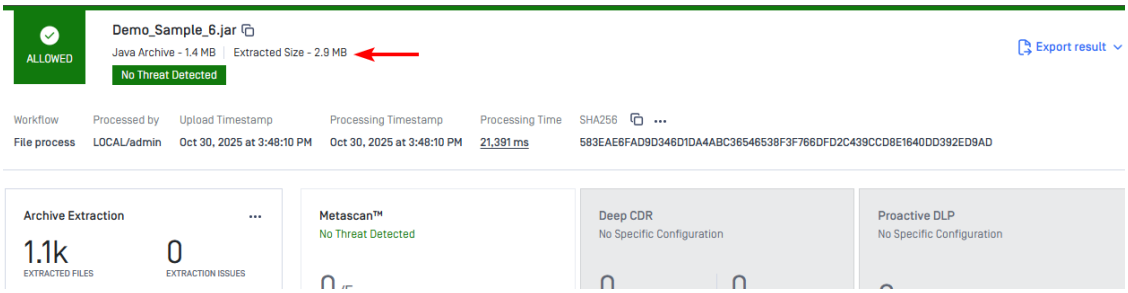
Admins can view the origin of each certificate, and both locally added and MOCM-synced certificates are fully supported in workflows and secure connection settings.



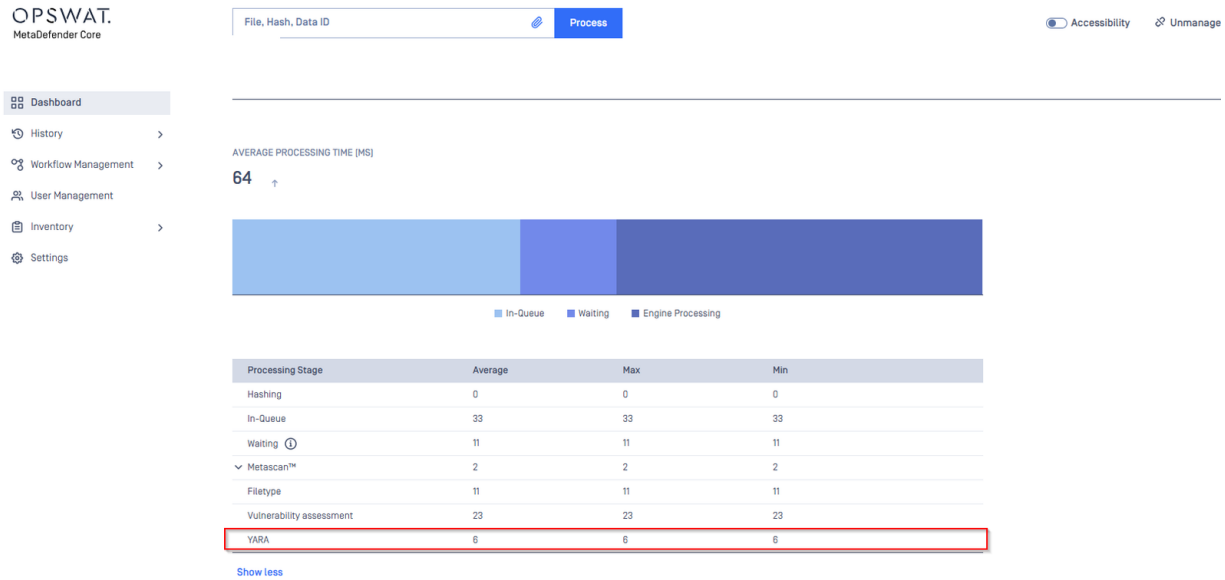
Further Enhancements

1) "InSights Threat Intelligence" has been renamed to "Threat Intelligence" across the MetaDefender Core platform. This name change applies to the following screens Workflow, Health Check, Modules, Processing Result, Export result, and Executive Report.

2) Display Total Extracted File Size for Archives: Begin to track and display the total extracted file size for archives, giving users a clear view of how much data is unpacked during file analysis.



3) YARA Processing Time is Visible in Executive Reports: The Executive Report and PDF export now display YARA processing time, providing transparency into how long YARA scans take during file analysis.



4) Real-Time Scan Queue Updates via readyz API: The readyz API provides real-time updates on the scan queue data, improving visibility for high-throughput environments like MetaDefender ICAP Server. Previously, queue size

information was refreshed every 5 seconds, which might be inaccurate under heavy load.

5) Parallel deployment of scanner packages to Metascan SDK has been improved, helping reduce deployment times.

6) Support proxy server entries with or without the `http://` or `https://` prefix. Admins can enter proxy addresses as plain hostnames, IP addresses, or with either HTTP or HTTPS schemes, and all formats will be accepted and handled correctly by the backend.

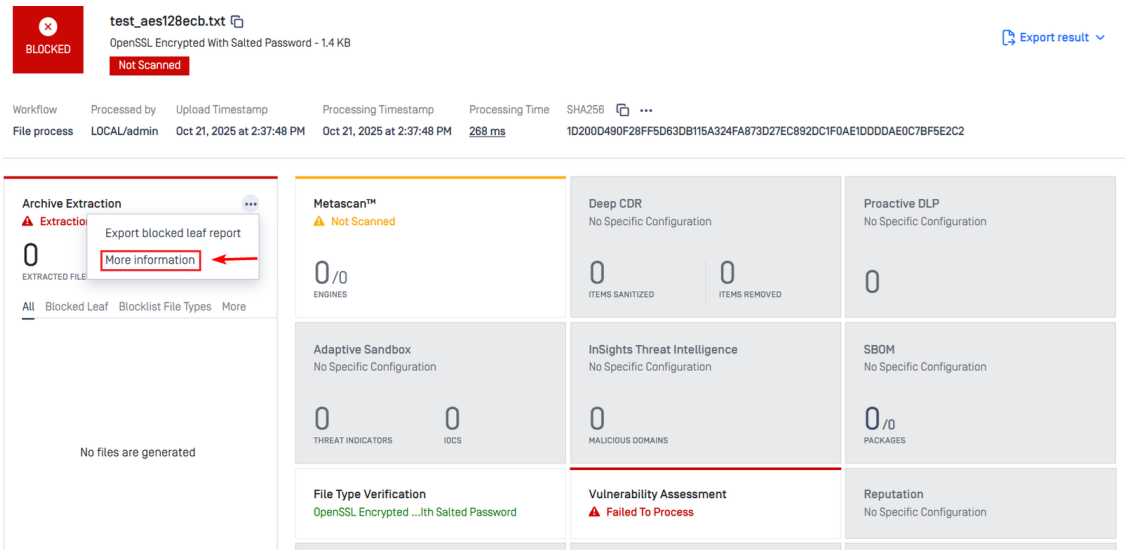
7) Support displaying multiple items from an object list in a single row under Workflow Rule, ensuring that up to three fields can be neatly aligned on one line. Prepared for future enhancements of engines.

8) Scan processing continues even if the client-identity JSON cannot be parsed, addressing rare but disruptive edge cases. Instead of halting the scan and returning an error, the system logs a warning and proceeds. The log level is also downgraded from ERROR to WARNING, making it clear that the issue is non-critical.

Log

```
[WARNING] Failed to parse client-identity's JSON, dataId='aa4ceafda8e242fca99eea4971472251',
..., jsonError.errorString()='...' [msgid: 7451]
```

9) Improved Error Visibility for CLI Extraction: displays extraction errors for files processed with the CLI Extraction feature. Previously, these error messages were hidden or hard to find, but now they are classified and surfaced as "General Unextractable Errors" when appropriate.



Security Enhancements

Upgraded libraries for vulnerability fixes:

- Protobuf v32.1
- OpenSSL v3.5.4

Permissions have been tightened so only users with the appropriate right can access and view the data of "Config History".

Adjusted Webhook worker count boundary:

- Setting the webhook worker count to a very high value would prevent MetaDefender Core from starting, causing it to hang.
- The worker count is now limited to a maximum of 100, and any higher values from previous versions are automatically adjusted during upgrade to ensure stable operation.

Logging Enhancements

Milliseconds are now included in CEF log timestamps, with a setting to maintain backward compatibility.

CEF Log

```
Sep 04 15:56:56.123 SE Asia Standard Time ... CEF:0|OPSWAT|MSCW|5.16.1|core.config|Request
change logger configs|3|OMSrequest=... OMSmsgid=6274
```

Bug Fixes

- Fixed an issue where error messages weren't shown correctly after a password reset if users entered the wrong password on the login page.
- Fixed an issue that allowed users to save configuration changes in Deep CDR file type handling modal even when required fields were left empty.
- Fixed an issue where users couldn't save workflow configuration changes after removing a row in Deep CDR file type handling section.
- Fixed an issue where HTML in workflow names could be injected into certificate expiration notification emails, affecting how the email was displayed.
- Fixed an issue where MetaDefender Core would crash when changing the database password on Windows and restarting itself, resulting in a crash dump and missing logs END.
- Fixed an issue where the blocked-leaves APIs did not consistently include YARA rule information in their JSON responses when a file was blocked due to a matched YARA rule.
- Fixed an issue where users could not navigate to the last page in processing history when filtering by "Object" request type, due to a database query error.
- Fixed an issue where audit log data retention was not working properly, causing old audit log entries to be removed much more slowly than expected.
- Fixed an issue where upgrading MetaDefender Core on Ubuntu 24 could cause the service to lose connection with Postgres due to a specific system warning.
- Fixed an issue where audit logs and syslog entries only displayed user ID instead of username when a local user was deleted.

Version v5.16.0

Release Date: 30 September 2025

Support for CIS Level 2

MetaDefender Core now aligns with CIS Level 2 benchmarks for Windows Server 2022. This enhancement supports

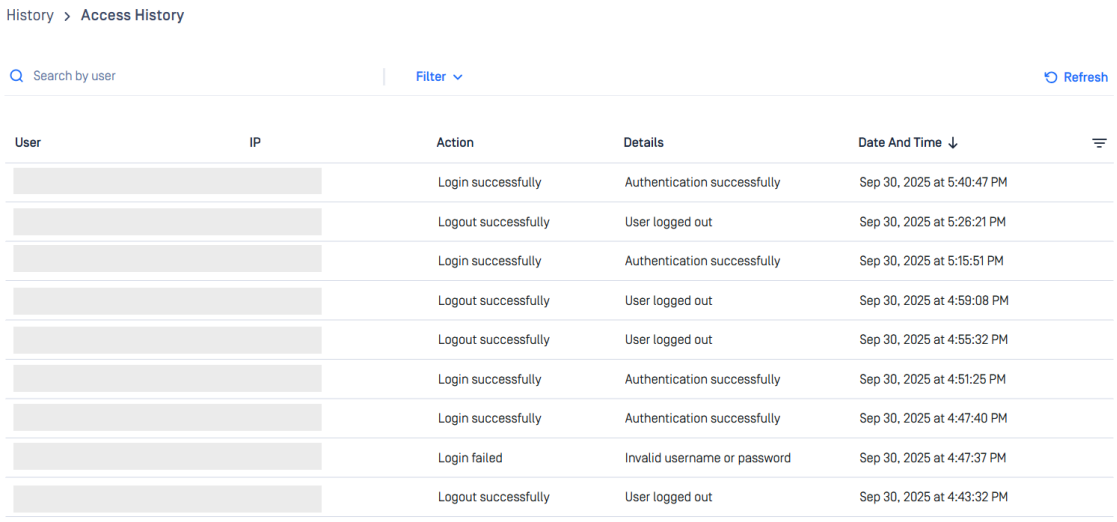
organizations that require stricter security controls.

Access History Screen, tracking Session Activities

A dedicated interface under the History menu displays a comprehensive log of user session activities, including:

- Sign-in and sign-out events with timestamps and IP addresses.
- Session timeout occurrences.
- Password change events.
- Role and permission modifications, with details of the initiating user.

Administrators can filter logs by date range, action type, IP or user to streamline investigations and reporting.

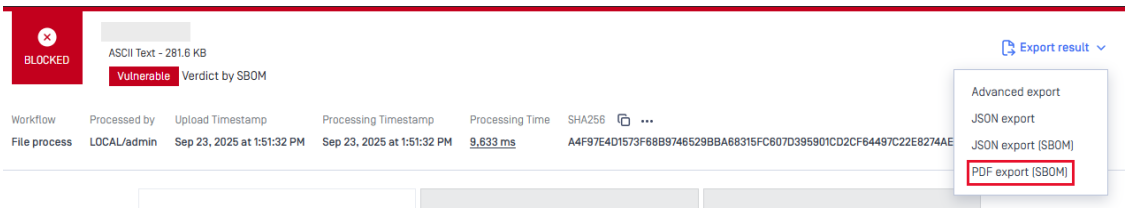


New Exportable SBOM Report

This version introduces a new PDF report specifically for Software Bill of Materials (SBOM) processing results. This dedicated report provides detailed library information including:

- Library metadata and versions
- Associated CVE (Common Vulnerabilities and Exposures) information
- License information and compliance details
- Component dependency relationships

This new report type enables users to receive complete SBOM analysis results in a shareable PDF format, addressing enterprise workflows where detailed library information should be distributed and verified for security assessment, vulnerability remediation, and compliance review.





OPSWAT.
MetaDefender Core

SBOM REPORT

Generated time	September 29, 2025 at 12:35:18 PM
Data ID	dcd96bf1db224a9faa041ab89aca1a06
Total number of files	1

Blocked Licenses 0 | Vulnerabilities 32 | Found in Packages 26/212

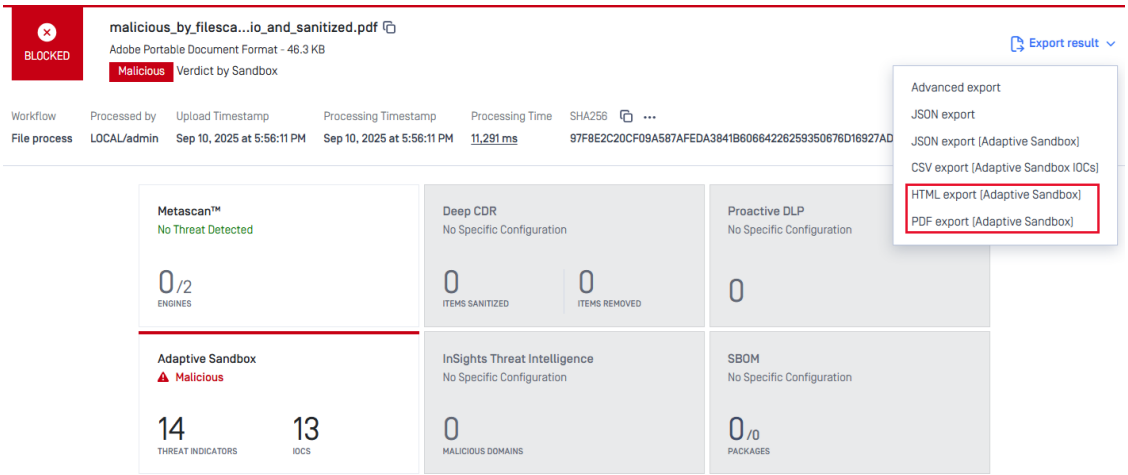
File 1 package-lock.json

Vulnerable Verdict by SBOM

Library Name	Version	Vulnerability	License
mongoose	5.9.18	CVE-2023-3696 CRITICAL Fix version: 7.3.3, 6.11.3, 5.13.20	MIT
mongoose	5.9.18	CVE-2022-2564 CRITICAL Fix version: 6.4.6, 5.13.15	MIT

Full Report of Remote Adaptive Sandbox in HTML and PDF Formats

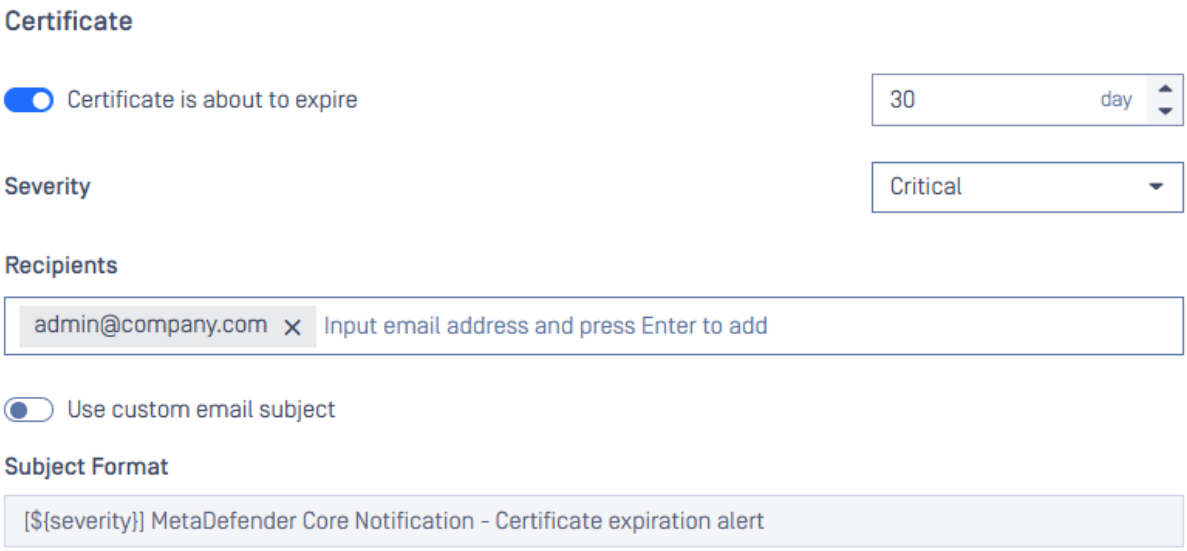
Previously, users had to retrieve the full report from Adaptive Sandbox interface. Now, the life is easier when users can directly download complete reports in PDF or HTML from MetaDefender Core web console with a single click.



SSL Certificate Expiration Notification

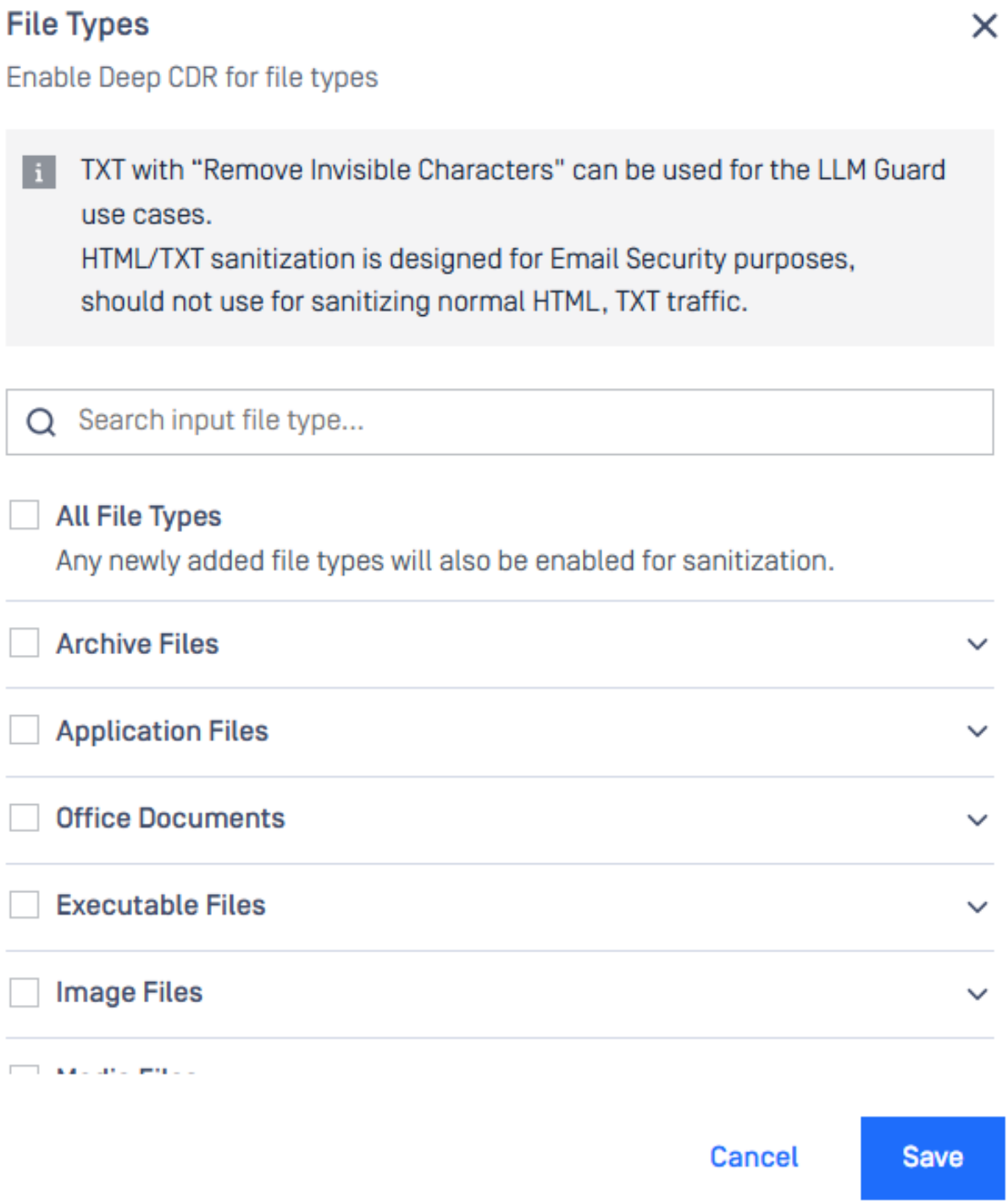
Administrators can now configure alerts for SSL certificates that are approaching expiration. The notification window is customizable. This helps prevent service disruptions and ensures secure communication remains uninterrupted.

These options can be found under Email Notification tab in Settings.



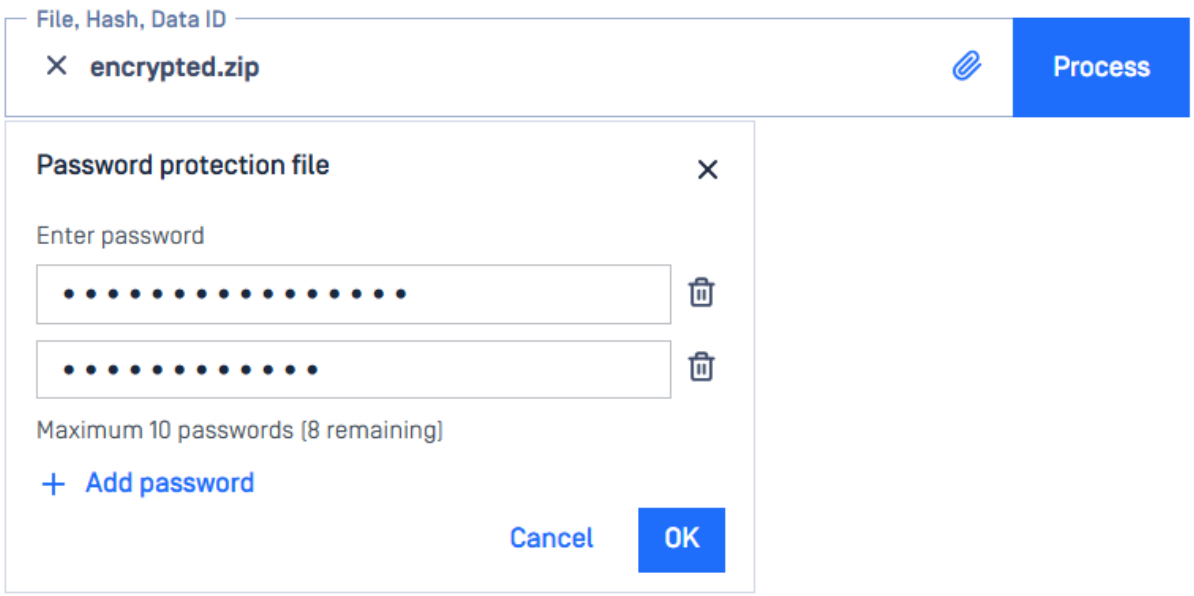
Better Experience for Deep CDR File Type Selection

A new search textbox enables users to quickly locate specific file types, such as JPG, by typing keywords and Enter. Info section and option to select all `All File Types` are also arranged.



Multi-Password Input Interface

Users can now specify a list of passwords during file upload processes on Web Console, allowing the system to iteratively attempt decryption against each provided password.

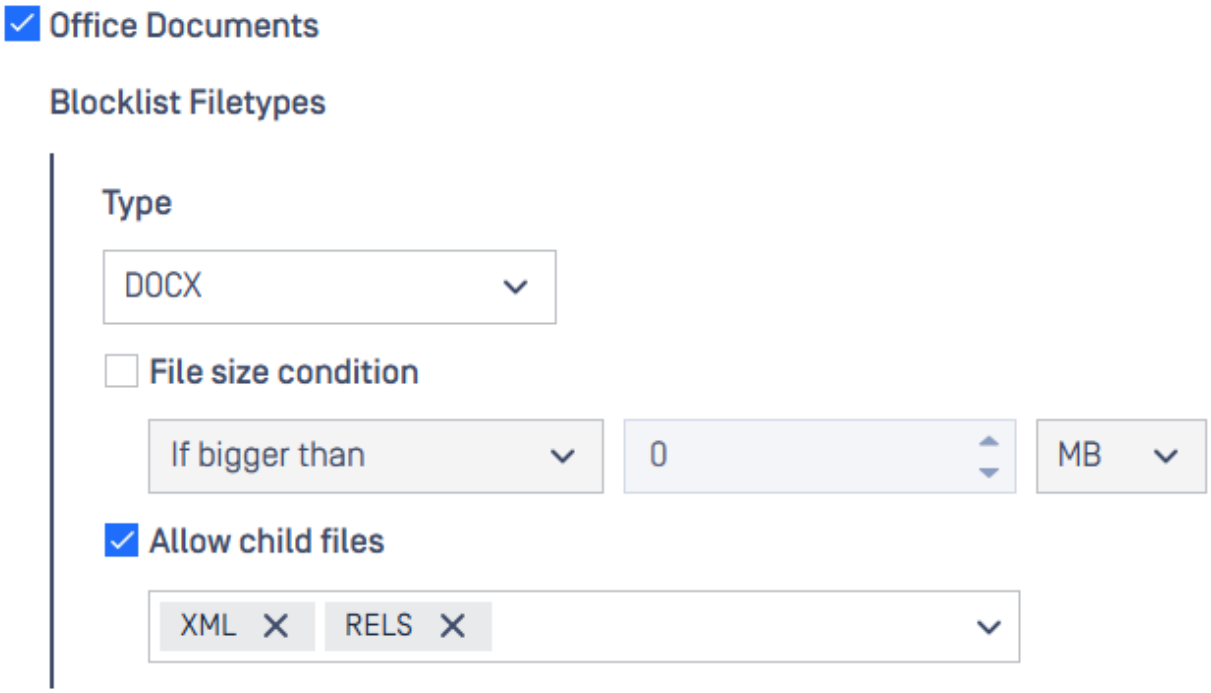


Blocklist All Except Child File Type, Extending Capability to more File Type Groups

The option `Allow child files` under `Blocklist All Except` feature allows users to define predefined child file types for further processing while instantly blocklisting others.

This update now, besides Archive Files, is covering more file type groups: Disk Image Files, Email Files, Encrypted Archive Files, Executable Files, and Office Documents.

For example, with the following settings, XML and RELS child files of DOCX file will be processed, meanwhile, other file types else will be blocklisted.



Logging Enhancements

1) When available disk space is low (< 10%), there will be a log message filed in core.log.

```
Code

[WARNING]... Disk is running out of space, drive=..., total_space_gb=..., used_space_gb=...,
folders=... [msgid: 7693]
```

2) Additional fields in msgid 82 for same-level extracted files and total extracted files.

- files_extracted_count: number of extracted files in the DIRECT level.
- total_extracted_files: total number of extracted files in ALL level.

```
Code

[INFO]... files_extracted_count='...', total_extracted_files='...',
totalProcessingTime='...' [msgid: 82]
```

REST API Enhancements

1) APIs GET - Fetch a list of blocked leaf files inside archive and Fetch a list of blocked leaf files inside archive by hash include

Hash-related fields:

- `md5`

- `sha1`
- `sha256`
- `sha512`
- `is_skip_hash`

And YARA rule fields: `yara_info`

2. Begin tracking Proactive DLP wait time: `dlp_wait_time` . This new field is included in Fetch Analysis Result and other result fetching APIs.

Further Enhancements

1) Improved workflow configuration handling to prevent save errors and validation issues while scanning engines are being updated.

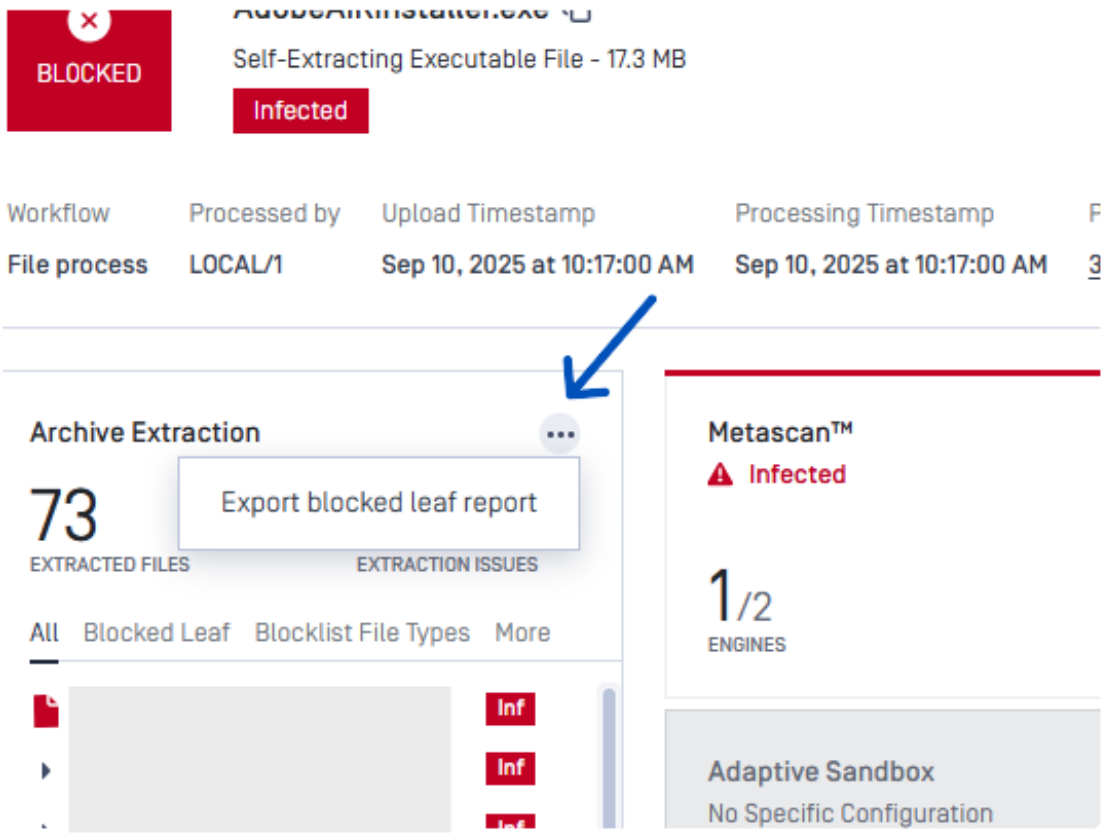
2) Quarantine feature now hides or grays out the option `Check with` when the respective module is inactive or unlicensed.

3) Quarantine feature offers options for searching for Technology results like SBOM, Adaptive Sandbox and sorting by Date Added.

4) Administrators may enable SSL connection for bundle PostgreSQL. More details at How to setup SSL connection for PG.

5) Session cookie sent or not sent with cross-site requests is configurable on web console under Security tab in Settings.

6) Ability to export results of all single files (non-archive) files within an archive file as JSON.



Security Enhancements

Upgraded libraries for vulnerability fixes:

- Libxml2 v2.14.6
- OpenSSL v3.5.2
- ng2-file-upload, ngx-bootstrap, ngx-infinite-scroll

YARA rules using HTTP are now subject to the following restrictions:

- Archive size limit: Maximum of 50 MB.
- File count limit: Maximum of 1,000 files.
- Folder depth limit: Maximum of 50 nested levels.
- Path length limit: Maximum of 255 characters, excluding the engine database folder path length.
- Extracted size limit: Maximum of 50 MB total after extraction.

Bug Fixes

- Addressed an issue that negated `data` type in the option `Allow child files` of Block All Except in Blocklist.
- Addressed a rare crash issue in the product that occurred after enrolled with MOCM.
- Addressed an issue that did not set correct customized IP and port to Desktop shortcut.
- Addressed an issue that caused a failure to Email Server authentication mode `None`.

- Addressed a UI issue that displayed an incorrect number of supported file types of Adaptive Sandbox in workflow configuration.
- Addressed an issue that incorrectly prohibited certain accepted special characters (e.g., underscores and hyphens) in usernames within the Email Server configuration settings.

Version v5.15.2

Release Date: 28 August 2025

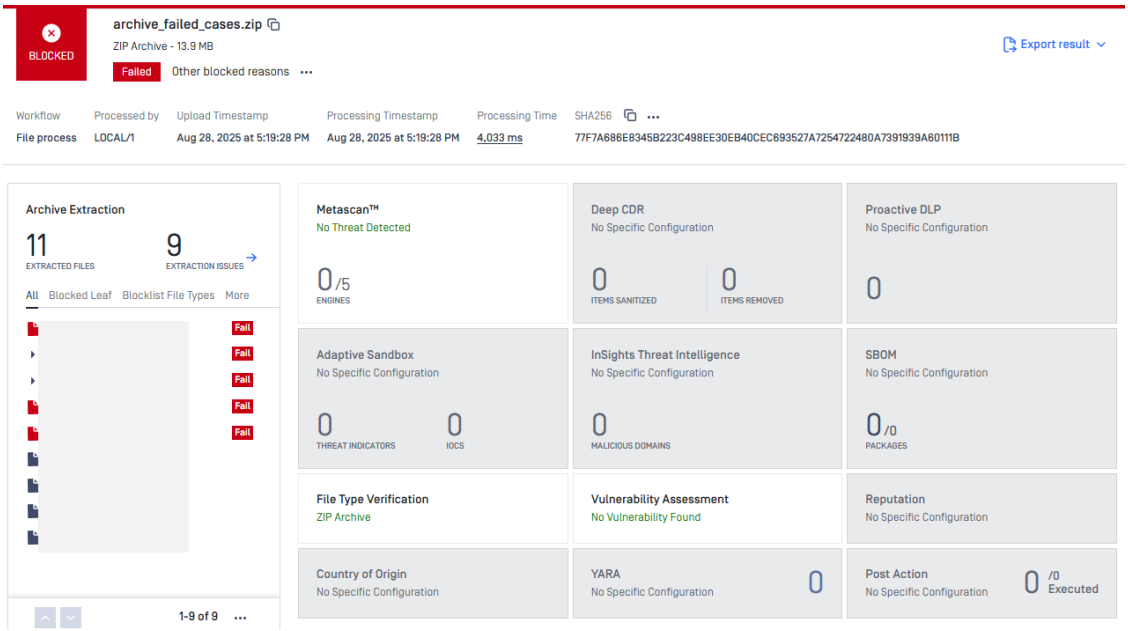
OAuth for Email Server Authentication for Microsoft 365

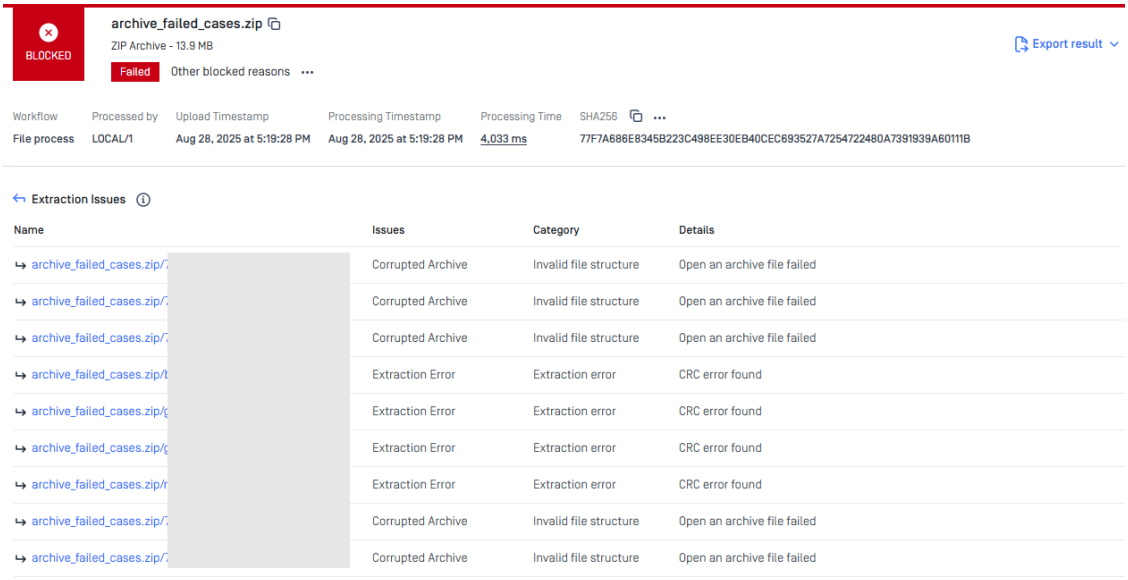
In March 2026, Microsoft will be retiring Basic authentication with SMTP AUTH, and the change will complete after April. That means applications will no longer be able to use this authentication method to send emails.

This change prompts us to prioritize implementing OAuth support for Email Server as soon as possible. If you are using Basic Auth, we encourage you to review this document Microsoft 365 OAuth and change your MetaDefender Core's configurations to OAuth.

Enhanced Extraction Issue Visibility for Nested Archives

Earlier, users might have struggled to determine the cause of extraction errors in nested archive files. This update introduces a additional interface that highlights any extraction issues, making it easier for users to identify problematic nested files quickly.

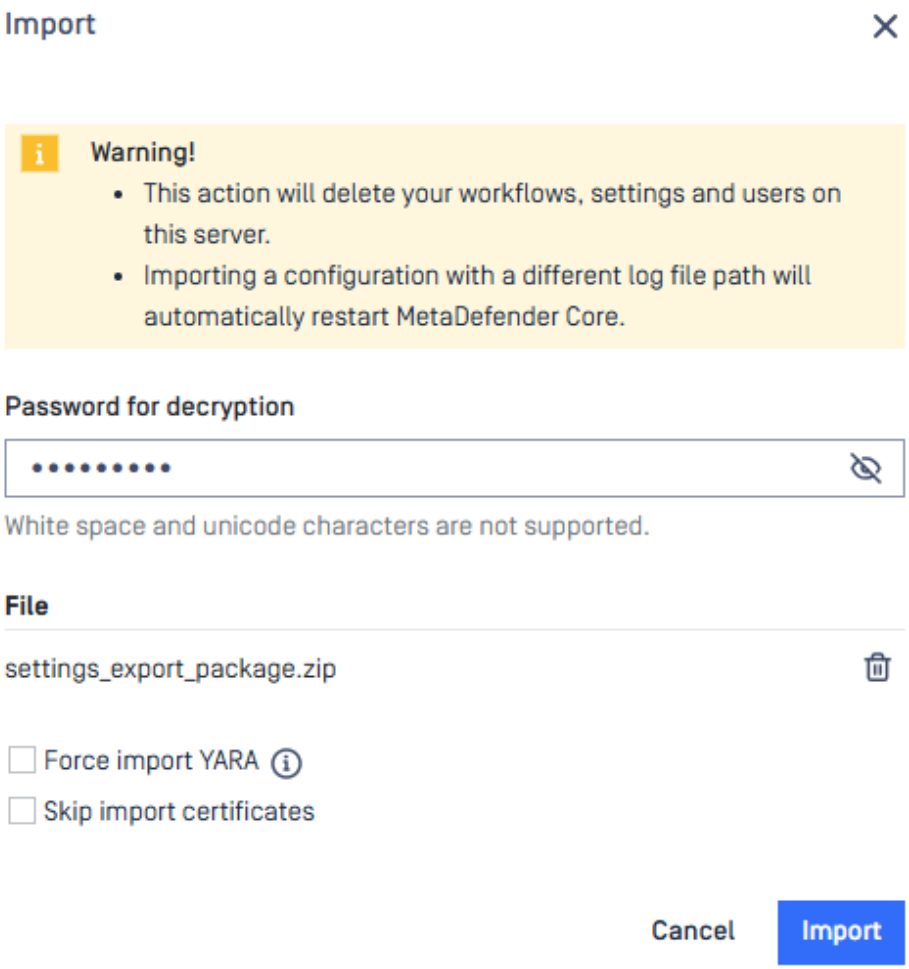




Configuration Import & Export Supports Certificates

The ability to export certificates with database mode simplifies the process of backing up certificates. Administrators no longer need to manually configure Certificates settings.

The export/import feature will export/import certificate data by default. Its compatibility checks, automatic routing based on database mode and automatic deletion of existing certificates before import ensure integration of certificates without conflicts and prevent duplication. When an import fails, the product restores the previous state.

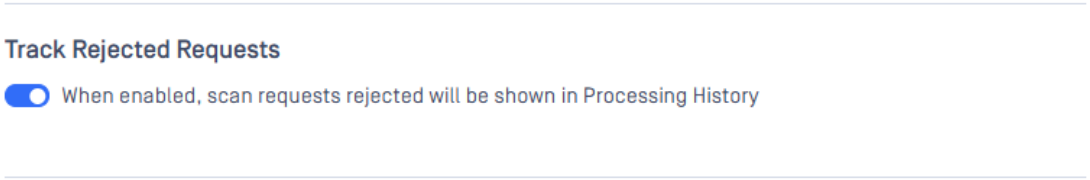


Rejected Scan Request Logging

Scan requests that are rejected such as those surpassing file size limits or occuring when local scanning mode is turned off, are not stored and displayed in the Processing History. A new optional setting now enables users to store these requests for later review and analysis.

Note: This feature only supports Standalone mode.

More details at Track Rejected Requests.



Non-TLS Communication for Local User Management Configuration in My OPSWAT Central Management

Added support for non-TLS communication with My OPSWAT Central Management (MOCM) for Local User Management configuration in specific on-premises deployments. This feature allows MOCM and MetaDefender Core to operate without TLS when both are deployed within the same isolated network.

- **Use Case**: This enhancement is designed for scenarios where MOCM and MetaDefender Core are running in a

secure, isolated network environment, and TLS is deemed unnecessary.

- **Recommendation**: While non-TLS communication is now supported, TLS is strongly recommended to ensure secure communication. Non-TLS should only be used in environments where security risks are minimal and well-understood.

Improved Stability for Group Configuration in My OPSWAT Central Management

In collaboration with My OPSWAT Central Management, MetaDefender Core begins to use dedicated endpoints and storage for configuration data and status, minimizing scenarios where a Core instance in a group fails to apply the group configuration. This should make life easier for system administrators managing large or complex groups.

REST API Enhancements

1) A dedicated API endpoint to fetch and monitor system's resources including Disk space usage, CPU and RAM utilizations. Document at GET - Get server system resource.

2) The feature Scan-from-Link can now skip its metadata check and proceed directly to file downloading when clients set the new header `skip-head-request` of POST - Analyze File (Asynchronous mode) to false.

Further Enhancements

1) Human-readable display for package URLs in engine processing results.

2) Filetype engine has been offering enhanced features and detection rates, which requires slightly more resources and processing time. Tracking the wait time at the Filetype engine will provide useful observability insights.

3) Setting private username for PostgreSQL can now have "-" (dash) character.

4) Slightly improved performance for scanning large archive files.

5) Retired the page Data Reporting from registration wizard screens since this feature was deprecated for years.

6) Enforced rate limiting on PIN entry attempts.

Security Enhancements

Upgraded libraries for vulnerability fixes:

- PostgreSQL v14.19
- NGINX v1.28.0
- 7zip v25.01

Bug Fixes

- Addressed an issue that poped up `No modification detected` message while assigning new workflow template to existing workflow rules.
- Addressed an issue that intermittently caused `Not Scanned` to file scannings with an error `Can't convert to engine message`. This issue was observed to occur only when two actions, posting a file and performing a local

scan for a file with the same hash, are executed.

- Addressed an edge case issue where the product failed to retrieve the workflow configuration for the Filetype engine, resulting in an error `Missing preferences.filetype_mismatch`.
- Addressed an issue that prevented editing of the Country of Origin's advanced settings while the engine was disabled or inactive.
- Addressed an issue where a negative value for `others_time` could occur after Compression sanitization execution.
- Addressed a comestic issue where Metascan tile was temporarily grayed out while Country of Origin was processing files.

Version v5.15.1

Release Date: 30 July 2025

New Operating System Support: Ubuntu 24.04

This release introduces support for Ubuntu 24.04 across most engines. The majority of engines are now compatible and operational on this new operating system. MetaDefender Core now also supports CIS Level 2 system hardening benchmarks on this OS. For more details, refer to CIS Level 2 - Linux Guidelines.

Few engines still require additional time for full compatibility with Ubuntu 24.04:

- Adaptive Sandbox
- File-based Vulnerability Assessment
- Reputation
- YARA

Further updates will be provided in future releases.

New Operating System Support: Windows Server 2025

This release of MetaDefender Core also introduces official support for Windows Server 2025, expanding platform compatibility for our suite.

Few engines still require additional time for full compatibility with Windows Server 2025:

- Adaptive Sandbox
- File-based Vulnerability Assessment
- Reputation
- YARA

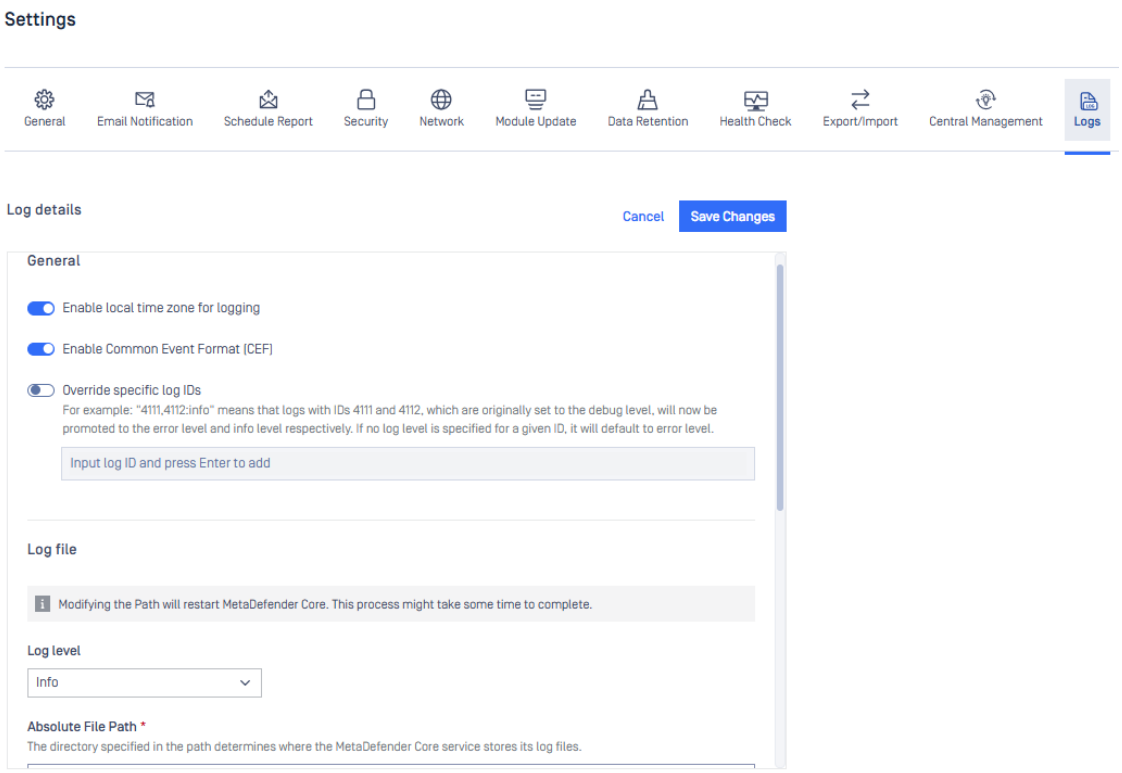
Further updates will be provided in future releases.

Support Advanced Log Configurations on Web Console

Starting with MetaDefender Core version 5.15.1, the following advanced log configurations including local time zone,

Common Event Format, product logs and syslog settings are supported.

Most settings, except for the product file path, do not require a service restart to take effect.



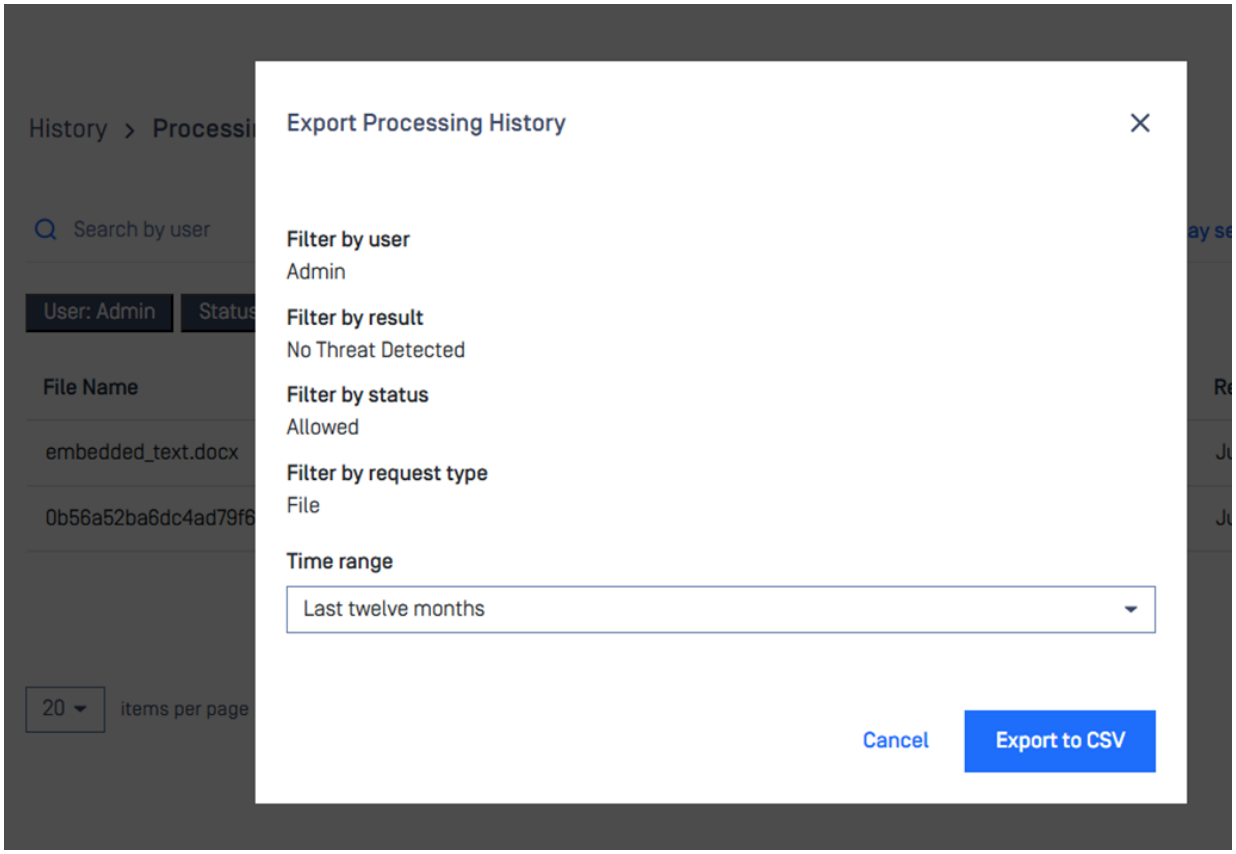
The product now offers three means of log configurations:

- Through MetaDefender Core web console (Advanced Log Configurations).
- Using REST API endpoints (PUT - Update log configuration).
- Enrolling and managing with My OPSWAT Central Management.

Processing History Export as CSV Supports More Filters

Added support for additional filters in the CSV export functionality for processing history. Users can now apply the following filters for CSV export:

- Result
- User
- Status
- Request Type (Batch, File, Object)



REST API Enhancements

Introduced a new field, `post_processing.sanitization_details.status`, to the response JSON when retrieving processing results. This field indicates the status of the sanitization process with the following possible values:

- `Queued`: The file is currently waiting in the MetaDefender Core's queue before being sent to the Deep CDR engine.
- `Processing`: The file has been sent to the Deep CDR engine and is currently being processed.
- `Finished`: The Deep CDR process has completed, with additional states available for further inspection.

Impacted REST API endpoints:

- GET /file/{data ID}
- POST /file/sync
- GET /hash/{hash value}
- GET /file/archive/{data ID}

Further Enhancements

1) Optimized PostgreSQL Query Performance: Enhanced the PostgreSQL query to improve performance when handling a large number of batches, resulting in faster data retrieval and processing.

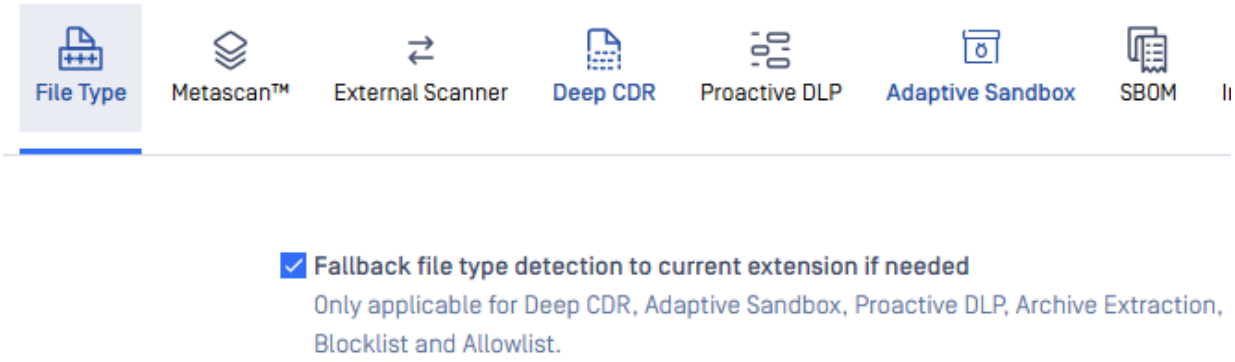
2) Enhanced Connection Persistence to My OPSWAT Central Management: Improved handling of connection persistence for MetaDefender Core to My OPSWAT Central Management (MOCM) in specific edge cases, enhancing user experience by avoiding the need for re-enrollment. Key scenarios addressed include:

- When the machine loses internet connectivity to MOCM and the service is subsequently restarted.
- When the system's root certificates are corrupted or invalid and are later updated or restored.

3) A tool for Admin to remove an existing PIN code or set a new one. More details at PIN Code Reset Tool.

4) Audit Log covers activities related to API PUT - Update list of custom response headers.

5) Extended the option "Fallback file type detection to current extension" to Adaptive Sandbox.



6) No longer processing zero-byte files with Adaptive Sandbox.

7) Enhanced engineprocess's thread pool mechanism to achieve immediate response post-termination and improved thread reusability. This increases the total number of tasks handled over time. Nevertheless, at the cost of performance boost, it could increase the load on the service if clients frequently request scan results.

Security Enhancements

Upgraded libraries for vulnerability fixes:

- Libxml2 v2.14.5
- 7z v25.0
- OpenLDAP v2.6.10
- Redis++ v1.3.14

Bug Fixes

- Resolved an issue that caused slowdowns in warehouse updates for the Executive Report and high CPU consumption when the service received a large volume of requests in a short period.
- Resolved an issue that caused Executive Report data to freeze following a sudden shutdown of the service.
- Resolved an issue that caused unnecessary retry attempts when Adaptive Sandbox returned a SKIPPED result.
- Resolved an issue where slots in the queue for Batch processing were not released and temporary files were not cleaned up when the client connection was closed, resulting in improper transmission of error responses.
- Resolved a potential issue that could cause the service to crash due to incompatibility between a library with Qt6.

Version v5.15.0

Release Date: 03 July 2025

Supports CIS Level 2 for Ubuntu 22.04

MetaDefender Core now supports CIS Level 2 system hardening benchmarks on Ubuntu 22.04. This enables organizations to meet more rigorous compliance requirements in regulated or high-assurance environments.

For more details, refer to CIS Level 2 - Linux Guidelines. Support for additional systems is planned for future releases.

Export & Import Engine Advanced Settings

A new feature has been added to support exporting and importing engine advanced settings. This enhancement allows administrators to:

- Export all engine settings, even for modules that are currently deactivated.
- Import these settings into a new installation, ensuring that module activation states and configurations are preserved.
- Restore previous settings for engines if they are re-enabled after import.
- Control the feature via the API header `engine_config`: GET - Export configuration.

This feature improves configuration management and migration scenarios, especially for customers who need to maintain consistent engine settings across deployments.

User Management via My OPSWAT Central Management API Endpoints

MetaDefender Core now supports local user management through the REST APIs of My OPSWAT Central Management when integrated with that platform. This allows administrators to automate user account provisioning and role assignments across distributed environments, enabling better alignment with centralized identity practices.

This support is available starting from **MetaDefender Core version 5.15.0** and **My OPSWAT Central Management version v10.1.2507.**

For more information, please refer to the official My OPSWAT Central Management documentation.

Additional File Size Limit Enforcement

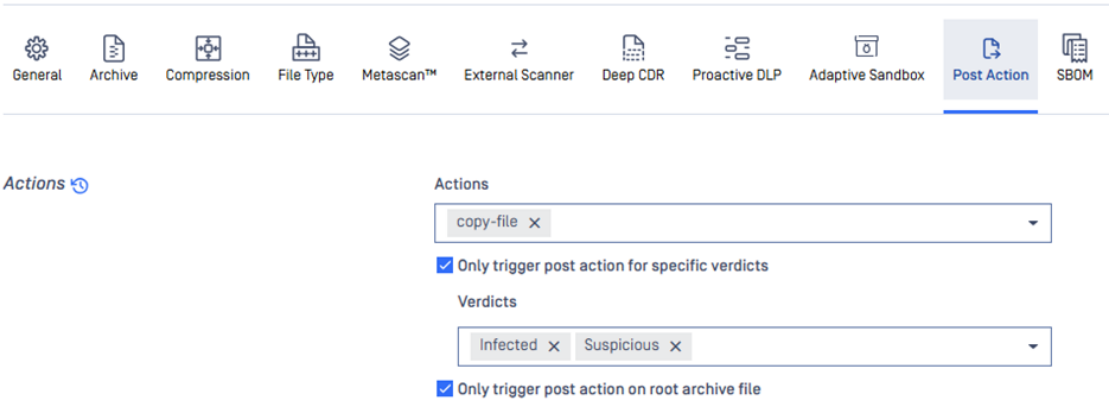
Administrators can now define a file size limit at the NGINX layer. This enables earlier enforcement of policy controls and supports more distributed or containerized environments.

This is configurable at the key `body_size` in ignition file. More details at Ignition file fields.

Post-Action Triggers Scoped to Root Archive Files

To improve processing efficiency, post-action triggers, such as copying or quarantining files, now apply to root archive files. This refinement avoids unnecessary processing of nested archive layers and aligns more closely with typical workflow logic.

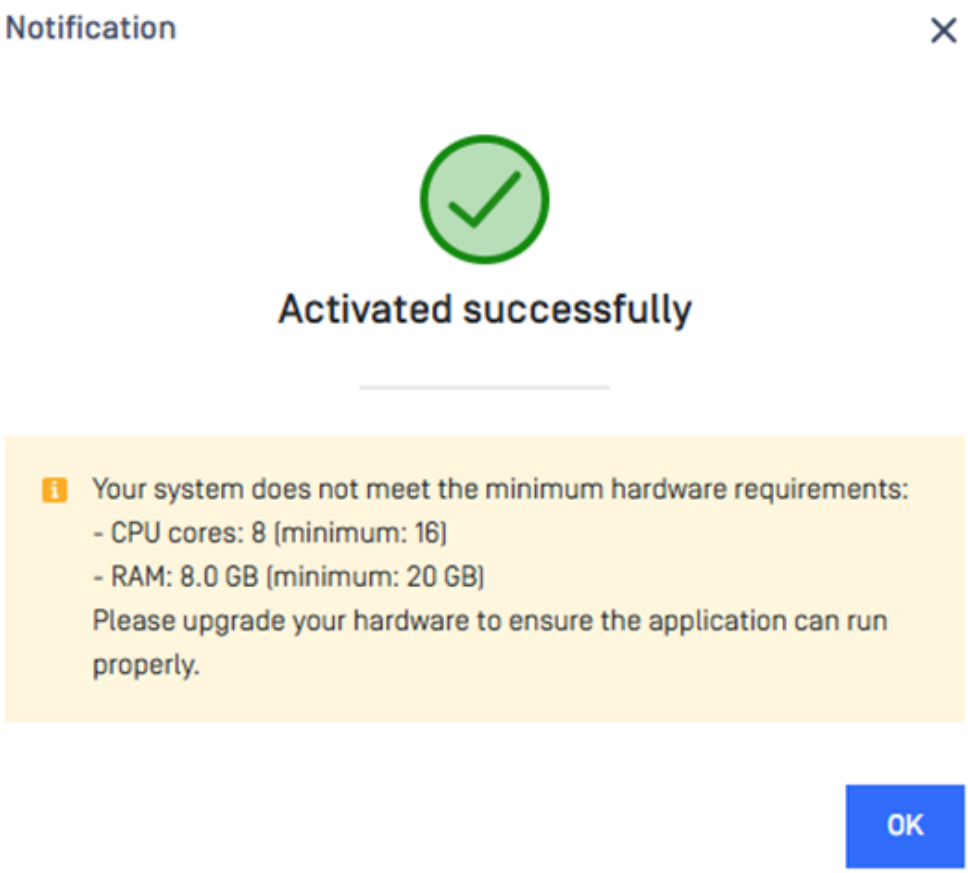
You can find the new option `Only trigger post action on root archive file` under tag Post Action in Workflow Rule.



System Specification Warning at License Activation

The product now displays a warning if the system does not meet the minimum recommended hardware specifications.

This post-activation check helps administrators verify whether their environment is properly sized. While continuing with an under-spec setup is allowed, doing so may lead to performance issues or instability under certain workloads.



Setting Up A Secure Connection During Installation

Administrators can configure HTTPS during the initial deployment process using ignition file, rather than post-installation on web console or via API call. This simplifies secure setup and reduces deployment time.

A new set of settings are introduced at HTTPS section in Ignition file.

Executive Report with Blocked Reasons Summary

Administrators can obtain number of objects blocked by several reasons.



Improved Local Folder Update Functionality

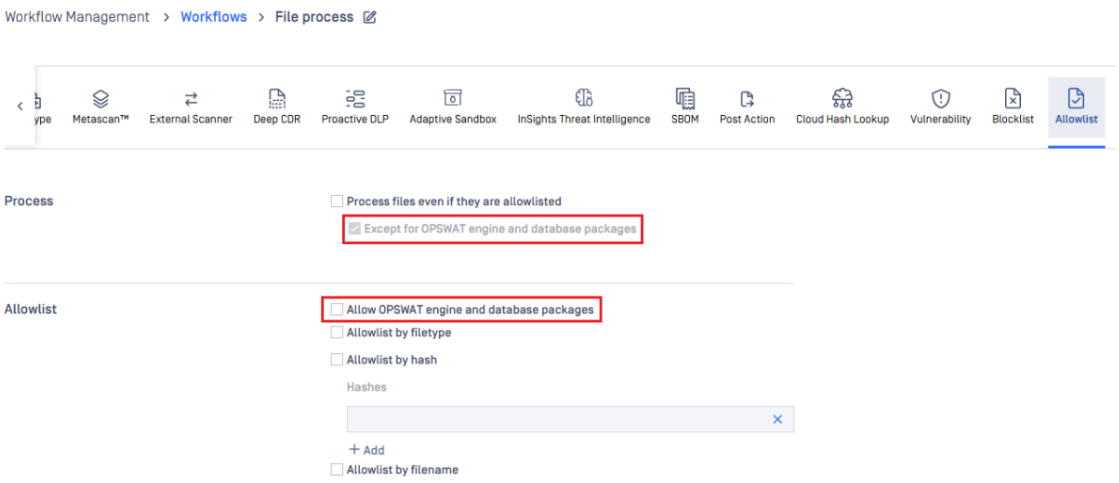
Using the Local Folder Update folder with a UNC path previously consumed significant network bandwidth.

An optimization for new update check in this version considerably reduces network bandwidth and hardware resource usage.

Allow OPSWAT Module Update Packages

MetaDefender Core always performs integrity checks and verification to ensure the packages remain untampered. Therefore, Administrators can allow trusted OPSWAT update packages to pass through policy filters, ensuring essential updates are applied reliably without triggering unnecessary scan and consuming additional resources for deep processing.

New options have been introduced in the Allowlist workflow configuration to automatically allow the OPSWAT engine and database packages.

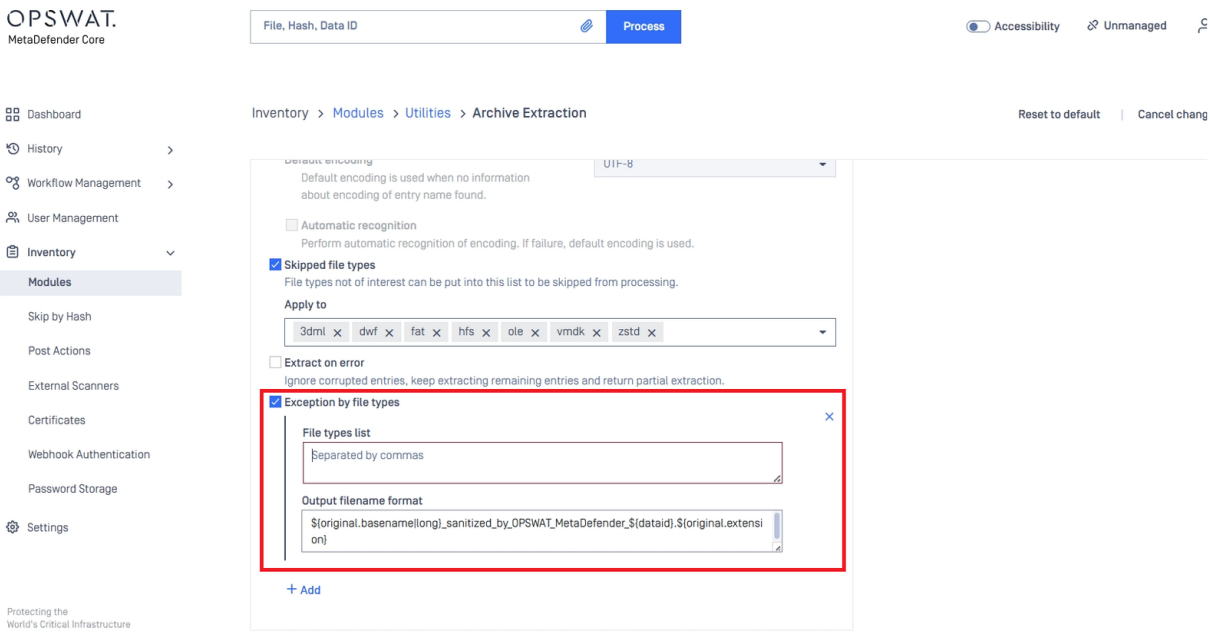


Usability Enhancements & Changes

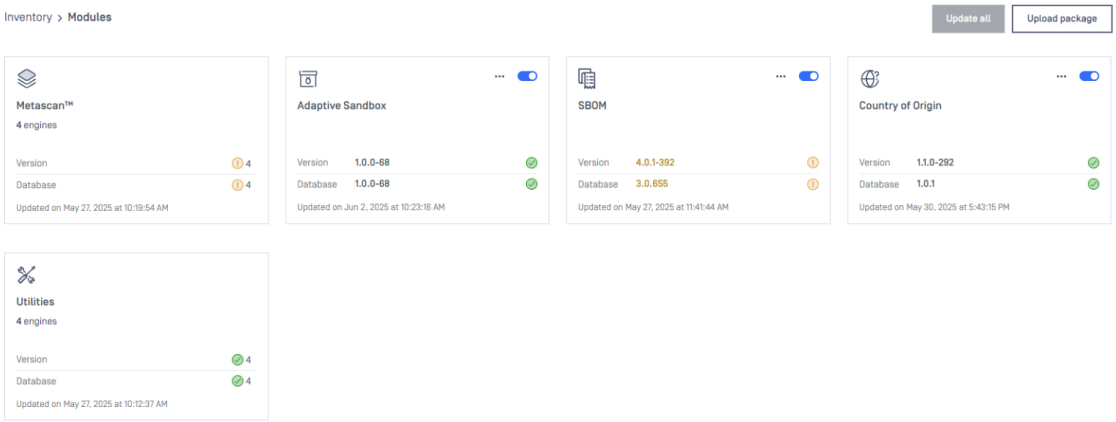
1) Automatic Cleanup for File Corruption: The product now includes automatic cleanup functionality to address file

corruption and residual files caused by external issues, such as connection loss, during the process of adding files to sanitizing storage. This enhancement ensures improved cleanliness of the storage environment.

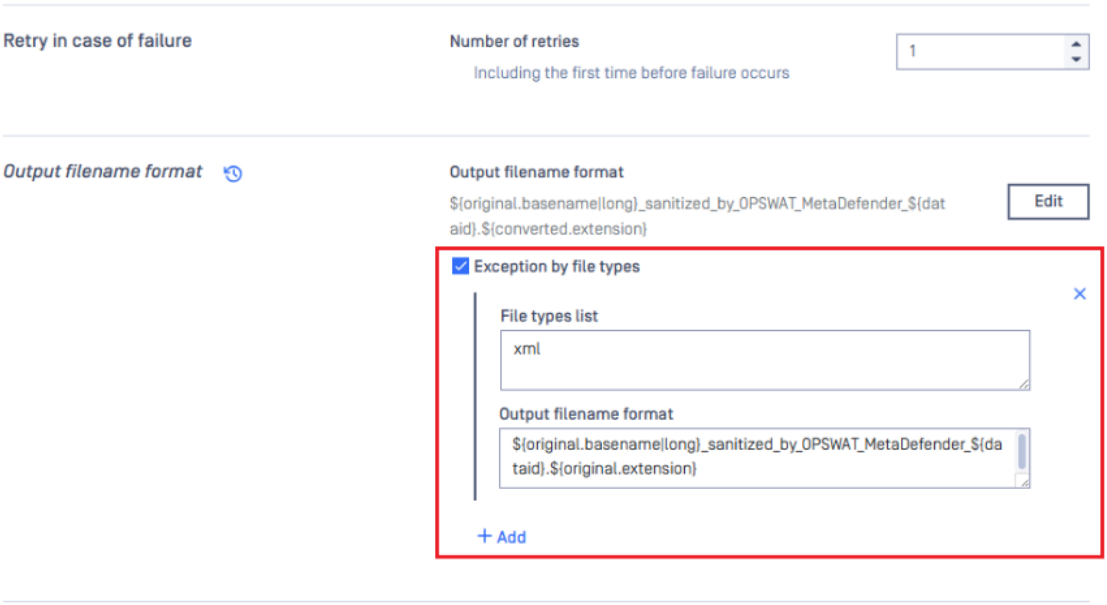
2) Improved the look and feel of multiple text boxes within lists in Workflow Rule.



3) When Local Folder Update is selected and in use, the button `Update all`, which manually triggers update of engine and database packages, becomes unnecessary and is therefore grayed out.



4) A new sub-option, **Exception by file types**, has been added under the `Output filename format` section in the Deep CDR tab of Workflow Rules. This feature allows users to control and customize the naming convention of sanitized files, particularly enabling the retention of original file extensions when required.



REST API Enhancements

1) Attach MetaDefender Core server information in the response header X-Core-Id of the following API endpoints:

- `POST` `/file/batch` - Initiate Batch
- `GET` `/hash` - Fetch Analysis Result By Hash

2) Supporting customizable parameters with new header `info` in the REST API GET /readyz allows users to query specific system metrics, such as CPU load thresholds, relevant to their use case. Available values:

- all: return all the information
- database: obtain database status
- license: obtain license status
- avg_duration_time: obtain average duration (processing) time
- cpu_load: obtain CPU load as of calling the API
- scan_queue: obtain scan queue stats
- number_active_av_engines: obtain the number of active AV engines threshold
- engines: obtain engine status

3) Introduce a new API endpoint to fetch a list of blocked leaf files inside the archive by Hash: GET /hash/{hash value}/blocked-leaves.

4) Introduce a dedicated API endpoint to validate PIN code: POST /admin/pincode/validate.

5) Provide a new API endpoint to trigger `Local Folder Update` even during business hours or pause update window: POST /admin/update/folder.

Logging Improvements

- Improved audit logs to include all activities on administrative configurations.

- To easily investigate network error while using SSO, we added more detailed log messages for authentication module.

Security Enhancements

Upgraded libraries for vulnerability fixes:

- PostgreSQL v14.18
- Angular v19
- yaml-cpp v0.8.0

Bug Fixes

- Fixed a crash issue that potentially happens during proxy setup. Network management is unavailable when the proxy configuration is applied, leading to a crash of MetaDefender Core. This scenario occurs with an extremely low likelihood.
- Fixed an issue where the setting `Processing results to be classified as Allowed` was not applied correctly, causing erroneous blocking of results.

Version v5.14.2

Release Date: 28 May 2025

Integration with MetaDefender Distributed Cluster v2.0

Enables integration with a new product, MetaDefender Distributed Cluster v2.0, giving organizations exclusive benefits:

- High availability: Distributed architecture ensures continuous inspection, even during instance-level outages or maintenance.
- Fault tolerance: Isolate failures and support automatic recovery, preserving uptime and service continuity.
- Load distribution across MetaDefender Core instances: Share file scanning tasks across multiple instances, eliminating performance bottlenecks.
- Parallel archive file handling: Extract and distribute archive contents across Core instances, reducing scan time for large files and enabling resumption of interrupted processing.
- Elastic scalability: Easy to add more instances as file volumes grow, scaling capacity seamlessly to meet organizational demand.
- Centralized orchestration: Manage all MetaDefender Core instances and API Gateways from a single Control Center web console. Provide a zero-downtime upgrade mechanism with just a few simple operations.

More details about this new product at MetaDefender Distributed Cluster Overview.

New Operating System Support: Amazon Linux 2023

This release introduces support for Amazon Linux 2023 across most engines. The majority of engines are now compatible and operational on this new operating system.

Few engines still require additional time for full compatibility with Amazon Linux 2023:

- NANOAV
- File-based Vulnerability Assessment
- RAR compression of Archive Compression

Further updates will be provided in future releases.

Client Identity Tracking

This release introduces support for sending client identity information with file scan requests into MetaDefender Core via a new header `client-identity`, enhancing visibility into the scan request lifecycle.

This optional header captures details provided by the clients a scan request passes through (e.g., MetaDefender KIOSK MetaDefender MFT MetaDefender Core), including:

- Client name
- Client host
- Username
- Client's deployment ID
- Received timestamp at client
- Client product version

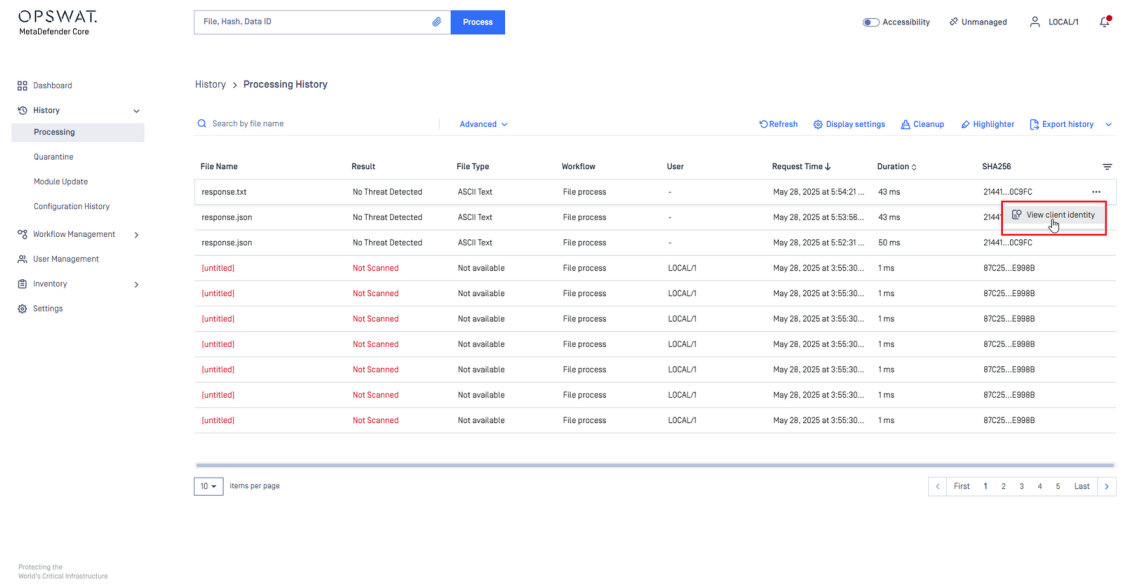
All the following methods support this header: asynchronous mode, synchronous mode, local scan, scan from link, batch scan, multi-part uploading.

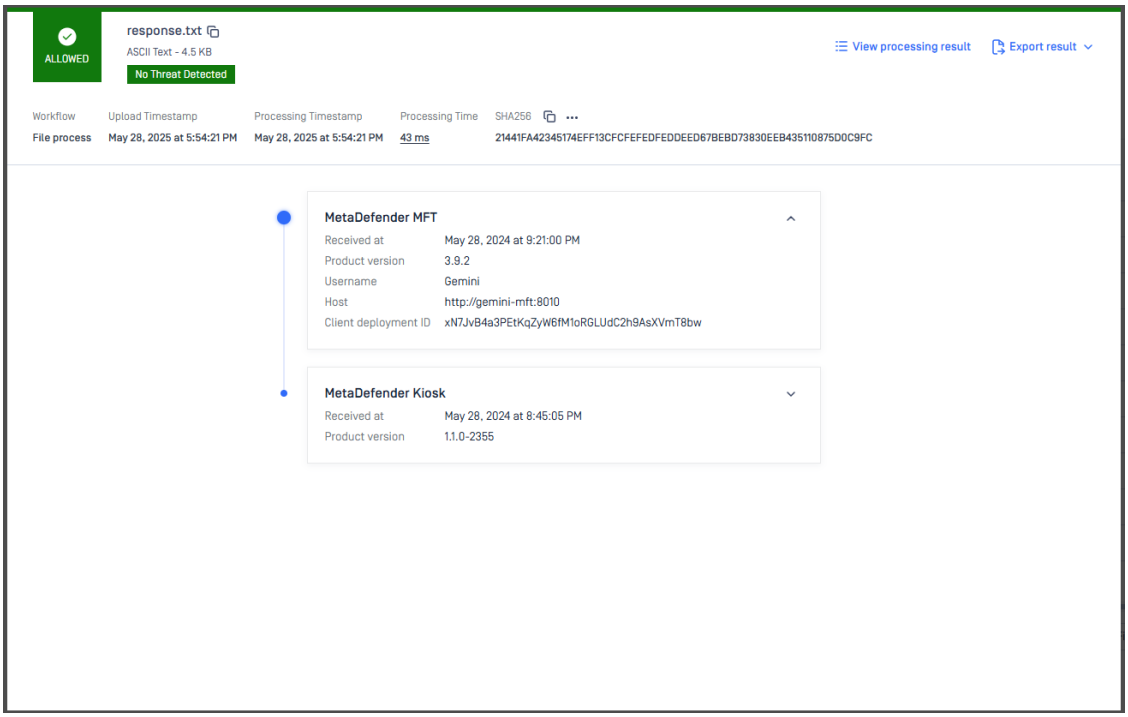
More details at:

- GET - Fetch Analysis Result
- GET - Fetch Analysis Result By Hash
- GET - Status of Batch Analysis
- POST - Analyze File (Asynchronous mode)
- POST - Analyze File (Synchronous mode)
- POST - Initiate Batch
- POST - Create multipart uploading session

The client identity details are accessible through:

- **MetaDefender Core web console**: View detailed scan request information directly in the user interface.
- **JSON Processing Result**: Client identity data is included in the processing result output.
- **My OPSWAT Central Management (MOCM)**: MetaDefender Core forwards client identity information as part of the scan report to MOCM for centralized management and reporting.





Archive Password Support for Multi-part Uploading

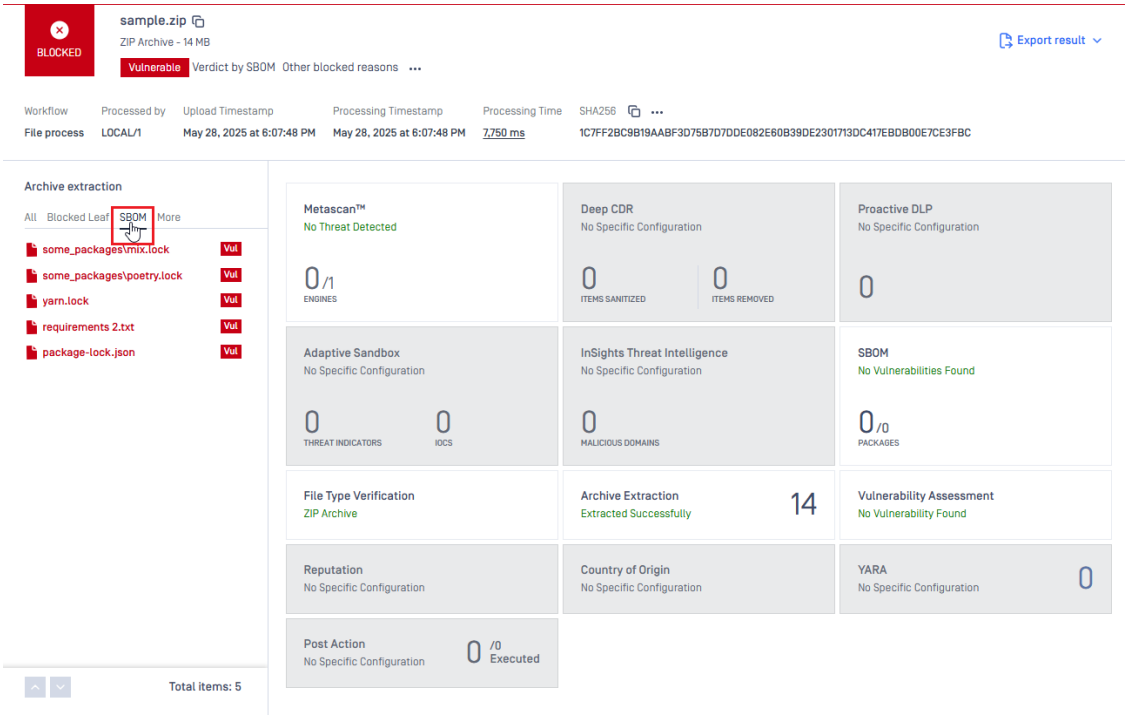
Starting with version 5.14.2, MetaDefender Core introduces support for archive password handling in multipart upload APIs, enhancing functionality for processing password-protected archives.

The passwords are transmitted via header `archivepwd` and are secured through at-rest encryption.

More details at POST - Create multipart uploading session.

SBOM Package Filter for Archive Files and Batch

A new SBOM filter functionality has been added to the Scan Results page. This feature enables users to efficiently filter and view results for nested files in archives and batch-processed files containing SBOM package information.



Usability Enhancements & Changes

- Setting "Block files if the process fails or times out" under Deep CDR workflow now flags a result as Blocked even when sanitized version of a file cannot be added to storage successfully.
- Ensure log rotation has unique filename and no remaining file.

Technology Integration Enhancements

- Enable engines to display hyperlink within its result details.
- Future versions of Archive Extraction engine will move away from the ID format `7z_` and provide a new format `extraction_` . Since this version of MetaDefender Core, the new format is supported along with the old one.

Security Enhancements

- Upgraded libraries for vulnerability fixes:
  - NGINX v1.26.3
  - Libxml2 v2.13.8
- Improved API endpoints' validation.
- Replaced deprecated functions of 3rd libraries.

Bug Fixes

- Fixed an issue that prevented access to web console after OS reboot. This issue happened with standalone mode and a remote PostgreSQL. When the issue happened, an error "Can't create connection" was found in log.
- Fixed an issue that stopped email sending after the machine lost internet connection.
- Fixed an issue that blocked modification of webhook configuration after enrolled into My OPSWAT Central Management.

- Fixed an issue that failed enrollment via ignition file.
- Fixed an issue that omitted blocked reason "Extraction Failed" when a temporary extracted file was absent.
- Fixed an issue that incorrectly disabled Country of Origin engine after importing a configuration with "Allowlist by vendor" setting enabled.

Version v5.14.1

Release Date: 28 April 2025

New OS support: Oracle Linux 9.5

This release of MetaDefender Core introduces official support for Oracle Linux 9.5, expanding platform compatibility for our suite.

New endpoint for API Key Management

To offer automated user management, MetaDefender Core provides a new REST API that Administrators can call to set or change a permanent apikey for local-type users.

API document at Update user's API key.

Securing syslog communication over TLS

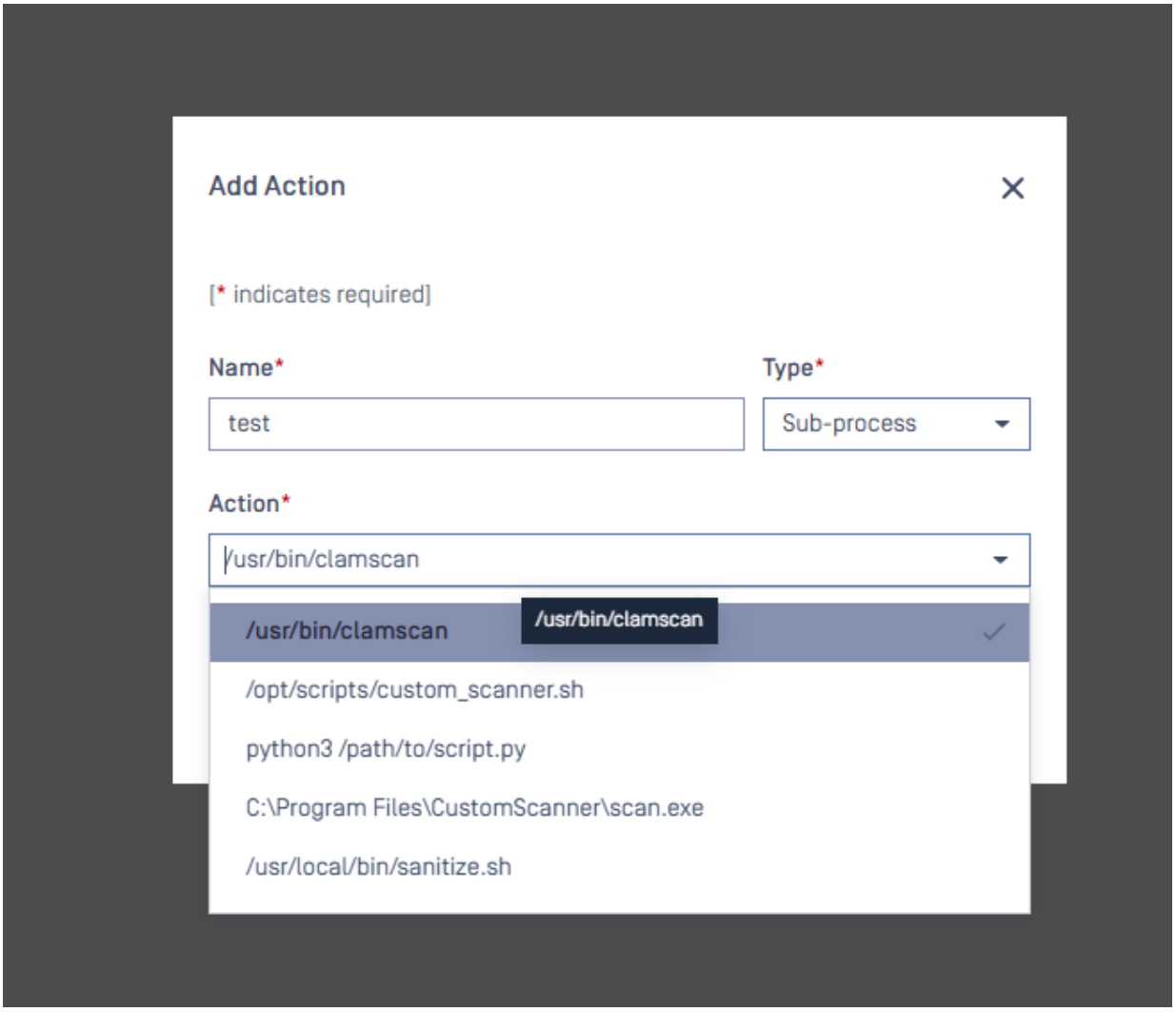
This version introduces support for securing syslog communication over Transport Layer Security (TLS), enhancing the confidentiality and integrity of log data transmitted across networks.

New protocol `tcps` is supported. Documentation at Logging, and API can be found at Apply syslog server configurations.

Post Action and External Scanner

Introduce a command restriction for Post Action and External Scanner features enabling server Administrators to specify a list of permitted commands that these features can only use.

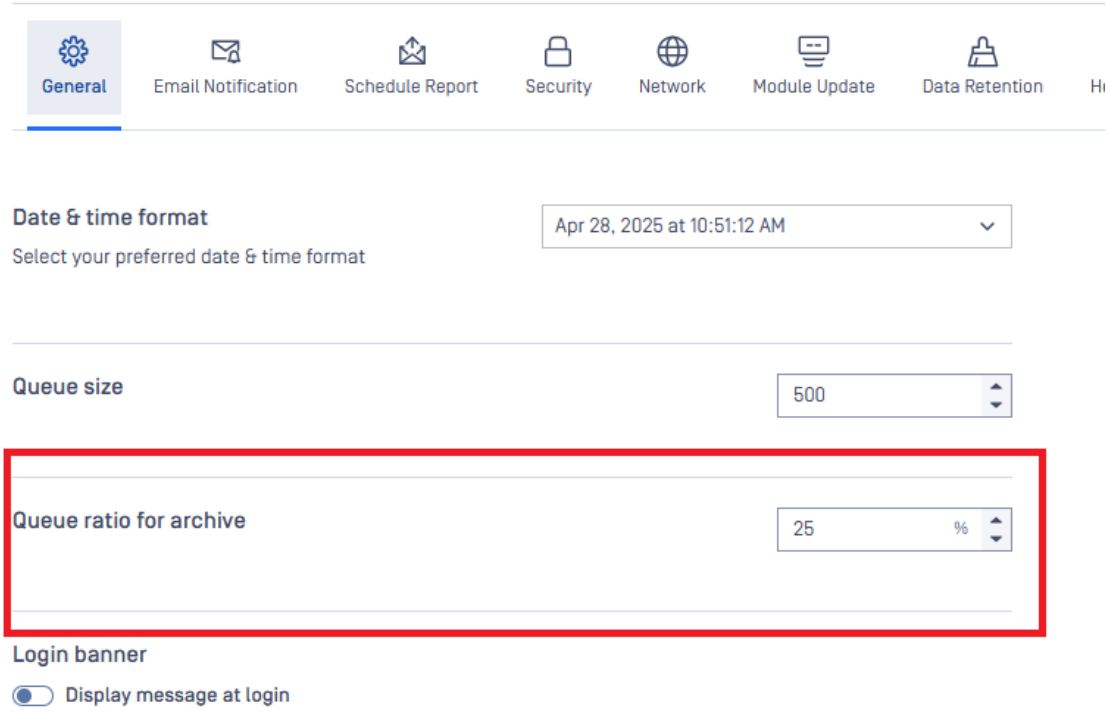
More details at Hardening feature for Post Action and External Scanner.



An ability to adjust maximum slots of queue capacity for archive processing

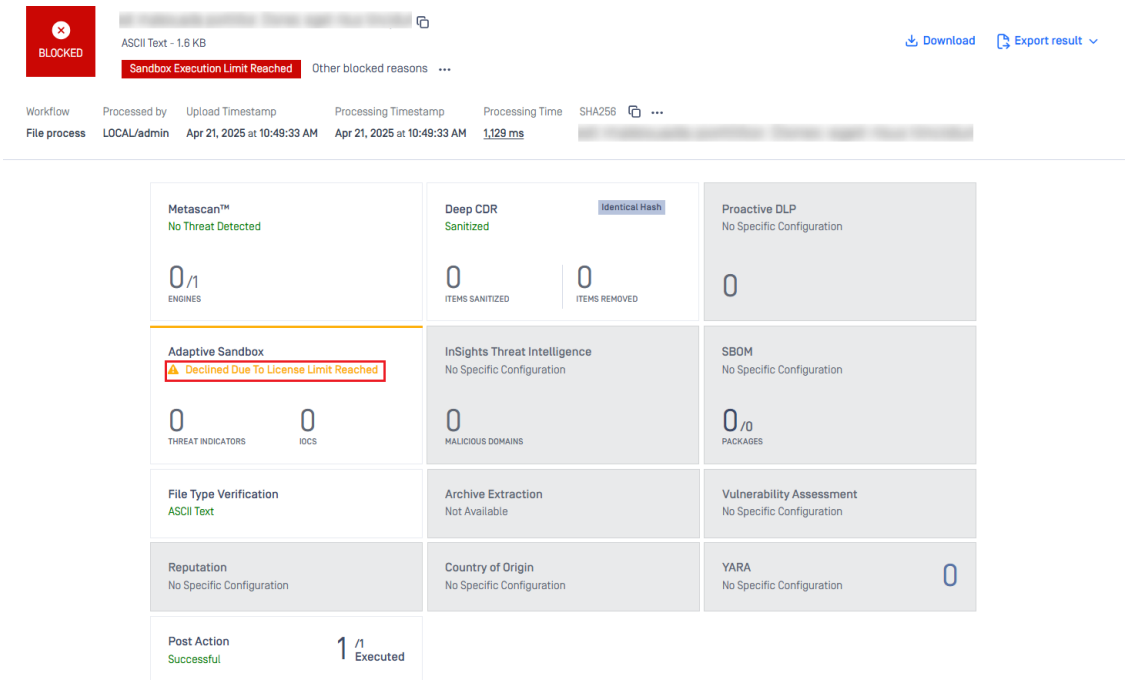
Previously, archive processing was fixed at 25% of the maximum queue size. Now, Administrators can configure this value within a range of 25% to 75%, providing better flexibility to optimize resource allocation for archive processing to improve efficiency in high-throughput environments.

You can find the setting `Queue ratio for archive` at Settings/General.



Embedded Adaptive Sandbox engine with Daily Scan Quota

When the daily scan quota is reached, Sandbox engine will reject scan requests and display a warning message `Declined due to license limit reached`.

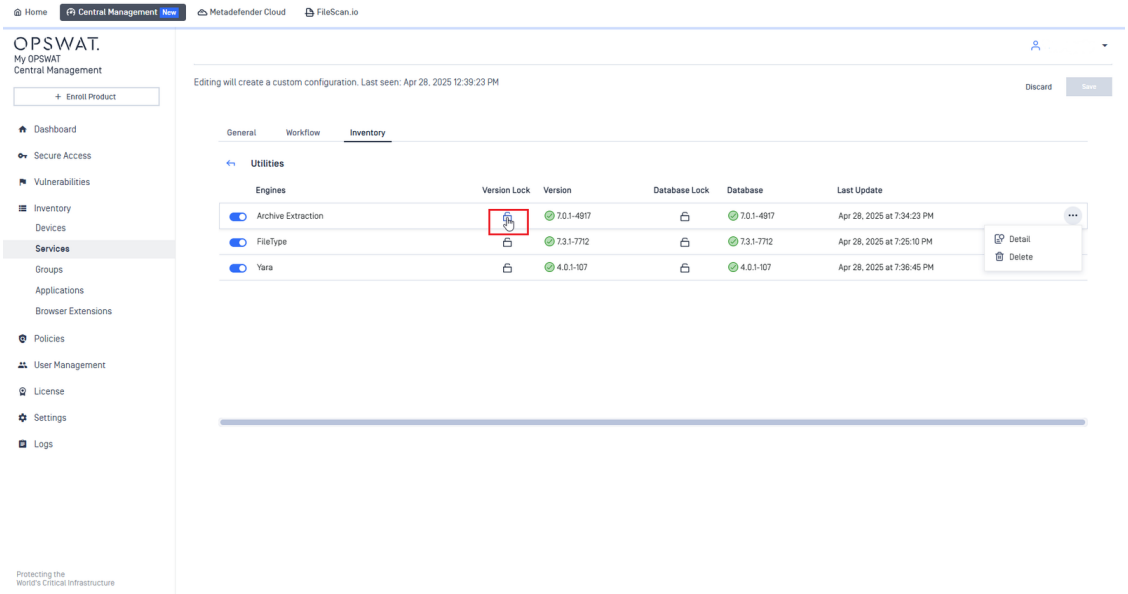


Users can decide to block files due to out-of-quota by ticking a new option `Block files if the execution limit reached`. It can be found in Workflow/Adaptive Sandbox.



Continuous Integration with My OPSWAT Central Management

To close the experience gap, the product now supports more features in Inventory/Modules of MetaDefender Core on My OPSWAT Central Management like engine activation, update lock & unlock.



Usability Enhancements & Changes

- Get prepared for a new engine ID of Archive Extraction engine.
- Changes in extracted object aggregation, `total_extracted_files` now excludes nested folder/directory.
- Handle an edge case where a missing test sample results in engine initialization failure.
- Improved the detection and redaction of support package generator tool. It coincidentally mistook engine versions for sensitive data.
- Provide new options for having Blocklisted and Allowlisted reasons in PDF result export.
- Introduce a new display option to switch between 12-hour and 24-hour clock format. This applies to either web console or exported PDF/CSV file. You can find the setting "Date & Time format" at Settings/General.
- No longer showing a misleading label `True Type` in result details of File Type engine when the result is "Not Available".
- "Processing results to be classified as Allowed" setting now included “Suspicious Verdict by Sandbox” by default.

Technology Integration Enhancements

- An ability to continue further processing based on results concluded by InSights Threat Intelligence engine.
- Adjust timeout value for the following engines (File Type Verification, YARA, SBOM, Reputation, COO, InSights

Threat Intelligence) to reflect correct remaining time. Only applicable when global timeout is enabled.

- Create permanent storage folder for internal jobs of Adaptive Sandbox engine.

Security Enhancements

- Restricted modifications to information of an Administrator account to exclusively be performed by accounts with Administrator privileges.
- Implemented secure random number generator.
- Improved endpoints' validation.

Bug Fixes:

- Fixed an issue that stopped periodic email notifications for out-of-disk-space and database maintainance events from being sent.
- Fixed an Email Notification issue that kept checking license expiration despite of that the setting was disabled.
- Fixed a display issue that blocked users to view workflow or user directory on touch screen.
- Fixed an Import issue that temporarily (~3mins) caused lack-of-permission issue for a user who previously had imported configuration to the product.
- Fixed a Data Retention issue that mistakenly cleaned up sanitized files and DLP-processed files. This issue only occured when feature "Skip hash calculation" is enabled.
- Fixed an issue that sent too frequently an event log "Synchronized configuration" to My OPSWAT Central Management (MOCM).
- Fixed a crash issue happening during unenrollment the product on Linux from MOCM.
- Fixed an issue that stopped syncing engine advanced settings to MOCM when an engine is being updated.
- Fixed issues that failed engine advanced setting synchronization after editing multiple engines' settings on MOCM.
- Fixed an issue that failed to apply engine advanced settings after several configurations have been edited on MOCM previously.
- Fixed an issue that failed to sync YARA source settings from the product to MOCM after performing import configuration.
- Fixed an issue that interrupted the group policy inheritance of MOCM and failed to apply those settings on the product.
- Fixed a display issue that showed up an empty screen of Inventory/Modules on MOCM after upgrading to v5.14.0.
- Fixed a display issue that sometimes showed up an empty screen of Workflow on MOCM after enrollment.

Version v5.14.0

Release Date: 31 March 2025

New features:

- Advanced SSL Settings Support: Configure and manage advanced SSL settings directly through the web console

for a seamless user experience and flexibility.

- Passphrase Support for Encrypted Private Keys in Certificates: An ability to add encrypted private keys to Certificates with the option to include a passphrase, providing an additional layer of security for sensitive data.
- Email Notification Feature Enhancements
  - Customizable Email Subjects: Tailor email notification subjects to suit your preferences or organizational needs.
  - Outdated-Definition Engine Alerts: Receive email notifications when engine definition becomes outdated, ensuring timely updates.
  - Disk Space Warnings: Get notified via email when your system is running low on disk space, helping you avoid disruptions.
  - Database Maintenance Reminders: Stay on top of database maintenance with periodic reminders displayed on the web console and sent via email.
- Quarantine Feature Enhancements
  - Stronger Encryption Algorithm: The Quarantine feature now supports a more robust encryption algorithm, ensuring greater protection for quarantined items. This protection is off by default to avoid affecting existing integration performance.
  - Configurable Quarantine Conditions: Define file size limit for files that should not be quarantined, giving you control over resource allocation and quarantine behavior. You can find this setting under Workflow Rule.
- Integration with My OPSWAT Central Management
  - Introduce first integration with My OPSWAT Central Management v10. Most of available features on v8 are migrated and retained on the v10 integration.
  - In addition, Administrator can manage and control Modules and advanced module settings seamlessly through My OPSWAT Central Management, streamlining administration and oversight from a single interface.
- Usability Enhancements & Changes
  - Support login with format `<domain>\<username>` for AD/LDAP user accounts.
  - An ability to export offline activation license for backup and restore. The exported license is safely stored in password-protected ZIP file.
  - Introduce a new child setting under 'Block files that exceed the file size threshold' to exclude good verdicts from specific technologies, preventing those files from being blocked due to exceeding the file size threshold.
  - Support nomination for hot-backup instance with OLMS activation.
  - Add more accurate load factors such as CPU load and Avg. duration time in the health check API. For more details, refer to Get health check status.
  - New API for AD-bound password rotation automation. For more details, refer to Change Password for AD/LDAP User Directory.
  - Reason details for blocklisted and allowlisted results.
  - An ability to search for users of user groups in Processing History.

  - New audit logs for JSON and PDF result export from Processing History.

- Technology Integration Enhancements
  - Provide YARA results to Sandbox engine for further analyzing.
  - Provide MIME type for sanitized and DLP-redacted files.
  - Support workflow settings for Archive Extraction and Archive Compression engines.

- Security Enhancements
  - Upgrade libraries for vulnerability fixes:
    - OpenSSL v3.4.1
    - quazip v1.4.0
    - libxml2 v2.12.10
  - Remove unnecessary .NET 6 from Windows Azure image.
  - Make Password Recovery feature more robust: Users now only receive an expirable token in password-reset email and use that token to set a new password. For more details, refer to Forgot Password.
  - Enhance error response to not return raw messages and HTML source code.

- UI updates
  - Display password strength meter.
  - Save "X items per page" as user preference.
  - Rebrand "My OPSWAT" to "My OPSWAT Central Management".
  - Display tooltip for items in workflow dropdown list.
  - Minor cosmetic UI changes.

Fixes:

- Addressed an issue that caused update failure to a version of FileType engine which supports result preference.
- Resolved a limitation that made compression failed while handling more than 300K file entries.
- Addressed an issue that caused JSON config import failure with message `an unexpected property: "dlp.filescan.certainty_threshold": "Low"` .
- Addressed an issue that displayed "mismatch db_api" while upgrading from a quite old Proactive DLP.
- Addressed an issue that omitted blocked reasons when a file is blocked by COO engine due to "Unsupported file type".
- Addressed an issue that caused Proactive DLP to return "Unsupported File Type" when processing .docx, .xlsx, .txt files in a random manner.
- Addressed an issue that caused a crash when stopping engine process during big archive file scanning.
- Addressed an issue that caused HTTP 500 Internal Server Error when filtering Processing History with null

parameters.

- Addressed an issue that caused product upgrade failure when upgrading from v5.11.1 to 5.12.x or 5.13.x with activated SSO. The error message is "Failed to init AES cipher".
- Addressed an issue that caused high CPU consumption on PostgreSQL process after product upgrade.
- Addressed an issue that caused a considerable delay in fetching results of a large batch scan.
- Addressed an issue that caused an incorrect proxy settings retrieval.
- Addressed an issue that caused Deep CDR unable to sanitize file after upgrading from product v4 to the latest.

Version v5.13.3

Release Date: 27 February 2025

New features:

- Enhanced Data Retention mechanism to perform effectively and reduce execution time.
- Upgrade PostgreSQL to v14.17 for vulnerability fix.
- Upgrade Angular to v18.
- Minor cosmetic UI changes.

Fixes:

- Addressed an issue that used up Linux file descriptor when using synchronous scan method.
- Enhanced query to avoid max stack depth error while cleaning up numerous depth levels of archive files.
- Improved NGINX logging to indicate initialization failure clearer.
- Resolved an issue that caused upgrade process suspended.

Version v5.13.2

Release Date: 21 January 2025

New features:

- Triggering condition for Post Action: To focus needed requests and save up resources, it is necessary to introduce a triggering condition for Post Action where Administrators can select and filter what verdicts to trigger the actions only.

Fixes:

- Addressed an issue that caused license activation failure in K8S when using MetaDefender Core with HTTPS enabled.
- Addressed an issue that caused automated license activation failure after Azure image instance was restarted.
- Addressed a crash issue that happened when checking invalid sanitized file existence.

Version v5.13.1

Release Date: 16 January 2025

New features:

- New technology: MetaDefender InSights Threat Intelligence
  - This technology can detect known malicious IP addresses and domains inside document files.
  - To enable the technology, you need to contact OPSWAT to update your license. Then you can just go to Workflow Rule, turn the switch on and explore its capability.
- New OS support: Debian 12
  - Since v5.13.1, we also replace base image Debian 11 with Debian 12 minimal base image.
- Prohibited from downloading produced files when original file is flagged as Blocked
  - Introduced a setting to add an extra layer of security that makes files produced by Deep CDR or Proactive DLP unavailable for download when original file's result is flagged as Blocked for any reasons. This setting is off by default and can be found under tab General in workflow rule.
- Deep CDR analysis mode
  - When analysis mode is turned on in workflow rule, Deep CDR no longer sanitize file, remove object and generate a new file. Instead, the technology will report objects inside the file for analysis purpose. Under this mode, you can see a tag flagging the mode on, number of detected objects and a neutral green message.
- Usability enhancements / changes
  - Precisely calculate "hash time" of nested files to have more relevant "Child files hashing" time.
  - In Linux non-persistent mode and local scan method, webhook callback is enhanced to ensure responses eventually to be sent even if there is permission issue or else with scanned local files.
  - Users now can download list of IoCs detected by Adaptive Sandbox from the web console.
  - New option to block files whose size is larger than file size threshold of Adaptive Sandbox.
  - Users now can granularly customize time period in Data Retention settings.
  - Enabled verdict searching in Advanced filter of Processing History.
  - Moderate UI changes to display hash values in full text as well as make other metadata result shown more clearly.
- Security enhancements
  - Upgrade OpenSSL to v3.4.0 for vulnerability fix.
  - Upgrade protobuf to v3.21.12 for vulnerability fixes.
- Modify Docker setup for Rocky Linux to ensure proper dependency checking for Proactive DLP.
- Minor cosmetic UI changes.

Fixes:

- Addressed an issue that caused service crash when validating non-existed ID of output files returned by Deep CDR and Proactive DLP.
- Addressed an issue that might cause memory leak when using scan-from-link feature.
- Addressed an issue that caused service unresponsive due to slow data retention process.
- Addressed an issue that made upgrade process and service hang in container while running with remote PostgreSQL v14 Windows.
- Addressed an issue that was "Fallback file type detection to current extension if needed" setting did not apply to Proactive DLP.
- Addressed an issue that caused misleading Proactive DLP result while processing archive file. It displayed "No Specific Configuration" meanwhile it should have been "Unsupported File Type".
- Addressed an issue that misvalidated setting file during Import process and made workflow rules inaccessible.
- Addressed an issue that caused SSL handshake error when downloading file with scan-from-link feature.
- Addressed an issue that caused product in Rocky container did not automatically activate license after HTTPS enabled.
- Addressed an issue that caused misleading Metascan result in PDF result of a batch. It displayed "No Threat Detected" whilst it should have been "Not Scanned".
- UI bug fixes:
  - On Average File Size chart, title sometimes was not truncated correctly.
  - Fixed an issue that caused UI infinite reload on Change Password screen.
  - Fixed an issue that caused Accessibility mode not be switched with Tab and Enter keys.
  - Fixed an issue that wrongly displayed "No Specific Configuration" at Deep CDR tile despite of the fact that result is "Sanitization Failed".
  - Other minor bugs.

Version v5.13.0

Release Date: 18 December 2024

New features:

- New OS support: Rocky Linux 9
  - We also provide a new Docker image based on official Rocky Linux repository. You can try it here: MetaDefender Core Rocky Linux image and Rocky Linux for non-root image.
- OPSWAT has discontinued support for CentOS 7 and RHEL 7 since December 2024 in MetaDefender Core and its associated engines. If you need some more time to transition from legacy OS, refer to this instruction Lock Your Engine and Plan for Migration.
- Report scheduling

  - MetaDefender Core provides a new feature that enables Administrators to schedule not only when executive report and processing history report are sent but also set recurring schedules for automated delivery.
  - This feature goes with abilities to specify email recipients, apply filters to tailor the data and define precise time range.

- Administrators now can directly generate and download support packages right on web console.
- Advanced rate limiting configuration via web console
  - For more details, refer to Request rate limit configuration and Request rate limit test scenarios.
- Syslog configuration via REST API
  - Apply new syslog server configurations
  - Remove syslog servers or clear all syslog configurations
- Ability to download summarized user list
- New chart and statistics to display how many percentages/files are reused by hash.
- Tool to hide specific data from support package.
  - Provides an ability to hide sensitive data from logs, support package, which may contain information such as IPv4, IPv6, MAC address and file path.
  - For instruction, refer to How to Create Support Package With Bundle Tools? and try the new parameter `-enable -redact` .
- Introduces an ability for Administrators to download a summary list of users and groups present on the product. You can find the Export button in User Management which generates a CSV file listing all the users and groups' information.
- Supports CIS Level 1 for Ubuntu 22.04 and RHEL 9. For more details, refer to CIS Level 1 - Linux Guidelines.
- PostgreSQL v16 support (remotely).
  - The product does not perform any version upgrade on customer's remote PostgreSQL.
  - Note: The bundled PostgreSQL version is v14.15 in this release.
- For files scanned with Adaptive Sandbox, result display now provides more details as well as adding informative records in "Quarantine" feature.
- Usability enhancements / changes
  - An ability to add description for users in the product.
  - Optimized UI rendering to improve display speed especially when rendering an immensely long list of file types in Blocklist.
  - Only display Unmanaged when My OPSWAT returns HTTP 404 and error "Instance not found".
  - Display correct involved disks and its purposes of use in System Health.
  - Display current number of finished files inside batch report.
  - Enhanced certificate loading to ensure Webhook communication over TLS.

  - Support "smaller than" operator for Blocklist and Allowlist by filename.
  - Reused scan results in Processing History now are tagged with a specified label to help users identify easily.
  - New changes in workflow setting "Exceptional verdicts": add "Cancelled" by default, and remove "Scan failed".
  - Unchecked "Detect file type mismatch" setting of "MetaDefender Software Supply Chain" workflow rule.
  - Enhanced retry mechanism to reconnect OLMS when receiving unregistered state.
  - Added new option to display 200 items per page in Processing History.
  - No longer trigger "Reuse processing result by hash" for files when all its type IDs are different.
  - Improved performance for exporting or importing an enormous list of hashes.
  - Retained deployment ID regardless of network interface card change, when the product is deployed on the same machine.
  - Sum up and display hash time of all directly nested child files.
  - Tracking first/last files received in batch and its timestamps.
- Enhancement for webhook use case including request receiving, webhook status storing and fetching, optimizing thread allocation.
- Security improvements
  - Upgraded bundle PostgreSQL to v14.15 for vulnerability fixes.
  - Upgraded NGINX to v1.26.2 for vulnerability fixes.
  - Upgraded 7z to v24.09 for vulnerability fixes.
  - Upgraded Npgsql to v4.0.17 for vulnerability fixes.
  - Applied patch to Qt for vulnerability fix.
  - Hardened OS command validation for External Scanner setup.
  - Improved login mechanism for SAML Single-SignOn.
  - Hardened regex validation.
- Engine integration
  - Support tab switching and breadcrumb in detailed result UI.
  - Populate tombstone flag to Archive Compression engine to enhance PST file sanitization.
  - New setting for Country of Origin engine to immediately block all unsupported file types.
- Docker enhancements
  - Eliminated unused components/libraries from DockerFile.
  - Gathered containerization information into support package.
- More detailed log messages for LDAP integration especially when not found a LDAP user.
- Added audit logs for Email Notification, Banner and Scan-from-link settings.
- New engine icons for Country of Origin, Adaptive Sandbox, and Threat Intelligence.

- Improved the login behavior to ensure that users can login even right after a successful product upgrade.
- Display extraction error code and details.
- Compression tab in workflow will be disabled when Deep CDR and Proactive DLP are both disabled. In addition, corresponding sections under Compression tab will be disabled depending on which engine is disabled.
- And many cosmestic UI changes.

Fixes:

- Addressed an issue that ignored filtering settings, causing Adaptive Sandbox to analyze sanitized or partially sanitized files that should not have been analyzed.
- Addressed an issue that caused detail view in Configuration History UI unresponsive after failed to import configuration.
- Addressed an issue that caused error "Invalid certificate responded" when using ForgeRock, KeyCloak, PingOne, and resulted in that user could not login to web console.
- Addressed an issue that was inproperly respecting "Fallback file type detection to current extension" setting for Proactive DLP.
- Addressed an issue that caused selected config file lost after clicking Enter at Password For Decryption textbox.
- Addressed an issue that was failing to extract encrypted archive file while integrated with Central Hub.
- Resolved an issue related to "Email Configuration" when using None as "Auth method" caused the setting to become misconfigured.
- Resolved an issue that might cause local user session repeatedly expired after its password was reset.
- Resolved an intermitten issue that caused file extension of downloaded file wrong.

Version v5.12.1

Release Date: 30 October 2024

New features:

- Support user login for nested AD groups
  - When enabling a new setting `Nested Group Login`, the product allows users who are indirect members of an AD group to log in and take on the corresponding roles and rights.
- MetaDefender Core now supports CIS Level 1 for RHEL 9. For more details, refer to CIS Level 1 - Linux Guidelines.
- Result display enhancements.
  - Introduces filtration options for the display of processing results, enabling users to quickly filter which file types violate the blocklist policy.
- Skip-by-hash enhancements
  - Boosts hash lookup performance.
  - Refines new REST API parameters for better functionality and usability.

  - Users can now download a CSV template for easy content input, simplifying data entry processes.

- Processing History enhancements
  - Ability to export history based on status `Allowed` or `Blocked`.
  - New column: `File size`.
  - New filtration option: by `Duration`.
- Selective analysis of files containing specific Active Content(s) with Adaptive Sandbox engine
  - Giving users more flexibility to analyze their own files with Adaptive Sandbox.
  - With this setting, only files containing specific Active Content(s) will be analyzed by Adaptive Sandbox engine.
  - Please note that this setting is only available if using Adaptive Sandbox version 2.1.0.
- Total size is sanitized and processed by Deep CDR in Executive Report.
- Usability enhancements / changes
  - Support new operator '**smaller than**' for file size condition in `Blocklist`& `Allowlist.`
  - REST API Fetch a list of blocked leaf files inside archive now includes AV engine name and detected threat name.
  - Addressed a case where Firefox browser sometimes incorrectly detects CSV file type, causing CSV uploading to fail with message '`File format is invalid. Please upload a CSV file.`'
  - Added more options for Data Retention of sanitized files and the Proactive DLP engine processed files.
  - Based on the override classification in `Workflow Rule` making blocking decision for the SBOM engine results.
  - Begin including engine state in export and import progress.
  - Support AD domain controller.
  - Removed PLPython from bundled PostgreSQL in Linux.
  - Improved validation for file scanning.
  - No longer deploy and initialize unsupported engine types.
  - Distribute SHA256 and file size data to the SBOM engine for analysis.
  - Move '`File download`' timeout setting from `MetaScan` tab to `General` tab.
  - Group settings '`Enabled file hash for only selected types`' and '`Skip Hash Calculation`'.
- Central Hub enhancements
  - Support a REST API to download digitally signed status reports for the entire batch. The specification is the same as Download Signed Batch Result.
  - Support a REST API to fetch the product version. The specification is the same as Get Product Version.
- Additional logging for the Single Sign-On (SSO) feature.
- Updated the UI to reflect engine state in case of inactivity.

- Ensured color consistency for table titles across UI modes.
- Implemented minor user interface refinements.

Fixes:

- Fixed an issue that caused the filename of the sanitized file to be assigned a different text than the workflow rule setting '`Output filename format`', when the '`Reuse processing result`' feature was enabled.
- Fixed an issue that caused a JSON attribute to be set incorrectly for encrypted archive files.
- Addressed a bug that caused newly supported file types to not be automatically checked after Deep CDR was updated.
- Improved an inconvenience with `Allowlist`/`Blocklist` settings where users were unable to save changes, and instead received an error message '`No modification detected`'.
- Fixed an issue that caused the `Revert`button to disappear sometimes.
- Fixed a problem that led to incorrect sanitization results due to a miscalculation of the tombstone files quantity.
- Fixed an issue that displayed an empty first page of `Module Update` history when no search criteria were specified.
- Fixed a potential data race in sending webhook result by multiple workers.
- Fixed an issue that caused activation to not be cleaned up properly in a Docker container.
- Fixed an issue that generated wrong database name when running in standalone mode with the parameter `IGNITION_JSON` in Docker container.
- Fixed an issue with Non-persistent mode that prevented the service from starting in Docker container.

Version v5.12.0

Release Date: 30 September 2024

New features:

- Email notification
- CSV upload for feature Skip-by-Hash
- Password storage for handling encrypted file
- By default, the workflow rule "MetaDefender Software Supply Chain" now includes SBOM license enforcement, enhancing compliance and security.
- Refines the user interface for Modules to enhance usability and streamline workflows.
- Now you can manage and configure "Post Actions", "External Scanners", "Webhook Authentication" and "Health Check" from My OPSWAT.
- Blocklist All Except treats child file types in separate ways
- Usability enhancements / changes:
  - Introduced a setting under Deep CDR in workflow rule to set Blocked to original file if sanitization is successful.
  - Optimized PostgreSQL query.

- Desktop shortcut of the product now can redirect to HTTPS.
- Users now can specify different workflow rules for files in a batch.
- Introduced an option to include vulnerability information into manifest file.
- An ability to export a specific date range from Processing History.
- Applied a better flow for Sandbox timeout handling.
- `outdated_data` now can reflect changes related to engine advanced settings.
- Display "Cancelled" for affected AV engines when client cancels a request in the middle.
- Rename Sandbox verdict "INFORMATIONAL" to "NO_THREAT".
- Added smaller options to Data Retention settings.
- Updated name of engines in Health Check settings.
- An ability to change file size unit in Blocklist and Allowlist.

- Security improvements
  - Enhanced key generation and destruction flow.
  - Improved login mechanism for SAML Single-SignOn.
  - Upgraded libxml2 to v2.12.9 for vulnerability fix.
  - Upgraded protobuf to v3.19.5 for vulnerability fix.

- Docker enhancements
  - Upgrade PostgreSQL to v15.8 for vulnerability fix.
  - Upgraded libldap to v2.5, python to v3.12.6 for vulnerability fixes.
  - Upgraded the preinstalled JRE to v17 in our Docker Images to support new requirement of Adaptive Sandbox v2.0.0+.

- Central Hub supports new header `scan-options`
- Made some settings' description more informative.
- Implemented minor user interface refinements.

Fixes:

- Fixed an upgrade issue when upgrading from v4.20.0
- Fixed an issue in container environment that product could not deactivate when PostgreSQL instance stopped or terminated.
- Fixed an issue that support package could not collect log files in Windows Server.
- Resolved an issue making temp files sometimes could not be removed when an engine crashed.
- Fixed an issue making obsolete functions for storing Sandbox and SBOM report left after an upgrade.
- Corrected Maximum and Total processed object size in PDF Executive Report.

Version v5.11.1

Release Date: 28 August 2024

New features:

- Support two standard formats for SBOM generation.
  - **SPDX** (Software Package Data Exchange): Developed by the Linux Foundation, SPDX is an open-source, machine-readable format designed to promote transparency and compliance in open-source and proprietary software management.
  - **CycloneDX** (CDX): Created by the OWASP community, CycloneDX is another open-source, machine-readable format that prioritizes ease of use and automation in SBOM generation throughout the software development lifecycle.
- New statistics in Executive Report
  - Introduces new settings to filter data by workflow, by user, by source in Executive Report.
  - There is an ability to switch statistics display on file or object.
- Introduces an updated system health dashboard that offers a new way to visualize scan queue and allocation.
- New scan-from-link settings
  - Introduces Max download queue setting which you can modify number of file downloads at a time for your use case and network. Also introduces pattern validation setting which users define blocklist and allowlist based on link pattern.

You can find these in Settings\Security tab. For more details: Scan-from-link Settings

- Usability enhancements / changes
  - Supports proxy for On-Premises License Management Server (OLMS).
  - Provides improved visibility for SAML SSO user directory settings.
  - Supports Windows 11, version 23H2.
  - Changes the display name for decrypted documents.
  - Allows users to export and import queue size settings.
  - Enhances the support package generation script to enable users to collect a variety of log files from newest to oldest. For more information, see How to Create Support Package With Bundle Tools?
  - Enables users to search for file types in Blocklist and Allowlist settings.
  - Exposes sanitized file sizes in the REST API, even when they are 0 bytes.
  - Allows users to switch between hours and days for setting the "Definition age threshold" under Workflow Rule \ Metascan.
  - Optimizes processing report synchronization for My OPSWAT.
  - Mode switching (from Standalone to Shared) during upgrade is no longer supported. Users are recommended to setup a Shared DB instance through a fresh installation instead.

- Adaptive Sandbox's supported archive file types
  - The Adaptive Sandbox engine now scans only supported archive file types that are selected by users in Workflow Rules.
  - This enhancement helps ensure that the Adaptive Sandbox engine focuses on processing relevant files, improving performance and overall system efficiency.
  - Nested files within supported archive file types will no longer be scanned by the Adaptive Sandbox engine.
- Security improvements
  - Implemented additional validations for administrative configurations and the scan-from-link feature.
  - Upgraded the bundled PostgreSQL database to version 14.13 for CVE-2024-7348.
  - Upgraded Qt to version 6.5.3.
  - Upgraded nghttp2 to version 1.61.0.
  - Upgraded curl to version 8.9.1 in the Docker image.
- Docker enhancements
  - Allows users to add license comments during activation.
    - Introduces a new optional parameter, MDCORE_LICENSE_COMMENT, type string.
  - Provides new settings for the On-Premises License Management Server (OLMS) proxy.
    - For more details: Docker run parameters
- Logging improvements
  - Optimized logging for unlicensing events.
  - Added more logs for synchronous operations.
- UI updates
  - Changed the YARA engine name.
  - Froze the header in the engine detail table.
  - Implemented minor user interface refinements.

Fixes:

- Database connection failure
  - Resolved a database connection issue that occurred after users upgraded to version 5.11.0. This issue did not affect all cases when upgrading to version 5.11.0. If the application was previously upgraded from version 5.5.1 (or older) to version 5.6.0 (or newer), this issue would occur when users upgraded to version 5.11.0.
- Fixes on product stability issues
  - Addressed an issue where the ometascan process consumed excessive memory when serving scan-from-link

requests.

- Resolved an issue that prevented users from searching Processing History for AD users.
- Fixed an issue that prevented header metadata from working with Hub.

- Implemented UI enhancements and resolved minor bugs.

Version v5.11.0

Release Date: 31 July 2024

New features:

- Multipart uploading support
  - This feature enhances upload efficiency and reliability for large data transfer.
  - **Seamless handling of large files**: Clients can now upload files seamlessly by breaking them into smaller parts.
  - **Simultaneous part uploads**: Clients can upload multiple parts of a file simultaneously, further boosting speed and efficiency of the upload process.
  - **Resumable part uploads**: In case of network interruption or even after a restart of MetaDefender Core service, clients can now resume part uploading from where it left off.
  - Multipart uploading support can work with Standalone mode and Shared mode. For Shared mode, client needs to ensure that parts of a file must be sent to 1 MetaDefender Core instance.
  - This mechanism is only compatible with Asynchronous scan, Batch scan and Webhook functionality.
  - Users can enable this mechanism in Workflow Rule and set a time-to-live for received parts.
  - Initiate multipart upload session: POST /file/multipart
  - Upload individual part of file to the conducted session: POST /file/multipart/{dataID}
  - Check uploading status: GET /file/multipart/{dataID}
  - Abort multipart uploading: DELETE /file/multipart/{dataID}
- Module Update settings and Data Retention settings are now managed in My OPSWAT product inventory.
  - After enrolled, these settings will be locked on MetaDefender Core management console, and users can only make changes on My OPSWAT console.
- My OPSWAT integration: PIN code required for configuration edit
- Central Hub offers the ability for client to cancel their scan requests submitted to MetaDefender Core instances behind it.
  - Single file and Archive file
  - Distributed archive file
  - Batch
- Queue size configuration

  - Size of the product queue (processing slots) now can be adjusted and defined on management console.
  - Note: setting this size too high could end up a potential risk of overloading MetaDefender Core and affecting the entire scanning service.

- New hash type: SHA-512
  - Introducing support for SHA-512 hash type for enhanced security.
  - You can toggle a new option in Workflow Rule to enable this new hash calculation.

- Definition age threshold rejection
  - This release introduces new settings for Metascan thresholds.
  - MetaDefender Core will reject file submission when there are too many outdated AV engines.
  - Error message in this case: "Failed to request scan. Insufficient number of up-to-date anti-malware engines"
  - You can find the settings in Workflow Rule > Metascan, and configure definition age (days) and number of outdated engines.

- Ability to display a custom notice during login process.
- Usability enhancements / changes
  - Non-persistent and Central Hub modes will not sync scan reports with My OPSWAT.
  - Optimize database partition creation for smoother performance.
  - Enhanced startup process to ensure service reliability, even with prolonged or failed temporary directory cleanup.
  - Added support for handling "General unextractable error" from Archive Extraction engine.
  - Expanded reuse of processing results by hash to include metadata header.
  - Corrected sanitization result for nested files in archives when reusing processing results by hash.
  - Improved backup method for database configuration files.
  - Accelerated Data Retention for Processing History.
  - Prevent engine initialization during paused update time when downloaded packages exist.
  - Added filter for batch scans in Processing History.
  - Introduced "Skipped assessment" verdict for File-based Vulnerability engine when "Skip hash calculation" is enabled in workflow rule.
  - Restructure workflow settings for Filetype engine.

- Performance improvements
  - Optimized ID generation process.
  - Enhanced report synchronization for improved performance impact when enrolled in My OPSWAT.
  - Improved dedicated slots allocation for efficient archive file scanning.
  - Standardized user and license validation for reduced costs and processing time.

- Security improvements
  - Implemented additional validations for administrative configurations.
  - Upgraded bundled PostgreSQL authentication method scram-sha-256 for enhanced security.
- PostgreSQL 15 support (remotely)
  - MetaDefender Core starts supporting remote PostgreSQL database v15, and now can work with PostgreSQL either v14 or v15.7
  - Product does not perform any version upgrade on customer's remote PostgreSQL.
  - Note: The bundled PostgreSQL version remains at v14.11 in this release.
- Prepared v15.7 PostgreSQL CLI tools.
- Logging improvements
  - Expanded logging for "Pause scheduled updates" feature.
  - Incorporated rootDataId into Hub log messages.
  - Implemented Workflow deletion tracking in logs.
  - Increased logging for the cleaning process during service startup.
  - Optimized log writing to reduce redundancy.
- Enhanced UI accessibility across multiple areas.

Fixes:

- Resolved an issue where temporary files were not cleaned up when the Archive Extraction crashed before MetaDefender Core could send a stop task.
- Fixes on product stability issues
  - Corrected discrepancy between actual and file submission counts on the Dashboard.
  - Resolved a service crash issue occurring during its restart after My OPSWAT enrollment.
  - Fixed missing "=== END ===" long marker on Debian when stopping the service.
  - Addressed inaccurate processed objects counts in Executive Report and Processing.
  - Correct pagination errors when adjusting the items per page.
  - Resolved reuse processing result by hash history incompatibility with URL encoding metadata headers.
  - Eliminated leftover files after import failures.
  - Resolved performance degradation in shared DB mode that occurred when more than 4 instances were running concurrently, particularly affecting handling of high traffic loads.
- Implemented UI enhancements and resolved minor bugs.

Version v5.10.1

Release Date: 27 June 2024

New features:

- Upgrade data at-rest encryption algorithm
  - Improved security by upgrading the at-rest encryption algorithm to AES (Advanced Encryption Standard).
  - Sensitive data stored in the system will now be protected using AES encryption algorithm, enhancing the confidentiality and integrity.
- Add new OS support for Ubuntu 22.04
- New SBOM detection mechanism
  - SBOM technology introduces enhanced capabilities for analyzing and detecting license issues to gain deeper insights into the licensing status of the software components they use, ensuring compliance and minimizing legal risks.
- Usability enhancements / changes
  - Splitting the waiting time for the Adaptive Sandbox engine from the "Others" time to provides better visibility into the Adaptive Sandbox analysis process, enabling users to identify and address any scan time effectively.
  - Users can now perform a look-up of quarantined files using hash values.
  - Extended the support for the API download quarantine functionality by accepting input SHA256 hash values as uppercase.
  - Displaying a "No License" message when enrolling MD Core to My OPSAWT without license.
- Performance improvements
  - Enhanced a case that dedicated slots of scan queue might not be applied when scanning archive files.
  - Standardized user validation and license validation to reduce the cost and time.
- Able to set External Scanner as mandatory engine
- Introducing a new workflow rule for streamlining the integration with the product MetaDefender Software Supply Chain.
- Upgraded to OpenLDAP v2.5.17 for vulnerability fix.
- Docker environment: Able to upgrade the application with a database password containing specific characters
- Cloud K8S environment: Skip checking available disk space on database instance when lacking of super-user permission
- Reducing size of engineprocess logs.

Fixes:

- Improved feature Reuse Processing Result by Hash
- Addressed an issue where tasks were not being released properly when they timed out on engine side.
- Addressed an issue that in certain circumstances, when MD Core sent a stop task, but Archive engine had not finished opening the corresponding file, temporary files in the Archive Extraction temp folder were not cleaned up

by the engine.

- Fixed an issue by ensuring that even if MD Core has realized a timeout and removed the stop extraction task, the response from Archive engine will still be processed correctly, and the associated temporary file will be removed accordingly.
- Resolved a corner case where task_id of next and stop tasks were duplicated, leading to the retention of the temporary file.
- In high-load situations, when Archive engine received a next task and proceeded to extract a file, but MD Core experienced a timeout while waiting for a response, later responses from Archive engine regarding successfully extracted files will still be processed and acknowledged to ensure proper cleanup of extracted files.
- Fixes on product stability issues
  - Addressed an issue where upgrading from older versions to version 5.7.1 or newer was not possible when using non-persistent mode in Linux environments.
  - Fixed an issue with the rotation of engine log files. It will now rotate correctly according to the configured rotation settings.
  - Resolved an issue where scan results were not updating correctly when canceling scans that involved External Scanners.
  - When using the Accessibility mode, there was a UI issue where only one AV engine appeared on the Modules screen, instead of displaying all available AV engines.
  - Fixed an issue where Webhook Authentication would fail after upgrading the system.
- Identified a memory leak issue where abandoned resources were not cleaned up when a request timed out in the queue.
- Fixes on My OPSWAT integration
  - Could not save configuration on My OPSWAT UI when session lifetime almost pasts.
  - Logs were flood with numerous "Config was changed on OCM" message when license is changed.
- Fixes on Non-Persistent mode
  - Failure to logging in with SSO.
  - Could not import configuration with v2 method.
  - Failed to add Webhook Certificate, External Scanner and Post Action.
  - Several redundant error logs appear when using this mode.
- Central Hub, unabled to create batch: Fixed an issue that client could not create batch to Central Hub

Version v5.10.0

Release Date: 21 May 2024

New features:

- Continuous support for My OPSWAT and Central Management v8 integration

  - Workflow rules configurations.
  - Product license activation.
  - Engine module update.
  - Processing history browsing.
  - Processing details of each scan.
  - Dashboard overview with processing statistics.
- Sandbox engine technology name updates
  - "Sandbox" engine is now displayed as "Adaptive Sandbox".
  - New filtration setting to allow triggering Adaptive Sandbox engine to run when hitting "Blocked Verdict by Deep CDR" result determined by Deep CDR.
- The default workflow rule for “MetaDefender for Secure Storage” is now renamed to "MetaDefender Storage Security".
- Managing and sharing YARA rules has never been easier with new export and import functionality.
- Security improvements
  - Upgraded to Cyrus SASL 2.1.27 for vulnerabilities.
  - Upgraded to libpng 1.6.43 for vulnerabilities.
  - Upgraded to zlib 1.3.1 for vulnerabilities.
- Significantly improved the base64 decoding performance.
- Usability enhancements / changes
  - Allow setting Multiscanning AV related timeout settings less than 1 minute (in seconds), applicable to both "Per AV" and "Global timeout" settings. ***Note***: upgrade process will convert 0 (min) value to 1 min.
  - Make "others_time" more granular and precise in case of processing archives.
  - Retry mechanism with License Manager to handle any potential unstable network issue better.
  - Distinguish Multiscanning AV's "wait_time" and "scan_time" in case of scan timeout.
  - Separate Multiscanning AV's wait_time from "others_time".
  - No longer flag Reputation engine's database outdated.
  - Default non-admin roles now can be deleted completely.
- Performance improvements
  - Optimized algorithm to boost parallel performance when functionning with 20+ Multiscanning engines.
  - Resolved slow responsing and reducing memory usage for file submission.
  - Optimized communication channel between MetaDefender Core service and its engineprocesses for product resilience under load.
  - Improved a mechanism to ensure engine initialization can be executed even when Windows ever runs out of

resources.

- Applied thread-pool mechanism in task management to reduce resource consumption and improve file processing performance.
- Optimized workflow logic to maximize the utilization of system resources.

- Containerization improvements
  - Supported non-root use case.
  - Supported to start shared-DB instances in a parallel way when using an internal (bundled) PostgreSQL user.
  - Supported YARA rule settings import.

- Logging improvements
  - Logs now can be rorated by day and size, at 1GB.
  - Data retention: keeping the last 30 days of log files.
  - Standardized key format in log file: "data_id".
  - Distinguished "Unexpected server response" from "Invalid deployment ID" error.
  - Ensure log message from Multiscanning engines should not mess up product log format.
  - Track timestamps of first chunk received and last chunk received in millisecond.
  - Track which AV engine takes most scan time (max_scan_time) and wait time (max_wait_time).

- Change the term "Waiting child files" to "Nested files".

Fixes:

- Addressed leftover temporary file issue in certain circumstances under high load.
  - When users cancel a request or terminate a connection in the middle, temp files of the cancelled/terminated requests could be left over.
  - When users cancel an archive file processing which is being extracted, it causes temp child files might be remaining while root file can be removed in advance.
  - When there are numerous archive timeouts, then required further commands cannot be sent to Archive Extraction engine for temporary file cleanup.

- Fixed issues caused by archive engine timeout and threshold exceeding that negatively impacted performance, and potentially caused memory leak issue.
- Fix issue that Multiscanning engines sometimes cannot active back after a timeout or failure termination.
- Status code and response body does not reflect Synchronous Scan timeout result.
- Fix issue that support package script overwrited new-location logs with the default-location logs.
- MetaDefender Core container cannot start when UID and GID are different.
- Fix issue that MetaDefender Core container could not auto deactivate.
- Multiscanning exclusion setting did not work properly when configured engine(s) are "Inactive".

Version v5.9.0

Release Date: 21 March 2024

New features:

- Reputation engine integration
  - The OPSWAT Reputation Engine enables instant threat identification by comparing hashes against a growing database cataloging known good and bad files.
  - The Reputation Engine utilizes hash reputation based on advanced analysis like metadata evaluation, content inspection, and contextual correlation to minimize false positives. This allows accurate threat detection with limited business disruption. Through database updates and hash contributions, the Reputation Engine enhances protection against new and evolving attacks.
  - In MetaDefender Core, customers need to opt-in to enable the Reputation Engine through the workflow management settings.
  - Learn more details at Overview
  - ***Warning:*** *The Reputation engine will be automatically added into all existing and new customers' license to fulfill our security protection committed to our customers. That means the engine will be downloaded and deployed when applicable, but users must opt-in to enable the engine's processing under MetaDefender Core workflow rule.*
  - *Without taking any action to enable it, this engine deployment should not cause any impact to our customers integration and scanning service. __*
- Country of Origin (COO) engine integration
  - The OPSWAT Country of Origin (COO) engine enables instant detection of a file's geographic source. It analyzes digital fingerprints and metadata to identify restricted locations and vendors automatically. This allows precise filtering that blocks unauthorized data access based on origin while ensuring cross-region regulatory compliance.
  - The COO engine provides organizations with visibility into file upload origins. With precise data origin insights, organizations can streamline audits, protect sensitive data, and prevent malicious file uploads. This enhances security and compliance at reduced cost.
  - By combining advanced technologies like Deep Content Disarm and Reconstruction (Deep CDR), Multiscanning, and Proactive Data Loss Prevention with origin-based access controls, organizations can secure intellectual property, maintain compliance, and block illegal or non-compliant foreign data. This mitigates breach risks and avoids substantial fines per regulations like GDPR.
  - The COO engine supports signed file types like PE files, MSI, and Self-extract.
  - Learn more details at Country of Origin
  - ***Warning:*** *For those customers who using older version of MetaDefender Core where we provided a feature for allowlisting files based on vendor, then those settings will be migrated to COO engine's settings under MetaDefender Core workflow rule on the latest version.*
  - *In that case (applicable to whom already using existing older version of MetaDefender Core's allowlisting file based on vendor), COO engine is not licensed by default for customers, so please reach out to OPSWAT to acquire license for COO engine and use that license* **before _ upgrading_** *__your MetaDefender Core to ensure settings*

migration to be made properly. __

- *Once upgraded, exppecting that COO engine will be displayed on MetaDefender Core inventory page, but inactive. It requires users to disable COO engine and re-enable it back. All previous allowlisting file based on vendor's settings will be then found under "Country Of Origin" tab in the corresponding workflow rule. __*

- Windows Server 2012 discontinued support
  - Windows Server 2012 has been out of support by the vendor (Microsoft) since Oct 2023 - Reference: https://learn.microsoft.com/en-us/lifecycle/announcements/windows-server-2012-r2-end-of-support
  - MetaDefender Core version 5.9.0 will no longer support this OS version, and recommend our customers migrating their systems to newer supported Windows Server version.
  - List of current supported Windows OS versions can be found at Supported Windows Operating Systems.

- API key protection
  - Strengthening MetaDefender Core's security by protecting users' API key (if created) in the database using SHA256 hashing. Then actual API key value could not be retrieved back for good.
  - By default, this feature is not enabled for backward compatibility. Administrators need to opt-in by enabling that. Expecting to see the following UI notification poped up upon signed in:
  - ***Warning:*** *Once enabled, administrators cannot turn back the feature for security enforcement. Before switching on this feature, it's strongly recommended to backup your users' API key in a secured vault / secrets manager.*
  - When this feature is enabled, API key will be hashed out in the database, and no one, including administrators, will be able to see actual API key value anywhere in the product.
  - In case API key is lost or not remembered, then administrators must re-create a new one. The new one will be displayed for only one time on the UI.
  - ***Note:*** *__Nothing should be changed from client side integration, client should keep using actual API key in plain text for REST API calls. __*

- PostgreSQL 14.11 supported
  - That also includes PosgreSQL 14.11 bundled with MetaDefender Core for local bundle database deployment. This upgrade addressed PostgreSQL own vulnerabilities in older versions, and helps protect our customers from supply chain attack.

- "Block all accept" under "Blocklist by file name"
  - Supporting a new option to allow users to block all except certain things against file name.

- Central Hub architecture updates
  - Distributed archive compression is supported.
  - Metadata header is supported for Proactive DLP while running in Central Hub
  - Country of Origin and Reputation engines are supported.
  - Secure API key is supported.
  - Including instance name in status response API.

- Enhanced the organization of files in an archive result and a batch result by sorting based on statuses and verdicts.
- Optimized batch processing in Central Hub.
- Optimized request header processing in Central Hub.
- Upgraded to OpenSSL 3.2.0 for vulnerabilities.
- Upgraded to Hiredis 1.2.0 for vulnerabilities.
- Upgraded to Nghttp2 1.58.0 for vulnerabilities.
- Upgraded to rabbitmq-c 0.13.0 for performance.

- Security improvements
  - New option to disable user information caching via using designated administrative REST API endpoint (PUT /admin/config/cacheuser). Details at MetaDefender Core API
  - Upgraded to PostgreSQL 14.11 bundled with the product for vulnerabilities.
  - Upgraded to OpenSSL 3.2.0 bundled with the product for vulnerabilities.
  - Handled URL encoded characters.
  - Removed Lodash version in product JS file.
  - Addressed minor potential security related issues.

- Usability enhancements / changes
  - Updated default parallel count setting value (maximum number of threads) for Sandbox engine to 5 (formerly 20).
  - Enhanced engine deintialization mechanism to ensure all dependent engine process(es) cleaned up and stopped gradually.
  - Prioritized to list and display all blocked nested files inside batch first over allowed items.
  - Supported to process scan request with local file path contains double backslash characters.
  - Changed the Content-Type header value for POST /file and POST /admin/update/upload (old: application/json; charset=utf-8 - new: application/octet-stream)
  - When setting value for global/tempdirectory key for MetaDefender Core temporary folder, if that folder does not exist, now the product no longer automatically create temporary folder. Instead expecting users to have that folder created upfront and set permission (write) on that folder.
  - Prevented the product upgrade proceeding when insufficient disk space is detected. Faiure reason will be caught in the product log when encountered.

- Performance improvements
  - Optimized PUT - Modify 'skip by hash' list to process a large volume of hashes data - up to hundreds of thousands of entries, which is not possible before the optimization.

- Logging improvements

- Added more information into the product log in the event of scan submission was blocked due to out of workflow time window (included filepath, filename, callbackurl, sanitizeurl, downloadfrom)
- Added data ID of root archive level to the log in the event of any child files of that root archive, regardless of any extraction level, is done with processing.
- Optimized to trim down unnecessary data captured in the product audit log when users modified a configuration comes with large content e.g. large list of hashes to be skipped.
- On Linux OS, supported log rotation for NGINX web server by MetaDefender Core built-in mechanism without relying on logrotate service on Linux OS.
- Added deployment ID into the product log in some relevant licensing events.
- Provided data retention setting for PostgreSQL bundled logs, by default: 1 month, and can be adjusted up to 12 months.

- UI updates
  - Display processing time details
  - Child (nested) checkboxes are no longer automatically checked/unchecked when their parent checkboxes are.

Fixes:

- Failed to process signed file via local file path mode (using filepath header).
- Failed to start MetaDefender Core service when using log overriden with multiple entries separated by semicolon.
- Mismatch values of Percentage column between the executive report PDF and actual product UI.
- Engines failed to re-download back when removing them while downloading definition packages.
- Failed to export Sandbox result JSON report on the UI with 403 error.
- Failed to update Deep CDR settings on Central Management 7.
- Scan could be hung while being processed with batch in Central Hub architecture.
- MetaDefender Core service could be crashed when using distributed load extraction with file size filtration setting in Central Hub architecture.

Version v5.8.0

Release Date: 20 December 2023

New features:

- Allow post action to override final processing verdict
  - The new setting under "Post Action" section in MetaDefender Core workflow rule (disabled by default) will let authorized users configure the product to scan for the pre-defined final output code returned by your custom post action script (or custom post action webhook server) to determine if the final processing verdict should be overriden as "blocked" by that or not.
  - This new setting create an opportunity for MetaDefender Core customers to customize even better their security policy with their own action's logic.

- Reuse processing result by hash
  - The new setting under "General" section in MetaDefender Core workflow rule (disabled by default) will let authorized users configure the product to automatically reuse eligible processed results with the same hash for all other in-progress submitted requests.
  - A great approach to tremendously boost up the entire processing throughput for certain use-cases such as Email Security, DevSecOps, Cross Domain etc. to avoid repeatedly processing same files within a short time period.
  - Learn more at Reuse scan results for the same hash
- Support Red Hat Enterprise 9
  - Enrich supported OS porfolio in MetaDefender Core with new support for Red Hat Enterprise 9.
  - **Note:** For Deep CDR engine, it is required to install all neccessary libraries mentioned in .NET 5 dependencies on RHEL 9
- Support PST archive file sanitization
  - When enabled, MetaDefender Core will remove infected and encryted child files inside original PST file, and create a sanitized version of PST without them.
- Prevent potential brute force attack with forgot-password entries by applying cool down period.
- Improve LOCAL scan mode on MetaDefender Core (by using scan request header `filepath`) to prevent unpredicted behavior when processed files are altered (moved / deleted), but they appear in many other different local locations. Beneficial for File upload or Cross domain use-cases.
- Allow you to track hostname of endpoint machine, and client side's logged-on user information on each MetaDefender Core scan session via using `current_logged_on_user` and `hostname`key under metadata header respectively, and this information will be displayed under MetaDefender Core processing history UI at "metadata" column (hidden by default).
  - For example, you can specify the information for client endpoint's host name and current logged-on user by writing those information in your scan request's header `metadata`as following: `{ "hostname": "<hostname>", "current_logged_on_user": "<whoami>" }`
- Set value of 0 (zero) to scan queue threshold under MetaDefender Core health check now means that, MetaDefender Core will disable checking scan queue status as an input for its health check logic (Legacy previous logic: setting 0 means when no current on-going scans, then it will meet the condition for heath check logic).
- Support URL encode for `metadata` header applied in file or batch processing request.
- Track the last login time for local and SSO user when they login via `POST /login` request.
- Include host name information via using new header `X-Core-Id` of the response HTTP 503 (scan queue is full) returned to `POST /file` request.
- Disable (uncheck) Deep CDR advanced setting `Remove conditional comment` belongs to HTML under MetaDefender Email Gateway Security workflow rule due to Outlook HTML parser limitations.
- Performance improvement: Cache user session information in memory for user validation in every applicable scan request, so MetaDefender Core can skip database reading and improve processing speed.

- Logging improvements
  - Log all SQL statements which took longer time than expecterd in bundled PostgreSQL database log.
  - Enhance MetaDefender Core log for the scan-from-download-link feature to write more information for both sucessful (HTTP 200) and unsucessful (non 200 HTTP) circumstances.
  - Include details of each engine process (path, process ID, working-set size) running under MetaDefender Core in the product support package.
- UI notification when MetaDefender Core configurations are locked due to being managed by Central Management.

Fixes:

- Import configuration with big amount of blocklist rules could freeze up MetaDefender Core.
- Quarantined files could be corrupted and unrecoverable under certain circumstance.
- MetaDefender Core randomly crashed at some points when running under "distributed extraction" mode as a part of Central Hub architecture.
- In K8S deployment, random MetaDefender Core instance restarted when starting all available MetaDefender Core instances at the same time in shared and remote PostgreSQL database mode.
- In docker mode, MetaDefender Core cannot start when making change on environmental variable `CORE_DATA _PATH` without attached volume.
- Some UI cosmetics and minor bugs are addressed.

Version v5.7.1

Release Date: 09 November 2023

New features:

- MetaDefender Core supports to receive engine update packages from Central Management v8.
- Custom time range for Executive Report.
- Upgraded bundled PostgreSQL to version 14.9 for vulnerability (CVE-2023-39417).
- Improved the integration between MetaDefender Core and Deep CDR to avoid wasting system resource working on timed-out tasks.
- Change Deep CDR retry count default value to 1.
- Introduce Sandbox engine file size limit.
- User password minimum length enforcement from 8 to 30 characters.
- Whenever an user modifies UI settings such as time range or processing history columns, the modified setting value will be remembered and retained for the next user login, without having to re-select again.
- Other blocked reasons added for batch processing.
- Decrypted time in the JSON scan result.
- Logging improvements.

Fixes:

- Submiting gzipped file failed with "Content-Encoding: gzip" if the file size is a power of 2 (2^n) bytes.
- Password protection was not retained when scanning a file without file name.
- Wrong status displayed for Threat Intelligence when processing a file bigger than 140 MB and timed out.
- Installation failed on Red Hat Enterprise 8 due to missing wkhtmltopdf library.
- The number of sanitization tasks might exceed than allowed threshold under high load.
- Failed to close large batch when file type detection timed out on files in batch.
- Temporary files generated by Deep CDR and Proactive DLP engines were not cleaned up properly when cancelling scan request.
- Failed to export scan result PDF report on Red Hat Enterprise.

Version v5.7.0

Release Date: 09 October 2023

New features:

- SBOM engine integration
- Upgraded NGINX to version 1.22.1 for vulnerability remediation
- Central Hub enhancements
  - Version 2 for MetaDefender Core NAS (file storage server) that supports continuous file streaming directly to MetaDefender Core instance over NAS (without waiting for file streaming to finish at NAS first), and data encryption capacity.
  - Optimized processing for archive extraction load distribution.
- Optimized scan result polling query (GET /file API) for faster response
- New scan results when engine stopped executing due to hitting global processing timeout
- Separated processing time for archive compression
- Separated processing time for external scanner
- Various improvements in MetaDefender Core log for a better visibility when processing files and troubleshooting.
- UI updates
  - Continuous improvements for UI accessibility for WCAG 2.0 and 2.1 level A, AA compliance.
  - Hash comparison after sanitization
  - Search processing results based on Data ID, using the existing search box at the top of the product management console.
  - In-product tagline, and other UI cosmetic updates.

Fixes:

- Deep CDR engine failed to install and became permanently failed when MetaDefender Core contains a lot of custom worflow rules.
- File processing could hang up when timing out during hash lookup with File Reputation.
- DLP result of archive file was "No Specific Configuration" instead of "Unsupported File Type" when containing allowed file.
- "Not scanned" returned by AV engines when processing archive file but disabling extraction mode.
- "Others_time" in the JSON scan result was mistakenly set to negative when a file is quarantined due to global processing timeout exceeded.
- "start_time" in the JSON scan result was set to 1970-01-01 for in-progress file processing.

Version v5.6.1

Release Date: 28 August 2023

New features:

- Integration with My OPSWAT portal
  - Get ready for My OPSWAT integration where MetaDefender Core can connect and be managed in My OPSWAT product inventory.
  - Stay tuned for My OPSWAT further updates (end of September) to get hold of this integration on My OPSWAT.
- Continuous improvement on UI accessibility
  - Various UI updates and remediations for UI accessibility support.
- New filtration sub-setting for file type detection fallback to extension
  - The new setting is under MetaDefender Core workflow - File Type tab. This new setting can be only enabled when the existing parent setting "Fallback file type detection to current extension if needed" is checked.
  - When enabled, MetaDefender Core will avoid using current file extension as a fallback when actual file type is not enabled for Deep CDR engine's setting.
- More comprehensive details for processing time in the JSON scan result
  - Introduce a new key `post_action_time` (separated from `others_time`) for actual time spent on post action (if applicable). This key is under `processing_time_details` key in the JSON scan result.
  - Introduce a new key `cdr_wait_time` to determine actual time spent when request is sitting in queue waiting for Deep CDR available to start processing. This new key is under `sanitization_details` in the JSON scan result.
  - Introduce new keys `cdr_storage_time`/ `dlp_storage_time`/ `quarantine_storage_time`to determine actual time spent on output file being saved in storage for sanitization / sensitive data processing / quarantine processing (applicable when Deep CDR, Proactive DLP and Quarantine is enabled for that scan). These new keys are under `processing_time_details`key in the JSON scan result.
  - Applicable to all of those above new processing time, we also support for data statistics for admins to view under Executive report UI when needed.

- Enhancement on database connection pool
  - Revamped back-end logic for database connection pool to mitigate potential bottleneck on database queries under extreme load.

Fixes:

- Sometimes failed to close batch under high load and problematic circumstances with files inside batch.
- Batch signature was not generated when batch contains hundreds thousands of files inside.
- Failed to send callback for sanitizeurl webhook requests.
- MetaDefender Core could rarely crash when repeatedly processing .docx large files.
- Temporary files could not be cleaned up when hitting global timeout.
- Memory increasing abnormally and leaking with data statistics against PostgreSQL.
- Other minor stability issues.

Version v5.6.0

Release Date: 14 July 2023

New features:

- Support PostgreSQL 14.8
  - MetaDefender Core now also supports PostgreSQL 14 (verified with 14.8).
  - Bundled PostgreSQL server comes with MetaDefender Core standalone deployment also comes with this new version.
  - **NOTE:** *For MetaDefender Core upgrade from older version 5.5.1 or older (using PostgreSQL 12) to version 5.6.0 (using PostgreSQL 14), it is required to run MetaDefender Core installer as admin privileges to ensure the database migration task finish successfully.*
- Priority for workflow
  - Provide better option for Quality of Service on workflow, MetaDefender Core now supports users to define certain priority for each workflow rule. This enhancement will help users control better business priority in file processing.
  - You can find this new setting under "General" workflow tab.
- Time availability for workflow
  - Only allow scan requests using certain workflow during pre-defined time windows. You can find this new setting under "General" workflow tab.
- Global processing timeout per workflow
  - MetaDefender Core provides one global timeout for entire processing applied to every scan request. This is expected to help client integration control its SLA for every scan at ease.

- This new setting could be configured separately for each workflow rule.
- When a scan request is timed out by this global processing timeout setting, the scan request will be blocked with "Global timeout exceeded" as blocked reason.
- **NOTE:** *This global processing timeout, when enabled, it should be greater or equal to the sum of all other engine related timeout settings such as Metascan timeout, Deep CDR timeout etc.*
- You can find this new setting under "General" workflow tab.

- Override global processing timeout for specific scan requests
  - Client also can override the new setting of global processing timeout per specific scan requests when needed, by using a new request header `global-timeout` for the file submission request.
  - Details: POST - Analyze File (Asynchronous mode)
  - If this header is not set, or its value is empty, then MetaDefender Core will use the pre-defined global processing timeout in corresponding workflow rule (only when enabled).

- Hash lookup supported for in-progress scan
  - Before having to submit another scan request for a same file to MetaDefender Core and duplicate the effort of scanning, client could now utilize the hash lookup API GET - Fetch Analysis Result By Hash to be acknowledged that a same file is actually being processed, so client could waive the same request to MetaDefender Core.
  - Introduce a new header `include-inprogress` for GET - Fetch Analysis Result By Hash to indicate whether or not client would want to receive "In-progress" (instead of not found result) in such circumstance.

- Allowlist by vendor filtration
  - Allow users to create a filter on MetaDefender Core to add PE (Portable Executable) files tied to certain vendors into the allowlist.
  - Furthermore, users can be supported to block those PE files when unsatisfied the pre-defined conditions. You can find this new setting under "Allowlist" workflow tab.

- Continuous improvement for Sandbox (formerly OPSWAT Filescan) engine integration
  - Rename "OPSWAT Filescan" engine to "Sandbox".
  - Add `extraction_info` to the scan result JSON of GET - Fetch analysis reports containing all files in archive
  - Adjust default enabled file types for Sandbox file processing under workflow:
  - Remote Sandbox: all file types are selected by default.
  - Local Sandbox: all file types except archive are selected by default.

- New option to skip sanitizing empty files
  - When enabled, MetaDefender Core will avoid sanitizing empty file (zero bytes). By default, it is disabled. You can find this new setting under "Deep CDR" workflow tab.

- Attach MetaDefender Core server information in scan result fetching response
  - When enabled via using a new supported API endpoint PUT - Update list of custom response headers,

MetaDefender Core will return allowable server information (deployment ID, server IP address and port) in designated header `X-Core-Id` in the scan result fetching response (e.g. GET /file/<data_id>).

- This new API endpoint also supports users to only return those pre-defined data to authenticated scan requests (where uses API key in the request).

- Throttling option for scanning webhook
  - Support new parameters `workercount`, `requestqueue` in the existing API endpoint PUT - Webhook set configuration to define better flexibility for scan webhook:
    - workercount: the total number of workers for the webhook responser (for sending callbacks to the webhook client). Default is 1. **Note:** a service restart is required for changes to take effect.
    - requestqueue: the maximum number of requests allowed for each worker. Default is 100.
  - Besides that, now users can set zero (0) for the existing parameter transfertimeout for the PUT - Webhook set configuration, that means, disable timeout for sending callbacks to webhook client.
  - **NOTE:** *These throttling options only apply to webhook mechanism that notify processing result, do not apply to the mechanism where MetaDefender Core sends sanitized file back to the client just yet.*

- Enhance sorting feature in processing history UI
  - Enhance the existing feature to allow sorting entire scan history across all pages (previously only allow sorting current UI page on MetaDefender Core version 5.5.1).

- Continuous improvement for UI accessibility
  - Address issues and enhance GUI for better accessibility readiness.

- Display file size and file type even when file is being processed
  - On processing result UI screen. Including when the file is being processed, or already finished.

- Display setting for scan duration on the UI
  - Allow users to specify preferred time format for scan duration to be displayed on the UI.

- Other UI enhancements
  - Move max file size setting from "Metascan" tab to "General" time in workflow.
  - Update to rename and logo from Cyren (AV engine) to Varist.
  - Other enhanced UI cosmetics.

Fixed:

- Callback responses got stuck
  - Under high load, MetaDefender Core could fail to send callback responses to webhook client.

- Health check API returned out of date data

  - Under high load, the heath check API GET - Get health check status could return out-of-date data for scan queue and engine status.

- Inconsistent scan result with Threat Intelligence under Quarantine
  - Scanning with Threat Intelligence under Quarantine could return inconsistent result.
- Leftover temporary files when timing out with Sandbox (formerly OPSWAT Filescan) engine
  - When a scan is done with timeout for Sandbox (formerly OPSWAT Filescan), temporary file was not cleaned up properly.
- Other minor bug fixes

Version v5.5.1

Release Date: 18 May 2023

New features:

- Continuous improvements on OPSWAT Filescan engine integration
  - New configuration to trigger OPSWAT Filescan processing when number of scan failures from AV engines' results exceeded certain amount of threshold.
  - Behavior changes: skip sending extracted child files of MS Office files to OPSWAT Filescan engine for processing.
  - Fixed some minor bugs related to UI display.
- Blocking file processing based on Deep CDR forensic analysis policy
  - New configuration to allow authorized users to block file processing based on Deep CDR forensic analysis policy. The policy will be managed and defined under Deep CDR engine advance settings.
  - ***Requires Deep CDR 6.6.0 or above (to be released at the end of Q2-2023).***
- Workflow rule cloning
  - Similar to workflow template, now allowing authorized users to clone / duplicate existing workflow rule.
- UI accessibility continuous improvements
  - Enormous changes and improvements towards UI accessibility support. This UI accessibility mode can be enabled on the top section of every UI page.
- Auto enabling any newly supported file types for sanitization
  - New setting to allow MetaDefender Core to auto enable any file option that newly supported by Deep CDR for sanitization.
  - This option is unchecked by default to keep backward compatibility. You can enable it under Deep CDR tab in workflow rule > Enable for file types section.

- New attribute added into existing key `extraction_info` in the JSON scan result structure
  - is_encrypted_file (optional, only accept "true"/"false"): a flag to indicate a file is encrypted or not.
  - See more details on GET - Fetch Analysis Result
- Handling MSG file with archive sanitization use-case
  - When enabling "Block unsupported file type" under Deep CDR section, then sanitization on MSG file failed due to msg.data file (blocked as unsupported).
  - This change on MetaDefender Core to avoid that situation, and instead let the archive compression engine take care of the MSG archive sanitization properly.
  - ***Requires archive compression engine version 6.2.3 or above.***
- Optimizing memory consumption by ometascan service under load
  - Possibly helped reduce up to 50% memory consumption with ometascan service.
- Sorting columns in table under processing history UI
  - Currently supported users to sort processing history UI based on start time and scan duration.
- Adding parsing digital signature processing time
  - New key named `parse_dgsg_time` added into the existing `processing_time_details` __to indicate actual processing time spent on parsing digital signature against file.
  - See more details on GET - Fetch Analysis Result
- Support retry configurations for webhook post action
  - Users now can customize retry mode and timeout setting for webhook post action in Workflow Rule.
- Including current process name, ram usage, cpu usage into the support package
  - New data will be collected and stored in new CSV files located in the support package.
- Improve query time for filter labels API
  - Helped improve search response time on the processing history UI page.
- UI enhancements

Fixed:

- Handle processing properly with content-encoding of gzip or x-gzip
  - When submitting file request, if mistakenly specified Content-Encoding header value = gzip or x-gzip, MetaDefender Core returned HTTP 500 error with unexpected JSON output.
  - For example: {"err":"Invalid gzipped body"}{"data_id":"a3179bc183a6494694768c315a21867d"}
  - Now addressed the issue to let MetaDefender Core return HTTP 500 error with proper JSON output, for example: {"err":"Invalid gzipped body"}

- Timeout occurred with synchronous submission requests
  - Addressed the issue when MetaDefender Core returned scan timeout immediately against synchronous scan requests.
- URL encoded "rule" header in scan request
- - MetaDefender Core did not decode values in the "rule" header when it is URL encoded (for example: "rule" = "File%20process")

Version v5.5.0

Release Date: 06 April 2023

New features:

- OPSWAT Filescan integration
  - OPSWAT Filescan is an unique adaptive threat analysis technology, enabling zero-day malware detection and more Indicator of Compromise (IOCs) extraction. Learn more at https://docs.opswat.com/filescan
  - MetaDefender Core now fully integrate with OPSWAT Filescan (as an engine module), and provide two separate engines:
    - **OPSWAT Filescan Embedded engine:** bundled in MetaDefender Core server, and the engine will process files locally itself.
    - **OPSWAT Filescan Remote engine:** the engine is supposed to send file requests to another remote OPSWAT Filescan system for processing. Requiring users to provide proper remote OPSWAT Filescan URL and API key.
  - We need separate MetaDefender Core license key for each engine option (Embedded vs. Remote).
  - Both OPSWAT Filescan engines require Java 8 JRE x64 (Windows and Linux) and Microsoft Visual C++ Redistributable for Visual Studio 2015 x64 (Windows only) as dependencies. Please make sure to have them installed on MetaDefender Core server before deploying OPSWAT Filescan engine.
  - The integration comes with two modes:
    - **Inline:** working as a part of MetaDefender Core processing workflow (real-time processing). Allowing users to block entire processing based on OPSWAT Filescan engine's outcome and decision.
    - **Out of band:** working as a part of MetaDefender Core quarantine (post-investigation processing).Providing additional option to analyze quarantined items along with the existing Threat Intelligence technology.
  - OPSWAT Filescan engine (both Embedded and Remote) provides configurations under workflow rule (for inline mode), and under engine module - Inventory (for out of band mode).
- UI accessibility
  - MetaDefender Core provides a new UI mode to support accessibility view. By default, this option is disabled.
- Licensing with On-prem License Management Server (OLMS)
  - This feature requires the upcoming new product (On-prem License Management Server) and our existing cloud-

based service (Activation Server) to operate.

  - New licensing managment model for MetaDefender Core to allow On-prem license management server (OLMS, to be released later) to manage the product's license status including activation, deactivation.

- Shared Authentication Session
  - In the shared database model, MetaDefender Core instances now can share same authentication session among them. That helps eliminate the need to manage separate authentication sessions on load balancer side.

This new feature supports local authentication, AD / LDAP authentication and also Single Sign-On (SSO).

- Customized Proxy Configurations
  - In constrast to what we provided in the past for proxy configuration, which is only one setting for all network involved modes on MetaDefender Core, now we provides a much more flexible way to define separate proxy configurations for each MetaDefender Core functionalities that require network connection.
- Webhook for post action
  - Besides the existing option to allow MetaDefender Core trigger each separate sub-processes to run your pre-configured post action script, now you are provided with another new option to run post action in multi-threading mode, as a part of workflow processing.
  - When configured under post action page (Inventory menu), for each file processing, MetaDefender Core will send pre-configured callback URL to your designated webhook server, and you can define to run series of actions on webhook server asynchronously. Please note you will want to setup webhook server on your own.
  - This new post action model helps improve overall performance.
  - Note: Currently, only HTTP protocol is supported for callback URL.
- SHA256 checksum for sanitized file
  - MetaDefender Core provides SHA256 checksum for sanitized file in the final JSON scan result. You can find it at`sanitization_details.sanitized_file_info.sha256`
- UI enhancements

Fixed:

- Archive engine integration issue
  - Fixed issues caused by archive engine integration that could make negative impact on archive extraction performance, as well as cause memory leak issue.
- MetaDefender Core service crashed under certain circumstance
  - When scanning some special files, MetaDefender Core could crash due to its file signature analysis feature.
- RootCA ceritificate store could not be loaded in Linux
  - When having *.0 files in rootCA trusted store.

- Some UI cosmetics and minor bugs are addressed.

Version v5.4.1

Release Date: 28 February 2023

New features:

- [Central Hub] High availability for distributed archive extraction
  - While handling distributed extraction, if a MetaDenfeder Core instance dies for whatever reasons, then distributed extraction task should keep running on another alive MetaDefender Core instance.
- [Central Hub] New setting to set file size limit (min - max) for distributed archive extraction
- Exception for file types based blocklist
  - A new setting under "Blocklist by file type" option for Blocklist configuration to allow users configure to block everything but excluding certain file types.
- Comprehensive session timeout settings on UI
- Security enhancements
  - Support TLS v1.3: Harden security policy with TLS v1.3 supported for HTTPS connection. By default, both TLS v1.2 and v1.3 are allowed for client handshake.
  - Upgraded to OpenSSL 3.0.8 for vulnerabilities.
  - Addressed vulnerability with security through obscurity, and information leakage.
  - Improved license enforcement mechanism for corner cases while communicating with the licensing server.
- Retain original file name with special encoding for archive extraction
  - Note: This feature requires Archive engine version 6.2 or newer.
  - By specifying correct encoding of original file name via engine-metadata header while doing file submission POST /file, MetaDefender Core will work with archive engine to retain original filename with the same designated encoding after extracted.
  - For example: engines-metadata={ "extraction": { "charset": "SJIS" } }
- Support to sign in MetaDefender Core via Identity Provider site (also known as IdP-initiated SSO)
  - IdP-initiated SSO allows users to login to MetaDefender Core dashboard directly from IdP homepage.
  - Users must enable this new setting under SSO user directory page on MetaDefender Core UI.
- Support proxy_pass for NGINX web server on MetaDefender Core
  - Use proxy_pass to redirect the URL to another designated location.
  - For example - redirect https://localhost:443 to https://localhost:8008 Add file *.conf under C:\Program Files\OPSWAT\MetaDefender Core\nginx

- Supported same date selection for processing history filtering
- Display "In-progress" label in processing history UI
- UI enhancements
  - Addressed UI accessibility issues
  - Sensitive Data Protection statistics on the Dashboard UI: Changed to calculate and display total number of sensitive result detections (previously: total number of hit objects).
  - Update UI labels and hyperlinks for My OPSWAT portal.

Fixed:

- [Central Hub] Hub service could be crashed while running under high load circumstance.
- [Central Hub] Failed to link files in batch
  - Hub refused to correlate files to a batch which was initiated by a MetaDefender Core instance (returned HTTP error 400 - Cannot scan in given batch)
- Deep CDR engine process could be crashed on Linux based system
  - While processing MS Office 2003 files, only impacted Linux based MetaDefender Core.
- Synchronous scan submission issue
  - "Require a min number of AV engines for the whole file processing" setting, if enabled and configured, did not work as expected for synchronous scan submission with local file scan feature.
- Data warehouse failed to refresh, when data contains non-Unicode characters.
- File type mismatched result was not displayed
  - When there was another higher blocked verdict (e.g. Sensitive Data Found) is also available.

Version v5.4.0

Release Date: 04 January 2023

New features:

- Central Hub enhancements
  - Applicable to Central Hub version 1.1.0 or above.
  - Load shared among MetaDefender Core instances for archive processing (extraction).
  - Shared authentication session among MetaDefender Core instances using Redis caching server.
  - Batch processing supported.
- Docker containerization enhancements
  - Supported MetaDefender Core upgrade with zero down time on containerization environment Kubernetes (blue/

green and rolling strategies).
  - Supported MetaDefender Core upgrade with data retention.
  - Log rotate service built in docker base image.
  - Supported proxy configuration for docker run.
  - Supported new option to enable "allowCrossIPSessions" when starting MetaDefender Core on container environment.
- UI enhancements
  - UI remediations and enhancements for accessibility support and compliance.
  - Backup license status displayed on UI.
  - Sanitized label as indicator in the processing history.
  - Search by name supported for workflows.
  - Appropriate UI result for vulnerability assessment when "skip hash calculation" setting is enabled.
  - Descriptive tooltip for "Fallback file type detection to current extension if needed" setting.
  - Changed setting name "Blocklist unsupported file types" to "Block unsupported file types".
- Security enhancements
  - Upgraded 3rd party libraries (gperftool, libxml2) for vulnerability risk.
  - Digest verification supported in FIPS enabled environment while installing MetaDefender Core.
- Logging enhancements
  - Avoided misleading (and unharmful) error messages in PostgreSQL logs.
  - Added batch ID and data ID information in NGINX access logs
  - Removed misleading warning messages in MetaDefender Core logs while updating engines via folder.
- Hash lookup performance improvement
- Support Windows 11
- New configuration to reject file submission based on engine availability
  - Only allowing MetaDefender Core to accept file processing submissions when certain crucial engines are available.
  - This setting can be found under workflow rule - General tab.
- New configuration to add exception for unsupported file types for sanitization
  - Adding exception list for unsupported file types for sanitization. By doing so, those file types configured in the exception list will not be blocked.
  - This setting can be found under workflow rule - Deep CDR tab.
- New configuration to block or allow file processing if smaller or bigger than certain file size

  - Allowing MetaDefender Core to block or allow file processing submission if its payload exceeds or smaller certain size.
  - This setting can be found under workflow rule - Blocklist and Allowlist tab.

- New configuration to skip processing fast symlink in archive file
  - By default, MetaDefender Core will try to process any files, including fast symlink in archive. However that could cause sanitization failure due to unsupported.
  - This setting can be found under workflow rule - Genertal tab.

- Import full configuration package (.zip) via the ignition file
  - Supported automation better with full configuration package (.zip) via the ignition file. Example: eula=true [config] import=settings.zip import_password=123 ....
  - Details: Ignition file

- Webhook callback retry mechanism enhancement
  - Enhanced webhook retry mechanism for resending callback result in case of either timeout or not receiving webhook server's response.
  - Timeout threshold could be defined via new key `transfertimeout` (by default, 30000 ms) with PUT - Webhook set configuration

- New CLI tool to clean up database created by no-longer-used MetaDefender Core instance in the shared DB mode
  - Using this tool to clean all scan data from MetaDefender Core instance(s) that you no longer use, applicable in the shared DB mode.
  - This tool can be found:
    - Windows: C:\Program Files\OPSWAT\MetaDefender Core\ometascan-db-cleaner.exe
    - Linux: /usr/sbin/ometascan-db-cleaner
  - Details: Database Cleanup Tool in Shared DB Mode

- New CLI tool to check remote PostgreSQL user compliance
  - This tool is to help your database system admin quickly determine whether or not your remote PostgreSQL database have sufficient user privileges as described at PostgreSQL User Privilege Requirements to work properly with MetaDefender Core application.
  - Details: Remote PostgreSQL User Privileges Checking Tool

Fixed:

- Retrieving file processing result hit 404 not found
  - Could happen when using archive compression with Proactive DLP

- Incorrect memory resource utilization report

  - Updated to recalculate memory available reported on the UI.

- Undesired behavior on the UI with API rate limiting
  - When hitting the limit, the UI hung at the uploading screen.

- Some UI cosmetics and minor bugs are addressed.

Version v5.3.1

Release Date: 07 November 2022

New features:

- Upgraded OpenSSL framework from version 3.0.5 to version 3.0.7 for its recent vulnerability patches (CVE-2022-3358, CVE-2022-3786 and CVE-2022-3602). More details: https://www.openssl.org/news/openssl-3.0-notes.html

Version v5.3.0

Release Date: 27 October 2022

New features:

- Central Hub new architecture (BETA)
  - Serving large scale deployments with better dynamic auto-scaling, high availability and fault tolerant.
  - The Central Hub architecture incorporates multiple newly crafted components running as service, and also containerization environment friendly:
    - MetaDefender Core.
    - PostgreSQL database server.
    - Central hub (known as "MetaDefender Core Hub").
    - File storage server (known as "MetaDefender Core NAS").
    - Redis (the open-source caching server).
    - RabbitMQ (the open-source messaging broker).
  - More details: MetaDefender Cluster Deployment

- PostgreSQL bundle upgrade
  - Applicable to PostgreSQL server bundled with MetaDefender Core installation. It is upgraded to version 12.12 (from old version 12.11) for vulnerability fixes.

- Engines integration improvements
  - Back-end related improvements against engines integration for more scalable, flexible and robust engines' new features and furture changes without MetaDefender Core upgrade requirement.

- New workflow setting for Proactive DLP
  - Allow users to configure MetaDefender Core to block file processing (final verdict = Blocked) if the Proactive DLP processing fails or times out.
- Vendor information added into processing result PDF report
  - Only applicable to files which are detectable for file signature.

Fixed:

- Product stability improvement
  - MetaDefender Core service could crash under certain circumstance (modifying workflow rules while actively scanning files).
  - Integration issue with Deep CDR that could possibly wipe up sanitized files by mistake when Deep CDR engine is stopped.
  - Engine ID was wrongly displayed in workflow rule for "excluded engines" section after removing that engine.
- Product UI fixes
  - Custom date and time feature: users could not set PM for start time, and AM for stop time as a range. For example: Sep 07 12:00:00 PM to Sep 08 12:00:00 AM

Version v5.2.2

Release Date: 06 October 2022

New features:

- External scanner and post action are now available configurable for the workflow
  - Use certain external scanners and post actions with corresponding settings in each workflow rule.
  - The import & export configuration feature will also support to cover external scanners and post actions defined in the workflow.
  - For MetaDefender Core upgrade scenario:
    - When you already have external scanners and post actions defined in the Inventory in older product version, then all of them will be automatically added to every available workflow rule (not template) in the new product version once upgraded.
    - Applied to shared DB mode, all external scanners and post actions settings will be auto shared among MetaDefender Core instances, unless the license for external scanner on each MetaDefender Core is different.
  - Details: External Scanner and Post Action
- File digital signature vendors in the executive report
  - Vendor statistics information for file digital signature is now calculated and displayed in the executive report.

- More details for file digital signature analysis information
  - Detect multiple signers (co-signers).
  - Verbose information of digital signature: version, issuer, serial number, digest algorithm, digest encryption algorithm.
- Archive extraction performance improvement
  - When processing archive file that includes empty folders.
- UI updates
  - Vulnerabilities Detection now is clickable.
  - Support to check enabled file types for sanitization in the workflow rule when MetaDefender Core is managed by OPSWAT Central Management.
  - Some other minor UI changes.

Fixed:

- Product stability improvement
  - Iterrupted connection with syslog server could crash the MetaDefender Core service.
  - Nested files was not be extracted correctly if its filename and its parent folder name are identical.
  - Did not display the result and error correctly when processing unsupported file types with Threat Intelligence.
  - User management UI loaded very slowly while handling numerous (thousands+) users.
- Product UI fixes
  - Too aggressive polling for file processing result when users navigated to some last pages.
  - UI cosmetic bug fixes.

Version v5.2.1

Release Date: 30 August 2022

New features:

- Compression tab in workflow
  - MetaDefender Core workflow is organized better with new compression tab to provide the archive compression related configuration for both existing Deep CDR use-case, and now, for the new-feature Proactive DLP use-case as well on archive files.
  - More context: archive compression technically covers the archive sanitization feature that we know so far, but now we take them out from Deep CDR workflow tab to mange everything relevant under the new Compression tab for both Deep CDR and Proactive DLP use-case.
- Archive processing (Compression) with Proactive DLP

  - MetaDefender Core now supports the archive compression for Proactive DLP use-case, along with already-supported Deep CDR so far.
  - That means, MetaDefender Core will also pack applicable output files produced by Proactive DLP as well for the new compressed archive.
  - By default to keep backward compatibility, this feature is turned off for Proactive DLP use-case, and you must enable it under the new Compression workflow tab.

- Hash lookup API new features
  - Hash lookup API is prefered to optimize the file processing especially when same files to be processed over again within a short time period.
  - MetaDefender Core now supports 2 new modes for the hash lookup API via new headers called `timerange` and `selfonly`:
    - `timerange` (number of hours): allow scoping down the recent number of hours that hash lookup task should start from till now, instead of searching the entire scan history in MetaDefender Core database.
    - `selfonly` (true/false): useful to archive hash lookup, allow specifying to only perform hash lookup against the original archive file self only, and skip searching all child files result within the original archive.
  - More details: Fetch Analysis Result By Hash

- Verbose file type analysis data
  - Support to expose verbose file type analysis data in JSON scan details. This can be done via setting `include_file___details` in the `engines-metadata` header for POST file submission request.

- Expose file type ID information
  - The info can be found in both JSON scan details, and on the scan result page UI. The file type ID will help indicate unique file type identification, e.g. PDF / TXT

- Engine dependency missing error response
  - Certain engines on MetaDefender Core require specific dependencies to run properly, otherwise failing on engine is expected. MetaDefender Core now make the dependency missing error more visible in both UI and log, and help troubleshooting easier and faster.

- Custom internal PostgreSQL user
  - MetaDefender Core once installed, it will automatically create an internal PostgreSQL user for operation purpose, and so far that user name is auto randomized by MetaDefender Core and non adjustable (`usr_<random characters>`).
  - Now MetaDefender Core supports database system admin to define and change to their own PostgreSQL user instead.
  - More details: Customize Internal PostgreSQL User

- Security enhancements

  - Continuing to harden and secure MetaDefender Core even better to prevent potential HTML injection and XSS attacks.

- UI updates
  - The setting page is auto refreshed itself after importing configurations.
  - Added copy button to copy entire file path on the scan result UI.
  - Pagination supported for "Skip by hash" UI.
  - Numbers on the dashboard are now clickable.

Fixed:

- Product stability improvement
  - MetaDefender Core on Linux crashed randomly while processing archive files.
  - Very long list of hashes in the Skip by hash caused slow response time on some admin API endpoints.
  - Regular expression validation failed with mode modifier (?i).
  - Display issue with pagination on the processing history UI.
  - Engine download could be hung up if keep hitting download error.
  - Failed to show blocked leaf files if searching by hash.
  - Filtering by post action was not correct on the processing history UI.
- Product UI fixed
  - UI cosmetic bug fixes.

Version v5.2.0

Release Date: 12 July 2022

New features:

- New OS support
  - Windows Server 2022
  - Debian 11
- Security enhancements
  - As a regular security practice, we upgraded 3rd party dependencies and development framework to prevent critical vulnerabilities:
    - Upgraded OpenSSL 3
    - Upgraded Qt framework 6

- Executive report in PDF
  - The executive report is now supported for PDF file export. Details: Executive Report
- File digital signature verification
  - File digital signature is now analyzed and recognized by MeDefender Core while processing. The vendor digitial signature is available on both processing details UI page and JSON scan result, and also calculated for the statistics found on Executive Report. Also supporting users to allowlist files based on specific vendor names (using regular expression) configured in the workflow rule (under "Allowlist" tab).
  - Currently only PE files are supported `.acm`,`.ax, .cpl, .dll, .drv, .efi, .exe, .mui, .ocx, .scr, .sys, .tsp` (Portable Executable)
- Blocked leaf files filtered in archive scanning
  - Leaf file (in archive scanning context): file that not having any successful extracted file inside.
  - Support to filter and display blocked leaf files in archive scanning on the processing details UI page ("Show blocked leaf files" checkbox). That helps narrow down the searching scope, and spot out exactly blocked files that should be investigated further.
- Blocked reasons exposure
  - All blocked reasons are exposed on the processing details UI page (if archive file, then also inheriting from child files' blocked reasons if available), that helps with further context on unclear failures, and troubleshooting.
- A setting to exclude sanitize version of blocked child files from archive sanitization
  - By default (Archive Sanitization) feature will pack following into sanitized archive file:
    - All allowed child files (or their sanitized version if available).
    - Sanitized versions of blocked child files (if blocked child files are sanitized successfully).
  - A new setting "Include sanitized version of blocked child files" under Deep CDR tab in MetaDefender Core workflow rule will allow you skipping not to include sanitized versions of blocked child files, so that you can conduct further investigation on those blocked files first.
- Sanitization tag on processing details UI page
  - Especially helpful to archive processing when you can tell which child files are sucessfully sanitized, and which files are not on one UI screen (without checking every single child file for details).
- Data retention for statistics
  - The new setting can be found under Settings page - Data Retention tab with named "Statistics". By default, the retention range is set to 6 months.
  - Details: Data Retention
- An installation option to skip startup UI screens
  - Better support for automation environments. A new setting `skip`**_startup _**`pages` supported in the ignition file to

skip all welcome and other startup pages (e.g. upgrade result) when installing / upgrading MetaDefender, and going straight to the product index default page.

  - Details: Ignition file

- Skipped sanitization result
  - A new sanitization result to indicate the santization is skipped due to certain reasons ruled by Deep CDR configurations.
- Child files processing time displayed in achive processing JSON result
  - Original archive file will want to wait for all child files processing to be finished before finalizing the overall processing.
  - Previously `others_time` covered that as well in JSON result, but now we make a dedicated new field for that in JSON result called `wait_child_``files_time`
- HTTPS configuration removal tool
  - Just in case you configured HTTPS via MetaDefender Core UI but due to unexpected failures on certificates, and thus being blocked to make any change because MetaDefender Core service is not running, thus you need to disable HTTPS configuration on MetaDefender Core using a dedicated CLI tool - (HTTPS Configuration Removal Tool
- Minor UI changes
  - Some UI updates on the executive reports, modules.

Fixed:

- Product stability improvement
  - Failed to process same files repeatedly with scan-by-download-link feature.
  - Failed to extract archive files occasionally.
  - SSL certificate issue on RedHat 8.
  - Memory leak during PostgreSQL disconnection and more scan requests are still coming.
  - Other minor product bug fixes.
- Product UI fixes
  - UI cosmetics bug fixes.

Version v5.1.2

Release Date: 27 April 2022

New features:

- Vulnerability CVSS v3 scoring

- File-based Vulnerability Assessment now supports CVSS v3 scoring, also comes with attack classification and impact details.

- Fallback for engine (module) deployment
  - Engine deployment on MetaDefender Core is now fault-tolerant support i.e. just in case a new version of engine ever failed to be deployed, the current (older) working engine version will be re-used and kept running well.
  - Notes:
    - Disable and re-enable engine: MetaDefender Core will try to re-deploy the new engine version again, if still failing, then the old-working engine version build will be re-used (fallback).
    - Remove engine: MetaDefender Core will remove all engine package(s) entirely, including new and old-working versions.
    - Not supported for custom engines which require separate engine installation setup

Fixed:

- Product stability improvement
  - Faster API responses when processing very long list of users (>= 1000 users).
  - Custom engines sometimes could be inactive after being updated.
  - Even when enabling processing of allowlisted file in Workflow Rule, the file could not be processed properly.
  - Fixed minor product bugs.

- Fixed some security vulnerabilities on the product:
  - Privilege escalation: allows non-admin users to gain unauthorized privileges on the product to execute harmful actions via the user management configuration. Details: CVE-2022-32272
  - Information leakage: an MetaDefender Core user without permission of accessing to the MetaDefender Core server could leverage the observable discrepancy attack method to expose file system structure on where the product resides via the certification configuration. Details: CVE-2022-32273

- Product UI fixes
  - The pie chart in the Executive Report page was not rendered and displayed correctly occasionally.
  - Some other UI cosmetic minor fixes.

Version v5.1.1

Release Date: 31 March 2022

New features:

- Product performance improvement
  - Up to 50% throughput increased while processing archive / office files. Applicable under certain circumstances,

configurations and data.

- Product security improvement
  - Remediate some potential security related issues while operating MetaDefender Core.
- Optimize processing flow for office files
  - All child nested files inside original office file are no longer sanitized and DLP processed separately.
- Bundle PostgreSQL upgrade (v12.10)
  - Applicable to the local PostgreSQL server bundled with MetaDefender Core installation.
- User's API key redeployment
  - Allow API key of an already-removed user to be assigned to another user, this is to ensure no interruption or changes in the cross-system integration.
- Health check settings supported for the configurations export/import feature
  - Health check settings (Health Check API Configuration) is now included in the MetaDefender Core configurations export/import feature
- Auto upgrade when connecting to an older database version
  - Allow MetaDefender Core auto detect and upgrade its database when connecting to an older database version as expected.
- Single Sign On (SSO) SAML enhancements
  - Support AES128-GCM and AES256-GCM encryption algorithms.
- Product UI/UX enhancements
  - Add OS information to the dashboard UI
  - Resizable column width on tables (workflow rules, workflow templates, workflow zones, modules)
  - Toggle configuration for proxy credentials

Fixed:

- Product stability improvement
  - Avoid potential Nginx socket read timeout, and MetaDefender Core crash issue under certain memory shortage circumstances.
  - MetaDefender Core service crashed when trying to load an empty advanced configuration of engines.
  - Some minor bugs.
- Product UI fixes
  - Pagination is broken when the page number is too large.

  - Inconsistency multiscanning engine names on the module page.
  - User Management page didn't display any record.
  - Cannot traverse the list of files in a batch on the scan result page.
  - Incorrect results are returned when filtering the datetime on Statistic page.
  - Scan result report in PDF text alignment issue.
  - Cosmetic minor fixes.
- Incorrect callback messages mistakenly sent
  - "In-progress" scan result callback message was mistakenly sent to webhook server while the scan is not yet finished.
- Disk space availability check omitted before processing file request
  - MetaDefender Core is supposed to check for disk space availability on where the product temporary folder is configured to ensure it could store file and serve processing request.
- Kaspersky and NanoAV engine failed to load on MetaDefender Core Linux
  - The issue was due to incorrect handling within symlink in the engine package.
- Engine manual upload issue
  - Re-uploading the same engine package again immediately after a failed attemp on the same engine might cause issue on engine upload feature until MetaDefender Core service is restarted.
- Processing failure with the encrypted archive ALZ/EGG file
  - Failed to sanitize and re-compress those file types.
- Failed to remove AD/LDAP users
  - Hit error message "Item does not exist" on the UI when trying to remove AD/LDAP users.
- Temporary files and folders cleanup
  - Obsolete temporary files and folders were not cleaned up when user configured to set a new tempdirectory location.
  - Temporary files downloaded from a link were not cleaned up when disabling engines during scan.

Version v5.1.0

Release Date: 23 February 2022

New features:

- New statistics data for executives report
  - Average file size based on processed file types

- Hash calucaltion time
- Archive extraction time
- Per AV engine processing time
- Details: Executives Report

- Exclusion list for proxy server configuration
  - Define an exclusion list of destination addresses to bypass going through proxy server configured on MetaDefender Core. Applied to connections triggered by MetaDefender Core for following activities:
    - Callback (webhook)
    - Engine update
    - Scan by download link
    - Single Sign On integration (Fetch metadata from URL and login)
    - Remote PostgreSQL database connection
  - Details: Exclusion List

- Support Ubuntu 20.04
  - See the list of supported OS at Recommended System Configuration

- API key in the ignition file
  - Local admin user's API key could be now definied in the ignition file for automation enhancement.
  - Details: Ignition file fields

- Network drive and UNC path for update folder pickup
  - Allow users to define UNC path and network drive for update folder pickup setting.
  - Details: Local folder update

- Timeout indication for Proactive DLP processing
  - Whenever Proactive DLP processing timed out, a clear and explicit timeout result is now used (instead of a generic failure reason).

- More logging comprehensive information for scanning activity
  - Absolute submitted file path for local scan feature.
  - Clear indication to distinguish between asynchronous and synchronous scan.
  - File path, file name, callback URL, sanitize URL, download-from URL for both asynchronous and synchronous scans.

- New simplified REST API to download output file after sanitized and Proactive DLP processed
  - Details: GET - Download either sanitized files or DLP processed files

- Clear indication for archive extraction in STDIN for post action
  - A new key of "extracted_files" is added into the STDIN for post action to indicate whether MetaDefender Core indeed extracted to child files from original file or not.
  - Details: Post Actions
- Resizable table column on the UI
  - Applied to tables on the UI including processing history, quarantine, user management, engine list, quarantine etc.
- Security enhancements
  - UI setting input validation.

Fixed:

- Product stability improvement
  - Fixed minor bugs regarding to product stability.
- Samesite cookie modification with SAML based SSO
  - Value of samesite cookie was not updated correctly while using SSO with SAML integration.
- Wrong error code and error message with user role and directory removal
  - Wrongly returned 404 (item does not exist) instead of 400 when attempting to remove non-existed user role and directory.
- Custom data directory configuration
  - MetaDefender Core did not point to custom data directory after upgrading on CentOS / RHEL.
- YARA engine failure
  - YARA engine could become permanently failed after updating.
- Minor UI fixes
  - Cosmetic bug fixes

Version v5.0.1

Release Date: 28 December 2021

New features:

- New fields supported for PDF scan report
  - Receive data timestamp (indicating when a first byte was uploaded to MetaDefender Core from client)
  - Upload timestamp (indicating when entire upload was finished)

  - Upload time (indicating total time spent for upload)

- Product UX improvement
  - Under MetaDefender Core modules page: Different color (Orange) for engine and database when they are still being downloaded and deployed, not reached yet to final state. And some other minor UI changes.

- Archive compression engine auto enabled after product upgrade from v4 to v5
  - This engine is supposed to be inactive and permanently failed on MetaDefender Core v4, when upgrading to v5.0.1, now this engine will be auto re-enabled back without any user action required.

- Hot backup register API behavior updated
  - Not possible now to register a MetaDefender Core instance as backup when `active-deployment-id` header value equals to that own instance's deployment ID. Otherwise, expecting to hit an error: `{ "error": "active-deployment-id cannot be the same with this Core's deployment ID" }`

- Security hardening
  - 3rd party vulnerabilities check and remediation.

Fixed:

- Sanitization of files without extension on Windows
  - Applicable to Windows
  - When configured to keep both original file name and extension `${original.basename|long}.${original.extension}` , but original file name does not have extension: Previously MetaDefender Core auto changed the dot (.) between file name and extension (which is empty) to underscore (_). Now it is no longer the case, expecting to have only original file name.

- User agent "webscan" took no effect
  - Adding user agent "webscan" to any workflow should now force MetaDefender Core to display that workflow for selection on processing scan UI.

- Engine communication was not resumed correctly after engine timed out or crashed
  - When happened, causing not scanned / not available result on engine processing.

- Missing Proactive DLP failure details
  - Previously it just showed failed to process for Proactive DLP, now the failure reason should be exposed correctly (e.g. file size limit exceeded).

- Wrong archive handling details description
  - Mistakenly changed from "Corrupted Archive" to "Unsupported Archive"

- Wrong sanitization time when timed out

  - When sanitization timed out, cdr_time was mistakenly set to zero (0).

- Leftover Node related folder / key
  - On Windows (`Computer\HKEY` _ LOCAL _ MACHINE\SOFTWARE\OPSWAT\Metascan Node``)
  - On Linux (`/etc/ometascan-node`)
  - They should be all cleaned up after upgrading to MetaDefender Core v5.
- Leftover archive engine temp folder
  - Only encountered when disabling archive extraction engine during extraction.
- Exporting scan history for all instances in shared database mode
  - Applicable to CSV / STIX export. The report only contained scan history records made on that current MetaDefender Core instance.
- UI bug fixes
  - Engine status color display on UI.
  - Processing scan highlighter.
  - Total number of processed objects displayed on Dashboard UI when being too big.
  - Error message on User Management disappeared when hitting Save changes button twice.
  - Unable to view sanitization details on UI when XFA form is sanitized.
  - MetaDefender Cloud API key was not redacted on Firefox.
  - Not displaying error message on UI when failing to import product configurations.
  - While file processing is happening, hitting details in DLP section was mistakenly redirected to AV section instead.
  - Error message was misleading when modifying a certificate being used by workflow rule or HTTPS setting.
  - Not able to continue testing LDAP configurations on UI after previous attempt was successful.

Version v5.0.0

Release Date: 30 November 2021

New features:

- Processing performance improvement
  - In typical use-cases, expecting better throughput on version 5 with architecture changes and processing flow optimization (30% - 200% depending on specific circumstances, configurations and data set)
- RedHat 8.x is supported (official)
  - Installation guide: Red Hat Enterprise Linux 8.x package (.rpm)

- Security hardening
  - Eliminated critical and major vulnerabilities found in 3rd party libraries as well as the application itself.
- Hot backup mode supported for high availability
  - Supporting to setup backup MetaDefender Core instance(s) to run in parallel with your production (active) instance. Backup instance(s) will be allowed to use the same license key as your active one without actually being accumulated into the activation slots, and allowed to have all engines downloaded and deployed, configurations pre-setup, share the same database with active one (except of processing files).
  - Any of backup instance(s) should be ready to take over the active one anytime as a part of your disaster recovery plan to minimize the down time as much as possible.
  - More details: Hot Swap Backup Configuration
- Health check API
  - Determining MetaDefender Core server's readiness to process upcoming files based on pre-defined criteria / policy that configurable by administrators. Allowing native support for a large pool of MetaDefender instances (load balancer in autoscaling groups. or liveness probe in Kubernetes, etc.)
  - More details: Health Check API Configuration
- Server active performance statistics API
  - Determining MetaDefender Core performance (processing time, throughput, queue time etc.) within specific time window to help you make better decision on which server is the best to route upcoming requests. Metrics that can be also collected and reported in external tools, not just via SIEM.
  - More details: GET - Active performance
- Bulk hash skipping modification API
  - No longer need to import one by one hash for either or both whitelisting and blacklisting. Allows you to import a list of hashes instead in a form of file.
  - More details: POST - Add new hashes to 'skip by hash' list
- Parallel engine deployment
  - Only applicable to bundled engines for now. Bundled engines will be deployed and initialized in parallel, to make MetaDefender Core get to ready state faster, and possibly reduce down time in dynamic environments (Containers, Cloud Autoscaling, etc.).
- More data points in log message for processing finished event
  - Applicable to log event generated upon a processing analysis task is finished. New information added:
    - Parent data ID
    - File size
    - File type description
    - Total processing time

- Estimated completion time on product upgrade
  - Upgrading time depends on various conditions including system specs itself, data size, characteristics etc. thus a roughly upgrade estimation time would be useful to set right expectation, especially to big size database that typically requires hours in upgrade. As a reminder, there is already an existing option on product to skip migrating processing data for upgrade when you do not really need it.
- Preferred encoding supported for sanitization and DLP processing
  - A new header `engines-metadata` for `POST /file` (asynchronous scan) and `POST /file/sync` (synchronous scan) to specify explicitly payload's preferred encoding (e.g. ISO-2022-JP) for sanitization and DLP processing. If not specified, then default encoding (UTF-8) will be applied.
  - More details: POST - Analyze File (Asynchronous mode)
- Proactive DLP integration enhanced
  - Helped expose new Proactive DLP features on MetaDefender Core workflow rules, and make the product workflow flexible for changes and improvements.
  - More details: Proactive DLP release notes
- SHA256 checksum of objects in Deep CDR analysis result details displayed
  - SHA256 checksum calculated on every sanitized objects processed by Deep CDR is now displayed on MetaDefender Core's processing result UI.
- Handle unsupported file type for archive compression
  - Under MetaDefender Core workflow rule - Archive tab. A new setting option where you can select to fail the scan with unsupported file type for archive compression / sanitization.
- Enhanced logic to help avoid MetaDefender Core initialization failed due to database failure
  - Possibly encountered in container based environment with remote PostgreSQL database in certain circumstances.
- A new option for unsupported compression method handling
  - Under Archive handling settings in MetaDefender Core workflow, a new option to allow failing archive extraction task for unsupported compression method.

Fixed:

- Advanced Deep CDR settings were reverted under certain circumstances
  - Only encountered when using workflow rule based on custom workflow template (this issue had happened since MetaDefender Core version 4.20.0) Changes made on Deep CDR advanced settings were reverted after MetaDefender Core service restarted or Deep CDR engine re-enabled.
- Vulnerability processing result was wrong in PDF scan report

  - When actual result was a potentially vulnerable file, but reported as no vulnerability found mistakenly in the PDF report.

- Fix memleak when handling Proactive DLP errors
  - Possibly happened when Proactive DLP encountered errors while processing files.

- Reinstalling MetaDefender Core could be failed
  - Uninstalling the product and then reinstalling back on the same machine but with different deployment ID.

Version v4.21.2

Release Date: 30 August 2021

New features:

- Mutual authentication for secured webhook (callback) mode
  - An additional security mode for callback HTTPS to authenticate requests indeed sent by MetaDefender Core, and help against man-in-the-middle attack. Details could be found at: Webhook Authentication

- Webhook (callback) for sanitized file download (BETA feature - not recommend to be used on production)
  - When registered by client, whenever a sanitized file is produced and ready, a new callback with sanitized file content will be sent back to client.

- Configurable relevant session cookie attribute aided in CSRF attack prevention
  - "*SameSite*" attribute on the session cookie "*session_id_ometascan*" with supported values: Lax (default), Strict, None (i.e. empty) to be configurable. Supported to configure via either administrative REST API, or ignition file. Details could be found at: Configure session cookie attribute

- Support SSL connection between MetaDefender Core and PostgreSQL
  - Along with a setup guidance for SSL database connection. Details could be found at: SSL Connection for bundled PostgreSQL

- SQL Injection prevention check and ensurement
  - On all supported REST APIs

- Adaptive mode when moving AD / LDAP users to other OU (Organization Unit)
  - Allowed users to change DN information in AD / LDAP directories

- Logging details enhancement on Webhook (callback) failures
  - Enhanced on generic failure (error code = 444) to expose more details in WARNING log level. Details could be found at: Webhook error message description

- Logging details enhancement when failing to open PostgreSQL service

  - In a circumstance of when failing to open PostgreSQL configuration file.
- Separate JSON configuration file now collected in the support package
  - For easier analysis and import back into MetaDefender Core when needed.
- Preinstall Deep CDR and Proactive DLP dependencies in MetaDefender Core docker image (libgomp, libgdiplus, mscorefonts)

Fixed:

- MetaDefender Core upgrade failures
  - Handled duplicated sanitization detail records
  - Handled empty key in registry when installing by commandline
  - Handled invalid value of type in ignition file
  - Handled invalid Unicode characters
- Initialization failure of Threat Intelligence engine on Ubuntu
  - Updated openssl linkage to address the issue
- Improper value type of "*archive_handling_details*" in JSON response
  - In a case of processing password-protected archive file, the value was set incorrectly, it should always be a map data type (not string).
- Lacked of compression failure detail (archive sanitization)
  - When MetaDefender Core fails to re-construct email file (.eml) in archive sanitization.
- Incorrect error message when recovering user password
  - In a case of the email server address configuration is not setup correctly for password recovery.
- Out of bound value of session timeout handling
  - Setting value on session timemout to 1 million or above resulted in authentication failure.
- Update hang up on engine in certain circumstance
  - Engine remained in "staging" status indefinitely if the configuration validation failed for engine.
- Missing Proactive DLP engine processing result
  - Making sure all engine results return correctly in both failure and success circumstance.
- Incorrect scan result in PDF report
  - In a case of using skip scan on engine setting
- License auto deactivation did not work properly in docker container deployment

  - When stopping docker container, the docker automation script in the toolkit (entrypoint.sh) did not handle license deactivation properly.

- Retry mechanism of Webhook (callback) did not work as expected in special circumstance
  - When MetaDefender Core service is restarted while operating, the callback was not sent as designated in callback retry configuration.

- Intermediate certificate was stripped out
  - HTTPS configured with full chain certificate scenario, the intermediate certificate could be stripped out before importing into the product

Version v4.21.1

Release Date: 13 July 2021

New features:

- A new setting to exclude password-protected documents from being blacklisted due to unsupported sanitization
  - The password-protected document is not supported for sanitization and possibly blacklisted when enabling "BLACKLIST UNSUPPORTED FILE TYPE" option under Deep CDR workflow rule.
  - Now you can exclude it via a new setting. By default, the new setting is not enabled to keep backward compatibility.

- Upgrade 3rd party libraries for security enhancement
  - (Built-in) PostgreSQL from 12.6 to 12.7
  - NGINX web server from 1.18.0 to 1.20.1
  - Remove outdated jsPDF 1.5.3

- A new CLI tool to analyze and deeply vacuum database
  - PostgreSQL database could be more bulky over time with high scanning traffic. The tool which is bundled with MetaDefender Core installation is to analyze database and vacuum database effectively, and simplify IT administration. Even though it is not required to stop MetaDefender Core service while running, but still recommended to run the tool when the scanning service is at rest, or not in peak hours to avoid scanning performance impact.
  - Details: Database Maintenance | MetaDefender Core 4.19.0 or above

- More data points collected for the support package
  - Including size of objects from MetaDefender Core database to help us determine better the characteristics of the data for troubleshooting purpose.

- Other minor changes
  - Enhanced integration logic on engine configuration handling to avoid potential configuration corruption.

- Enhanced installation script in Linux-based installer to avoid misleading warning.
- When the Vulnerability assessment module is not enabled, the relevant vulnerability result on the MetaDefender Core scan result UI will be now as "not configured" (instead of no vulnerability found).

Fixed:

- OPSWAT Central Management integration issues
  - Workflow rule list order was different between MetaDefender Core and OPSWAT Central Management UI.
  - Last configuration import timestamp was invalid.
- Deep CDR's advanced settings was not applied
  - Occurred when creating a new workflow rule on MetaDefender Core.
- Negative duration processing time
  - Under certain corner circumstances, the wait_time and queue_ time in JSON scan result could be mistakenly calculated.
- User on old database was left out during the MetaDefender Core upgrade
  - When upgrading MetaDefender Core with standalone database mode to shared mode.
- Leftover files when canceling batch
  - Occurred when handling archive files.

Version v4.21.0

Release Date: 14 June 2021

New features:

- Docker container support
  - A new deployment support for MetaDefender Core on docker container environment.
  - Supporting for Linux based environment (CentOS / RedHat, Ubuntu / Debian). Windows based more to come later.
  - Comes with 2 options: (1) Docker images hosted by OPSWAT (2) A build kit with docker file and scripts to use with your own docker image.
- Synchronous scan
  - One more new scan API interface for Core with a sync mode (blocking mode). That helps ease integration hassles on some of your web applications.
  - The connection of file submission request must be held by client side until MetaDefender Core returns processing result of individual scan when finished, or return data ID associated to that request if timed out (which is configurable).

- Scan performance improvement
  - Optimized for resource conflict handling, even better under high scan rate with many AV engines.
- Azure AD supported with OIDC
  - Continuous Sigle-Sign-On (SSO) improvement with OIDC supported for Azure AD.
- New scan result "Potential Unwanted" for PUA/PUP
  - Applicable to AV engine scan result.
- Crash dump auto collected for support package
  - Only if crash dump is available for MetaDefender Core related processes.
- Statistics page enhancement
  - New section"Custom Regular Expression" applicable to Proactive DLP.
- Proactive DLP settings limit range extended
  - Extended maximum character length limitation of Proactive DLP settings under workflow rule.
- Secure the integration with Central Management product
  - The integration credentials meet complexity requirement.
- Retain password protection for RAR file sanitization
  - A new option for RAR file could be found under CDR tab in workflow rule, applicable to password protection for archive sanitization.
- More data points for /stat/nodes API endpoint
  - Including:
    - Total disk space
    - Total scan queue
    - Service uptime
- Upgrade 3rd party libraries
  - Built-in PostgreSQL 12.3 to 12.6
  - Lodash 4.17.10 to 4.17.21
- Product upgrade process enhancement
  - For back-end processing efficiency when upgrading MetaDefender Core from standalone to shared DB mode.

Fixed:

- Product upgrade issues
  - Fixed some failures when upgrading the product with malform data in database.
- Default workflow template mistakenly modified with Central Management integration
  - On Central Management, changing any setting in a new workflow rule unexpectedly resulted in changing associated workflow template.
- Wrong data in PDF report for processing result
  - Under certain circumstances, some information in PDF report could be invalid / misleading.
- Incorrect blocked_reason in tombstone file
  - When processing an archive file contains at least one child file returning Sanitization Timed Out / Sanitization Failed.
- Minor fixes
  - Some other minor bug fixes.

Version v4.20.1

Release Date: 31 Mar 2021

New features:

- MetaDefender Core's PostgreSQL database admin credentials management
  - Supported via both web management UI and CLI tool to change Core's PostgreSQL database information.
  - The CLI tool could be used for older PostgreSQL based Core versions as well (4.19.0 to 4.20.0)
- MetaDefender Core upgrade mechanism changes
  - Ensure the old product database is always kept regardless of upgrade result, and easier to rollback when needed.

Fixed:

- MetaDefender Core upgrade failures
  - Addressed various upgrade failures occurred with Core version 4.20.0
- Scan stuck until timeout hit under certain circumstances
  - Possibly happened with Core 4.20.0 when the data retention is enabled for sanitized / DLP processed file, and only when the system keeps processing the same file over again while running the data retention.
  - This issue could only be applicable when CDR and(or) DLP feature is used.
- MetaDefender Core service failed to start

  - Encountered when the Core's PostgreSQL database admin password contains special characters /-+"?<>:{}&^% $#@!~`';.,\

- Invalid batch signature
  - Due to a bug in 3rd party library, batch signature could be invalid.
- MetaDefender Core service issue on docker
  - Only applicable when replacing systemctl to start Core service within docker container.
- MetaDefender Core service failed to start automatically after reboot
  - Thus sometimes required manual start action.
- Minor fixes
  - Some other minor bug fixes.

Version v4.20.0

Release Date: 08 Feb 2021

New features:

- Centralized (or shared) database mode
  - A complete new database design to support to use single PostgreSQL database shared among all connected MetaDefender Core applications. That helps auto-sync all product configurations (except Single-Sign-On that requires unique settings for each Core), processing histories, live/on-going scan result query, audit, statistics data. All product functionalities are updated to support both standalone and shared database deployment model.
  - Database migration (configurations and processing history) is also supported for older Core versions upgrade.
- Workflow based for Deep CDR advanced configurations
  - Deep CDR advanced configurations now could be defined differently for each workflow rule on MetaDefender Core.
- HEAD request is now optional for the scan-from-link feature
  - MetaDefender Core no longer requires to send HEAD request to check for metadata of file before downloading with scan-from-link feature. Some storage web services such as AWS S3 with signed URL, it does not support unauthorized HEAD request.
  - When HEAD request is omitted, expecting no support for download progress status and insufficient disk space pre-checking.

Fixed:

- Statistics data failed to migrate after Core ugprade

  - Upgrading from old Core version 4.17.3 or below failed at statistics data migration.

- Processed file content duplicated
  - The issue could sometimes occur when we have two or more same file to be processed.

- Minor UI fixes
  - Tooltip for processed file cleanup explanation

Version v4.19.2

Release Date: 07 Dec 2020

New features:

- Non-persistent scan mode
  - A stateless scanning solution to improve performance. When triggered, MetaDefender Core will not write any scan result into its database.
  - Note: Webhook scanning fashion must be used from client side to retrieve scan result back from MetaDefender Core.

- Interactive engine removal on UI
  - Any engine module could be removed on the management console (instead of using CLI tool). No service restart is required, no command to run.

- New Identity Provideres (IDP) supported and verified for Single Sign On (SSO)
  - Microsoft Azure AD Connect
  - Ping Identity
  - Idaptive
  - Xecurity
  - OneLogin

- Data retention supported for statistics
  - Data stored in statistics warehouse could be configured to clean up.

- Minor supports / changes
  - File type processing time included in final JSON response
  - DLP image cropping feature support (Only available starting Proactive DLP version 2.6.0)
  - Remove BETA from Statistics page
  - UI clarity with scan result page
  - Unicode characters better support for DB migration (product upgrade)

- Better performance for statistics page
- Support package to include engine logs (only available when engines start writing it)
- Workflow rule setting values validation
- Case sensitive support in header "filepath" for POST /file request

Fixed:

- Performance degradation issue mitigation
  - Archive scanning performance
  - Hash lookup slow response
  - Batch close request potential bottleneck
  - Database dynamic query optimization
- Memory leak
  - Both could happen on ometascan and ometascan-node processes under certain circumstances.
- Incompatible Nginx cipher algorithm
  - Default acceptance cipher algorithm list modified to be less strict.
- Missing vulnerability detailed result
  - Vulnerability detailed result could be missing from final scan report in some cases.
- Very high CPU continuous usuage occupied by node process
  - Might happen when extraction failed and then affecting to the completion of hash calculation process, result in high CPU usuage continously.
- Potential service crash
  - While cancelling archive scanning
  - Engine package download
- Issues with scan result report
  - Wrong AV and Deep CDR result
  - Missing Proactive DLP section
  - User right issue (i.e. who is allowed to download the scan report)
- Proactive DLP setting issue with unicode characters
  - The unicode setting value could be malformed after Core / Proactive DLP upgrade
- Other product functional issues

- Blocked files was not stored in quarantine with local file path scan, or scan from link fashion.
- Wrong data calculation on statistics page

Version v4.19.1

Release Date: 21 Oct 2020

New features:

- Processing file with input as a download link
  - Supporting users to process file by just specifying a direct download link, which means no required for the payload in HTTP(S) request body.
- Tombstone file supported for archive sanitization
  - Whenever any child file in original archive file are not sanitized successfully, the product supports to append tombstone file into sanitized archive file to indicate which original child file(s) are removed from sanitized archive output file with details.
- No hash calculation configurable setting
  - Support an option to skip hash calculation on every processing file (using dummy hash values instead) regardless it is individual file or a part of archive.
  - Expecting to be used against giant file processing to significant reduce overall processing time.
- Nginx web server component ugprade
  - Nginx web server has been upgraded to its latest stable version 1.18.0.
- Comprehensive individual scan report in PDF
  - Support users to download a detailed scan report in PDF format on any processed individual file.
- PostgreSQL performance improvement on scan result query
  - Improved scan result query against PostgreSQL server in certain circumstance (query by hash value).
  - Enhanced mechanism in selecting relevant database connections for serving data related requests.
- PostgreSQL connections configurable setting is supported (for scaling up)
  - A new setting called " db_connection" to define a maximum number of concurrent connections allows MetaDefender Core to open and work with PostgreSQL database server.
- UNC path supported for local file scan
  - Local file scan feature now accepts UNC path specified on desired workflow rule's setting
- New processing time in JSON scan report for each stage in workflow processing
  - The JSON response for scan result will include processing time on each major workflow processing step.

- New workflow rule configuration supported for file-based vulnerability assessment
  - File-based vulnerability assessment will be supported to be managed under each workflow rule, comes with new configurable settings.
- New workflow rule configuration supported for file type analysis
  - File type analysis will be supported to be managed under each workflow rule, comes with new configurable settings.
- New configuration to skip further processing (and block) in certain malware scan results
  - Support an option to skip any further processing (e.g. sanitization) EXCEPT post action run (if configured) on every processing file if the AV scan engines' final result matched to any supported and checked scan result in the list.
- New configuration to skip further processing (and block) if file type detection failed
  - Enabling it to let MetaDefender Core skip further processing if actual file type could not be detected by file type engine for following reasons:
    - File type engine process crashed
    - Waiting tasks in queue aborted when file type engine process terminated
    - File type analysis task timed out
    - User cancelled entire processing while analyzing file type
  - When triggered, the final scan verdict will be “Not scanned”.
- Enhance existing configuration to skip further processing (and block) if archive engine is not available
  - Support an option to skip any further processing (e.g. sanitization) and fail any processing archive file if archive engine was not available for any reasons (disabled, crashed, timed out…) before trying to extract that archive file. The final processing verdict will be “Failed”.
- Proper blocked reason in JSON scan result when sanitization failed or timed out
  - With Deep CDR's workflow setting "Block files if sanitization fails or times out" is enabled, when the sanitization result meets those conditions, then now the "blocked_reason" key value would represent actual sanitization failure result (instead of other scan results such as No Threat Detected which could cause misleading to client integration).
- Override whitelist certain file types in archive file scanning
  - When triggered, the whitelist setting will be ignored if pre-configured file types belong to archive file processing.
- Proactive DLP default threshold value changed to high
  - The previous threshold value was low under Proactive DLP workflow rule settings. For those customers already tweaked this setting value, this change will not revert their pre-configured setting.
  - This change is to address false positive concern on DLP processing against files inside document by default.

- Email server configuration now accepts no authentication option
  - That helps integrate with external email server without authentication required.
- MetaDefender Core could be activated via offline mode without any network card enabled
  - In an isolated environment, we understand that your machine's network card could be disabled completely for security reasons. Hence now MetaDefender Core could be activated under that circumstance with offline yml file upload as usual.
- Log enhancement on webhook callback failure on HTTPS mode
  - When the callback failed to send to client side over HTTPS, the status code will be 444, and now the product log is enhanced to reveal more why the secure connection was not made successfully.
- Improved search performance on MetaDefender Core processing history page
  - Earlier users could observe the slowness in search by username while MetaDefender Core was running data retention.

Fixed:

- Advanced engine settings mistakenly reset back to default values
  - The issue could encounter in following circumstances:
    - Upgrading MetaDefender Core
    - Engine reloaded (disable and enable back)
    - Engine updated (and then reloaded)
- Ignition file with configuration file location import issue
  - Using ignition file for configuration file location import was not working.
- Sanitized archive compression failed with same password retainment
  - Archive sanitization could be failed with same password retain feature enabled under Deep CDR workflow settings.
- Potential memory leak issue
  - Leaking on Node component (ometascan-node) while updating engines online, or via Central Management.
- Input validation
  - Enhanced and fixes bad input validation for data queries.
- Service crashed in certain circumstance
  - Core component could be crashed when stopping the service immediately upon the product service start command.

- Update from folder feature issue
  - Update from folder trigger failed to skip in what pre-configured for update pause time frame.
- Failed to download big sanitized files
  - Failure could be exposed (failed to download) against big sanitized files in GB in size.
- Product database migration issue
  - Failed for statistics data migration.
- Proactive DLP regex validation failed for Unicode characters
  - Fixed to support encoded Unicode characters in Proactive DLP's regex (Proactive DLP version 2.5.0 or above is required)
- Sanitized archive file contained blocked watermarked file
  - The issue could only happen when Proactive DLP was used with watermarking triggered on child files inside archive sanitization.

Version v4.19.0

Release Date: 27 Aug 2020

New features:

- New Database Management System (PostgreSQL) to replace SQLite
  - PosgreSQL is now MetaDefender Core's new database management system to replace its predecessor SQLite. That expects to step by step help the product easily scale out, network based database support, gain better performance, migrate high load bottleneck and native high availability. MetaDefender Core supports users to create a local PostgreSQL server running in the box, or allow leveraging a pre-installed remote PostgreSQL server.
  - Data migration auto runs in background upon product uprade.
  - For large database migration, MetaDefender Core comes with a web-based data migration to walk users through quick steps to move all your SQLite data to PostgreSQL at ease.
- FIPS-140 security compliant
  - We are now FIPS-140 compliant with a new support for RSA186-4 on OpenSSL
- Native proxy management with authentication support
  - MetaDefender Core now will allow users to control proxy settings for product (instead of using system configuration), and also support authentication for proxy which is not possible on older Core versions.
  - That comes with a UI configuration support on MetaDefender Core management console.
- Harden Nginx web server settings for security

  - Secure MetaDefender Core web server even more based on nginx vendor guideline to protect your MetaDefender server from being vulnerable (i.e. cross site scripting, MIME sniffing, TLSver 1.1 or below forbidden)
  - Still we keep all supported functionalities working as expected.
- Nginx web server statistics (on web server report)
  - Support to enable better statistics for the HTTP Server for web server healthcheck and debugging.
- Origin client source address retrieval when running via load balancer
  - MetaDefender Core now will be able to retrieve your origin client source address even when the client communicates over a load balancer.
- Enhanced search for processing history page
  - Searching by attributes represented via corrensponding column on the list.
- Enhanced user experience on statistics page
  - Instant statistics processing data calculated and visualized on UI
- Logic improvement to handle better against sanitization timed out
  - New logic implemented on MetaDefender Core to offload concurrent tasks on Deep CDR engine, and to reduce sanitization timed out as a result.
- Enhanced log messages
  - Log events enhanced with more sufficient and clear information, easier for traceability while troubleshooting.
- Pre-check mode for file submission
  - Refuse file upload immediately when MetaDefender Core does not have enough disk space to handle, expecting to hit error 400. This is to avoid wasting upload time on big files.
- Sanitized file information appended into JSON scan result
  - Including sanitized file size and its SHA-256 hash value.
- Blacklist overidden on nested files within archive
  - New configurable setting to allow overridening blacklist enablement on nested files within an archive.
- New engine sweeper tool bundled into the product
  - New engine sweeper troubleshooting tool tailored for PostgreSQL, and now it is bundled into MetaDefender Core product (not a separate download tool).

Fixed:

- Setting inputs validation

  - Threat detection threshold, wizard password and SSO profile settings affected.

- Batch signature sometimes contained redundant characters
  - That could make signature becomes invalid.

- Processing time of nested files in archive not calculated correctly
  - Processing time of nested files could be very big due to incorrect caculation (but actually they are processed much faster).

- Statistics page to support multiple users simultaneously
  - The statistics UI now could handle multiple users query at the same time.

- Proactive DLP timeout setting mistakenly reset upon engine restart
  - The setting was reset back to default value (3 minutes).

Version v4.18.0

Release Date: 26 May 2020

New features:

- Single Sign On (SSO) Authentication
  - Additional to already-supported various authentication models (Local, Active Directory, LDAP), now MetaDefender Core also supports authentication using SSO with widen integration coverage for most of Identity Providers (IDP) via SAML 2.0 and OpenID Connect 1.0 standard support.

- Brand New MetaDefender Core API Guide (Sample Codes Available)
  - Brand new design and standardized API documentation (following OpenAPI V3 specification), auto-generated sample codes on various programming languages supported helps your API integration even easier.

- Database Defragmentation and Optimization
  - When your scan database grows big, it might cause performance degradation (e.g. timeout on client requests). Now MetaDefender Core administrators can be notified on the UI (also warning logs), and you are supported to perform database defragmentation and optimization including multiple stages to vacuum and defrag your database without loss of actual scan data.
  - As a result, your database file size could be reduced which helps boost processing performance tremendously over usage time.

- Comprehensive Statistics On Processing Data
  - Featured in an interactive UI help you gain deeper insights on your processing filtered by every workflow rule, breaking down into each file type. Last but not least, you are also supported to select time range to calculate statistics data.

- Data Reporting (Business Intelligence)
  - When enabled, MetaDefender Core will auto-sync your historic processing data to OPSWAT dedicated servers. That helps us gain more visibility on your processing load and how our product is being used, and thus we could improve our product to accommodate your use-case better. You are supported to customize which piece of information should be shared with OPSWAT, and when to share.
  - By default, this feature is disabled to respect your privacy rights and save performance impact.
- Webhook Continuous Improvement
  - New setting mode to control callback timeout and retry (configurable via REST API)
  - Stability improvement to avoid being stuck on callback and crashing on Node service
- High Load Processing Improvement
  - When running under high load, file type usually returns "Not Available" caused by various reasons. Product logic enhanced to elaborate causes, and improved stability on the product.
  - We keep working on this matter to ensure our customers have the most stable product running under high load as much as possible.
- Configurable Behavior On Archive Extraction Failure
  - Configurable settings on workflow rule to let you tweak and decide MetaDefender Core final scan verdict when a processing archive file failed to extract for some reasons.
  - The default selection on each designated extraction failure reasons (invalid file structure, extracted partially,...) will be different on each workflow rule depending on use-case characteristics. Please make sure you are aware of the new settings and adjust them accordingly tailored to your security demand.
- Archive Extraction Failure Exposure
  - Archive extraction failure reasons exposed to both REST API response and UI.
- Encryption on Archive & Document Sanitized Files
  - Help retain password protection on supported archive and document files (.zip, .7z, .pdf, MS Offices) upon sanitized successfully.
- Sanitization Forensic Details Enhancement
  - When sanitized successfully, an even more comprehensive forensic available on both UI and REST API level letting you know all processed object details (e.g. what exact hyperlink was sanitized).
- Processing File Information Enhancement (File Type Category)
  - File type category is now available on REST API response along with other already-supported file information.
- MetaDefender Core Log Rotation Experience Improvement
  - This feature is now enabled by default applicable to both upgrade and fresh install scenario.

- Workflow Rule For MetaDefender For Secure Storage
  - With the best practice to serve MetaDefender For Secure Storage use-case, we have a new dedicated workflow rule with designated configurations.
- Archive Processing Result Retrieval API Enhancement
  - Applicable to pagination fashion polling *GET /stat/log/scan?first={start_item}&size={number_of_items_next}* , now the action ran information available in JSON response
- FIPS Object Module 2.0 Bundled
  - Operating product in FIPS mode enabled on Operating System

Fixed:

- File Scanning Process Stuck
  - When the custom engines stopped its process for reasons (updating while scanning / crashed), the running scans on Core could not be finished and stayed at 95% forever.
- MetaDefender Core Service Crashed (Webhook Mode)
  - When using webhook mode, and callback can't be sent back to client, the MetaDefender Core service could be crashed.
- Memory Leak While Updating Engines Automatically
  - The memory could be leaked on *ometascan-node* process while updating engines in online mode.
- Scan Details Missing From Recursive Scan Results
  - While fetching scan results on all nested files in big archive file *GET /archive/{data_id}* the "scan_details" field from the top-level root archive was empty.
- Input Field Overflow On Management Console UI
  - Preventing invalid values putting in UI configuration controls (Deep CDR, Archive)
- Archive Timeout File Skipped For Scanning
  - None of AV engines actually scanned archive file when archive timeout occurred

Version v4.17.3

Release Date: 06 Apr 2020

New features:

- Configurable setting to run database optimization
  - Database optimization has been introduced since Core 4.17.0 to help run database queries faster. The

downside is while running (for a few seconds), Core queries hold up causing possibly timeout on client side.

  - This new setting allows users setting specific time to run database optimization task (to avoid peak hours), or just disable to prevent this task from running (to avoid performance degradation while running). Learn more how to configure: Startup Core Configuration)

- Scan database rollback mechanism
  - In some circumstances (e.g. Core crashes, out of disk/memory etc.), the atomicity of product database could be compromised causing inconsistent processing scan history returned. Rollback mechanism helps retain that atomicity of database.

- Logging improvement with configurable settings
  - Log rotation for Core, Node, Nginx web server logs (Configurable settings supported).
  - More comprehensive support package (to include engine and database info, Nginx web server info).
  - More informative log message on sanitization related tasks.
  - Sensitive info redacted (on debug level logging mode).
  - Performance impacted warning on both MetaDefender Core GUI and logs when the scan database (ometascan.db.sqlite) starts growing up big (>10 GB).

- Webhook mode continuous refinement
  - Retry to send scan results to client upon network interim disconnected.
  - Resend file scan results to client after Core service restarted.

- New download mechanism for Processing History on MetaDefender Core management console
  - Support IE / Edge web browsers to download processing history report.

- MetaDefender Drive use-case better support when engine packages corrupted
  - Support to re-new engine packages to re-download engine packages again when corrupted (due to upon unexpected reboot).

- Central Management v7 support to revert download source when unhooked
  - Respect skipped scan settings (Whitelist / Blacklist) to keep backward compatibility, and also save Core resource for processing files as well.

- RoleIDs JSON field validated when creating / modifying user
  - Effective to *POST_*/admin/user_ and *POST /user* endpoint REST APIs. Role ID value must be an array of strings according to current user guide.

- Account name value validated on Core wizard setup
  - Effective to "Admin User Setup" screen during wizard setup, "Account name" validated against special characters (e.g. @ &)

- Better support for sanitized file download when under load
  - When under load and certain circumstance with system write failed, the sanitized download on the same file might return 404 HTTP response (not found) to client. Enhanced our Core caching mechanism to ensure next time sanitization on the same file will not rely on the previous failed time.
- Minor UI changes
  - Hide "Edit Workflow" button in "Workflow Templates Management" screen
  - Remove space between date and time in "Definition date" field on "Modules" screen

Fixed:

- Data tunnel between Node and engines could be lost under high load
  - When occurred, expecting to see " process communication timed out" message repeatedly in Node log, and none of engines could be able to scan files.
- Node crashed when swapping engines during update
  - Node could be crashed under certain circumstances, applied to swapping engine instances during update.
- File processing was stuck at 95%
  - Encountered when custom engine stopped its process, all of running tasks on that engine becomes stuck, or when ClamAV engine can't return consistent scan result during its engine update.
- Overflow issue with unexpected inputs
  - Overflow value issue could occur within product causing unexpected behavior or result.
- Unexpected result with non-ASCII password protected document scanning
  - When occurred, document file could not be sanitized properly.
- Proactive DLP displayed wrong result within archive scanning
  - When occurred, Proactive DLP engine could return misleading result (Not scanned) while archive file processing result is "Sensitive Data Found".
- Session expired on IE / Edge web browser
  - When session cleared out, authenticated users could be logged out repeatedly due to session expired error on MetaDefender Core management console.

Version v4.17.2

Release Date: 03 Mar 2020

New features:

- Quarantine cleanup task no longer blocks Core service starting procedure
- Empty file submission is no longer be blocked at REST API level
  - Retain same behavior on Core 4.16.3 or older, to support back some corner use-cases from MD Kiosk and ICAP
- Custom engine initialization enhancement
  - Increased timeout to 10 minutes to support engine deployment on under-specs hardware (formerly 1 minute)
- Processing history report enhancement
  - Added "username" column to the processing history export from MD Core
- Validation mechanism on file scan and batch init REST API changed
  - When using via REST API, no longer validate session cookie, only API key header is validated when exists (same behavior on Core 4.16.3 or older)
- Configurable Proactive DLP timeout is supported
  - Support to adjust timeout for Proactive DLP handling (formerly fixed on 3 minutes)
- Respecting whitelist and blacklist configurations
  - Respect skipped scan settings (Whitelist / Blacklist) to keep backward compatibility, and also save Core resource for processing files as well.
- Response for POST /login no longer returns cookie back to client
  - When using via REST API, by default the response for POST /login no longer returns cookie back to client (same behavior on Core 4.16.3 or older, to avoid breaking F5 LTM scenario with cookie header is auto added)
- Removing failed dummy scan results on Core processing history UI due to upload failure
  - When file upload is failed for some reasons (e.g. network corrupted) between clients and MD Core, dummy record results are still available and displayed on Core processing history UI, but actually MD Core never processed those files, and client never got results from MD Core on those files. Those dummy records will be removed since this version to avoid misleading.

Fixed:

- Deadlock could possibly happen when engine update task is timed out
  - When encountered, all files happens in "Failed" result with "Not available" result for file type analysis after timeout hit (~ 70 seconds), and only Node service relaunched can bring the scanning be operational back.
- Node service could be crashed when archive engine crashed
  - When archive engine crashed for some reasons, Node service could be crashed as well (but not happened all the time)
- Core and Node service could be crashed when under high load

  - Core and Node service could be crashed when under high load
- Archive file extraction when timed out, or failed to extract, the original archive itself could not be scanned by AV engines
  - When archive extraction hits timeout or failed to extract, the original archive itself could not be scanned by AV engines
- Core could return 404 not found HTTP(S) response to client for sanitized file download API request
  - When processing the same file many times on Core, it could return 404 not found HTTP(S) response to client (e.g. MD Email) due to file sync issue between Core and Node
- Memory leak issue on Core process
  - The process ometascan could be leaked on memory with auto update mode enabled and Proactive DLP engine is enabled
- Anonymous user can't submit file scan to MD Core web scan UI
  - When not logged in, anonymous user can't submit file scan to MD Core web scan UI (error: Invalid session ID given)
- Core service can't restart due to configurations corrupted while running Proactive DLP engine
  - Core service can't restart due to configurations corrupted while running Proactive DLP engine
- Document files inside sanitized password protected archive file could not be sanitized
  - When document file is also treated as an archive file, there was a bug on archive compression level calculation to prevent document files not being sanitized inside original archive file
- Timeout on hash calculation task resulted as Blocked regardless of "override scan results classified as allowed" setting
  - When this task timed out, regardless what users set on "override scan results classified as allowed" setting, the final verdict were "Blocked"
- Webhook continuous fixes and updates
  - Duplicated callbacks returned to client
  - Support retry mode for sending callback to client (when client is temporarily unresponsive etc.)
  - Enhance validation callbackurl header against IP version 6 and domain format
- Minor UI fixes
  - Added margin to bottom edge of scan result UI
  - Name of rights under user management did not match to Inventory

Version v4.17.1

Release Date: 06 Jan 2020

New features:

- Archive extraction details
  - Available on both scan result UI and
- Advanced engine configurations enhancement
  - UI interactive and schema based for advanced engine settings
- Proactive DLP engine integration enhancement
- MetaDefender Cloud integration enhancement
  - MetaDefender Cloud API version 4 upgraded
- Engine integration enhancement to avoid product crash
- User validation update for file and batch processing
  - File scan and batch init endpoint API is now validated on API key input when that key information is available.
- More ready for adding password back to sanitized archive and document files
- Minor UI update

Fixed:

- Core could become unavailable to clients when under high load
  - Data communication channel between Core and Node service could be broken when under high load
- Processing giant files (> 50 GB) could be stuck at hash calculation
  - Hash calculation with pre-set timeout value (10 minutes) could be exceeded when processing giant files (> 50 GB) and then stuck at 5% forever
- Scan could be failed with "not available" result for File type analysis when under certain circumstances
- Memory leak issue on Core process
- Temporary files not cleaned up when archive extraction timed out
- Webhook continuous fixes and updates
  - Redundant warning log messages populated even when not using webhook mode
  - Core could be crashed itself when trying to close a not-found bath with callback
  - Callback sent to client with wrong status when Core is restarted
- Visibility level smaller than full details might break batch result display
- Minor UI fixes

- The field "File Password" on the file processing UI not cleared up after empty file selected

Version v4.17.0.1

Release Date: 27 Nov 2019

Fixed:

- Deadlock issue on batch handling
  - Under certain circumstances, a deadlock issue could encounter locking database from being queried (timeout on REST requests)

Version v4.17.0

Release Date: 14 Nov 2019

New features:

- Callback URL (Webhooks) for file and batch scanning (to avoid polling result from client)
  - Support for individual file and batch scanning to eliminate polling mechanism i.e. MetaDefender Core will notify client based on designated / configurable callback URL whenever an individual scan finished or a batch can be closed.
- Security enhancements
  - Harden MetaDefender Core management console against security vulnerabilities found on pen-test's result
- Log correlation from parent archive file to child files
- Comprehensive failure reason on archive extraction (available on JSON response)
- Sanitization output name on password protected document fully respects value set on UI
  - No longer appended with fixed value "decrypted_document" in output name
- UTF8-encoding password for file scan request via REST
  - File scan REST API now supports "archivepwd" header with encoding password
- Total number of files inside archive (all recursive levels), available on scan result UI
- Archive scanning enhancements
  - Better integration logic with archive engine
  - Support empty folder inside archive engine
  - Not try to extract archive file if extracted size exceeded is anticipated
- Database query optimization
- Processing input refinement

  - Empty file scan request no longer is accepted at API level
- MetaDefender Core's nginx log location no longer requires double backslashes
- UI enhancements
  - Password field supported for password-protected archives or documents on the UI (web scan)

Fixed:

- Relaunching Proactive DLP engine process after timeout could crash Node service
- Nginx custom configuration file and certificates is unexpectedly erased when upgrading MetaDefender Core
- Overall failure on scan could encounter when system goes wrong while analyzing file type
  - When something wrong occurred while analyzing file type, the scan process could be immediately stopped and ended up as overall failed.
- Override scan result setting did not apply properly to empty batch
- Changes on workflow template could interfere Core service
  - Excluding engines in workflow template could make Core service failed to start
- Return incorrect REST response code when closing batch with invalid API key
- Fail to create local user directory under certain settings
  - Creating "Local" user directory type could be failed when "Enhance password policy" setting is unchecked
- Nginx access log location customized on registry is not retained when upgrading MetaDefender Core
  - When upgrading MetaDefender Core, nginx log location (nginx_logfile) could be unexpectedly reverted back to default value, not retained to what users configured.
- Minor UI fixes
  - Some display and hyperlink minor issues related to table and navigation
  - Hitting cancel batch button on UI causing error

Version v4.16.3

Release Date: 16 Oct 2019

New features:

- Support new header (metadata) for file submission API
- Enhance MetaDefender Core service starting procedure
- Enhance engine update procedure
- Remove restriction on Core version retrieval REST API

Fixed:

- MetaDefender Core service on Linux could not be started when running on FIPS mode
- MetaDefender Core service could be unexpectedly restarted when engines repeatedly crashed
- Uninstalling MetaDefender Core did not terminate its processes properly (nginx)
- Password protected document could not be decrypted properly for data sanitization
- Uninstalling MetaDefender Core did not clean up its leftover data folder

Version v4.16.2

Release Date: 10 Sep 2019

New features:

- Restrict APIs based on user roles (configurable)
- Support displaying and filtering username on processing history UI
- Enhance logging with YARA matched rules appended
- Upgraded nginx web server component to latest version 1.16.0
- Add new scan result - Unsupported file type
- Refined JSON output when users want to quarantine items which are already in quarantined folder
- Updated UI (minors)

Fixed:

- In-progress files could be deleted mistakenly, causing failures when scanning
- Engines repeatedly disable and re-enable
- File processing could be stuck until archive timeout value reached
- Non UTF-8 characters were not displayed correctly when exporting process history via UI
- Dependency installation issue on Ubuntu 18 & Debian 9

Version v4.16.1

Release Date: 12 Aug 2019

New features:

- Supported to pin & unpin engines and their database on the UI to prevent auto update being applied
- Gently handled timeout on Archive and Deep CDR engines
- New logging mode for archive processing troubleshooting
- Enhanced logic for non-archive file processing
- Limited number of characters on some applicable text fields on the UI
- Enhanced security with unquote service exploit

Fixed:

- Node crash issue when under high load
- Issue with resource manager with in-use temp files
- Memory leaking issue on archive engine process
- Memory leaking issue on Node process
- Batch handling issue causing failure on batch
- Stuck scan issue at 5% when parallelcount_7z_extract is set with definitive number
- Detection issue on Proactive DLP engine with regrex rule applied
- UI issue where YARA result is not displayed
- UI visibility issue on Internet Explorer (IE) web browser
- Some other minor UI issues
- Wrong timezone set on exported CSV scan report

Version v4.16.0

Release Date: 08 July 2019

New features:

- Proactive DLP engine (ver 2.0) integration
- Password policy enforcement
- Support archive partial sanitization for Vault and Email integration
- New REST API for local update server source
- Better handle archive sanitization timeout
- Support configurable settings for archive extraction and compression parallel count
- Enhance syslog message format
- Retouch UI
- Better logging with timeout on engines
- Enhance logic to apply engine definition files

Fixed:

- Wrong outcome when archive engine process unexpectedly stopped
- Wrong UI result on sanitization timeout
- Memory leak issue on engine package uploading

Version v4.15.2

Release Date: 19 June 2019

Fixed:

- Stability issue
  - Potential deadlock issue on batch scan handling prevents querying batch information
- Usability issue
  - Enhanced error log messages when the engine process is terminated due to engine timeout
  - Exposed log messages on warning level when there is an archive extraction failure

Version v4.15.1

Release Date: 06 June 2019

New features:

- Partial sanitization use-case for archive file types
- Clarified error messages for terminated engine processes
- New REST API for cleaning up idle batch scans
- UI improvement
- License EULA update

Fixed:

- Stability issue
  - Potential memory handling issue that could cause the node service to crash
  - Empty and read-only files are no longer extracted
- Usability issue
  - Not able to remove abandoned temp. files of archive files when they are empty and read-only
- Security issue
  - AD user credential is not masked properly on the audit log while sending over to AD server for authentication
- Scanning batch REST API issues
- Engine custom configuration
- UI issues
  - Dashboard refresh button sometimes did not work as expected
  - List of processing records didn't show when changing "number records per page" while not staying at first page
  - Not user-friendly error messages when adding duplicate hashes to a backlist

- Typos on the UI

Version v4.15.0

Release Date: 06 May 2019

New features:

- Data Sanitization details displayed on Core management console
- User password recovery and reset enforcement
- API rate limiting
- Support Windows Server 2019 (The support is still on beta)
- Suspicious results returned by engines are now configurable to be handled as a different circumstance (infected, ignore)
- Improve usability
  - Return zero for definition dates on non-AV engines' database
  - "Select all" option added to the Data Sanitization page
- Improve handling on node
  - Improve cleanup mechanism on nodes to avoid deleting files in use
  - Improve validation process when starting the node service, support to try creating temp. folder with a configurable timeout

Fixed:

- Fixed stabilization issues that possibly caused Node services to crash
- Scan batch API closing issues
  - No longer returns total time of -1 in response
  - Should not randomly fail due to " 400 - One or more scan is still in progress" even when all linked scans already finished
- Upgrading Core when installed in a non-default installation path prevented users from choosing another folder path by mistake
- UI issues
  - The "Process File" button no longer disappears in case of sanitization failed
  - Max recursive level under archive handling tab must equal 1 or greater
  - non-Unicode file name displayed on web scan UI encoded properly
- Sanitizing empty archive file no longer returns failed

Version v4.14.3

Release Date: 01 Apr 2019

New features:

- Support built-in integration with OPSWAT Central
- New setting for archive sanitization timeout
- Add process time field into CSV exported history report
- Effectively wipe out necessary data from support package
- Revamp Inventory UI page with "Utilities" group
- More relevant REST error message for scan request where file is non-existed / inaccessible
- Syslog message for scan-finish event more comprehensive
- Consolidated scan info for archive scan result fetching
- Add libcurl4 as alternative dependency to libcurl3 for better support on Ubuntu 18.04
- Outputs and indicators for Threat Intelligence feature on Quarantine UI page more relevant and informative

Fixed:

- Node becomes unstable under high load processing
- Closing batch with ongoing scans could result in failed verdict on batch
- Inconsistent behavior with password protected document
- Temporary files are not cleaned up when cancelling an ongoing scan
- Inconsistent returned error message between batch and file scanning via REST
- DLP verdict returns incorrect value for some cases
- Logs in support package did not handle non-Unicode characters

Version v4.14.2

Release Date: 28 Feb 2019

New features:

- New result page, n ew look and more informative badge

Fixed:

- Engine configurations could not be saved
- Make error message more relative for case where file exceeded the size limit
- Pinning engines and their databases independently

Version v4.14.1

Release Date: 31 Jan 2019

Fixed:

- Missing "pinned" option from "/stat/packages" JSON response
- Inconsistent "progress_percentage" and "result" values
- Hash validation (blacklist/whitelist)
- Upload performance

Version v4.14.0

Release date: 19 Dec 2018

New features:

- Send quarantined files to MetaDefender Cloud for scanning
- Automation support:
  - Support ignition file to automate the welcome wizard
  - Configuration API functions have been documented
- Enhanced password policy can be enabled for local users
- Files with Failed to sanitize result can be set to be blocked

Version v4.13.2

Release date: 21 Nov 2018

New features:

- Tiles on Dashboard are linked to the corresponding pages
- More options to filter Processing History (Post Actions and CDR)

Fixed issues:

- In case of an engine hangs, the communication channel is blocked between the Node and the Core, so more engines can time out
- Clean-up mechanism removes files still in use
- Various engine handling issues

Version v4.13.1

Release date: 31 Oct 2018

Fixed issues:

- YARA and DLP tasks are not stopped on cancelling a processing

- Batch processings cannot be cancelled via web management console
- "Can't process shared resource file" error message did not contain the file name

Version v4.13.0

Release date: 16 Oct 2018

Important:

- YARA engine integration

New features:

- Processing history entries can be colorized
- Files can be marked as suspicious if less than a given number of engine mark it as infected
- Processings can be cancelled via web management console
- Default rules are added for MetaDefender Email Security
- Bulk operations in quarantine

Fixed issues:

- Extracted files are left behind
- On Debian based systems, on upgrades, engines are deleted and disabled engines are re-enabled

Version v4.12.2

Release date: 3 Oct 2018

Fixed issues:

- In case of archive processing, sometimes clean-up mechanism removes some extracted files before processing is finished

Version v4.12.1

Release date: 26 Sept 2018

New features:

- Files can be whitelisted/blacklisted by their checksums
- More specific log entries for CDR

Fixed issues:

- Details of scan result for nested archives (for the file itself not for the content) is not propagated to the top level
- The value, set in "MAX TOTAL SIZE OF EXTRACTED FILES" is handled incorrectly
- Older configs cannot be imported into v4.12.0

Version v4.12.0

Release date: 15 Sept 2018

Important:

- Data Loss Prevention functionality

New features:

- Possibility to set the number of engines that required to start file processings (per workflow)
- Possibility to exclude engines from processings (per workflow)
- Improved user interface performance
- Possibility to blacklist/whitelist files by file types besides file type groups
- Re-designed workflow tab list appearance
- Possibility to set timeout for sessions regardless of user activity

Fixed issues:

- On Node details page, every issue appears multiple times
- Despite not detecting any vulnerabilities, the vulnerability tab appears
- On hash lookup page, empty hash can be searched
- Sanitized output file name validation can cause user interface stalled

Version v4.11.3

Release date: 30 Aug 2018

Fixed issues:

- Whitelist page under Inventory menu does not exist (only UI issue)

Version v4.11.2

Release date: 29 Aug 2018

New features:

- The access_log Nginx directive now can be overridden
- The parallel count parameter now can be set per engine
- Minor changes on user interface for better user experience

Fixed issues:

- A critical CSV injection vulnerability in the CSV export functionality (issue reported by Wojciech Reguła, SecuRing)
- Archives can be sanitized even in case of partial processing (e.g. exceeded archive size, exceeded archive file

number)

- In some cases, blocked results can be overwritten by an allowed result with higher priority
- Inconsistent operation of MetaDefender Cloud integration
- Typos on the user interface
- Abandoned files left behind after processings

Version v4.11.1

Release date: 8 Aug 2018

Fixed issues:

- Unexpected Core and Node service restart in some corner cases
- Using remote syslog server slows down the product in case of missing PTR record in DNS
- Empty files are skipped in archives
- Incomplete archive extraction issue happened on heavily overloaded systems

Version v4.11.0

Release date: 11 July 2018

New Features:

- Exceptions (by mime-type) from whitelist/blacklist
- New engine page called Technologies
- Support for user-friendly engine configuration (depends on the engine version)
- Welcome wizard

Fixed issues:

- Slow clean-up mechanism
- Abandoned files after uninstall in Windows
- Temporary files are left behind after processings
- Wrong sanitized output file name in some cases
- Default workflows can be overridden on config import
- Core crashes

Version v4.10.2

Release Date: 27 June 2018

Fixed issues:

- Uninstall not properly cleans the system
- The "whitelisted" and "blacklisted" results are overriden by "infected" result
- Node crashes
- Inconsistent results in case of archive processing: In case of processing an archive more times, the result may be different by cases (infected/exceeded archive file number/exceeded archive size)

Version v4.10.1

Release Date: 23 May, 2018

New features:

- Data Sanitization engine time-out and retry count is now configurable
- REST API: process info contains the name of the last scanned file when scanning archive file types
- REST API: Configurations that may change the final scan result since the time of processing will be included in the process info response (i.e., outdated definitions)
- Hash based result lookups can be filtered by rule name

Fixed issues:

- Sanitized DB integrity issue
- On the dashboard, category names of doughnut charts were truncated
- In case of archive processing, the "Not scanned" result to a file is not propagated to a higher level (overall verdict)

Version v4.10.0

Release Date: 2 May, 2018

Important:

- Added support for the LDAP directory type
- Syslog messages can now be sent to multiple log aggregators
- MetaDefender installers no longer use eicar test files

New features:

- AD and LDAP directories can now be configured with multiple servers
- Sanitization failures are marked with a badge in the scan session summary
- Admin's will be notified if a third party solution is blocking MetaDefender from working as expected
- Users can now be granted API keys manually
- Paginated archive results
- HTTPS can now be enabled from web management console

Fixed issues:

- Improved license status info
- In some cases, sanitized files had faulty names
- Suspicious scan results were not always at the top of the list in archive file types
- Inappropriate handling of user rights in the Whitelist page
- AD group members did not have user profiles
- Misleading license information

Version 4.9.1

Release Date: 28 February, 2018

New features:

- New-looking user interface
- Workflows based on the default one (not edited by workflow editor) will be kept and upgraded on version upgrade in the future
- It is allowed blacklisted/whitelisted files to be processed

Fixed issues:

- Security zone: IP address validation
- Cancelled batches are displayed as in-progress
- Removing certificates from the inventory caused policies to disappear
- Memory leak in Node
- Access via Active Directory is not logged
- Sluggish pages under Policy menu

Version 4.9.0

Release Date: 13 December, 2017

New features:

- IPv6 support
- Global whitelist by hash
- Whitelist by file type group
- Display more security related information on dashboard
- Changed default port for external nodes to 8007
- New default security rule for Metadefender Secure File Transfer (SFT)
- Performance tuning of processing history

- Improved resource handling on Node
- On Linux, multiple nginx worker processes for better scaling

Fixed issues:

- Upgrades overwrite existing configuration (IP, port, etc.)
- Resource folder clean up after data sanitization
- Update timing settings affect manual updates
- Poorly handled invalid update files
- Poorly handled UTF-8 characters in output file name for sanitized files
- /hash API can give "in progress" result

Version 4.8.2

Fixed issues:

- Fixed a memory leak caused by failed update download
- Fixed a possible crash issue at Scan history manual cleanup in case of high load
- Fixed a memory leak in case of recurrring failed database deployment on Node

Version 4.8.1

Release Date: 5 October, 2017

New features:

- Improved engine/database update distribution to nodes
- Improved archive extraction limit handling
- Improved engine monitoring
- More precise time duration measurement for requests
- API for canceling scans (file/batch scans)
- Option to disable archive extraction of office documents
- For batch scans, certificate validity interval can be set
- Improved scan result badge

Fixed issues:

- Fixed issue of scans stuck in "in progress" state
- Fixed possible product crash during archive scanning
- Fixed update bug where incorrect packages left behind
- Fixed failed quarantine handling

- Fixed handling unavailable engine during scans
- Scan result JSON now contains file name in UTF-8 format
- Limited number of parallel Post Action and External Scanner scripts
- Archive handling parameters now have upper bound
- Improved archive handling
- Archive related failure handling

Version 4.8.0

New features:

- Quarantine for blocked files
- Scanning files in batch (REST API)
- Certificate and key handling for scan batch signing
- Configurable sanitized file name
- Post action commands gets the result JSON with final verdict included
- Increased scan history export interval
- Improved archive bomb handling
- Added eng_id to scan_results.scan_details (REST API)
- Showing in-progress files in "extracted files" list of archives
- Added "scan_all_result_a" into "extracted_files" (REST API)

Fixed issues:

- Fixed case insensitive username comparison in Active Directory integration
- Process workflow revamped (post actions run every time)
- Fixed non-updated policy user interface after added new user roles
- Fixed handling of database upgrade errors in linux package installers
- Fixed error handling when scan target was sent in the body and via filepath (/file REST API)
- Fixed disconnected ghost node issue displayed on user interface

Version 4.7.2

Issues fixed:

- Fixed bug that could cause policies to not contain any elements and forbid user to create new items
- Fixed bug where Core could download older version of engines where newer one was already downloaded

Version 4.7.1

Issues fixed:

- Fixed upgrade of scan configuration
- Fixed ghost nodes appeared on Inventory Nodes page

Version 4.7.0

New features:

- Active Directory integration
- Custom post actions
- Redesigned user interface
- External (customer developed) scanner integrations
- Policies export/import
- Archive sanitization
- Individual log message level override
- Aggregated archive scan result in Scan History
- Self-lockout protection, admins can not delete themselves
- gzip and base64 encoding now supported on /file REST API
- Able to navigate through archive hierarchy
- Timezone changed to local in log messages

Issues fixed:

- Fixed scanning of .lnk files on Windows
- Fixed blacklisting of Unicode filenames
- Automatically downloads packages again if the previous download failed
- Fixed order of extracted files on scan details view
- Fixed rare temporary file leak during archive scan

Version 4.6.3

Issues fixed:

- Improved scan result fetching performance for big archives

Version 4.6.2

Issues fixed:

- Improved archive extraction performance

- Fixed a race condition in /file/<data id> REST API that could provide access error in some cases
- Fixed advanced engine config reload for Data sanitization engine
- Fixed login issue which happened when many login request was initiated concurrently
- Fixed calculation of extracted file count

Version 4.6.1

New features:

- List of path for local filescan can be blacklist / whitelist with specific error message on REST

Issues fixed:

- Invalid external Node listening IP/port config stops product startup
- Connection to remote syslog is reactivated on network error
- If user has no right to use a rule, following rules in order will still be checked
- sending HEAD request where GET should have been sent will not lead to product crash
- Ensure resource file deletion on Microsoft Windows when a scan engine locks file further than expected
- Scan history CSV export uses comma as separator
- Fixed potential Node service crash when stopping during scanning
- More specific error message when uploaded file size limit exceeded
- Fixed a rare race condition in update downloader component
- Fixed login issue when Core v3 like URL is used by the admin

Version 4.6.0

New features:

- Multiple user roles introduced with different access rights
- Scan Agent has been renamed to Scan Node
- Role (user group) based rule availability configuration
- Role based scan result visibility with different level of details exposed
- Ability to export part of scan history into STIX/Cybox format
- Ability to export part of scan history into CSV format
- Filter on rule and source added into Scan history
- Configurable lockout feature against brute force login attack
- Official support introduced for Ubuntu 16.04
- Detection threshold (suppress threat detection if less then X engines detected a threat)
- Custom engine configuration via user interface

- Free text search functionality in user guide
- Suspend engine testing/deployment to Node when 3rd party security software blocks access to malware files
- Successful login / unsuccessful login / lockout events are logged
- Option to send engine issue count info during update
- [REST API] /file/{data_id} response for scan results now contain process info block for extracted files
- Initiating local scan is faster as no wait for hashing is required

Issues fixed:

- [REST API] /file/{data_id} blocked reason change to mirror V3 API
- Fixed handling of archive extraction depth
- More flexible and stable internal database upgrade when upgrading product
- Custom engine update timeout increased to one hour to deal with slow engine updates
- Archive engine fixes (non-ASCII filenames in archive)
- Engine handling fixes, improved handling of engine deinitialization
- More precise engine cleanup when removing engines
- Fixed bug where random connections were rejected every 2 min
- Fixed bug regarding updates handling (conflicting names)
- Filesize is now correctly displayed on scan result user interface
- Support package generator now includes auditlog db

Version 4.5.1

Issues fixed:

- Fixed possible crash of Agent when there is database which is handled by engine
- Fixed possible crash of Core that could occur when updating a package

Version 4.5.0

New features:

- Data Sanitization of files to protect against unknown threats
- Filetype mismatch detection
- Improved user interface responsiveness for small screens
- Real filetype based blacklist option in rules/workflows
- Improved licensing for offline deployments
- Added product specific proxy settings in the Linux version
- Advanced configuration for allowed/blocked file scan result types

Issues fixed:

- Fixed local scan option user interface for new rules
- Fixed Scan History auto cleanup collision with manual cleanup
- Potential issue fixed for update file upload
- /apiversion interface is added to easily determine REST API compatibility level

Version 4.4.1

New features:

- Added several features/improvement for better Metadefender Kiosk integration
- Full audit log about any configuration changes via Web user interface or REST API
- Able to disable applying update in user configurable time periods
- Core can act as an update source for OESIS product line
- Detect if the analyzed binary is a part of any vulnerability detection
- Improved scan engine status monitoring and auto recovery
- Custom directory can be set for storing temporary files
- Able to set up apikey for every user for easier REST API integration
- Improved hardware detection in license component

Issues fixed:

- Fixed message content format in Windows Event log
- Fixed system wide proxy usage on Windows
- Improved browser cache handling in case of product upgrades
- Fixed a path specification issue in local file scanning feature on Windows
- Fixed engine counting on Agent details page (do not count utility type engines)
- Fixed lost agent connection handling
- Fixed handling of unsupported Transfer-Encoding on REST API
- Patched internal nginx web server to fix CVE-2016-4450
- Fixed archive timeout handling and user interface
- Fixed scan results in case of archive related findings
- Improved logging of proxy usage
- Improved handling of slow file uploads
- Detailed logging in case of SSL connection issues
- Improved auto-recovery of engines running under Emulated Windows

Version 4.3.0

New features:

- Introduced official support for Microsoft Windows 7 or newer and Microsoft Windows Server 2008 R2 or newer
- Added offline update picker feature to make it easy to apply offline updates without user interaction or scripting
- Able to scan local files stored on server without transferring the content via REST API
- Added hardware related info into generated support package
- Created a framework in Linux version to be able to run Windows scan engines on Linux server
- Option added to log to a remote syslog server
- Inventory / Scan Agents page extended with more detailed agent information
- Parameter workflow renamed to rule in some REST APIs
- Improved system issue notification on Web Management Console
- Added detection of 3rd party anti-malware products that break operation of Metadefender Core
- Improved scan performance of various engine integrations

Issues fixed:

- Improved documentation of multiple REST APIs
- Fixed failed scans during some engine or database update
- Removed unmeaningful database age display of non-anti-malware engines

Version 4.2.0

New features:

- product name has changed to Metadefender Core
- able to use scan results from metadefender.com
- workflow options can be configured from Web Management Console
- workflow options can be overridden from rule editor window
- support for system wide HTTPS proxy
- it is possible to configure maximum file size of scanned files
- filtering security rule by user agent is now possible
- eliminate limitations on the size of scanned files
- improved scan related log messages
- deployment can now be deactivated on the License page
- automatic deployment reactivation of online installations if license becomes invalid
- Metascan v3 URLs (/management and /metascan_rest) are now redirected to the proper v4 URLs
- check disk space before/during scan requests

Issues fixed:

- fixed encrypted communication error with activation server on Ubuntu 12.04
- fixed temporary folder cleanup
- fixed support data collector scripts
- do not download database without the corresponding engine package
- number of engines and maximum file size is now reflect the current status

Version 4.1.0

New features:

- https support for REST API and for Web Management Console
- update history to track every database/engine change
- new option to globally disable or enable specific scan engine
- reworked result page for archive files
- user guide is available within the product
- no scan downtime while updating engine/database (if engine supports)

Issues fixed:

- more descriptive communication error messages instead of error codes in logs
- proper handling of update download issues
- fixed handling of scan engine crashes
- fixed manual update package upload
- fixed unwanted warning message after successful activation

Version 4.0.1

New features:

- new script to help log collection for support
- inform the user if browser is not HTML5 compatible
- show a spinner if loading a page takes too much time
- support lower screen resolution for web interface
- support for non-ascii character filenames in archives

Issues fixed:

- fix stability issue in update downloader
- optimize database queries

- do not check for updates at product startup if auto update is off
- fixed a page auto refresh issue with Internet Explorer

Version 4.0.0

New features:

- Able to to monitor Metascan v4 for Linux instances
- Able to to monitor Metascan v3 for Windows instances
- Collect Files scanned and Infections found stats from managed instances
- Deploy scan engine database updates to Metascan v3 for Windows instances
- Deploy scan engine and scan engine database updates to Metascan v4 for Linux instances

Metascan release notes

V 4.2.19

Release date: 11/12/2026

- Engine update for Bitdefender - Windows
- Engine update for ClamAV - Linux
- Upgrade ClamAV Linux updater
- Engine update for ESET - Linux
- Engine update for Webroot - Linux
- Engine update for Webroot - Windows
- Engine update for Aurora - Linux
- Engine update for Aurora - Windows
  - x7.6 - x2 time performance improvement following curl dependency upgrade

V 4.2.18

Release date: 11/11/2026

- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.6.1
  - Added PDF embedded object scanning
  - Added byte-pattern signatures

  - Fixed certificate detection for signed PEs
- Upgrade OPSWAT Predictive Alin AI Windows SDK to 3.6.1
  - Added PDF embedded object scanning
  - Added byte-pattern signatures
  - Fixed certificate detection for signed PEs
- Upgrade OPSWAT Predictive Alin AI Cloud Linux SDK to 3.6.1
  - Added PDF embedded object scanning
  - Added byte-pattern signatures
  - Fixed certificate detection for signed PEs
- Upgrade OPSWAT Predictive Alin AI Linux updater
- Upgrade OPSWAT Predictive Alin AI Windows updater
- Upgrade OPSWAT Predictive Alin AI Cloud Linux updater

V 4.2.17

Release date: 11/4/2026

- Upgrade Xvirus Anti-Malware Linux SDK to 5.1.3
  - New config options for scan only executables and Cloud scan
  - Refactored the heuristics engine
- Upgrade Xvirus Anti-Malware Windows SDK to 5.1.3
  - New config options for scan only executables and Cloud scan
  - Refactored the heuristics engine
- Upgrade ESET Windows updater

V 4.2.16

Release date: 10/28/2026

- Upgrade Vir.IT ML Windows SDK to 9.5.106
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.106
- eEngine update for Emsisoft Windows
  - Heuristic Scan config option is always enabled

V 4.2.15

Release date: 10/22/2026

- Upgrade Avira Linux SDK to 4.15.32
- Upgrade Avira Linux updater
- Upgrade Avira Windows SDK to 4.15.32
- Upgrade Avira Windows updater
- Upgrade Bitdefender Linux SDK to 3.0.1.418
- Upgrade AhnLab Linux SDK to 3.30.2.3
- Upgrade AhnLab Windows SDK to 3.30.2.3
- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.6.0
- Upgrade OPSWAT Predictive Alin AI Windows SDK to 3.6.0
- Upgrade OPSWAT Predictive Alin AI Cloud Linux SDK to 3.6.0
- Upgrade Gridinsoft Windows SDK to 1.0.255
- Upgrade Gridinsoft Windows updater

V 4.2.14

Release date: 9/16/2026

- Upgrade Gridinsoft Windows updater
  - Delta check added
- Engine update for Trellix - Linux
  - Result mapping adjusted
- Engine update for Trellix - Windows
  - Result mapping adjusted
- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.5.0
- Upgrade OPSWAT Predictive Alin AI Windows SDK to 3.5.0
- Upgrade OPSWAT Predictive Alin AI Cloud Linux SDK to 3.5.0

V 4.2.13

Release date: 8/31/2026

- Upgrade ClamAV Linux SDK to 1.4.6
  - Fixed a security issue
- Upgrade ClamAV Linux updater

- Upgrade ClamAV Windows SDK to 1.4.6
  - Fixed a security issue
- Upgrade ClamAV Windows updater
- Upgrade Gridinsoft Windows SDK to 1.0.252
- Upgrade Gridinsoft Windows updater
- Upgrade Lionic Linux SDK to 8.31
- Upgrade Lionic Windows SDK to 8.31
- Upgrade Avira Linux SDK to 4.15.31
- Upgrade Avira Linux updater
- Upgrade Avira Windows SDK to 4.15.31
- Upgrade Avira Windows updater
- engine update for Webroot - Linux
  - Updated OpenSSL to the latest version
- engine update for Webroot - Windows
  - Updated OpenSSL to the latest version
- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.4.3
  - Added support for per-filetype models
  - Added support for embedded-script IOC scanning
  - Added support for tuning deepscan
  - Added support for the XML file family
- Upgrade OPSWAT Predictive Alin AI Windows SDK to 3.4.3
  - Added support for per-filetype models
  - Added support for embedded-script IOC scanning
  - Added support for tuning deepscan
  - Added support for the XML file family
- Upgrade OPSWAT Predictive Alin AI Cloud Linux SDK to 3.4.3
  - Added support for per-filetype models
  - Added support for embedded-script IOC scanning
  - Added support for tuning deepscan
  - Added support for the XML file family

V 4.2.12

Release date: 8/18/2026

- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.4.2
- Upgrade OPSWAT Predictive Alin AI Linux updater
- Upgrade OPSWAT Predictive Alin AI Windows SDK to 3.4.2
- Upgrade OPSWAT Predictive Alin AI Windows updater
- Upgrade OPSWAT Predictive Alin AI Cloud Linux SDK to 3.4.2

V 4.2.11

Release date: 8/3/2026

- New scan engine in Linux: BKav Pro
- Multiple dependencies upgrades for the listed engines
  - engine update for AhnLab - Linux
  - engine update for Aurora - Linux
  - engine update for Avira - Linux
  - engine update for Bitdefender - Linux
  - engine update for BKAV - Linux
  - engine update for ClamAV - Linux
  - engine update for CMC - Linux
  - engine update for CrowdStrike - Linux
  - engine update for ESET - Linux
  - engine update for IKARUS - Linux
  - engine update for K7 - Linux
  - engine update for Lionic - Linux
  - engine update for OPSWAT Predictive Alin AI - Linux
  - engine update for OPSWAT Predictive Alin AI Cloud - Linux
  - engine update for Quick Heal - Linux
  - engine update for RocketCyber - Linux
  - engine update for Sophos - Linux
  - engine update for TACHYON - Linux
  - engine update for Trellix - Linux
  - engine update for Varist - Linux
  - engine update for Webroot - Linux

  - engine update for Xcitium - Linux
  - engine update for Xvirus Anti-Malware - Linux
- Engine update for Huorong - Windows
  - Upgraded multithread support, major scan speed improvement using more threads
- Upgrade ClamAV Linux SDK to 1.4.5
  - CVE fixes for CVE-2026-20217, CVE-2026-20213, CVE-2026-20216, CVE-2026-20214, CVE-2026-20243, CVE-2026-20215, CVE-2026-20244
  - Hardened `clamscan`, `clamdscan`, and `clamonacc` quarantine actions against time-of-check/time-of-use races
  - Raised the minimum required CMake version to 3.17
- Upgrade ClamAV Linux updater
- Upgrade ClamAV Windows SDK to 1.4.5
  - Resolved multiple security vulnerabilities
- Upgrade ClamAV Windows updater
- Upgrade AhnLab Linux SDK to 3.30.1.4
- Upgrade AhnLab Windows SDK to 3.30.1.4
- Upgrade Sophos Linux SDK to 3.98.2-4.5.1
  - Enabled PDF scanning by default in SUSI mode
  - Added support for reporting all TFT types
  - Improved handling of Unicode characters
- Upgrade Sophos Windows SDK to 3.98.2-4.5.1
  - Enabled PDF scanning by default in SUSI mode
  - Added support for reporting all TFT types
  - Improved handling of Unicode characters

V 4.2.10

Release date: 6/2/2026

- Upgrade IKARUS Windows SDK to 6.5.5
  - Redesigned mailbomb handling
  - Tampered archive parsing fixes
  - Improved unpacking

  - Improved scanning for different filetypes: Packed executables, Rust binaries, Office files, .NET executables
  - Script identification enhancements
  - Windows long filename support
  - Multiple memory and performance optimizations
- Upgrade Gridinsoft Windows SDK to 1.0.246
- Upgrade Gridinsoft Windows updater
- Upgrade Bitdefender Windows SDK to 3.0.1.405
- Upgrade Xvirus Anti-Malware Windows SDK to 5.1.2
- Upgrade BKAV Windows SDK to 8.2.42
  - Fixed the crash issue when scanning some PE files
  - Fixed a memory leak issue when scanning compressed files
  - Fixed the crash bug when scanning some Office files
  - Increased the malware detection rate
  - Decreased the false positive rate
- Engine update for ESET - Windows
  - PUP/PUA config separated to detection of unsafe / unwanted / suspicious applications
- Engine update for Lionic - Windows
  - Crash fix at initialization
- Upgrade Avira Windows SDK to 4.15.30
- Engine update for K7 - Linux
  - a crash bug fixed

V 4.2.9

Release date: 5/19/2026

- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.0.0
- Upgrade OPSWAT Predictive Alin AI Windows SDK to 3.0.0
- Upgrade OPSWAT Predictive Alin AI Linux SDK to 3.0.0
- Upgrade CrowdStrike Linux SDK to 1.13.1
- Upgrade CrowdStrike Windows SDK to 1.13.1

V 4.2.8

Release date: 5/18/2026

- Engine update for CrowdStrike - Linux
  - New config option - `Prefilter unsupported filetypes`
- Engine update for CrowdStrike - Windows
  - New config option - `Prefilter unsupported filetypes`
- Engine update for Aurora - Linux
  - New config option - `Prefilter unsupported filetypes`
- Engine update for Aurora - Windows
  - New config option - `Prefilter unsupported filetypes`

V 4.2.7

Release date: 4/20/2026

- Upgrade Bitdefender Linux SDK to 3.0.1.399
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.103
  - Improvements on the heuristic engines for detecting MSIL malwares
- Upgrade Vir.IT ML Windows SDK to 9.5.103
- Upgrade AhnLab Linux SDK to 3.30.0.6
- Upgrade AhnLab Windows SDK to 3.30.0.6
- Upgrade Gridinsoft Windows SDK to 1.0.245
- Upgrade Gridinsoft Windows updater
- Upgrade SentinelOne Linux SDK to 26.2
  - The Nexus SDK v2 API is now available
  - Improved detection of PE, ELF, MACH-O, DotNet, PDF, DOC files
  - Improved runtime performance
- Upgrade SentinelOne Windows SDK to 26.2
  - The Nexus SDK v2 API is now available
  - Improved detection of PE, ELF, MACH-O, DotNet, PDF, DOC files
  - Improved runtime performance

V 4.2.6

Release date: 4/20/2026

- Upgrade Trellix Linux SDK to 6900
  - Switched to native STL implementation for enhanced security and memory reliability
  - Extended VHD support to EFI partitions
  - Integrated additional capabilities for Excel files
  - Optimized system performance and resolved bug fixes
- Upgrade Trellix Windows SDK to 6900
  - Switched to native STL implementation for enhanced security and memory reliability
  - Extended VHD support to EFI partitions
  - Integrated additional capabilities for Excel files
  - Optimized system performance and resolved bug fixes

V 4.2.5

Release date: 4/14/2026

- Upgrade Bitdefender Windows SDK to 3.0.1.392
  - Minor bug fixes
- Engine update for Aurora - Linux
- Engine update for Aurora - Windows

V 4.2.4

Release date: 4/9/2026

- Upgrade ClamAV Linux SDK to 1.4.4
  - Resolved multiple issues in ClamAV
- Upgrade ClamAV Windows SDK to 1.4.4
  - Resolved multiple issues in ClamAV
- Upgrade ClamAV Linux updater
- Upgrade ClamAV Windows updater
- engine update for Avira - Windows
  - fix for immediate config change
- engine update for Avira - Linux
  - fix for immediate config change
- Upgrade Gridinsoft Windows SDK to 1.0.242

- Upgrade Gridinsoft Windows updater

V 4.2.3

Release date: 3/24/2026

- Engine update for OPSWAT Predictive Alin AI - Linux
  - FileType info provision fix
- Engine update for OPSWAT Predictive Alin AI - Windows
  - FileType info provision fix
- Engine update for OPSWAT Predictive Alin AI - Linux
  - FileType info provision fix
- Upgrade OPSWAT Predictive Alin AI Linux updater
- Upgrade OPSWAT Predictive Alin AI Windows updater
- Upgrade OPSWAT Predictive Alin AI Cloud Linux updater
- Engine update for SentinelOne - Windows
  - New config option - `Process archive files`
- Engine update for SentinelOne - Linux
  - New config option - `Process archive files`
- Upgrade Xvirus Anti-Malware Linux SDK to 5.1.1
  - Fixed different detection cases with Linux variant
- Upgrade Xvirus Anti-Malware Windows SDK to 5.1.1
  - Fixed different detection cases with Linux variant
- Upgrade Xvirus Anti-Malware Linux updater
- Upgrade Xvirus Anti-Malware Windows updater

V 4.2.2

Release date: 3/13/2026

- Upgrade Gridinsoft Windows SDK to 1.0.238
- Upgrade Gridinsoft Windows updater
- Upgrade Microsoft Defender Windows updater

V 4.2.1

Release date: 3/9/2026

- Upgrade IKARUS Windows SDK to 6.4.23
- Upgrade IKARUS Linux SDK to 6.4.23
- Upgrade Vir.IT ML Windows SDK to 9.5.102
  - Improvements on the heuristic engines for detecting MSIL & AutoIT malware
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.102
  - Improvements on the heuristic engines for detecting MSIL & AutoIT malware
- Upgrade Avira Windows SDK to 4.15.29
  - Brings new improvements and security updates, further enhancing its stability and performance
- Upgrade Avira Linux SDK to 4.15.29
  - Brings new improvements and security updates, further enhancing its stability and performance
- Upgrade ESET Windows SDK to 1.5.5.0
- Upgrade ESET Linux SDK to 1.5.5.0
- Upgrade Trellix Windows SDK to 6800
  - Decomposition support for Virtual Hard Disk (VHD) files
  - Repair enhancements for infected Excel spreadsheet
  - DAT Memory Protection feature for ENS Windows
  - Quality improvements and malware threat enhancement
- Upgrade Trellix Linux SDK to 6800
  - Decomposition support for Virtual Hard Disk (VHD) files
  - Repair enhancements for infected Excel spreadsheet
  - DAT Memory Protection feature for ENS Windows
  - Quality improvements and malware threat enhancement
- Upgrade IKARUS Windows updater
- Upgrade IKARUS Linux updater
- Upgrade Avira Windows updater
- Upgrade Avira Linux updater
- Upgrade ESET Windows updater
- Upgrade ESET Linux updater

V 4.2.0

Release date: 2/4/2026

- Upgrade SentinelOne Windows SDK to 26.1
  - Improved detection of PE, ELF, MACH-O, DotNet, PDF, DOC files
  - Added support for ISO, MSI and MSIX files
- Upgrade SentinelOne Linux SDK to 26.1
  - Improved detection of PE, ELF, MACH-O, DotNet, PDF, DOC files
  - Added support for ISO, MSI and MSIX files

V 4.1.19

Release date: 1/21/2026

- Upgrade Systweak Windows SDK to 1.0.0.2-2147
- Upgrade Gridinsoft Windows SDK to 1.0.234-470
- Upgrade Gridinsoft Windows updater

V 4.1.18

Release date: 12/18/2025

- Upgrade BKAV Pro Windows SDK to 8.2.39-602
  - Improved malware detection rate and reduced false positives
- Upgrade AhnLab Windows SDK to 3.29.1.1-2455
- Upgrade AhnLab Linux SDK to 3.29.1.1-2414
- Upgrade IKARUS Windows SDK to 6.4.20-2363
- Upgrade IKARUS Linux SDK to 6.4.20-2252
- Upgrade ClamAV Linux SDK to 1.4.3-2653
- Upgrade ESET Windows SDK to 1.5.4.0-2268
- Upgrade ESET Linux SDK to 1.5.4.0-2282
- Upgrade Bitdefender Windows SDK to 3.0.1.371-2289
- Upgrade Bitdefender Linux SDK to 3.0.1.342-2179
- Upgrade ESET Windows updater
- Upgrade ESET Linux updater

V 4.1.17

Release date: 11/24/2025

- Upgrade Vir.IT ML Windows SDK to 9.5.100-2207
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.100-2354
  - Improved heuristic engines to detect LNK and MSIL Malware

V 4.1.16

Release date: 11/24/2025

- Upgrade NETGATE Windows SDK to 25.0.650.0-1982
- Upgrade Sophos Windows SDK to 4.23-3.96.1-2179
- Upgrade Sophos Linux SDK to 4.23-3.96.1-2089
- Upgrade Avira Windows SDK to 4.15.28-2389
- Upgrade Avira Linux SDK to 4.15.28-2348
- Upgrade Avira Windows updater
- Upgrade Avira Linux updater

V 4.1.15

Release date: 11/11/2025

- Upgrade Webroot Windows SDK to 1.6.2-2015
- Upgrade Webroot Linux SDK to 1.6.2-1980
- Upgrade Sophos Windows SDK to 4.23-3.95.1-2170
- Upgrade Sophos Linux SDK to 4.23-3.95.1-2076

V 4.1.14

Release date: 11/05/2025

- Upgrade TACHYON Linux SDK to 20251029-2033
- Upgrade ClamAV Windows SDK to 1.4.3-2553
- Upgrade ClamAV Linux SDK to 1.4.3-2619

V 4.1.13

Release date: 10/20/2025

- Upgrade Sophos Windows SDK to 4.23-3.95.1-2159
- Upgrade Sophos Linux SDK to 4.23-3.95.1-2068
- Upgrade Trellix Windows updater
- Upgrade Trellix Linux updater

- Upgrade Xcitium Windows updater
- Upgrade Xcitium Linux updater
- Upgrade Aurora Windows updater
- Upgrade Aurora Linux updater

V 4.1.12

Release date: 10/09/2025

- Upgrade Bkav Pro Windows SDK to 8.2.38-568
- Upgrade Bkav Pro Windows updater
- New Enhancements and Bug Fixes for Bkav Pro:
  - Increased the malware detection rate
  - Decreased the false positive rate
  - Added the SDK for detecting Polymorphic virus
  - Fixed the crash bug when scanning some PE files
  - Added an SDK for malware detection based on code analysis and heuristics
  - Fixed the stability issue when multi-threaded scanning polymorphic malware files

V 4.1.11

Release date: 10/02/2025

- Upgrade XVirus Windows SDK to 5.0-1905
- Upgrade XVirus Linux SDK to 5.0-564
- Upgrade XVirus Linux updater
- Upgrade XVirus Windows updater
- New configuration options for Xvirus:
  - Signatures based scan
  - Heuristics aggressive level 1-5 scans
  - MaxHeuristicsPeSscanLength
  - MaxHeuristicsOtherSscanLength
- McAfee engine has been renamed to Trellix
- Comodo engine has been renamed to Xcitium
- Cylance engine has been renamed to Aurora

V 4.1.10

Release date: 09/15/2025

- Upgrade Webroot Windows SDK to 1.6.1-1973
- Upgrade Webroot Linux SDK to 1.6.1-1941
- Upgrade Gridinsoft Windows SDK to 1.0.209-393
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.97-2318
- Upgrade Vir.IT ML Windows SDK to 9.5.97-2169
- Upgrade Scrutiny Windows SDK to 3.2.4-1425
- Upgrade Webroot Linux updater
- Upgrade Webroot Windows updater
- Upgrade Gridinsoft Windows updater
- Resolved a security vulnerability in the Webroot engine

V 4.1.9

Release date: 08/29/2025

- Upgrade Comodo Windows SDK to 6.5.0.1206-2390
- Upgrade SentinelOne Windows SDK to 25.3-92
- Upgrade SentinelOne Linux SDK to 25.3-83
- Upgrade TACHYON Windows SDK to 20241014-2232
- Upgrade Avira Windows SDK to 4.15.27-2341
- Upgrade Avira Linux SDK to 4.15.27-2314
- Upgrade Comodo Windows updater
- Improved detection of PE and ELF files, enhanced runtime performance, and fixed crashes caused by specific malformed ZIP files in the SentinelOne engine
- Improvements and security updates, further enhancing its stability and performance in the Avira engine

V 4.1.8

Release date: 07/28/2025

- New Config Option in Sophos (Windows & Linux) Max Recursion Depth

V 4.1.7

Release date: 07/14/2025

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.96-2299
- Upgrade Vir.IT ML Windows SDK to 9.5.96-2151
- Upgrade AhnLab Windows SDK to 3.27.2.4-2383

- Upgrade AhnLab Linux SDK to 3.27.2.4-2358
- Upgrade Avira Windows SDK to 4.15.26-2318
- Upgrade Avira Linux SDK to 4.15.26-2281
- Upgrade Ikarus Windows SDK to 6.4.12-2283
- Upgrade Ikarus Linux SDK to 6.4.12-2193
- Upgrade Ikarus Linux updater
- Upgrade Ikarus Windows updater
- Upgrade Avira Linux updater
- Upgrade Avira Windows updater

V 4.1.6

Release date: 07/3/2025

- Set default temp directory to OPSWAT's temp dir for Emsisoft - Windows

V 4.1.5

Release date: 06/23/2025

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.94-2286
- Upgrade Vir.IT ML Windows SDK to 9.5.94-2140
- Upgrade Varist Windows SDK to 6.6.2-2254
- Upgrade Varist Linux SDK to 6.6.2-2219
- Upgrade ESET Windows SDK to 1.4.2.0-2186
- Upgrade ESET Linux SDK to 1.4.2.0-2178
- Upgrade ClamAV Windows SDK to 1.4.3-2460
- Upgrade ClamAV Linux SDK to 1.4.3-2545
- Upgrade ClamAV Linux updater
- Upgrade ClamAV Windows updater
- Resolved a security vulnerability in the ClamAV engine

V 4.1.4

Release date: 06/12/2025

- Upgrade ESET Windows SDK to 1.4.2-2178
- Upgrade ESET Linux SDK to 1.4.2-2172
- Upgrade Filseclab Windows SDK to 1.0.2.2132-2078
- Upgrade Bkav Pro Windows SDK to 8.2.31-503

- Upgrade ESET Linux updater
- Upgrade ESET Windows updater
- Fix the issue where Bkav Pro fails to detect malicious files in SFX installers
- Fix the issue where Filseclab caused a system shutdown bug

V 4.1.3

Release date: 06/02/2025

- Upgrade ClamAV Windows SDK to 1.4.2-2439
- Upgrade ClamAV Linux SDK to 1.4.2-2525
- Customers can now configure the Avira engine to treat PUP/PUA detections as infected or not
- Fix ClamAV Windows when MDCore's temp directory is set to a RAMDISK

V4.1.2

Release date: 04/25/2025

- Upgrade ClamAV Windows SDK to 1.4.2-2411
- Upgrade ClamAV Linux SDK to 1.4.2-2516
- Upgrade Comodo Windows SDK to 6.5.0.1195-2316
- Upgrade AhnLab Windows SDK to 3.27.1.1-2344
- Upgrade AhnLab Linux SDK to 3.27.1.1-2330
- Upgrade Avira Windows SDK to 4.15.25-2248
- Upgrade Avira Linux SDK to 4.15.25-2248
- Upgrade Bitdefender Windows SDK to 3.0.1.306-2159
- Upgrade Bitdefender Linux SDK to 3.0.1.297-2093
- Upgrade BKAV Pro Windows SDK to 8.2.30-477
- Upgrade CMC Windows SDK to 2.3.5-389
- Upgrade CrowdStrike Windows SDK to 1.10.1-1945
- Upgrade CrowdStrike Linux SDK to 1.10.1-1669
- Upgrade Cylance Windows SDK to 1.2.1-587
- Upgrade Cylance Linux SDK to 1.2.0-629
- Upgrade Emsisoft Windows SDK to 2021.05.7597-2036
- Upgrade Eset Windows SDK to 1.0.0-2127
- Upgrade Eset Linux SDK to 1.0.0-2126
- Upgrade Filseclab Windows SDK to 1.0.2.2123-2030
- Upgrade Gridinsoft Windows SDK to 1.0.209-329

- Upgrade Huorong Windows SDK to 1.0.0-1835
- Upgrade Ikarus Windows SDK to 6.3.23-2230
- Upgrade K7 Windows SDK to 4.0.0.7-2280
- Upgrade K7 Linux SDK to 4.0.0.7-2354
- Upgrade Lionic Windows SDK to 8.23-1876
- Upgrade Lionic Linux SDK to 8.23-1676
- Upgrade McAfee Windows SDK to 6700-2144
- Upgrade McAfee Linux SDK to 6700-2098
- Upgrade Microsoft Defender Windows SDK to 1.0.0
- Upgrade NANO AV Windows SDK to 1.0.146.25796-1980
- Upgrade NETGATE Windows SDK to 25.0.650.0-1865
- Upgrade QuickHeal Windows SDK to 18.0-2086
- Upgrade QuickHeal Linux SDK to 18.0-2049
- Upgrade RocketCyber Windows SDK to 02_01___2025-1915
- Upgrade RocketCyber Linux SDK to 09_07 _2024-363
- Upgrade Scrutiny Windows SDK to 3.2.4-1336
- Upgrade Sophos Windows SDK to 4.21-3.93.1-2035
- Upgrade Sophos Linux SDK to 4.21-3.93.1-1979
- Upgrade Systweak Windows SDK to 1.0.0.2-1980
- Upgrade TACHYON Windows SDK to 20240927-1965
- Upgrade Varist Windows SDK to 6.6.1-2210
- Upgrade Varist Linux SDK to 6.6.1-2189
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.81-2262
- Upgrade Vir.IT ML Windows SDK to 9.5.81-2113
- Upgrade VirusBlockAda Windows SDK to 5.3.2-1970
- Upgrade Webroot Windows SDK to 1.4.114-1876
- Upgrade XVirus Windows SDK to 4.2.3-1778
- Upgrade Zillya Windows SDK to 1.2.0.11-2029
- Upgrade Antiy Windows SDK to 3.0.3.1-2063
- Upgrade ClamAV Linux updater
- Fix the issue in VirusBlockAda where PDFs are incorrectly flagged as suspicious
- Fix ClamAV integration issue to ensure compatibility with Debian 9
- The BKAV Pro SDK update has improved malware detection, reduced false positives, and enhanced scanning of packaged files

V4.1.1

Release date: 03/05/2025

- Upgrade ClamAV Windows SDK to 1.4.2-2385
- Upgrade ClamAV Linux SDK to 1.4.2-2475
- Upgrade AhnLab Windows SDK to 3.27.1.1-2322
- Upgrade AhnLab Linux SDK to 3.27.1.1-2312
- Upgrade K7 Windows SDK to 4.0.0.7-2257
- Upgrade K7 Linux SDK to 4.0.0.7-2335
- Upgrade RocketCyber Windows SDK to 02___01_2025-1897
- Upgrade RocketCyber Linux SDK to 09_07_2024-349
- Upgrade Avira Windows SDK to 4.15.25-2224
- Upgrade Avira Linux SDK to 4.15.25-2234
- Upgrade Sophos Windows SDK to 4.21-3.93.1-2014
- Upgrade Sophos Linux SDK to 4.21-3.93.1-1965
- Upgrade Gridinsoft Windows SDK to 1.0.209-308
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.81-2244
- Upgrade Vir.IT ML Windows SDK to 9.5.81-2090
- Upgrade ClamAV Windows updater
- Upgrade ClamAV Linux updater
- Upgrade Avira Windows updater
- Upgrade Avira Linux updater
- Upgrade Gridinsoft Windows updater

V4.0.98

Release date: 02/28/2025

- The Tachyon license has been updated

V4.0.97

Release date: 02/03/2025

- Upgrade AhnLab Windows SDK to 3.27.0.8-2308
- Upgrade Bitdefender Windows SDK to 3.0.1.306-2122
- Upgrade CMC Windows SDK to 2.3.5-353
- Upgrade Emsisoft Windows SDK to 2021.05.7597-2001

- Upgrade ESET Windows SDK to 1.0.0-2093
- Upgrade Huorong Windows SDK to 1.0.0-1800
- Upgrade K7 Windows SDK to 4.0.0.6-2240
- Upgrade McAfee Windows SDK to 6700-2106
- Upgrade QuickHeal Windows SDK to 18.0-2046
- Upgrade Sophos Windows SDK to 4.20-3.92.0-1996
- Upgrade TACHYON Windows SDK to 2020.4.22.1-2115
- Upgrade Varist Windows SDK to 6.6.1-2178
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.80-2229
- Upgrade Vir.IT ML Windows SDK to 9.5.80-2075
- Upgrade VirusBlockAda Windows SDK to 5.3.1-1932
- Upgrade Zillya! Windows SDK to 1.2.0.11-1992
- Upgrade BKAV Pro Windows SDK to 8.2.25-422
- Upgrade Gridinsoft Windows SDK to 1.0.203-281
- Upgrade Gridinsoft Windows updater
- Resolved an issue where the Zillya! engine was locking files after a scan
- Resolved an issue where BKAV Pro engine crashed with certain files
- Added support for scanning Office file formats in BKAV Pro engine
- Improved malware detection rate in BKAV Pro engine
- Reduced false positive rate in BKAV Pro engine
- Resolved an issue with multi-thread scanning in Vir.IT engine

V4.0.96

Release date: 01/16/2025

- Upgrade Scrutiny Windows SDK to 3.2.4-1278
- Upgrade Scrutiny Windows updater

V4.0.95

Release date: 12/18/2024

- Reverted the VirusBlokada SDK engine update to address and resolve the false positive detection issue in PDF files

V4.0.94

Release date: 12/16/2024

- Upgrade XVirus Windows SDK to 4.2.3-1717
- Upgrade XVirus Linux SDK to 4.2.3-354
- Upgrade Ikarus Windows SDK to 6.3.23-2165
- Upgrade Ikarus Linux SDK to 6.3.23-2110
- Upgrade Ikarus Widnows updater
- Upgrade Ikarus Linux updater
- Upgrade Comodo Windows SDK to 6.5.0.1195-2241
- Upgrade Vir.it eXplorer Windows SDK to 9.5.76-2194
- Upgrade Vir.it ML Windows SDK to 9.5.76-2037
- Performance Improvement in XVirus for Windows and Linux
- Added a new heuristic engine in Vir.IT to detect AutoIT malware and LNK files
- Added new functions to heuristic engines in Vir.IT for detecting MSIL malwares

V4.0.93

Release date: 11/28/2024

- Upgrade K7 Linux SDK to 4.0.0.5-2276
- Upgrade K7 Windows SDK to 4.0.0.5-2195
- Upgrade VirusBlokAda Windows SDK to 5.0.0-1888
- Upgrade VirusBlokAda Widnows updater
- Resolved a vulnerability in the K7 engine
- Add preflight script to check OS version for K7 engine (CentOS 7 is not supported anymore)

V4.0.92

Release date: 11/20/2024

- Upgrade Xvirus Linux SDK to 4.2.2-341
- Upgrade Systweak Windows SDK to 1.0.0.2-1785
- Upgrade Systweak Windows updater
- Upgrade Ahnlab Linux SDK to 3.26.1.4-2253
- Upgrade Ahnlab Windows SDK to 3.26.1.4-2261
- Upgrade BKAV Pro Windows SDK to 8.2.23-379
- Upgrade Avira Windows SDK to 4.15.23-2168
- Upgrade RocketCyber Linux updater
- Upgrade RocketCyber Windows updater
- Upgrade Lionic Linux SDK to 8.23-1600

- Upgrade Lionic Windows SDK to 8.23-1793

V4.0.91

Release date: 11/18/2024

- Upgrade Filseclab windows SDK to 1.0.2.2123-1948
- Upgrade Filseclab Windows updater
- Switched the Filseclab engine process to 64-bit for enhanced performance
- Disabled the QProcess timeout in Filseclab to improve stability

V4.0.90

Release date: 10/22/2024

- Upgrade Cylance Linux SDK to 1.2.0-539
- Upgrade Cylance Windows SDK to 1.2.1-484
- Upgrade RocketCyber Linux SDK to 22_05_2024-262
- Upgrade RocketCyber Windows SDK to 12_02_2021-1818
- Upgrade Webroot SMD Linux SDK to 1.4.114-1767
- Upgrade Webroot SMD Windows SDK to 1.4.114-1770
- Upgrade Avira Linux SDK to 4.15.23-2163
- Upgrade Avira Windows SDK to 4.15.23-2147
- Upgrade TACHYON Linux SDK to 20240927-1880
- Upgrade Avira Windows updater
- Upgrade Avira Linux updater
- Add a UI option to the Cylance engine to configure the malware detection threshold
- Resolve the issue causing the TACHYON engine to crash on certain files

V4.0.89

Release date: 10/15/2024

- Upgrade CMC Windows updater
- Upgrade CMC Linux updater
- Upgrade BKAV Pro Windows updater
- Upgrade Cylance Windows updater
- Upgrade Cylance Linux updater
- Upgrade Emsisoft Windows updater
- Upgrade ESET Windows updater

- Upgrade Lionic Windows updater
- Upgrade Lionic Linux updater
- Upgrade NANOAV Windows updater
- Upgrade NANOAV Linux updater
- Upgrade NETGATE Windows updater
- Upgrade Systweak Windows updater
- Upgrade Webroot SMD Windows updater
- Upgrade Webroot SMD Linux updater

V4.0.88

Release date: 10/14/2024

- Upgrade Gridinsoft Windows SDK to 1.0.187-191
- Upgrade Bkav Pro Windows SDK to 8.2.22-323
- Upgrade Varist Windows SDK to 6.6.1-2103
- Upgrade Varist Linux SDK to 6.6.1-2095
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.72-2148
- Upgrade Vir.IT ML Windows SDK to 9.5.72-1989
- Upgrade ClamAV Windows SDK to 1.4.1-2297
- Upgrade ClamAV Linux SDK to 1.4.1-2392
- Varist updates include multiple new features and bug fixes:
  - Fixed memory corruption issue
  - Resolved endless loop in DOS malware scans
  - Improved URL detection and recovery for Office malware
  - Enhanced encrypted password feature for archive scanning
  - Improved scan speed and stability across various formats
- Improved BKAV engine for better malware detection and fewer false positives
- Enabled support for image fuzzy hashing in ClamAV

V4.0.87

Release date: 09/12/2024

- Upgrade Cylance Windows SDK to 1.2.1-437
- Upgrade Cylance Linux SDK to 1.2.0-474
- Fixed false positive verdicts when scanning unsupported files with Cylance

V4.0.86

Release date: 09/04/2024

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.66
- Upgrade Vir.IT ML Windows SDK to 9.5.66
- Upgrade CMC Windows SDK to 2.3.5
- Upgrade CMC Linux SDK to 2.3.5
- Upgrade CMC Windows updater
- Upgrade CMC Linux updater
- Updated the Vir.IT engine to patch the vulnerabilities
- Disable scan by cloud option by default on CMC engine

V4.0.85

Release date: 09/02/2024

- Updated the Vir.IT engine to patch the vulnerabilities
- Upgrade Vir.IT eXplorer Windows SDK to 9.5.65
- Upgrade Vir.IT ML Windows SDK to 9.5.65

V4.0.84

Release date: 08/20/2024

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.64
- Upgrade Vir.IT ML Windows SDK to 9.5.64
- Upgrade K7 Windows SDK to 4.0.0.3
- Upgrade Avira Windows SDK to 4.15.22
- Upgrade Avira Linux SDK to 4.15.22
- Upgrade Avira Windows updater
- Upgrade Avira Linux updater
- Upgrade AhnLab Windows SDK to 3.26.0.5
- Upgrade AhnLab Linux SDK to 3.26.0.5
- Improved Windows Vir.IT eXplorer scan speed for EML files
- Prevent Windows K7 extracting files to system temp directory

V4.0.83

Release date: 08/13/2024

- Upgrade Gridinsoft SDK to 1.0.183

- Improve Antiy's scan speed
- Fix Comodo failure when repeating scan a file

V4.0.82

Release date: 07/23/2024

- Improve encrypted files recognition in Cylance

V4.0.81

Release date: 07/18/2024

- Comodo new configurations on Windows and Linux:
  - Max scanned script size - Text files larger than the threshold won't be scanned

V4.0.80

Release date: 07/15/2024

- Upgrade Cylance Windows SDK to 1.2.1
- Upgrade Cylance Linux SDK to 1.2.0
- Upgrade Lionic Windows SDK to 8.17-1658
- Upgrade Lionic Linux SDK to 8.17-1499
- Improve archive handling in Cylance

V4.0.79

Release date: 07/09/2024

- New scan engine in Windows: Gridinsoft
- Upgrade Sophos Windows SDK to 4.18-3.90.0
- Upgrade Sophos Linux SDK to 4.18-3.90.0
- Upgrade RocketCyber Linux SDK to 22-05-2024
- Upgrade ESET Windows updater
- Upgrade ESET Linux updater
- Upgrade Avira Windows updater
- Upgrade Avira Linux updater

V4.0.78

Release date: 07/04/2024

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.61

- Upgrade Vir.IT ML Windows SDK to 9.5.61
- Upgrade Bkav Windows SDK to 8.2.21

V4.0.77

Release date: 06/21/2024

- Upgrade K7 Windows updater
- Upgrade K7 Linux updater
- Upgrade McAfee Windows updater
- Upgrade McAfee Linux updater
- Upgrade VirusBlokAda Windows updater
- Upgrade Varist Windows updater
- Upgrade Varist Linux updater

V4.0.76

Release date: 06/18/2024

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.59
- Upgrade Vir.IT ML Linux SDK to 9.5.59

V4.0.75

Release date: 06/12/2024

- Upgrade CrowdStrike Falcon ML Windows SDK to 1.10.1
- Upgrade CrowdStrike Falcon ML Linux SDK to 1.10.1
- Upgrade Cylance Linux SDK to 1.2.0

V4.0.74

Release date: 04/24/2024

- Upgrade Avira Windows SDK to 4.15.21.77
- Upgrade Avira Linux SDK to 4.15.21.77
- Performance Improvement in Comodo for Windows

V4.0.73

Release date: 04/18/2024

- Upgrade NANOAV Windows SDK to 1.0.146.90815
- Upgrade NETGATE Windows SDK to 25.0.650.0

- Upgrade Vir.IT eXplorer Windows SDK to 9.5.0.68
- Upgrade Vir.IT ML Windows SDK to 9.5.0.68
- Upgrade Bitdefender Windows SDK to 3.0.1.306
- Upgrade Bitdefender Windows updater
- Upgrade Bitdefender Linux updater
- Temporary folder config in AhnLab - Linux
- Temporary folder config in AhnLab - Windows

V4.0.72

Release date: 03/26/2024

- Upgrade QuickHeal Windows SDK to 18.0
- Upgrade IKARUS Windows SDK to 6.03.11
- Upgrade IKARUS Linux SDK to 6.03.11
- Upgrade Avira Windows SDK to 4.15.20.156
- Upgrade Avira Linux SDK to 4.15.20.156
- Upgrade Anhlab Windows SDK to 3.25.1.4.34589
- Upgrade Anhlab Linux SDK to 3.25.1.4.34589
- Upgrade Vir.IT Windows SDK to 9.5.0.59
- Upgrade Vir.IT Linux SDK to 9.5.0.59
- Upgrade CMC Windows updater

V4.0.71

Release date: 03/06/2024

- New scan engine in Linux: CMC
- Upgrade IKARUS Windows SDK to 6.03.07
- Upgrade IKARUS Linux SDK to 6.03.07
- Upgrade Sophos Windows SDK to 4.17-3.0090
- Upgrade Sophos Linux SDK to 4.17-3.0090
- Upgrade TACHYON Windows updater
- Upgrade TACHYON Linux updater

V4.0.70

Release date: 02/06/2024

- Upgrade AhnLab Windows SDK to 3.25.0.6.34367

- Upgrade AhnLab Linux SDK to 3.25.0.6.34367
- Upgrade Cylance Windows SDK to 1.1.2
- Upgrade BKAV Windows SDK to 8.2.20
- Upgrade McAfee Windows SDK to 6700
- Upgrade McAfee Linux SDK to 6700

V4.0.69

Release date: 01/22/2024

- New scan engine in Windows: BKAV

V4.0.68

Release date: 01/18/2024

- Resolved issue where log rotation failure on Windows ClamAV
- Resolved issue where log rotation failure on Windows Cylance
- Resolved issue where log rotation failure on Windows Filseclab
- Set log level during scanner initialization on Linux Bitdefender
- Set log level during scanner initialization on Windows Bitdefender
- Set log level during scanner initialization on Windows Filseclab
- The Metadefender MetaScan MAX package for both Windows and Linux has been updated to remove Kaspersky
- The Metadefender MetaScan 20 Windows package update replaces Kaspersky with Xvirus and Antiy, with Lionic

V4.0.67

Release date: 01/16/2024

- The Aegislab engine has been renamed to Lionic
- Upgrade Lionic Windows SDK to 8.7
- Upgrade Lionic Linux SDK to 8.7

V4.0.66

Release date: 01/04/2024

- Upgrade Clamav Windows SDK to 0.103.11
- Upgrade Clamav Linux SDK to 0.103.11
- Upgrade Sophos Windows SDK to 4.17-3.89.0
- Upgrade Sophos Linux SDK to 4.17-3.89.0
- XVirus new configurations on Windows and Linux:

- EnableHeuristics - Enables heuristics scanning of files. Default: true
- MaxScanLength - Maximum file size to be scanned in bytes. If set “null” then there is no limit. Default: null

V4.0.65

Release date: 12/18/2023

- Upgrade Vir.IT to 9.5.43
- Upgrade RocketCyber 03_11_2023
- Upgrade Kaspersky to 8.10.1.1311
- Fix Cylance, Vir.IT to return not scanned for password protected archives
- Improve integration to avoid DLL hijack

V4.0.64

Release date: 12/7/2023

- New scan engine in Windows: CMC
  - Upgrade CMC Windows SDK to 2.3.1

V4.0.63

Release date: 12/4/2023

- Upgrade Cylance Windows SDK to 1.1.0
- Upgrade Cylance Linux SDK to 1.1.0
- Make archive scanning depth configurable in Cylance for Windows and Linux

V4.0.62

Release date: 11/20/2023

- New scan engine in Linux: RocketCyber

V4.0.61

Release date: 11/13/2023

- Upgrade IKARUS Winnows SDK to 6.2.7
- Upgrade IKARUS Linux SDK to 6.2.7
- A bug fix has been released for the IKARUS Windows SDK

V4.0.60

Release date: 10/23/2023

- Upgrade Avira Windows SDK to 4.15.19
- Upgrade Avira Linux SDK to 4.15.19

V4.0.59

Release date: 10/19/2023

- Increased the maximum number of concurrency to the Cylance engine
- Introduced a timeout feature for the Cylance SDK requests

V4.0.58

Release date: 10/9/2023

- Enhance the script for Windows Defender

V4.0.57

Release date: 9/25/2023

- Enhance the script for Windows Defender

V4.0.56

Release date: 9/7/2023

- Introduction of Linux Max Package
  - It includes 19 engines (AhnLab, Avira, Bitdefender, ClamAV, Eset, QuickHeal, K7 Security, Tachyon, Varist, Ikarus, Kaspersky, McAfee, Nano, Sophos, Webroot, CrowdStrike, AegisLab, Cylance, and XVirus)
- New scan engine in Linux: Xvirus
- New scan engine in Linux and Windows: Cylance
- Update Avira Linux SDK to 4.15.17
- Update Avira Windows SDK to 4.15.17

V4.0.55

Release date: 9/6/2023

- Update Sophos Linux SDK to 4.16.3.88
- Update Sophos Windows SDK to 4.16.3.88
- New config “Scan all file types” added to Sophos (Linux) and Sophos (Windows)

V4.0.54

Release date: 8/29/2023

- Update Xvirus SDK to 4.2

V4.0.53

Release date: 8/23/2023

- Updated VirIT ML SDK to 9.5.34
- Updated VirIT eXplorer SDK to 9.5.34

V4.0.52

Release date: 8/9/2023

- Improved the speed of ClamAV

V4.0.51

Release date: 6/28/2023

- Updated VirIT ML SDK to 9.5.27
- Updated VirIT eXplorer SDK to 9.5.27

V4.0.50

Release date: 6/22/2023

- Fixed the issue where engines fail to scan .dat and .ink files

V4.0.49

Release date: 6/19/2023

- Updated Ahnlab SDK to 3.23.3.2
- Updated IKARUS SDK to 6.1.10

V4.0.48

Release date: 5/22/2023

- The Cyren engine has been renamed to Varist

V4.0.47

Release date: 5/22/2023

- All Windows Docker updaters have been updated
- Fixed an issue reported by customer when Avira returned Failed to scan result to a sample file

V4.0.46

Release date: 4/26/2023

- Fixed an issue that caused K7 Windows to permanently fail on some environments

V4.0.45

Release date: 4/20/2023

- Updated ClamAV to 0.103.8 to increase performance

V4.0.44

Release date: 3/29/2023

- Updated Adaware CLI that caused the engine to fail initializing

V4.0.43

Release date: 3/23/2023

- Upgrade IKARUS SDK to 6.1.6
- Upgrade Vir.IT SDK to 9.5.25
- Upgrade Avira SDK to 4.15.15

V4.0.42

Release date: 3/22/2023

- Updated the TACHYON updater utility to patch vulnerabilities in their 3rd parties

V4.0.41

Release date: 3/20/2023

- Added description to all engine configurations

V4.0.40

Release date: 3/7/2023

- Fixed an issue where ClamAV Linux had a missing dependency

V4.0.39

Release date: 3/1/2023

- Fixed an issue, where Huorong updater did not generate databases

v4.0.38

Release date: 3/1/2023

- Upgrade AhnLab SDK to 3.23.1.2
- Upgrade Sophos SDK to 4.15-3.87.0
- Upgrade CrowdStrike SDK to 1.8.1
- Fixed an issue where AhnLab may not initialize on specific systems
- Fixed an issue with Systweak update generation

v4.0.37

Release date: 2/21/2023

- Updated ClamAV SDK to 1.0.1 to resolve vulnerabilities

v4.0.36

Release date: 2/16/2023

- Updated XVirus SDK to 4.1

v4.0.35

Release date 1/9/2023

- Updated McAfee SDK to 6600
- Fixed Avira issue that caused previous release to fail permanently when updating

v4.0.34

Release date 12/6/2022

- Upgrade Bitdefender to 3.0.1.297
- Upgrade ClamAV Windows to 0.103.7-2
- Significantly improved the multi-thread speed of Sophos Windows
- Fixed an issue where Avira did not detect infected files in nested archives

v4.0.33

Release date 11/21/2022

- Upgrade XVirus Engine

v4.0.32

Release date 10/27/2022

- Upgrade Vir.IT SDK to 9.5.22

v4.0.31

Release date 10/20/2022

- Upgrade Cyren SDK to 6.5.5

v4.0.30

Release date 10/10/2022

- Upgrade Bitdefender Windows SDK to version 3.0.1.289
- Upgrade IKARUS SDK to 6.0.28
- Upgrade Sophos SDK to 4.14-3.86.1
- Introduced Max File Size configuration in CrowdStrike

v4.0.29

Release date 9/12/2022

- Upgrade ClamAV SDK to version 0.103.7
- Improve K7 detection

v4.0.28

Release date 7/28/2022

- Upgrade Avira SDK to version 4.15.10
- Upgrade IKARUS SDK to version 6.0.26
- Upgrade McAfee SDK to version 6500
- Upgrade Cyren SDK to version 6.5.4

v4.0.27

Release date 7/4/2022

- Upgrade K7 SDK to version 4.0.0.2

v4.0.26

Release date 6/9/2022

- Fix wrong definition date in K7 Linux

v4.0.25

Release date 6/1/2022

- Upgrade Avira to version 4.15.8
- Upgrade Vir.IT SDK to version 9.5.18

v4.0.24

Release date 5/26/2022

- Lavasoft will be renamed to Adaware on 6/1
- Upgrade ClamAV SDK to version 0.103.6

v4.0.23

Release date 5/23/2022

- Upgrade AhnLab to version 3.21.0
- Upgrade Avira to version 4.15.7

v4.0.22

Release date 5/9/2022

- Upgrade Scrutiny to version 3.2.4
- Upgrade Sophos to version 3.85.1
- Upgrade Vir.IT to version 9.5.11

v4.0.21

Release date 5/2/2022

- Upgrade TACHYON Linux to version 20220408
- Upgrade Vir.IT to version 9.5.14
- Ensure all engines return "No Thread Found" for 0 byte files

v4.0.20

Release date: 4/19/2022

- Upgrade IKARUS to version 6.0.13
- Upgrade Vir.IT to version 9.5.13
- Upgrade ClamAV Linux to version 0.103.5

v4.0.19

Release date: 4/13/2022

- Update K7 SDK to version 4.0
- Update Bitdefender Windows to version 3.0.1.285
- Update Avira to version 4.15.5

v4.0.18

Release date 3/29/2022

- Update Avira SDK to version 4.15.3
- Updated IKARUS SDK to version 6.0.5
- Updated Vir.IT SDK to version 9.5.12

v4.0.17

Release date: 3/14/2022

- Update Scrutiny SDK to version 3.2.3

v4.0.16

Release date: 2/28/2022

- Updated Systweak SDK to 2.3.1001.26067
- Updated Sophos Windows SDK to 3.84.1

v4.0.15

Release date: 2/16/2022

- Upgrade Vir.IT SDK to 9.5.11
- Return default threat name for Scrutiny
- Fix Symantec Malformed container detection

v4.0.14

Release date: 2/9/2022

- Upgrade Avira SDK to 4.15.2
- New ESET version that doesn't need to change temp folder on the hardened Linux OS
- Increased memory utilization capabilities of the 32 bit engines
- Fixed an issue that could potentially cause ClamAV to hang
- Increased internal timeout for SUPERAntiSpyware that could potentially cause it to fail

v4.0.13

Release date: 2/2/2022

- Upgrade ClamAV SDK to 0.103.5
- Add NETGATE Heuristic scan option

v4.0.12

Release date: 1/31/2022

- Upgrade new engine SDK:
  - McAfee 6400
- Configuration to allow TrendMicro/TrendMicro HouseCall to change batch scan wait time
- Improve ClamAV stability

v4.0.11

Release date: 12/7/2021

- A configuration to adjust confidence level for CrowdStrike Falcon ML, Webroot SMD to resolve false positive issues
- Significant TACHYON (Windows) performance improvement
- Upgrade new engine SDK:
  - IKARUS 5.7.5
  - Vir.IT 9.5.04
  - AegisLab (Linux) 7.7
  - Kaspersky (Windows) 8.9.3.1319
- Kaspersky and Systweak are able to return PUP/PUA scan result
- Add new logging mode configuration: Webroot SMD, AegisLab (Linux), TACHYON (Linux), K7 (Windows)
- Add max archive scan file size configuration for Avira

v4.0.10

Release date: 12/1/2021

- Engine names are changed regarding on this announcement
- Add new logging mode configuration: AhnLab, HAURI, NETGATE

v4.0.9

Release date: 11/24/2021

- Upgrade Cyren SDK to version 6.5.1

- Add new engine configurations:
  - McAfee: Logging mode, detect PUP/PUA
  - VirusBlokAda: Logging mode, detect PUP/PUA, heuristic scan, deep scan, Max archive scan file size,
  - Zillya!: Max recursive extraction, max archive scan file size

v4.0.8

Release date: 11/11/2021

- New scan engine: Scrutiny
- Upgrade Cyren SDK to version 6.5.0
- Upgrade Vir.IT to version 9.5.02
- Add new engine configurations:
  - Comodo, FilsecLab: logging mode
  - CrowdStrike: Max extracted file size, Max number of extracted files
  - TrendMicro / TrendMicro House Call: Extract Archive, Max recursive extraction
  - FilsecLab: Max recursive extraction

v4.0.7

Release date: 11/01/2021

- Upgrade Zillya! SDK to version 1.2.0.11
- Upgrade Sophos SDK to version 3.38.3
- Better handling ClamAV temp files
- Add new configuration for:
  - QuickHeal, Sophos, Tachyon, Kaspersky, NANOAV, IKARUS, Antiy: logging mode
  - Cyren: Heuristic

v4.0.6

Release date: 10/18/2021

- Improve TrendMicro and TrendMicro House Call scan throughput
- Add new configurations for:
  - ESET Linux: Max recursive extraction, logging mode
  - Bitdefender: Max recursive extraction, Max scan size (archive)
  - ClamAV: Recursive match limit (PCRE), Max file size (PCRE), Max recursive extraction, logging mode

- Avira: logging mode

- Upgrade Avira SDK to version 4.15.1

v4.0.5

Release date: 9/28/2021

- Fix Symantec initialization failed in some specific case.
- Upgrade Crowdstrike Falcon ML SDK to version 1.6
- Improve Lavasoft scan throughput
- Fix Systweak scan issue when input files containing semicolons

v4.0.4

Release date: 9/14/2021

- Upgrade Sophos SDK to version 3.83.2
- Upgrade Vir.IT SDK to version 9.5.01

v4.0.3

Release date: 8/31/2021

- Upgrade Bitdefender SDK to version 3.0.1.280
- Upgrade Vir.IT SDK to version 9.5.00
- Upgrade CrowdStrike Falcon ML SDK to version 1.5
- Fix Kaspersky locked file issue
- Fix TrendMicro, TrendMicro HouseCall slowness issue

v4.0.2

Release date: 8/17/2021

- Symantec can use different port if the port 1344 is being used
- Remove Eicar test file in the NANO AV package
- Improve code for several engines: TrendMicro, TrendMicro HouseCall, VirusBlokAda, Zillya

v4.0.1

Release date: 8/3/2021

- Upgrade Emsisoft SDK to version 2021.05.7597
- Improve code for several engines: RocketCyber, Systweak

Deep CDR release notes

v8.1.3

Release date: 11/18/2026

- Supported FDF file format:
  - Removed JavaScript and actions.
  - Removed embedded objects.
- Supported UTF-16 encoded JSON files.
- Improved Analysis mode:
  - Reported the action actually applied to each object, instead of reporting no action, for Office, RTF, text, IDML, Project, 3MF, Cell, and Show files.
  - Produced consistent results for AEM, DBF, DWFX, PBIX, and URL files.
- Displayed processed objects in the MetaDefender Core interface when processing fails.
- Improved processing speed for Word, Excel, and PowerPoint files that contain many images.
- Improved HEIC support and fixed valid HEIC files being rejected.
- Fixed invalid file structure when processing AutoCAD files with a large number of pages.
- Fixed timeouts when processing PDF files that contain numerous comment lines.
- Fixed a path traversal issue when processing Office OOXML and flat XML Office files
- Upgraded bundled third-party components, including the .NET runtime and OpenSSL, to address known vulnerabilities.

v8.1.2

Release date: 11/4/2026

- Supported AutoCAD R14 drawings in DWG sanitization.
- Removed image content hidden underneath redaction annotations in PDF.
- Extended PDF sanitization coverage:
  - Removed Screen and Projection annotations.
  - Processed embedded objects referenced by Rendition actions.
  - Removed page thumbnails and the document information dictionary completely.
  - Removed alternate presentations from the name dictionary.
- Applied the Skip Allowlisted Objects setting in Analysis mode, so allowlisted objects are no longer blocked when sanitization is not performed.

- Improved image sanitization consistency across Microsoft Office formats.
- Fixed engine initialization failure on hardened Linux systems where font directories are not readable by the service account.
- Fixed an application error and missing images when sanitizing JTD and JTDC files.
- Upgraded the OpenSSL library bundled with the HTML conversion component to 3.6.3 on Linux.

v8.1.1

Release date: 10/14/2026

- Supported HL7 file sanitization:
  - Removed embedded objects from HL7 files.
  - Applied image processing to images embedded in OBX segments.
  - Sanitized embedded files recursively.
- Improved sanitization performance for image-heavy Office 2007 files by processing embedded images in parallel.

v8.1.0

Release date: 9/23/2026

- Supported hidden data removal in Word, Excel, and PowerPoint 2007 files.
- Supported encapsulated object removal and recursive sanitization in DICOM files.
- Supported AFF4 compression.

  Supported header and footer removal in Word 2007 and Excel 2007 files.
- Supported digital signature removal in Office 2007 files.

  Supported video object removal in PowerPoint 2007 files, including orphaned thumbnails shared with audio objects.
- Supported attached template reference removal from Word 2007 files.
- Extended IDML sanitization to remove unused data and all externally linkable content, and to allowlist hyperlinks.
- Improved NEF processing by validating all image file directories and removing the invalid ones from the output.
- Preserved non-text objects in hidden rows and tables when removing hidden data from Word 2007 files.
- Preserved the visual appearance of PDFs with an invalid ToUnicode CMap.
- Separated image removal reporting for images with and without object metadata.
- Reported old Excel 2 workbooks as an unsupported version instead of an invalid file structure.
- Supported complete comment removal in PowerPoint 2007 files.
- Supported complete hyperlink removal in Office 2007 files.
- Patched FFmpeg 8.0 to address CVE-2026-8461.

- Fixed a crash caused by heap corruption during WDP watermark sanitization.
- Fixed PDF forms no longer working after sanitization.
- Fixed sanitized JTD files that could not be opened in Ichitaro.
- Fixed PDF to TIFF conversion keeping only the first page.
- Fixed MOV files becoming unplayable after sanitization.

v8.0.3

Release date: 8/26/2026

- Fixed memory leak when converting from DXF to PDF.
- Fixed a stability issue when processing PPTX files.
- Fixed an issue with JTD sanitization.

v8.0.2

Release date: 8/13/2026

- Supported hyperlink processing and metadata removal for IDML files.
- Supported embedded audio removal in PowerPoint 2007 files.
- Supported showing of hidden text with invisible Unicode characters.
- Added details of removed DTD declarations to the XML sanitization report.
- Improved image removal in Excel 2007 files by preserving thumbnails of embedded objects.
- Fixed corrupted output for Word 2007 files when Remove Structured Document Tags is enabled.
- Fixed invalid file structure reported for DWF (containing Group 4 fax images), XLS (non-standard compound file header), WAV (invalid fmt chunk size)

v8.0.1

Release date: 7/22/2026

- Supported for Adobe InDesign Markup (IDML) file.
- Enhanced Microsoft Office sanitization:
  - Added removal of document protection elements from Word, Excel, and PowerPoint 2007 files.
  - Enhanced complete revision data removal from Excel (XLSX) files.
  - Improved processing efficiency for Office 2007 documents, covering image, comment, embedded object, structured document tag, and field code handling.
  - Fixed an issue where a thumbnail image was not removed when its embedded object was removed in Office 2007 documents.

- Enhanced PDF sanitization:
  - Added sanitization of inline images embedded in PDF content streams
  - Fixed an issue where the last pages of some PDF files became blank due to image resource key normalization.
- Fixed broken output when sanitizing DRC files extracted from B3DM.

v8.0.0

Release date: 7/2/2026

- Supported sanitization for JT, XML-XLS file.
- Enhanced PDF sanitization:
  - Added removal of additional annotation objects.
  - Added removal of internal /GoTo actions.
  - Improved XFA processing to handle non-compliant XML structures.
  - Added encoding support for the ASCII85Decode filter.
- Enhanced Microsoft Office sanitization:
  - Added removal of complex and standard field codes from Word files.
  - Added removal of endnotes and footnotes from Word files.
  - Added removal of notes slides from PowerPoint files.
  - Preserved embedded object thumbnails when removing images from Word and PowerPoint files.
  - Supported allowlisting of embedded binary content in Excel files when using the Block file option.
  - Applied the enhanced embedded object and chart processing workflow to Microsoft PowerPoint 2007 and Excel 2007
- Supported template removal in RTF files.
- Supported dynamic redirect handling for hyperlinks in QR codes.
- Added blocking of outbound network connections during HTML conversion.
- Improved sanitization performance for Microsoft Office 2007 files, delivering up to a 50% speed increase and reducing memory usage by up to 2× for large files.
- Upgrade Windows Compiler from VS2019 to VS2022.

v7.8.2

Release date: 5/20/2026

- Supported 3MF file sanitization.
- Supported image removal in PowerPoint 2007 and Excel 2007 files.

- Supported removing control document variables in Word 2007.
- Supported dynamic safe URL redirection in PDF and OpenOffice documents.
- Added option to skip image loading in HTML to PDF conversion.
- Improved processing performance and memory usage:
  - Reduced memory consumption when converting PDF to TIFF.
  - Optimized calcChain elements processing for XLSX with many external sheets.
  - Streamlined Word, PowerPoint, and Excel 2007 sanitization with single-pass processing.
- Fixed newline preservation in HTML attribute values.

v7.8.1

Release date: 4/15/2026

- New file type support: IFC, LYRX, BLZ.
- MS Office 2007 Enhancements
  - Improved Excel 2007 processing performance with optimized single-pass SmartTag handling.
  - Removed ActiveX definitions in PowerPoint 2007 and Excel 2007.
  - Enhanced content type processing for Excel 2007.
  - Improved external link handling in Excel 2007.
  - Added support for removing embedded font extension definitions in Office 2007 files.
  - Removed embedded object definitions and zeroed content in PowerPoint 2007.
  - Added support for removing template field codes in Word 2007 files.
  - Added support for dynamic safe URL redirect addresses in HTML/RTF/TXT/Office 2003/Office 2007 files.
- Added support for preserving special TIFF tags (Artist, Copyright, Make, SMinSampleValue, SMaxSampleValue, ...)
- Improved CRLF line ending handling in HTML output for compatibility with Microsoft Exchange.
- Improved BOM-based encoding handling in XML files.
- Fixed empty RTF output when removing metadata containing the rempersonalinfo node.
- Fixed an issue where JWW files became unreadable after sanitization.
- Allow skip loading images in conversion of HTML to images.

v7.8.0

Release date: 3/25/2026

- MS Office 2007 Enhancements:
  - Added support for image removal in Word 2007.

- Removed structured document tags in Word 2007.
- Removed author information inserted through Field Codes in Word 2007.
- Removed embedded objects referenced via "oleLink" in Excel 2007.
- Removed Relationship nodes in Definition path, Usage of embedded object when removing Embedded objects.
- Enhance removal of embedded object with multiple definitions.
- Handle document properties with exceptional paths.

- PDF Enhancements:
  - Handled XSS in PDF text fields.
  - Handled concatenated PDF files.
  - Implemented custom ratio scaling for PDF to rasterized PDF conversion.
  - Enhanced processing flow for innocent actions in PDF files.
  - Disassociated "Document open action" from "Block File > Script objects" section.
  - Prevented hyperlinks in comments from being re-added during re-sanitization.
- Support namespace in SVG sanitization.
- Enhanced XML Base64 processing for attribute values and self-closing nodes.
- TIFF Enhancements:
  - Preserved compression method for TIFF64 files.
  - Preserved EXIF tag RPCCoefficientTag in TIFF images.
- Added QR Sanitize support for cgBI PNG format.
- Fixed inconsistent sanitized image count behavior in RTF files.
- Added support for obfuscated emoji detection in TXT sanitization.
- Support Debian 13.
- Upgrade OpenSSL to 3.6.1.

v7.7.2

Release date: 2/5/2026

- Supported new file type: YENC.
- MS Office 2007 Enhancements:
  - Supported removal of custom property data.
  - Improved error messages for invalid file structures.
  - Removed all linked data when removing comments.
- PDF Enhancements:

  - Removed DTD data when removing XFA forms.
  - Added support for removing watermark annotations.
- Improved VCF Sanitization:
  - Handled QUOTED-PRINTABLE encoding.
  - Handled hyperlinks in text form.
  - Extended Base64-encoded data sanitization to arbitrary fields.
- Other Improvements:
  - Handled script tags with namespaces in SVG.
  - Improved handling of hyperlinks in RTF files.

v7.7.1

Release date: 1/14/2026

- Added support for VCF, JAR, WAR, LZ4, MAR.
- Added support for converting CgBI PNG files to other formats.
- MS Office 2007 enhancements:
  - Added support for converting DOCX/XLSX files to rasterized PDF.
  - Added support for removing image metadata in Microsoft Office 2007 files.
  - Added support for removing database field codes, DataBinding element in DOCX files.
  - Enhanced OLE object removal in DOCX files.
  - Reported hlinkfile and hlinkpres as hyperlinks in PPTX files.
- PDF enhancements:
  - Added support for removing text, popup, and free-text annotations.
  - Improved handling of PDF files containing /ObjStm objects.
  - Completely removed all related information when removing embedded objects.
  - Added dynamic fallback from Very Low to Low image quality.
  - Replaced invalid header versions with a valid one (1.7).
- Removed beta labels for several file types.
- Added CIS Level 2 support for Windows Server 2022.

v7.7

Release date: 12/16/2025

- Added support for High Efficiency Image File Format (HEIF).
- Added support for cgBI PNG image format.
- Improved LNK file sanitization, with command details shown in specific scenarios.
- More accurate reporting of comment counts in Microsoft Word and Excel 2007.
- Improved comment removal for PowerPoint (PPTX) files.
- Updated the "Allowlist External Image" feature to exclude other types of external objects.
- Added audio frame data validation for MP3.
- Allow configuration of processing threads for video conversion.
- Fixed rare omsMediaService crashes when processing HEIC files.
- Reduced Linux package size by up to 20%.
- Upgraded the third-party conversion tool to a newer version.
- Improved VSIX sanitization with enhanced structural validation and removal of untracked files.
- Improved removal of embedded OLE objects in MPP files.

v7.6.3

Release date: 11/19/2025

- Fixed backward compatibility issue with the old MetaDefender Core (5.6.1 or older).

v.7.6.2

Release date: 11/12/2025

- Support new sanitizations: STEP (STP), AV1 Image File format (AVIF), Microsoft Project (MPP). 3D Mesh Layer (3DML), Visual Studio Extension (VSIX), 3D Assembly (EASM).
- PDF Improvements:
  - Enhance processing of JP2 images.
  - Improve hyperlink handling.
  - Return "action" details when removing Javascript.
  - Add "Very Low" quality option for PDF to Image conversion.
  - Expose details about "unused objects".
- XLSX improvements:
  - Enhance XLS to XLSX sanitization to ensure compliance with a specific validation tool.
  - Improve hyperlink handling in case of non-standard formular structure.
- Add a new option to configure output dimensions when converting TIFF to BMP.
- Provide more detailed feedback when sanitizing a broken GIF file.

- Enhance WMV output when input files lack a video stream.
- Add support control framerate in video conversion.
- Allow character encoding fallback in XML sanitization.
- Add support for removing "foreignObject" elements from SVG files.
- Improve invisible text detection in TXT files.
- Return detailed conversion and sanitization information, including source and target file types.

v7.6.1

Release date: 10/14/2025

- Expose suspicious data removed during GIF sanitization.
- Handle empty X-ALT-DESC fields in ICS files.
- Add support for "Display Hyperlinks in Comments" in PDF sanitization.
- Add support for removing unapproved tags in HTML sanitization.
- SVG sanitization improvements:
  - Return more detailed information when SVG sanitization occurs.
  - Reclassified the removal of Windows event attributes under JavaScript.

v7.6.0

Release date: 9/24/2025

- Support RIBC data table format (RIBC).
- Support PPT to PPTX conversion and OLE-based FLA to WAV.
- Skip signed PDF files based on provided certificate files.
- Enhance XML sanitization:
  - Support validating XML files against an XSD schema.
  - Support recursively sanitizing base64-encoded data in specified XML paths.
- Enhance DWFX sanitization:
  - Support removing embedded fonts.
  - Sanitize hyperlinks in .fpage and XML files.
- Apply syntactic checks to multiple XML structure files in MS Excel/PowerPoint 2007.
- Support DocuWorks version 10.
- Support sanitizing objects under a foreignObject element in SVG.
- Support analysis mode for “Process ASCII-based QR code” in MS Word.

- Return returning header and footer validation results for MS Word 2007.
- Return raw data information in image sanitization (when Include Processed Objects is enabled).
- Support validation of the RIBC Table in UTIB sanitization.
- Support JSON files in UTF-8 with BOM.
- Enhance image processing speed in MS PowerPoint 2007.
- Enhance JP2 image sanitization for PDF.
- Handle escaped characters in script nodes properly.
- Distinguish between email addresses and hyperlinks in several file formats.

v7.5.2

Release date: 8/19/2025

- Support for new file types:
  - MPP to PDF, MPP to XLSX,
  - Archive file: LZIP, ASAR.
  - Other: B3DM, DWFX.
- New configuration in HTML sanitization to replace "." with "[.]" in hyperlinks for domains.
- Support sanitizing ASCII-based QR codes in TXT files.
- Support removing "href" and "xlink:href" in the "script" node during SVG sanitization.
- Support "Excluded file types for Recursion" in MS Word sanitization to remove embedded files instead of performing recursive sanitization.
- Support allowing User-Defined Named Actions in PDF.
- Support base64 data in XML.
- Added syntactic validation for Custom XML property files in MS Office documents.
- Improved scan results to show structure validation outcomes more clearly.
- Fixed an issue where Deep CDR's skip allowlisted feature was not working in some cases.
- Added support for Windows Server 2025 and Ubuntu 24.04.

v.7.5.1

Release date: 7/24/2025

- Support new file types:
  - AVC to AVC (the file type is enabled on-demand, please contact Customer Support or Sales team if you want to use it).
  - Convert XPS to PDF, PPTX to PPT.

  - Convert OPUS to WAV, FLA to WAV.
- Detect and report ASCII QR code in HTML.
- Improve sanitization for HTML with UTF-16 content.
- Upgrade OpenSSL to 3.3.3.

v7.5.0

Release date: 6/25/2025

- Support new file type:
  - WSQ, ODG, HEVC, TTML, MPD.
  - Convert PDF to rasterized PDF, VSD to PDF.
  - Convert MPP to WAV, PAF to WAV, M4A to WAV, CAF to WAV.
- Support QR code created using ASCII characters in RTF.
- Support QR code for PDF.
- Enhance sanitization:
  - Validate MS Excel 2007 file structure against zip format.
  - Handle a specific non-standard multiple HTML tags case.
  - Support new version of HEIC.
  - Better handling big MS Excel 2007, TIFF files.
  - Improve PDF, ICS, MS Powerpoint sanitization.
- Upgrade .NET to version 8.
- Drop support CentOS 7/RH 7/Debian 10/Ubuntu 20.

v7.4.2

Release date: 5/28/2025

- Support new filetype conversions:
  - Video: MTS to MP4, OGG to MP4, WMV to MP4, MP4 to AVI, AVI to MP4, AVI to WMV, MTS to WMV, SWF to WMV, WMV to AVI, MP4 to WMV
  - Audio: M4A to MP3, OPUS to MP3, WAV to MP3, WAV to MP4
  - Others: ICO to BMP, ICO to PNG, DOCX to DOC, XLSX to XLS, ODT to PDF, OTT to PDF, PDF to PPTX
- Enhance Word 2007 sanitization:
  - Validate many XML structure files against the XSD schemas.
  - Validate file structure against zip format.

- Enhance WAV sanitization.
- Support Amazon Linux 2023.

v7.4.1

Release date: 4/24/2025

- Support new file types:
  - Sanitization: URL, MDZip
  - Conversion: XLS to XLSX, TIF to PDF
- Support validating footer and header in MS Word, Excel 2007.
- Support QR code created using ASCII characters in MS Word 2007.
- Optimize resource usage when removing comments.
- Enhance timeout handling in case of recursive sanitization.
- Improve ICS sanitization in case of invalid html content.

v7.4

Release date: 3/27/2025

- New options for MS Office 2007:
  - Remove Smart Tag.
  - Validate Document Properties.
  - Validate Template Name.
- Support returning "Reason for action" to explain why Deep CDR removes/ sanitizes/validates an object (support Smart Tag, Document Properties, Template Name only)
- New configuration in HTML to allow output with pretty-printing style.
- New file type support:
  - convert MJ2 to MP4, MOV, AVI and WMV.
  - SCDOC
- Skip signed Macro.
- Support sanitization of base64 encoded data in JSON.
- Support skipping sanitization in special cases (Unsupported version, Exceeded file size, password protected, Invalid file structure).
- Support QR code created using ASCII characters in MS Word 2003.
- Better handling MP4, ICS, EML, PDF, CSV sanitization.

- Improve WEBP sanitization speed.
- Improve VSTO removal in DOCX.
- Update sanitization description to be clearer.

v7.3.2

Release date: 2/19/2025

- Support ZSTD, YZ1 sanitization.
- Allow users to input exceptional characters when removing control characters in TXT sanitization.
- Optimize resource usage for handling very large MP4 files.
- Enhance metadata removal for MP4, MOV, OGG files.
- Support metadata removal when converting from MP3 to MP4.
- Improve handling of charset values with double quotes (Content-Type: text/html;charset="...") in HTML sanitization.
- Improve `/FontMatrix` attribute processing to address CVE-2024-4367.
- Fix PDF form processing to prevent the removal of more objects than necessary.
- Update error messages for image sanitization when exceeding the size limit.
- Remove the Beta label for several file types.

v7.3.1

Release date: 1/15/2025

- Support removing invisible characters in TXT.
- Better handling HTML encode.
- Improve ODP sanitization to avoid missing images.

v7.3

Release date: 12/18/2024

- Support TNEF sanitization.
- A new configuration in MS Office 2007 to allow external images by given paths.
- A new speed mode for image sanitization
- Add "hidden watermark" to several file types to prove that Deep CDR validated the file structure.
- Add "pixel data" in sanitization details for image sanitization.
- Improvement:
  - OpenOffice (ODP, ODS, ODT) sanitization: better handling charts, images.

  - Preserve color palette in PNG.
  - Preserve HTML style tag "<!--" in HTML sanitization.
  - Handle MS Excel 2003 only contains a Book stream without a WorkBook stream.
  - Remove redundant data in AVI.
  - Handle NULL data in DictionaryObject in PDF.
  - Faster process XLSX contains large sheets.
- Support Rocky Linux 9.4, Debian 12.

v7.2.3

Release date: 11/13/2024

- Support MHT to MHT sanitization.
- Support new file type conversions:
  - MKV, MOV and OPUS to MP4.
  - MKV, M4V, 3GP and MTS to WMV.
  - MKV, M4V, MPEG, MTS and MOV to AVI.
- Support QR sanitization for MS Office.
- Enhance hyperlink processing for MS Office, **SHOW** file.
- Enhance process hyperlinks in HTML.
- Enhance EmbeddedPress removal towards preserving WordArt in JTD.
- Distinguish internal hyperlinks and external hyperlinks for the PowerPoint 2007.
- Optimize PDF hyperlink processing to use less system resource.
- Digitally sign all binary files.

v7.2.2

Release date: 10/23/2024

- Fix Japanese encoding in the HTML sanitization.

v7.2.1

Release date: 10/8/2024

- Support new file types:
  - EML, MSG, MBX and ESZ to ZIP.
  - 3GP, MP3, MPEG, M4V to MP4.

  - OTT to PDF.

- Extend Analysis mode for Audio, Image, Video.
- Support removing Book stream, printerSettings and hidden sheets in MS Excel.
- Enhance DDE detection in MS Excel.
- Handle the DDE with Japanese full-width characters in CSV.
- Handle HTML with ISO-2022-JP properly.
- Enhance VBA project processing in MS Office sanitization.
- Enhance OLE object removal in RTF sanitization

v7.2

Release date: 9/18/2024

- Support new file types:
  - TGA, ACM, MBX, JNLP.
  - GPKG, B3DM to ZIP.
  - CSV to XLSX.
- New configurations in the Blocking mode:
  - Blocking CVS files with formulas
  - Blocking PDF files with invalid digital signature
- New configuration in PDF to allow users to keep embedded objects base on file types.
- Handle white colored QR code.
- Handle "**onbegin**" event in HTML and SVG.
- Report DDE objects in MS Office more accurately.
- Distinguish regular hyperlink and HYPERLINK functions in MS Excel.
- Enhance TIFF, JPG, and DWF sanitization.

v7.1.1

Release date: 7/24/2024

- Sanitize Windows Compressed Enhanced Metafile (EMZ) files.
- Block files with "**Invalid file structure**" in the `Blocking mode`.
- Sanitize XML-DOCX files.
- Support QR code with transparent background in BMP, PNG.
- Validate digital signature in signed PDF files.

- Remove Beta labels for several file types.

v7.1

Release date: 6/25/2024

- Support new file type: Microsoft Power BI (PBIX), PowerPoint XML Presentation (XML-PPTX), Broadcast Wave 64 (BW64), Multiple Network Graphic (MNG), and Bean Markup Language (BML).
- Support DWG version 2000.
- Support macro **allowlist** for MS Office 2003.
- Support more objects in PDF, including:
  - PostScript,
  - Reference XObject, and
  - Named action.
- Support Japanese full-width formula signs in CSV sanitization.
- Support Shift-JIS encoding for TXT to PDF conversion.
- Support Shift-JIS binder ins XBD.
- Support Ubuntu 22.04.
- Enhance metadata removal for embedded images in MS Office.
- Enhance MS Word 2007 sanitization with documents missing main rels.
- Enhance image sanitization in PDF.
- Improve HTML encoding in EML sanitization.

v.7.0.1

Release date: 4/22/2024

- Resolve service startup failures.
- Improve HTA hyperlink handling.

v7.0

Release date: 4/3/2024

- Support DWF to DWF sanitization.
- Support new file types: Adobe Illustrator Template (AIT), Log (LOG), and Research Institute on Building Cost (RIBC).
- Support allowlisting of macros for Microsoft Word 2003.
- Improve sanitization for Microsoft Excel, Microsoft Office 2007, HTA, and EML.

- Return metadata object details for several file types.
- Upgrade the engine to use .NET 6.

v6.8.2

Release date: 2/5/2024

- Enhance signed PDFs with compressed catalog processing.

v6.8.1

Release date: 1/16/2024

- Enhance macro processing for Microsoft Excel.
- Enhance hyperlink processing for PDFs.

v6.8.0

Release date: 12/19/2023

- Support new file types:
  - Archive files: e-dossier (ES3), System Center Configuration Manager Package Files (PCK).
  - Shape format (DBF), Printing (PRN), and MS Excel Add-ins (XLAM).
- QR code enhancements:
  - Support BPM format.
  - Add the capability to block images if they are QR codes.
  - Improve processing of QR code with rotation.
- Add a new hyperlink configuration for PDFs to prevent PDF readers turning text to hyperlinks
- Return metadata object details for MS Office 2003/2007 and PDFs.
- Improve attachment handling for EML sanitization.
- Enhance custom XML removal.
- Improve JWW sanitization.
- Fix the temp file removal issue during high load.

v6.7.2

Release date: 11/9/2023

- Better handling Japanese texts in the PDF sanitization.
- Improve non-English encoding when converting MS Office.

- Fix a race condition issue during a high-load situation.

v6.7.1

Release date: 10/5/2023

- Fix crash issue when processing a specific webp file

v6.7

Release date: 9/27/2023

- Support QR code sanitization
- New file type support: Material Exchange Format (MXF), V-nas BFO (BFO), Storyist Document (STORY), HTML Archive (HAR), VTXT, CAB to CAB
- Return base64 data in CDR details for OLE objects (it is configurable)
- Add watermark to images structural data inside documents
- Support removing windows event attributes on XML file format
- Show the reason of Blocked by Deep CDR
- Remove BETA label for EGG, WMA, OGG, OPUS, HEIC, JWC, JHD, APNG, TSV, OFT, BWF, W64, RF65, AIFF
- Fixed sanitization failure for a zip file with specific LNK files in it

v6.6.2

Release date: 8/9/2023

- Support DocuWork Container (XCT) sanitization
- Fix engine crash when processing TIFF file

v6.6.1

Release date: 7/19/2023

- Support AltChunk resource in MS Office
- Improve PDF validation rule
- Support new encoding in XML sanitization

v6.6

Release date: 6/20/2023

- Block file based on sanitization details (require MetaDefender Core 5.5.1+)
- New file type support: GeoTIFF, OneNote (ONE), Advanced Audio Coding (AAC), Flash Video (FLV), MPEG-2

Tranmission (MTS), AutoCAD DXF (DXF), Shapefile (SHP, SHX), Powerpoint add-on (PPAM), Tableau workbook (TWB), DocuWorks Binder (XBD), Resource Description Framwork (RDF)

- Support removing customXML in DOCX/XLSX/PPTX
- Improve sanitization for JSON, WAV, MP3, MP4, Images
- Improve logging

v6.5.2

Release date: 4/13/2023

- Optimize the sanitization process during highload
- Improve AutoCAD sanitization
- Improve logging

v6.5.1

Release date: 4/2/2023

- Enhance URI checking in Calendar file sanitization
- Handle big PDF Outline tree

v6.5

Release date: 3/15/2023

- New file type support: Audio Interchange File Format (AIFF), Justsystem Hanako (JHD).
- Support RTF font table processing
- Support TXT with ISO/IEC 8859-1 encoding
- Improve the dependencies check to display proper messages
- Improve WMV processing
- Enhance external image removal in Excel 2007
- Enhance hyperlink processing in ICS
- Split the processors to services to improve the stability

v6.4.1

Release date: 2/13/2023

- Support new file types: Tableau Packaged Workbook (TWBX), Tableau Datasource (TDS), ShadowProtect Full Backup File (SPF)
- Enhance JPEG sanitization to prevent file size from increasing

v6.4

Release date: 1/11/2023

- New file type support: BWF, W64, RF64
- Support encoding Shift-JIS in TXT sanitization
- Handle SVG bomb attack

v6.3.2

Release date: 12/2/2022

- Support new file types: OFT, OPUS
- Fix an issue when converting a big SVG to BMP
- Preserve EML header when sanitizing files

v6.3.1

Release date: 11/3/2022

- Support DOCX to RTF conversion
- Enhance HTML sanitization
  - Better Zero-font removal
  - Support "base" tag and relaitive link
- Apply image size setting to BMP
- Improve PDF sanitization

v6.3

Release date: 10/5/2022

- Support Analysis mode
- Support APNG converstion to PNG, Ogg Vorbis Compressed Audio File (OGG), Tab-Separated Values (TSV) sanitization
- Support "Skipped Signed File" configuraiton for MS Office
- Allow user to set log level in the web management console
- Support recursive sanitization for PDF
- Introduce size limit settings for TIFF
- Display an error message when missing engine dependencies (MetaDefender Core 5.2.1+)
- Enhance comment removal in HTML

v6.2.2

Release date: 8/22/2022

- Support JW CAD (JWC), Windows Media Audio (WMA) sanitization
- Remove linked objects in ODT sanitization
- Remove Content Description, Extended Description and Content Branding when removing metadata in WMV/WMA

v6.2.1

Release date: 8/2/2022

- A configuration to skip processing digital signed MS Office Word/Excel 2003 and MS Office 2007
- Enhance WMV sanitization
- Allow whitelist macro hash with case insensitive
- Better handling signature in PDF file

v6.2

Release date: 6/29/2022

- Support new file types:
  - MPEG-4 format for Audio (M4A)
  - High Efficiency Image File Format (HEIC), the file type is enabled on-demand, please contact Customer Support or Sales team if you want to use it.
- Better handle executable actions in PDF (all annotations and actions)
- Sanitize Visual Studio macro (VSTO) in MS Office documents
- A new GIF configuration (GIF Layer Count Limit) to limit the number of GIF layers
- Improve LNK sanitization to show sanitization details better
- Better timeout handling
- Handle loading external resources in MS Office processing
- Remove BETA label for WEBM, VDX, VTX, VSX,TIFF64

v6.1.3

Release date: 6/2/2022

- Improve PDF sanitization

v6.1.2

Release date: 5/16/2022

- Process more objects in PDF sanitization
- Preserve empty encoding declaration in XML
- Fixed line folding issue for ICS sanitization

v6.1.1

Release date: 4/27/2022

- Report JavaScript Hyperlink as JavaScript in PDF sanitization details
- Using LZW compression for BMP to TIFF, PNG to TIFF, DWF/DXF to TIFF conversions
- Optimize PDF sanitization resource

v6.1

Release date: 3/28/2022

- Support JSON, CRL file format (BETA)
- Remove BETA label for several file types
- A new configuration to return hyperlink without processing for Microsoft Office and PDF
- Support whitelisting hyperlink for Open Office, PDF, Hancom, JustSystems Ichitaro, Visio, HTA, XDW
- Improve PDF, TXT, Excel, SVG, LNK, HTML, JWW sanitization
- Improve debug log: log location is moved to the engines log folder, support log rotation

v6.0.2

Release date: 2/10/2022

- Enhance stability
- Fix HTML encoding issue

v6.0.1

Release date: 1/10/2022

- Change Blacklist/Whitelist to Blocklist/Allowlist
- Remove unnecessary files in the package

v6.0

Release date: 12/15/2021

- Significant improvement on Linux version:
  - Performance increases 50% in average
  - Add 200+ file type conversions
- Support whitelisting hyperlink for MS Office, RTF, HTML, Calendar file
- Remove BETA label for several file formats
- Support big TIFF, Nikon Raw Image **(**NEF), Hangul Word Processor 2010 Document (HPWX) file format
- Handle more image formats in PDF
- Improve timeout handling
- libgdiplus 6.0.5 or above is required for some file type conversions

v5.14 (Windows Only)

Release date: 11/22/2021

- Improve PNG sanitization to avoid increasing file size
- Enhance RTF/PDF sanitization to handle unexpected cases
- Support new file type conversion: EGG to the other archive file types

v5.14 (Windows Only)

Release date: 10/4/2021

- Support new file types (beta):
  - VDX to VDX, VSX to VSX, VTX to VTX,
  - WebM
- Support MPEG-1 encode in MPEG sanitization
- Remove beta label for several file types, please check the Supported File Types page to see the full beta list
- Better macro handling in MS Office 2007
- Enhance HTML sanitization with specific encoding and non-ASCII characters

v5.13.1

Release date: 8/17/2021

- Handle MPEG-1 encode in MPEG sanitization
- Improve PDF/HTML sanitization to handle corner cases, none standard format.

v5.13

Release date: 7/5/2021

- Remove BETA label for AI, JTDC, HWT, DWT, DWS, WAV, SVG, VCS, ICS, TXT, PPS, XML
- Performance improvement on Linux
- Support strict validation configuration for WAV
- Support new file type: XDW (DocuWorks Image). Customers need to install and active DocuWorks 9 or newer on the system to use this feature
- Several minor bug fixes
- libgomp (Centos), libgomp1 (Ubuntu) is required

v5.12.3 - hotfix

Release date: 6/15/2021

- Fixed the configuration issue when upgrading from Core 4.19.x to Core 4.20.x

v5.12.3

Release date: 6/3/2021

- Support strict validation configuration for MP4/MOV
- Remove BETA label for MP4/MOV
- A new configuration to control image quality for Image sanitization
- Improve PDF sanitization

v5.12.2

Release date: 5/18/2021

- Support MOV sanitization
- A new configuration to control JPEG file size/quality in PDF file
- Improve m4a sanitization

v5.12.1

Release date: 4/27/2021

- Improve XML sanitization performance on Linux: 10x faster
- Fix PDF hang issue
- Better handling Think Cell object in Microsoft Office

v5.12

Release date: 4/8/2021

- End of support Windows 7, 8, and 8.1; Centos 6+; and RedHat 6+
- Recursively sanitize PST (require .NET Core 3.1 or newer to be installed on Linux
- Support new file types: JW CAD (JWW), SXF Feature Comment (SFC), P21 (STEP Data Model)
- A new configuration to process CDATA in XML
- Improve CSV, DWG sanitization on Linux

v5.11.2

Release date: 3/1/2021

- Improve XLSB sanitization
- Support to remove event attribute in HTML file
- Fix sanitization failure when a TIFF file has unknown custom tags
- Fix compound file corrupted after sanitization

v5.11.1

Release date: 1/27/2021

- Fix a crash issue when sanitizing LNK file
- Fix sanitization failures with some specific XLM, PDF, PNG files.

v.5.11

Release date: 1/11/2021

- Per-file configuration will be available as part of Workflow (requires MD Core 4.20.0 and above)
- Whitelist Macro based on Macro content
- Enhance engine communication on Linux
- Support XLS to CSV conversion

v5.10.3

Release date: 12/07/2020

- Reduce resource usage when sanitizing XML files
- Fix a crash issue when sanitizing Excel files

v5.10.2

Release date: 11/16/2020

- Fixed a hang issue with XLSX sanitization
- Fixed a PDF timeout issue

v5.10.1

Release date: 10/28/2020

- A better way to remove VBA macro in XLS
- Fixed a bug that causes some specific PDF files cannot be opened after sanitizing
- Fixed memory leak issue in MP3 sanitization

v5.10

Release date: 09/28/2020

- Support new file types:
  - Open Office Spreadsheet (ODS, OTS)
  - Open Office Presentation (ODP, OTP)
  - Hangul Word template (HWT)
- Support recursive sanitization for MS Office 2003 (DOC, XLS, PPT) , Open Office (ODT, OTT, ODS, OTS, ODP, OTS) , Rich text format (RTF)
- New configuration to remove JavaScript hyperlink in HTML
- Preserve object thumbnail after removing OLE objects in Office 2003
- Improve Chart object handling in XLSB
- Better hyperlink handling in RTF: remove hyperlinks properly, keep bookmark links
- Improve XFA form sanitization in PDF
- Enhance processing PpAction (mouse over llink and mouse click link) objects in Powerpoint 2007

v5.9.3

Release date: 9/1/2020

- Better handling JSON details
- Improve TXT sanitization
- Better HTML redirection in cases a hyperlink is escaped
- New file type conversion: XLSM to CSV, XLTX to CSV, DOC to RTF, DOCM to RTF, HTML to TXT

v5.9.2

Release date: 8/10/2020

- Support new file types:
  - File type conversions on Linux: PPTX/PPSX/SVG to BMP
  - TIP Test selection engine (TSE, TSEC, TSEZ)
  - WinACE archive format (ACE) conversion to other archive file formats
- Better handling embedded PDF file in MS Excel 2007
- Improve PDF, TIFF, RTF sanitization

v5.9.1

Release date: 7/15/2020

- Hotfix to work with MetaDefender Kiosk Secure Image

v5.9

Release date: 7/7/2020

- Support new file types:
  - Digital Imaging and Communications in Medicine (DCM)
  - AutoCAD Drawing Standards (DWS), AutoCAD Drawing Template (DWT),
  - Microsoft Powerpoint Slide 2007+ (SLDX), Microsoft Office PowerPoint Slide 2007 - Macro-Enabled (SLDM)
  - Windows Shortcut File (LNK)
  - HD Photo (WDP), Google Web Image (WebP)
  - DOC to DOCX, DOT to DOTX conversion
- A new configuration to allow hyperlink redirection in Microsoft Offices, Microsoft Visio documents, Hancom documents, PDF, RTF, ICS, VCS, ODT
- Better timeout handling for Microsoft Office, PDF sanitization
- Improve HWP sanitization: better image handling, separate chart configuration with embedded objects, fix memory leak
- Better handling Tiff images, unused objects, 3D objects in PDF sanitization

v5.8.1

Release date: 5/20/2020

- Support ALZip (ALZ) sanitization

- Support removal of thread comment in Microsoft Excel
- A new configuration to allow zero font in HTML file
- Improve SVG, XML, UTF-16 HTML sanitization

v5.8

Release date: 4/14/2020

- Microsoft Visual C++ 2017 Redistributable is a new requirement
- New configurations:
  - A configuration to skip a PDF file with digital signatures
  - A configuration to process a HTML file which has multiple <html> tags in it.
  - A configuration to keep/remove chart in Microsoft Office files
- Support new file types: HTML Application (HTA), Adobe Illustrator (AI), WinRAR (RAR to RAR)
- Improve Microsoft Office sanitization: better file path handling, improve parser performance
- Better PDF timeout handling
- Improve image quality when converting from PDF to TIFF

v5.7.4

Release date: 3/23/2020

- Better handling special signs (",+) in CSV
- Improve hyperlink removal in RTF

v5.7.3

Release date: 3/5/2020

- Handle Meta Refresh tag in HTML
- Better OLE object handling in DOCX

v5.7.2

Release date: 2/18/2020

- Better hyperlink handling in PDF
- Improve embedded font removal in DOCX
- More stable JPG, ICO, CUR sanitization
- Minor changes in configuration UI

v5.7.1

Release date: 1/29/2020

- Fixed compatibility with FIPS compliance
- Better handling PDF/A-2
- New configurations to control image quality in PDF
- Return hyperlink list when choosing "Display hyperlink with domain only" in HTML sanitization

v5.7.0

Release date: 12/26/2019

- New configurations to sanitize hyperlinks in HTML
  - Return hyperlinks without processing them
  - Replace with text
  - Add a prefix URL to hyperlinks
- More details about the sanitized objects for MS Office, PDF, HTML
- New UI Configuration (available in MetaDefender Core 4.17.1 or newer)
- New support file type
  - Icon file format (.ico)
  - Cursor file format (.cur)
  - Hancom Cell (.cell)
  - Hancom Show (.show)
- Improve PDF sanitization speed
- Better disk I/O usage

v5.6.5

Release date: 12/2/2019

- Fixed additional break line when sanitizing HTML
- Better handling metadata removal in Microsoft Office 2007

v5.6.4

Release date: 11/19/2019

- Improve PDF sanitization: betting handling metadata, none standard PDF files

v5.6.3

Release date: 11/7/2019

- Better processing Cross Reference Table (xref) in PDF
- Improve multi-threads handling for JTD sanitization

v5.6.2

- Preserve newline character when sanitizing CSV
- Better handling PDF Form with empty decoded data

v5.6.1

- Better handling invalid file structure for TIFF
- Better handling MP3 version 2.5

v5.6.0

- Better handling CSV formulas starting with special characters
- Microsoft Office sanitization improvement
  - Sanitize recursively all supported file types embedded in Microsoft Office 2007 documents
  - Better handling file path when processing
  - Improved invalid file structure detection for DOC
- PDF sanitization improvements:
  - Supported sanitize PDF file generated by Adobe Illustrator
  - Improved PDF form sanitization
- Optimized system resource usage (RAM, I/O)

v5.5.3

- Supported TXT to TXT sanitization
- Introduced new options to process formulas in CSV
- Improved memory usage when sanitizing big HTML files
- Better handling invalid structure JPG files

v5.5.2

- Better resource usage for image processing
- Better disarming process for Microsoft Office sanitization

v5.5.1

- Supported new file types:
  - Text file (TXT)
  - Audio Video Interleave (AVI), Moving Picture Experts Group (MPEG)
  - Outlook Personal Folder (PST)
- Supported a new configuration to process image link in HTML
- Improved 20% PDF sanitization speed

v5.5.0

- Deep CDR as new name
- Supported new file types:
  - Media file format: MPEG-1 Audio Layer-3 (MP3) , Waveform Audio (WAV)
  - Archive file: Cabinet Archive (CAB), ARJ Archive (ARJ), LZH Archive (LZH), LZMA Archive (LZMA)
  - Email file: Outlook Message (MSG)
- Improved XLS/XLSM sanitization: better Macro/VBA handling
- Better embedded object removal for XLSX/PPTX sanitization
- Improved ODT sanitization: remove linked images; hidden text, paragraphs, comments
- Improved Deep CDR details

v5.4.3

- Improved file handling for HTML, CSV, images sanitization
- Improved process management in Linux sanitization
- Better encode handling for RTF sanitization
- Improved logging

v5.4.2

- Better DDE handling for DOCX
- Improved PDF sanitization: better handling MetaData, XFA form, Security setting
- Better temp file handling for JTDC sanitization

v5.4.1

- Improved Linux sanitization stability
- Improved ODT sanitization: font, embedded sheet handling
- Better DDE handling for DOC/XLS/XLT
- Better link handling for DOCX

v5.4.0

- Supported new 26 file formats, more than 100 sanitization types:
  - Video file format: MPEG-4 Part 14 (MP4), Windows Media Video (WMV)
  - Email file format: Electronic Mail (EML)
  - Microsoft PowerPoint: Microsoft PowerPoint (97-2003) Show (PPS), Microsoft PowerPoint (97-2003) Template (POT), Microsoft PowerPoint Macro-Enabled Show (PPSM), Microsoft PowerPoint Template (POTX), Microsoft PowerPoint Macro-Enabled Template (POTM),
  - Microsoft Excel: Microsoft Excel (97-2003) Template (XLT), Microsoft Excel Template (XLTX), Microsoft Excel Macro-Enabled Template (XLTM)
  - Microsoft Visio: Microsoft Visio Drawing (VSDX), Microsoft Visio Macro-Enabled Drawing (VSDM), Microsoft Visio Drawing Stencil (VSSX), Microsoft Visio Template (VSTX), Microsoft Visio Macro-Enabled Template (VSTM), Microsoft Visio Macro-Enabled Stencil (VSSM), Microsoft Visio Stencil XML file type (VSX), Microsoft XML for Visio Template (VTX), Microsoft Visio XML Drawing (VDX)
  - OpenDocument Document Template (OTT)
  - MIME HTML (MHT)
  - AutoCAD: Drawing Interchange Format (DXF), Design Web Format (DWF)
  - 3D files: Digital Asset Exchange (DAE), 3D Studio (3DS), Universal 3D (U3D), Google Draco (DRC), AVEVA Plant Design Management System Model (RVM)
  - RTF to PDF
- Return forensic info to show which objects were sanitized, removed
- Improved image sanitization speed, memory usage
- Improved PDF sanitization: better error classification, metadata handling, script handling, form handling
- Improved XLS, PPT sanitization: duplicated object handling, macro handling, image handling
- Improved HTML sanitization: handle hyperlink better, preserve content in "pre" tag, remove redundant data
- More stable Linux sanitization for DOCX, PPTX

v5.3.4

- Better handling empty HTML file
- Improved PDF parser to handle file structure better

v5.3.3

- Improved handling hidden content in DOCX sanitization
- Improved ODT sanitization in Linux

v5.3.2

- Improved error message for Microsoft XML Spreadsheet 2003 (.xml) sanitization
- Improved PDF sanitization: XFA form, Metadata parser

v5.3.1

- Enhanced XLS sanitization in case files only contain Macros
- Improved forensic info in HWP sanitization

v5.3.0

- Sanitization of recursively embedded documents in Microsoft Office 2007 documents
- Sanitization of Calendar data files: iCalendar (.ics) & vCalender v1.0 (.vcs)
- Enhanced EMF/WMF sanitization
- Improved image sanitization to prevent advanced steganography attacks
- Extended file support that now includes Ichitaro Document to Compressed format (JTDC), a compressed version of JTD
- Enhanced PDF sanitization functionality, including handling of digital signatures, JavaScript, form fields, and layers

v5.2.12

- Enhanced handling DDE with RTF file type
- Improved handling external objects in Microsoft Office document
- Improved handling Forms in PDF
- Improved error message for password protect documents

v5.2.11

- Enhanced the EMF sanitization inside PPT for Linux

- Improved error messages for invalid file structure in DOC, PPT
- Improved the TIFF sanitization inside PDF
- Better CDATA node handling in HTML sanitization
- Better temp file handling in Microsoft Office 2007 sanitization

v5.2.10

- Better OLE handling for HWP
- Enhance file structure validation for PDF, Microsoft Office

v5.2.9

- Better failure handling for PDF, DOCX sanitization
- Enhanced Microsoft Office sanitization: remove more linked objects, control stream.

v5.2.8

- Improved PPT sanitization speed
- Improved CSV sanitization speed in Linux
- Improved WMF/EMF sanitization to prevent device independent bitmap (DIB) copy vulnerability (CVE-2017-0190) and buffer overflow vulnerability (CVE-2005-2123, CVE-2005-2124)
- Improved forensic info and error message

v5.2.7

- Sanitization detail is available on MetaDefender Core REST API (version 4.12.1 and newer) (Beta)
- Improved WMF/EMF sanitization to prevent code execution
- Improved WMF/EMF sanitization for images inside document files
- Support Windows Metafile (WMF), Enhanced Metafile (EMF), Microsoft Visio Drawing (VSDX, VSDM) sanitization on Linux (Beta)
- Better embedded object handling for Microsoft Office file types
- Supported external media objects removal for PPT sanitization
- Better removing PDF metadata
- Improved HTML sanitization speed on Linux
- Added performance report for Linux

v5.2.6

- Enhanced steganography sanitization for PNG

v5.2.5

- Support Windows Metafile (WMF), Enhanced Metafile (EMF) sanitization on Windows (Beta)
- Better non-standard objects handling for PDF
- Enhanced HWP sanitization: content, image handling.

v5.2.4

- Improved Microsoft Office sanitization speed on Linux
- Improved PDF sanitization speed on Linux
- Improved HWP sanitization speed on Linux
- Improved JTD sanitization speed on Linux

v5.2.3

- Supported Microsoft Visio Drawing (VSDX, VSDM) sanitization on Windows (Beta)
- Improved hyperlink sanitization for HTML
- Improved hyperlink sanitization for HWP

v5.2.2

- Supported external media objects removal for PPTX/PPSX sanitization
- Improved hidden text removal for DOCX/DOTX sanitization

v5.2.1

- Fixed the issue of loading customized configuration on a 64bit system
- Handle NULL character in content for HTML to HTML sanitization
- Preserve UTF-8 BOM encoding for CSV sanitization

v5.2.0

- Handle big file size sanitization on Windows
- Improved PPSX sanitization: better OLE object, Command handling
- Improved file size with Microsoft Office files contain EMF/WMF images
- On Linux, additional file types support: DOCM2DOCM, DOTM2DOTM, DOTX2DOTX, XLS2XLS, XLSM2XLSM,

XLSB2SXLSB, PPT2PPT, PPSX2PPSX, PPTM2PPTM, RTF2RTF, HTML2HTML, WMF2JPG, XML-DOC2PDF, XML-XLS2PDF, XML-DOCX2PDF, JTD2JTD, HWP2HWP, DWG2DWG

v5.1.20

- Fixed XML, SVG sanitization issue

v5.1.19

- Enhanced Linux CDR performance
- Supported more file types in Linux CDR: XML2XML, SVG2SVG, DOT2DOT, GIF2GIF

v5.1.18

- Removed Linux BETA label
- Improved HWP sanitization: better image handling, stream processing.
- Fixed HTML sanitization issue when sanitizing links, scripts

v5.1.17

- Increased HWP sanitization speed
- Handling missing fonts for DOC sanitization on Linux
- Handling attachment removal for PDF with non-standard format
- Better handling of temporary files for DOCX, PPTX, XLSX sanitization

v5.1.16

- Fixed moniker type issue in PPT
- Fixed some handling embedded font in PDF

v5.1.15

- Improved Linux sanitization for DOCX
- Added support iso-2022-jp for HTML sanitization
- Enhanced CSV sanitization
- Enhanced PNG sanitization inside Microsoft Office documents

v5.1.14

- Enhanced Macro removal for AutoCAD (Beta)
- Adding support additional potentially malicious objects in DOCX
- Supported none Unicode encoding for HTML sanitization
- Enhanced sanitization of some non-standard format PDF
- Enhanced sanitization of some non-standard format RTF

v5.1.13

- Supported SVG to SVG sanitization (Beta)
- Adding support additional Potentially Malicious Objects (PMO) object types in PPTX
- Adding support additional PMO object types in XLSM
- Improve handling file structure integrity with a specific DOCX type

v5.1.12

- Enhanced PPTX, PDF sanitization
- Better handling of temporary files for image sanitization

v5.1.11

- Supported AutoCAD (Beta)
- Introduced new type of process_hyperlink_behavior for HTML
- Enhanced PPT sanitization
- Fixed bug on PDF sanitization: inverted colors in Linux, remove_form, ...
- Fixed image sanitization failure on Linux
- Fixed embedded object removal for JTD

v5.1.10

- Updated omsFileTypeConversion.exe icon for rebrand
- Fixed JTD sanitization lost images and objects
- Fixed bugs on Microsoft Office files: DDE, Macro,...
- Handled OLE Link for Microsoft Office files

v5.1.9

- Improved Linux sanitization

- Fixed bugs on PDF sanitization: corrupted files, crashed
- Distinguished Generic XML and Microsoft Office XML sanitization

v5.1.8

- Fixed bugs on PDF and DOCM sanitization

v5.1.7

- process_tag in HTML configuration was changed to process_hyperlink_behavior, default value is 1
- Removed metadata in RTF file
- Enhanced JPG, PDF, HTML sanitization

v5.1.6

- Supported DDE sanitization for DOC, XLS and CSV file format
- Enhanced sanitization on ODT file format
- Enhanced DLP support for DOC and DOCX sanitization (hidden text)
- Enhanced Microsoft Office 2007 file format (DOCX, XLSX, PPTX) sanitization (comment, revision)
- Enhanced stabilization for HTML, XLS and PPTX file format
- Enhanced validation for HTML and RTF file format

v5.1.5

- Improved PDF sanitization (reduced sanitized file size, optimized image processing time in PDF,...)
- Supported ODT sanitization (for both Windows and Linux)
- Improved JPG sanitization (reduced processing time)

v5.1.4

- Supported DDE sanitization for DOCX and XLSX file format

v5.1.3

- Enhanced metadata sanitization for Microsoft Office 2007 file format (Title, Subject, Comments, Author, Last Modified By, Company, Modified Date)
- Improved TIFF to TIFF sanitization

v5.1.2

- Improved DOC, PDF, HTML sanitization for validation
- Supported Form object sanitization in PDF

v5.1.1

- Supported metadata sanitization for Microsoft Office 2003 file format (DOC, XLS, PPT)

Proactive DLP release notes

v3.6.0

Release date: 11/10/2026

- Added OCR support for Simplified and Traditional Chinese
- Added built-in Airline Booking Reference (PNR) detection in TXT, DOCX, PPTX, and PDF files, supporting English, Hebrew, Arabic, Russian, and German
- Improved font-size obfuscation detection in Excel WordArt and PowerPoint SmartArt
- Enhanced ISIN validation and detection accuracy to reduce false positives
- Fixed Hebrew regex detection issues in PDF files, including reversed text, line boundaries, and multiline matching
- Improved PDF text extraction and scanning for Type 3 fonts and special characters
- Fixed incorrect hit column positions in large text files
- Reduced false positives involving hexadecimal values, Israeli National IDs, and incorrectly grouped ISIN numbers
- Fixed text extraction issues involving ligatures in LibreOffice generated PDFs
- Updated third-party dependencies to address security vulnerabilities

v3.5.0

Release date: 10/28/2026

- Added detection capabilities for retinal scans and fingerprints across the backend and UI, organizing biometric detection (including face detection) under a unified category
- Enabled redaction (masking and substitution) of sensitive data found in document metadata properties across Microsoft Office and RTF formats (DOC/DOCX/DOCM, XLS/XLSX/XLSM, PPT/PPTX/PPTM, RTF) supporting predefined types and custom regex rules
- Implemented backend support for advanced file type configurations and handling
- Added 20 new predefined sensitive information types across multiple countries and regions, including national identification numbers, tax identifiers, social security numbers, healthcare identifiers, and Passport Number. Each new type includes validation rules, contextual keyword support, and parser test coverage to improve detection

accuracy and reduce false positives (The full list can be found here: LINK)

- Upgraded the engine runtime and dependencies to .NET 10
- Minimum MetaDefender Core version raised to 5.14.0
- Fixed an issue where redacting RTF documents containing Right-to-Left (RTL) text mistakenly applied RTL alignment to unrelated sections
- Resolved an engine crash that occurred when scanning large text files with AI detection enabled
- Fixed an issue where regular expression pattern matching and sensitive data detection failed for Hebrew text inside PDF documents
- Fixed scan failures on large sector-split text files
- Fixed a Linux engine crash when several PowerPoint presentations were scanned at the same time

v3.4.2

Release date: 9/10/2026

- Added 42 new predefined sensitive information types across multiple countries and regions, including national identification numbers, tax identifiers, social security numbers, healthcare identifiers, and IPv6 addresses. Each new type includes validation rules, contextual keyword support, and parser test coverage to improve detection accuracy and reduce false positives (The full list can be found here: LINK)
- Added the optional `engine.build_time` and `engine.support_duration_months` fields to the package manifest. MetaDefender Core can use this information to determine an engine's lifecycle status and alert administrators when an engine is approaching or has reached end of support
- Added detection response handling and support for the new file type ID used for MISP-tagged files identified by FTA
- Fixed an issue with MetaDefender Core 5.18.1 where disabling the DLP engine could incorrectly generate crash notifications in the event log
- Fixed a Windows concurrency issue during custom regex TSV file reloads that could cause `engineprocess.exe` to crash and trigger repeated engine recovery or installation attempts

v3.4.1

Release date: 8/10/2026

- File name scanning and redaction is now a setting in its own right, and a file name that matches can be redacted by hashing the whole name rather than only the part that matched. The extension is preserved, including multi-part ones such as `.tar.gz`. Redacting only the file name now produces an output file, where previously none was written.
- Context word validation: Optional validation could already reject a detection when unwanted words appeared nearby, it can now require words to appear as well, so a match only counts when its surrounding text supports it.
- Detection policy expressions can now count distinct values rather than occurrences, so a rule can ask for three different card numbers instead of three copies of one: `count<hit_text>{type == "ccn"} >= 3`. The detected text is itself a field policies can match on, and a hit filter can be left empty to match every detection, so `any{}`

tests whether a file has any at all.

- Valid PDFs that use Adobe reflow content regions were rejected as malformed and skipped. They are now processed normally.

v3.4.0

Release date: 7/15/2026

- Arabic language is added to OCR with further accuracy improvements
- Metadata scan is now available for AVI, MP4, WAV files
- Human face redaction on ID cards
- Custom redaction for CCN - keep last 4 or last 4 and first 6 digits options

v3.3.1

Release date: 5/18/2026

- Added FIPS support and enhanced structural PDF validation capabilities
- Improved PII AI model performance and overall detection accuracy
- Resolved multiple stability and reliability issues
- Fixed several scanning and detection issues, including Excel AI scanning handling

v3.3.0

Release date: 4/21/2026

- Added support for resizing text that is too large or too small
- Added capability to highlight detected hits
- Added support for redacting text that is too large or too small
- Introduced threshold configuration for large font detection
- Introduced threshold configuration for cropped image detection
- Added support for font size and invisible text detection in PowerPoint files
- Extended support to WordArt and Shapes in Excel files
- Fixed an issue with incorrect redaction positioning on rotated PDF pages

v3.2.0

Release date: 1/26/2026

- Custom regex TSV files are now shown on a pop-up, so the user can decide which defined regexes they want to use

- Generic phone number detection
- E-mail address detection with both regex and AI methods
- Electronic Authorization Number detection with both regex and AI methods
- Name detection improvements utilizing AI

v3.1.0

Release date: 11/05/2025

- Security and privacy regulations such as GDPR and PHI can now be configured
- DLP now supports scanning filenames
- Users can choose whether to anonymize private tags and other user-defined tags in DICOM files
- Multiple TSV files can now be added for custom regex scanning
- The OCR section has been moved to the root configuration level, and language selection is now available
  - OCR now supports English, Hebrew, Hindi, French, Italian, Arabic, and Japanese, and can also detect MRZ (Machine Readable Zone) data from ICAO-compliant ID cards.
- Added text-encoding support for French, Italian, Arabic, and Japanese

v3.0.0

Release date: 07/14/2025

- Tagging system - DLP is now able to add metadata tags to processed PDF, Word, Excel, PowerPoint files and images
- AI scan is now available for Word and PowerPoint files as well
- DLP supports redaction for Json files from now on
- MDB and ACCDB file types are now supported for scanning
- Indian PAN and Aadhar cards are added as pre-defined options

v2.23.1

Release date: 05/12/2025

- Added support for Amazon Linux 2023 and Oracle Linux
- Resolved DLL loading issue on Windows Server 2016
- Fixed failure in NSFW scan under specific conditions
- Improved stability and error handling for large PDF scans

v2.23.0

Release date: 04/08/2025

- It is now possible to HASH the hits in the output document for Word, PDF and plain text files
- Toxic text detection is now available in more languages, including French, Spanish, Turkish, Italian, Russian, and Portuguese
- More NSFW categories introduced, including guns and violence
- More PIIs are available via AI scanning, including NHS and UK Electoral Roll Number
- Resolved an issue where enabling AI scan previously disabled regex scan. Both scans now function concurrently

v2.22.1

Release date: 02/05/2025

- DLP no longer uses "/tmp" directory in order to fully support non-root Docker
- Result UI page became more comprehensive, detailing scan types and processing results
- Switching on AI scan won't interfere with other features anymore

v2.22.0

Release date: 01/13/2025

- Intentional Leakage Detection
  - **Small Font Size Recognition**: Detect intentional data leaks by finding text hidden using very small font sizes in PDF, Word, and Excel files
  - **Invisible Text Recognition**: Detect intentional data leaks by identifying hidden text where the text color and background color are the same or very similar in PDF, Word, and Excel files.
- New predefined sensitive information types are now supported for detection and redaction:
  - Turkish Passport Numbers
  - Turkish Phone Numbers
  - Turkish ID Numbers (TC Kimlik)
  - IMEI/IMEISV (International Mobile Equipment Identity)
  - Israeli ID Numbers
- Fixed an issue where AI scanning disrupted regex-based scanning

v2.21.1

Release date: 11/04/2024

- Debian 12 and Rocky Linux 9.4 are now supported
- AI scan failure in case a PDF file has an empty page is now fixed

- Hit validation failure when only a TSV file is used for PDLP is now fixed

v2.21.0

Release date: 10/02/2024

- Anonymization is now supported for PCAP and PCAPNG files
- PII detection using Artificial Intelligence
  - Supported PII: Driver's license, passport number, and national ID number
  - Languages Supported: English, French, Spanish, German, Italian, and Portuguese
  - Supported File Types: PDF and text files
- Detection policy update fixed

v2.20.0

Release date: 07/25/2024

- Support the .jp2 file type.
- New predefined sensitive information types are now supported for detection and redaction:
  - Australia Medicare Number
  - Australia Company Number
  - Australia Business Number
  - Australia Tax File Number
  - UK NHS Number
- The localization select list in the UI has been replaced with a country-specific sensitive information select list.
- The US-SSN has been divided into US-ITIN and US-SSN (they are still referred to as SSN in the result JSON).
- The JP-SSN has been renamed to JP-MYNUM (it is still referred to as SSN in the result JSON).

v2.19.1

Release date: 06/13/2024

- Improve PDF metadata dictionary processing.
- Detection Policy improvements include: precedence, new operators, and parenthesis usage.
- Custom regex file updates no longer require an engine restart.
- Fix date regex issue in Excel files using TSV files.

v2.19.0

Release date: 04/11/2024

- Support multi-frame GIF files with OCR.
- Support .loc files, FDF files, and Acrobat Forms.
- Support recursive PDF processing (for embedded PDF only).
- Improve Microsoft Office 2007 object processing, including linked objects, tracked changes, background images, SmartArt.
- Improve PDF stamp annotation and form processing.
- Fix watermark scan bug.

v2.18.1

Release date: 02/15/2024

- Implement more sophisticated TSV file handling related to the "**custom regexes from file**" feature.
- Refine detection of SWIFT codes, reducing both false positives and false negatives.
- Resolved interruption issues related to hit limits during embedded image processing.
- Removed duplicate results that occur when using the best quality OCR.
- Fix the independence of redact settings from TSV file loading.
- Fix a memory leak

v2.18

Release date: 01/04/2024

- Importing custom regular expressions from a file is now supported.
- New predefined sensitive information types are now supported for detection and redaction:
  - ABA Routing Number
  - U.S. Bank Account
  - International Banking Account Number (IBAN)
  - International Securities Identification Number (ISIN)
  - SWIFT Code
- OCR capabilities have been improved.
- SSE4.2 is now also accepted by Proactive DLP (in addition to SSE4.1) as the required CPU instruction set for OCR.
- File metadata handling for scan requests have been improved.
- Proactive DLP's UI workflow settings are now unified across all officially supported MetaDefender Core versions.
- Proactive DLP's UI workflow settings for detection have been redesigned to provide a more compact look.

v2.17

Release date: 10/20/2023

- Announcing Document Identification
- Not Safe For Work (NSFW)
  - Detect "Not Safe For Work" content in text and images
  - Redact textual hits and blur images with NSFW content
- Personal document classification
  - Detect personal ID on images
- Detection of Generic Password and Generic API Key secrets have been improved

v2.16

Release date: 07/18/2023

- Announcing DICOM Anonymization:
  - Anonymize patient information according to the Basic Attribute Confidentiality Profile
  - Remove sensitive burned-in annotations from DICOM images
- Patterns for US SSN numbers have been updated
- Fixed issue when DLP engine crashes when files are sent for scanning right after engine initialization

v2.15.1

Release date: 05/22/2023

- Improved generic password and generic API token detection (secret detection)
- Fixed issue when processing of large text files took very long time
- Fixed issue when CCN hits contained extra character at the end
- Fixed sensitive info substitution in hyperlinks
- Fixed issue when PDLP was unable to detect certain software secrets in XML files
- Allowlist and custom validators now properly working with XML files

v2.15

Release date: 03/31/2023

- Support more secrets:
  - Generic passwords

  - Generic API tokens
  - PostgreSQL credentials
  - MYSQL credentials
- DOCM and XLSM file types are now supported in scanning
- Redact sensitive information in CSV and XML files
- Text substitution can be configured instead of redaction in the following document types:
  - MS office documents (word, excel, slides)
  - PDF files
  - Text files
- Fixed issue when valid hits can be invalidated due to a bug
- Fixed issue with duplicate character validator that resulted in more false positive

v2.14

Release date: 12/19/2022

- Support more secrets:
  - Private keys (PEM, PPK)
  - IBM Cloud key
  - IBM API Connect Credentials
  - IBM COS HMAC Credentials
- Reorganize the workflow configurations
- Improve MS Word processing to reduce false positive detection
- Fixed issue when valid hits could be lost when other hits are redacted during PDF processing
- Fixed log retention

v2.13.1

Release date: 11/2/2022

- Allowlist feature has been added to all sensitive info types
- Log retention period can now be configured from the engine configuration
- Encoding detection can be configured in the Workflow rule setting instead of the engine configuration
- Fixed issue where DLP engine updates were failing permanently due to unrecognized OS
- Fixed issue when encoding detection settings were lost during configuration export and import
- Fixed issue where an empty regular expression field in the workflow rule settings caused DLP engine update to fail

permanently

v2.13

Release date: 9/31/2022

- Secret detection in text files supporting AWS, Azure and GCP secrets
- New optional validators for sensitive data
  - Exclude prefix
  - Exclude suffix
  - Exclude beginning characters
  - Exclude ending characters
  - Duplicate characters (only for custom regexes)
- DLP log level is now selectable in the engine configuration
- External dependencies are checked before engine initialization
- Some descriptions have been streamlined in the workflow rule settings
- Fixed issue when local scan fails on read-only PDF files
- Fixed issue when SSN localization could be saved without value

v2.12

Release date: 7/5/2022

- Improved PDF processing speed
- General improvements in performance due to update from .NET Core 3.1 to .NET 6.
- Improved OCR capabilities
- Improved product logging to enhance product diagnostics
- Fixed issue when the quality of jpeg images dropped after metadata removal
- Fixed issue when DLP fails to process EMF images embedded in a document
- Fixed issue when embedded files can't be opened during recursive PPT processing
- Fixed issue when xls files could not be processed during MD Core local scan

v2.11.1

Release date: 5 /11/ 2022

- A configuration to choose a default fallback encoding
- Optimize system resource usage for PDF processing

- PDF Watermark: Added line breaks to long texts

v2.11

Release date: 3/30/2022

- Support watermark feature for MS Word
- Allow the customers to set certainty for the regular expression
- Support encoding detection (for Japanese, Hebrew and UTF8 encodings)
- Processing all annotation object types in PDFs
- Processing all stamp object types in PDFs
- Keeping the original image quality for output files when removing Metadata

v2.10.1

Release date: 2/16/2022

- Enhance PPT file processing
- Improve metadata processing for image file types
- Fix line break issue with DOC/DOCX when applying watermark
- Fix metadata removal with TIFF file format
- More plain text file support
- Improved image processing (performance)

v2.10

Release date: 12/22/2021

- Add more supported encodings (for Email Security Gateway use case)
- A configuration to set limit file size per workflow
- Scan and remove Metadata recursively
- Support processing cropped images for XLS/XLSX/PPT/PPTX
- Fixed memory leak issue

v2.9.1

Release date 12/1/2021

- Detection Policy is available on MetaDefender Core v5 or newer
- A configuration to allow a file if it is redacted (MetaDefender Core v5 or newer)
- Optimize memory usage

- Better encode handling between Email Security Gateway and Proactive DLP

v2.9

Release date 10/13/2021

- Support watermark EMF/WMF/SVG
- Detect and Redact sensitive info in several objects
  - MS Excel sheet name
  - Defined Name object in MS Excel
  - Image alternative text in MS Office
  - Comment Author in MS Word and Excel
  - Header and Footer in MS Word and Excel
  - Track changes in MS Word and Excel
  - Alternative images in PDF
  - Form fields in PDF
- Improve recursive processing
  - Support RTF as an embedded file
  - More details about hit location
- Upgraded Qt framework to version 5.15.2

v2.8.1

Release date: 8/18/2021

- Improve performance for several file types (MS Excel, text, etc ...)
- Fix MS Excel redaction failure in some cases

v2.8

Release date: 7/8/2021

- A configuration to remove embed objects if recursive processing fails
- Fixed FILE SIZE LIMIT configuration issue
- Fixed embedded image OCR processing
- Improve large plain text file processing

v2.7.1

Release date: 6/3/2021

- Support unlimited depth and number objects in recursive document processing
- Allow users to set a limit number of returned sensitive info
- Added "," to the default delimiter list
- Improve Excel processing
- Fixed Chinese regular expression in text files (CSV, TXT, ...)
- Fixed missing keyword configuration when upgrading to DLP 2.7

v2.7

Release date: 4/27/2021

- Drop supporting MetaDefender Core older than 4.17.1
- Recursive scan and redaction of embedded files in MS Office files
- Localization support for Japanese SSN
- Support watermark, metadata detection, OCR for BMP format
- Support "Delimiter" as an optional validator
- Context detection around hits in PDF files has been improved
- Chart detection and redaction has been introduced in Excel
- Improve OCR detection quality
- Improve redaction function for MS Word files

v2.6.1

Release date: February 8, 2021

- Fix detection issue when an empty cell has a comment (Excel)
- Improve MS Office validation in some regular expression cases

v2.6.0

Release date: January 11, 2021

- Process the hidden areas in a cropped image (DOCX, DOC)
- Support OCR for standalone image file types (JPG, PNG, TIFF)
- Support OCR for embedded images in DOC, DOCX, XLS, XLSX
- Support remove metadata for document files (PDF, DOC, DOCX, XLS, XLSX)
- Support redaction for RTF

v2.5.1

Release date: November 12, 2020

- Improve Japanese string detection/redaction in PDF
- Fix a detection issue when a regular expression contains a Hebrew string
- Fix a crash issue when scanning DOC file on Linux

v2.5

Release date: October 1, 2020

- Metadata removal, Watermark, OCR are available on Linux
- Advanced watermark configurations: font size, text opacity, text position
- New configurations
  - Stop the process if found enough sensitive info
  - Quality configurations for OCR (Normal, Best)
- Support HTML and TXT redaction
- 10x faster when processing text file

v2.4.1

Release date: August 8, 2020

- Improved memory usage
- Improved IPv4 and CIDR search
- Added threaded comment search and redaction in Excel files
- Up to 40% speedup when scanning Excel files

v2.4

Release date: July 7, 2020

- Utilize column and row header to improve certainty level in Excel
- Detect sensitive info in file properties with regular expressions
- Custom keyword list for regular expression
- Support redaction feature on Linux
- Performance improvement: faster processing, less resource usage
- New system requirements on Linux
- End of support Centos 6, Debian 8

v2.3.2

Release date: May 20, 2020

- Better context calculation for Excel and PDF
- Improve IPv4 detection in TXT
- Distinguish between "Failed to detect" and others

v2.3.1

Release date: April 21, 2020

- Threaded comment redaction in Excel files.
- Slightly increased PDF scan performance.
- Improved certainty calculation for MS Office and PDF files.
- Fixed wrong context when a single cell in an Excel file contained the same hit multiple times.

v2.3.0

Release date: April 7, 2020

- Support Optical Character Recognition (OCR) for PDF (Windows only)
- Redact sensitive information for Microsoft Office Excel (XLS/XLSX)
- Better detection method, reduce false positive

v2.2.1

Release date: Feb 12, 2020

- Improve IPv4/CIDR detection performance
- Better handling temp files
- Remove "Parse Binary" option

v2.2

Release date: Jan 6, 2020

- Supports watermark addition for PDF
- Redact sensitive information for Microsoft Office Word (DOC/DOCX)
- Support DLP in Linux with limited functions (work with MetaDefender Core 4.17.1 or newer)
- Redact sensitive information based on certainty level (work with MetaDefender Core 4.17.1 or newer)
- Sample Regular expressions to detect Personally identifiable information (PII): email, address, full name, date of

birth, driver license, phone number, bank account number

v2.1.2

Release date: November 27, 2019

- Better error message when an input PDF file is corrupted

v2.1.1

Release date: October 31, 2019

- Better displaying the words before and after a hit in PDF

v2.1

Release date: September 8, 2019

- Supports IPv4, Classless Inter-Domain Routing (CIDR) detection
- Supports remove metadata for TIFF, GIF file
- Better CCN detection

v2.0.1

Release date: August 15, 2019

- Better watermark and redaction handling when a system is under high load
- Improve CCN detection

v2.0

Release date: June 28, 2019

- Proactive DLP as new name
- Certainty score for sensitive data detection
- Redact sensitive information for text-based PDF file
- Watermark addition for JPEG, TIFF, PNG, GIF
- Supports remove metadata for JPG, PNG file

v1.0.3

Release date: February 18, 2019

- Improve detection for Microsoft Access format

- Improve context for hits
- Improve processing speed (20%)

File Type Module release notes

v8.1.3

Release date: 11/18/2026

- Supported NITF and COMTRADE single-file format detection.
- Added a configurable ZIP signature search range.
- Improved Adobe Illustrator detection accuracy.
- Improved detection performance for AEM, VTAR, and CRX files.
- Fixed arbitrary data being detected as LZMA and blocked as failed to extract.

v8.1.2

Release date: 11/4/2026

- Supported NFO, CXS, and WDP detection.
- Improved DBF detection accuracy with a new native analyzer.
- Improved detection by Machine Learning:
  - Reduced false-positive classification of plain text files as batch files.
- Improved detection performance:
  - Reduced detection time for files that no analyzer recognizes.
  - Returned results earlier for WMF and ICO files.
  - Skipped third-party detection when a native analyzer already has the best possible result.
  - Ran custom detection rules independently, so a single custom rule no longer slows down every file.
  - Limited how much data is read from large unrecognized files, preventing scan failures on very large files.
- Supported the new AI policy license flag that keeps AI-backed features turned off for customers who are not permitted to use AI.
- Fixed an engine crash when a file is removed while it is being detected.

v8.1.1

Release date: 10/14/2026

- Supported SZIP, IDX, and CAT detection.

- Improved detection accuracy for machine learning model files in Parquet, ONNX, and SafeTensors formats.
- Enhanced GGUF detection to cover big-endian files.
- Enhanced LPKX and MPKX detection for large packages.
- Increased detection confidence for WEBP files by validating the RIFF container structure.

v8.1.0

Release date: 9/23/2026

- Supported GDBTABLE, K5M, LPKX, MPKX, and ONNX detection.
- Increased the number of supported file types through machine learning to 190 formats.
- Improved JPEG detection to cover more structural variants.
- Improved GIF detection performance.
- Increased detection confidence for JAR, WAR, and XPI files.
- Improved module performance.
- Fixed a potential data race when looking up file type IDs concurrently.

v8.0.3

Release date:8/26/2026

- Supported detection of ECD, HL7.
- Enhanced PHP detection.
- Improved module performance.

v8.0.2

Release date: 8/13/2026

- Increased the number of supported file types through machine learning to 170 formats.
- Fixed CSV file detected as MTS.

v8.0.1

Release date: 7/22/2026

- Added a type hierarchy field to expose parent-child file type relationships.
- Added the detection method and confidence score to the scan output UI.
- Improved Base64 detection for files containing line breaks.
- Reduced misclassification of plain text files as machine-learning-detected types.
- Fixed a JT detection bypass caused by a crafted false-positive sample.

- Fixed FAT boot sector dumps being misidentified as FAT boot sectors.

v8.0.0

Release date: 7/2/2026

- Supported detection of SBN, SBX, JT, Microsoft Delta Compression, and DPA file formats.
- Supported regular expressions in the "Accepted extensions" setting for more granular file extension rules.
- Improved detection by Machine Learning with expanded support for 151 file type classes.
- Improved detection accuracy for OLE, XML, ZIP, TXT, and JSON sub-types.
- Improved detection of large SafeTensors files.
- Improved TAZ, disk image files containing MBR/GPT partitions detection.
- Fixed CSS files being incorrectly detected as CSV when the ML module is disabled.
- Fixed crashes caused by out-of-bounds memory access in the XML and OLE detectors.
- Upgrade Windows Compiler from VS2019 to VS2022.

v7.8.3

Release date: 5/20/2026

- Supported OTF, LST, Mach-O, and IDML detection.
- Improved GPT detection to distinguish GPT files from GPT metadata.
- Improved DLL archive detection performance.
- Improved detection accuracy so CSV and YAML files are correctly identified instead of being returned as generic TXT.
- Reduced false positives:
  - Refined MBX, EML and ERB detection.
  - Refined PS1 detection to avoid misclassifying when Machine Learning is available.

v7.8.2

Release date: 5/7/2026

- Fixed a crash that occurred when detecting data files larger than 7GB.

v7.8.1

Release date: 4/15/2026

- New file type support: Web Archive (WARC), SQUASHFS, LYRX, WIM, BLZ.

- Reduced memory usage and improved performance.
- Adjusted confidence scoring for TXT files detected by third-party engines to reduce false positives.
- Improve LM model detection for TXT and JSON.

v7.8.0

Release date: 3/25/2026

- Added support for CGR, CATPART, INDB, TXF, ICA, and IFC file formats.
- Improved detection accuracy for PostScript and HTML files.
- Enhanced detection speed for JSON and password-protected PDF files.
- Detected embedded files in Microsoft Office 2007 as OLE format.
- Fixed stability issues to improve overall reliability.
- Allowed users to define which file formats are subject to file type mismatches.
- Upgraded OpenSSL to 3.6.1.
- Added support for Debian 13.

v7.7.2

Release date: 2/5/2026

- Supported LZOP, YENC, PEM CMS detection.
- Improved PKG detection.
- Improved custom rule detection performance.
- Improve detection by Machine Learning:
  - Added support for 100+ text-based format.
  - Reduced overfitting to weak or misleading patterns for more reliable results.

v7.7.1

Release date: 1/14/2026

- Supported new detections for Z, X3P, MAR, LZ4.
- Better DTD, CSV, AVIF detection.
- Added CIS Level 2 support for Windows Server 2022.

v7.7

Release date: 12/16/2025

- Support detecting specific file types only for use cases that do not rely on file type identification.
- Add new detections for High Efficiency Image File Format (HEIF), Safari Web Archive (WEBARCHIVE), Parquet (PARQUET), Keyhole Markup Language (KMZ) and NIST Biometric (NIST) files.
- Enhance ISO file detection.
- Support a minimum threshold length for a file to be detected as Base64.
- Support additional attributes in engine-metadata.
- Improved detection support for large TAR archive, HEIC, Pytorch Archive files.

v7.6.2

Release date: 11/12/2025

- Support new detections for QEMU Copy-On-Write (QCOW1/QCOW2/QCOW3), AV1 Image file format (AVIF), eDrawings Assembly (EASM), COM.
- Include the "sourceName" attribute in the like_type_ids.
- Improve detection accuracy for: PICKLE, SWF, ZIP, ISO.
- Improve the Machine Learning detection.

v7.6.1

Release date: 10/14/2025

- Block files based on their properties.
- Support the RDP file format.
- Improve the detection performance of XML-based files.

v7.6.0

Release date: 9/24/2025

- Update ML model to OPSWAT LM v3 to have a better detection, please check this page to see a full list of supported file types by ML.
- Support new file types: WMZ, L01, EVTX, WOFF2, DES, OLE-based FLA, VBA, MATLAB, REG, JUPYTERNB, LUA, PERL, JAVA, VBS, GO, R, COFFEESCRIPT, CSHARP, RUST, MARKDOWN, RUBY, CSPROJ, TYPESCRIPT, TEX, MAKEFILE, SQL.
- Enhance TAR , WMF, MP3, RIBC, ZIP

v7.5.3

Release date: 8/20/2025

- Support for new file types: UFDR, AppImage, HDF5 (v0, v1), and encrypted Hancom Office documents.

- Improve OVA detection for large files.
- Enhance MP3, CAB, ASN1, JSON detection.
- Add support for Windows Server 2025 and Ubuntu 24.04.

v7.5.2

Release date: 7/24/2025

- Improve detection for: ICO, ASN1 DER, AC3, XML, ZIP-based.
- Enhance memory and string handling to resolve an engine crash issue.
- Optimize engine load time with the Machine Learning module.

v7.5.1

Release date: 7/15/2025

- Support new file types: VMware OVA (OVA), PyTorch (PT), HDF5 version 2 (HDF5), and ONNX (ONNX).
- Properly handle the import of engine configurations.

v7.5.0

Release date: 6/25/2025

- Support new file types: Python Zip (PYZ), Atom Shell Archive (ASAR), LZip (LZ), MS Access (MDB), Microsoft Project File (MPP), Ensoniq's Paris Audio File (PAF), Core Audio Format (CAF), COMSOL Multiphysics Binary (MPHBIN), RIBC Data Table (utibwktmp, mthbwktmp), Wavelet Scalar Quantization FBI fingerprint format (WSQ), Advanced Video Coding (AVC, H264), High Efficiency Video Coding (HEVC), Media Presentation Description (MPD), Timed Text Markup Language (TTML), OpenDocument Drawing (ODG), Flash (FLA).
- Improve detection for NSISBI, MP4, WAV, HTML, APK, ASN1 PEM.
- Remove Beta label for "Extract Metadata" feature.
- Drop support CentOS 7/RH 7/Debian 10/Ubuntu 20.

v7.4.3

Release date: 5/28/2025

- Support new file types: Inno Installer.
- Improve detection for HTML with ISO-2022-JP encoding.
- Support Amazon Linux 2023.

v7.4.2

Release date: 5/7/2025

- Utilize "score" in the custom rule to overwrite some native detections.
- Enhance AI detection feature.

v7.4.1

Release date: 4/23/2025

- Support new file types: Pickle (PKL), BRD, URL, MDZIP, PyTorch (Zip-based), new APK version.
- Return "detected_by" value to identify how a file type is detected.
- Enhance detection rule feature.
- Improve engine initialization process.
- Fix GPG detection for big file size.

v7.4

Release date: 3/27/2025

- Support file type detection with custom rules.
- Support new file types: PRTCAD, SafeTensor (STS), variant of JPG2000 (J2K, MJ2, JPM)
- Add ZERO filetype id for zero byte file.
- Improve Git Pack, OVA, P7M, EOT detection.
- Upgrade the AI model to have better detection.

v7.3.1

Release date: 2/19/2025

- Support new file types: OVA, DAV, DXF, R12
- Support detect encryption for encrypted-entry RAR.
- Enhance Yml/YAML detection.
- Support engines-metadata for remaining settings.
- Enhance Json detection with "Classify with Machine Learning".

v7.3

Release date: 12/18/2024

- Support new file type detections: YZ1, DrawIO, SCDOC.
- Support Rocky Linux 9.4, Debian 12.
- Migrate configurations from global to workflows.
- Improve detection for text files that have control characters.

- Improve detection for text with Shift-JIS file encoding.
- Improve WMZ, ZLIB detection.

v7.2.2

Release date: 11/13/2024

- Support new file type detections: NAR, Git Pack, PCAPNG, PBIVIZ, 3DML.
- Enhance TGA detection for big files.
- Improve JTDC, EML detection.
- Mismatch detection: Support asterisk (*) in the "Accepted extensions" field.
- Digitally sign all binary files.

v7.2.1

Release date: 10/8/2024

- Support the detection of NVR file.
- Improve PCAP, WMZ, EML, JS, and CSS file detection.

v7.2

Release date: 9/18/2024

- Allow users to configure accepted extensions to avoid mismatches.
- Support the detection of base64 text files.
- Support new file types: SQLite, B3DM, EXT, GPKG, PCAP, Hikvision DVR (HIK), ACM, TGA 1.0, JNLP, ZUGFeRD XML, and MBX.
- Improve LOC, ICS, DWF, and PDF file detection.
- Improve PERL, YML, BAT, ASPX file detection with AI.

v7.1.1

Release date: 7/22/2024

- Enhance detection for TGA, VMDK, VMX, and VMXF files.

v7.1

Release date: 6/25/2024

- A new configuration to utilize Google Magika to improve text-based detection.
- Support new file types: Bean Markup Language (BML), Power BI (PBIX, PBIT), Broadcast Wave 64 (BW64), and

CompoundFile (OLE).

- Improve DM and Git-bundle processing.
- Support detecting MS Office documents larger than 5GB.
- Support Ubuntu 22.04.

v7.0.2

Release date: 5/22/2024

- Fix detection for the ZED and DSK file formats.

v7.0.1

Release date: 4/22/2024

- Improve detection for JSON and LOC file formats.

v7.0

Release date: 4/3/2024

- Support new file types: Encrypted ZED, PowerPoint add-in (PPA), Forms Data Format (FDF), New Technology File System (NTFS), Multiple Image Network Graphics (MNG), GPS location data (LOC), TrueType Font (TTF), GGUF, Design Web Format XPS (DWFX), Adobe Illustrator Template (AIT), Research Institute on Building Cost (RIBC), and Web Open Font Format (WOFF).
- Improve file type detection for tar POXIS, MTS, EXE, DLL, and OpenPGP files.
- Upgrade Exif tool to a newer version.

v6.5.2

Release date: 2/5/2024

- Support new file types: PYC and VIB with multiple payloads.

v6.5.1

Release date: 1/16/2024

- Enhance detection for TXT file mismatches that occur with PRN and CSS file.

v6.5.0

Release date: 12/18/2023

- Support new file types: Digital Negative (DNG), Report Definition Language (RDL), Microsec E-Szigno (ESZ), Nuget

(NUPKG), 3D CAD (STP), Parasolid Model Part Text Format (PXT), Parasolid Model Part Binary Format (PXB), Initial Graphics Exchange (IGS), VSIX package (VSIX), Microsoft Excel Workspace (XLW), Virtual Hard Disk v2 (VHDX), JasperReports Format (JRXML), MS Excel Add-in (XLAM)

- Improve TAR, JHDC detection
- Display MISP metadata in better format
- Better handling big PDF file

v6.4.2

Release date: 12/5/2023

- Fix TAR file detecion timeout

v6.4.1

Release date: 11/9/2023

- Support VOX file type detection

v6.4

Release date: 9/27/2023

- Support new file types: Git Bundle (BUNDLE), Encapsulated PostScript(.EPS, .EPSF, .EPSI), ActiveMime(.MSO), VTXT, Storylist document (STORY), DocuWork Container (XCT), Society Of Exploration Geophysicists (SGY, SEGY), Printer (PRN)
- Return EXIF metadata for more file formats
- Return custom metadata from MISP files
- Enhance IPA detection rule

v6.3.2

Release date: 8/9/2023

- Fixed engine crash on Linux when analyzing some CERT files

v6.3.1

Release date: 7/19/2023

- Support DocuWork container detection

v6.3

Release date: 6/21/2023

- Detect new file format: Material Exchange Format (MXF), Advanced Audio Coding (AAC), Shapefile (SHP and SHX), MPEG-2 Transport Stream (M2TS), DocuWork Binder (XBD), Vnas CAD Format BFO (BFO), Split GZ (GZ), Apache NIFI (NIFI)
- Better clean up temporary files when analyzing DMG files
- Distinguish between WAR and JAR
- Enhance Exiftool process monitoring
- Improve JSON detection
- Optimize detection flow for media file types
- Detect encrypted archive faster

v6.2.3

Release date: 5/22/2023

- Improve performance when scanning text files in some cases

v6.2.2

Release date: 4/25/2023

- Detect encrypted HWP file
- Enhance DAT detecttion
- Improve performance for Winrar SFX detection

v6.2.1

Release date: 4/3/2023

- Enhance the filetype engine to handle big files

v6.2

Release date: 3/15/2023

- Detect new file format: AppleSingle/AppleDouble, Arcserve ShadhowProtect (SPF), Google Chrome Plugin (CRX)
- Improve logging

v6.1.1

Release date: 2/13/2023

- Detect new file format: Firebird database files (FDB), Tableau Data Extract (TDE), VTAR

- Better way to manage EXIF process
- Enhance CSV, JSON, ICS, VCS detection
- Fix file type output contains none UTF-8 characters

v6.1.0

Release date: 1/11/2023

- Support to return Exif info for images, MS Office 2003, MS Office 2007, PDF (Beta, require MetaDefender Core 5.3.0 or later)
- Detect ATC, EFI, JHD, JHDC, AIFF, BWF, RF64, W64, INF, PPK files
- Optimize performance for big XML files
- Enhance detection of MS Office 2007, JSON files

v6.0.2

Release date: 12/2/2022

- Detect .p12 and .pfx as cert file extension
- Fix text detection slowness when files have many spaces
- Enhance dection of encrypted PowerPoint 2007, Excel 2007, OPUS, ATR
- Enhance detection rule for files starting with Draw string

v6.0.1

Release date: 11/7/2022

- Improve OpenPGP, LZMA detection
- Support UTIB detection
- Fix crash issue on invalid certifcate files

v6.0.0

Release date: 10/5/2022

- VC++ Redist 2017 is required
- Support new file type detection: MPEG Transport Stream (MTS)
- A global configuration to allow including file info details in the result.
- Dependency check when intializing the engine

v5.3.21

Release date: 8/22/2022

- Support engines-metadata header (require MetaDefender Core 5.2.1 or newer)
- Enhance VDX detection
- Improve performance for detecting big Text, AI
- Improve logging

v5.3.20

Release date: 8/2/2022

- Enhance detection for EML, ICS/VCS, CPIO and MS Office 2007

v5.3.19

Release date: 7/11/2022

- Better classification for MPEG-4 Part 14 file types: .mp4, .m4a, .m4p, .m4b, .m4r and .m4v
- Enhance JSP, XLSB detection
- Support Executable and Linkable Format (ELF) file detection

v5.3.18

Release date: 5/30/2022

- Enhance Text file detection
- Classify EICAR file as a text file
- Enhance CERT and CRL detection

v5.3.17

Release date: 4/27/2022

- Support new file type detection: JW CAD (JWC)
- Enhance PDF detection to detect restricted PDF
- Support logging

v5.3.16

Release date: 3/23/2022

- Improve INI, JSON, DLL, DBF, Excel 2 worksheet, PPAM
- Support new file type detection: FlashPix (FPX), Revenant Game Archive (RVM), Microsoft OneNote (ONE)

v5.3.15

Release date: 2/10/2022

- Support new file type detections: Audio Codec 3 (AC3), OGG, Animated Portable Network Graphics (APNG), 3D manufacturing format **(**3MF), E57, Downloadable Sound (DLS), QualComm PureVoice (QCP), Revit (RVT), DJVU, High Efficiency Image (HEIC),
- Detect encrypted PDF files protected by Microsoft Information Protection (MISP)
- Support TSD, ALZ, EGG encrypted file detection
- Improve INI detection

v5.3.14

Release date: 12/20/2021

- Fix PGP false positive detection
- Support new file type detections: e-book file format (EPUB), Targa raster graphic (TGA), ZSTD

v5.3.13

Release date: 12/10/2021

- Support new file type detections: Silverlight app (XAP), Email template (OFT), Autodesk Filmbox (FBX), Universal 3D (U3D), CAD software (STL)

v5.3.12

Release date: 11/22/2021

- Support new file type detections: Bourne Again Shell (Bash), EGG, Security Certificate (CER), Windows Media Player Skin Package (WMZ), Certificate Revocation List (CRL), Big TIFF, Encrypted OpenOffice XML documents, Argus (AVUX), PCX, PGP, Nikon Raw Image (NEF), Acronis True Image (TIB)
- Adjust MIME-Type for CSV, TSV, EMZ

v5.3.11

Release date 9/29/2021

- Support new file type detections: Restricted-permission message (RPMSG),
- Improve detection for ARC, encrypted MS Office 2007, PS1

v5.3.10

Release date: 8/10/2021

- Support new file type detections: Windows Compressed Enhanced Metafile file (EMZ), iOS application archive (IPA), Matroska Multimedia Container (MKV), WebM
- Enhance PRT, MSG detection

v5.3.9

Release date: 7/1/2021

- Support OST file type detection

v5.3.8

Release date: 6/3/2021

- Support new file type detections: X12 EDI, Tableau (TDSX, TBM, TDS, TWB, TWBX)

v5.3.7

Release date: 4/27/2021

- Improve TAR, DLL, AIP (Azure Information Protection), DBF (ArcGIS) file type detection

v5.3.6

Release date: 4/1/2021

- Support detection for P7M

v5.3.5

Release date: 3/1/2021

- Improve detection for HAR, DSN, CELL, encrypted PDF

v5.3.4

Release date: 1/27/2021

- Support P21 file type detection
- Improve detection for JavaScript file format
- Fix a crash issue when reading an invalid compound file

v5.3.3

Release date: December 29, 2020

- Improve detection for Cimplicity Binary Format (CIM), F4V, SXF Feature Comment (SFC), Jw_cad (JWW), DocuWorks (XDW)

v5.3.2

Release date: December 1, 2020

- Better detection for TSV (Tab-separated values), DSD (DART DSD) , XLS (Excel binary) file format

v5.3.1

Release date: October 26, 2020

- Support detection for Associated Signature Containers - Extended (ASiCE) and Binary Document (BDOC) file format

v5.3.0

Release date: October 1, 2020

- Reduce 50% resource usage while processing speed is 2x faster
- Better password protection detection for OpenOffice files

v5.2.26

Release date: September 1, 2020

- Better detection for Microsoft CAB and Installshield CAB file format

v5.2.25

Release date: August 10, 2020

- Improve JPEG, ACE detection
- Improve mismatched detection logic for text file format

v5.2.24

Release date: July 6, 2020

- Support new file type detections: DCM, WEBP, LNK, WSP, JP2, ODS, OTS

v5.2.23

Release date: May 20, 2020

- Support new file type detections: OpenSSL encrypted with a salted password, WDP, ALZ.
- Improve DWT and DWS detection, Encrypted Microsoft Office file detection

File Structure Validation release notes

v2.1.3

Release date: 11/18/2026

- Supported SZIP file structure validation, including validation of its sub-components.
- Supported schema-based JSON validation.
- Detected unlisted entries in ZIP files:
  - Added an option to extract unlisted entries so they can be scanned by other engines.
- Reported additional object types in Office files:
  - ActiveX controls and VSTO macro objects in Office Open XML and legacy Office files.
  - DDE objects in legacy Office files.
- Extracted analyzed objects from RTF files, including images, hyperlinks, DDE, embedded objects, embedded fonts, and metadata.
- Supported detection of stacked polyglot files for the DICOM format.

v2.1.2

Release date: 11/4/2026

- Reported metadata and embedded fonts found in PDF and Microsoft Office 2003 documents.
- Reported DDE objects found in Microsoft Office documents.
- Delivered analysis results sooner for files that produce objects gradually, instead of waiting for a full batch.
- Supported validation of big-endian GGUF files.
- Applied the Selected Object Types setting earlier for Microsoft Office documents, so object types that are not selected are no longer extracted.
- Fixed extracted files being blocked because of an extension mismatch.

v2.1.1

Release date: 10/14/2026

- Reduced Stack Polyglot false positives
- Returned orphaned objects found by the PDF analyzer.

- Improved the XML schema validation output when validation fails.
- Reported more types of embedded content as soon as it is found.
- Fixed polyglot detection failures, slow processing when many files were scanned at the same time.
- Fixed the Schemas Validation setting having no effect when disabled.

v2.1.0

Release date: 9/23/2026

- Supported structure validation for HDF5, HL7, GDBTABLE, LPKX, MPKX, and Pickle AI model files.
- Supported DICOM encapsulated object extraction, including handling of preamble data.
- Supported returning embedded objects and external image link from Microsoft Office 2003 files.
- Supported streaming of returned objects to improve performance and less disk I/O.
- Added standard validation references to the validation result.
- Added schema upgrading logic for the selected object types setting.

v2.0.0

Release date: 8/27/2026

- Validates file structure across 160+ supported formats.
- Extracted embedded objects from the file and applied the full processing pipeline to each.
- Blocked files based on object-level policies.
- Supported polyglot file detection and blocking.
- Supported XML Schemas validation.

Archive Module release notes

v8.1.3

Release date: 11/18/2026

- Supported sanitization details for compressed containers:
  - Reported what was sanitized inside each entry of an archive.
  - Added support for EML and MSG attachments.
- Upgraded the bundled .NET runtime to address known vulnerabilities.

v8.1.2

Release date: 11/4/2026

- Supported Microsoft Purview Information Protection decryption for PDF files.
- Upgraded the bundled System.Security.Cryptography.Pkcs library to address CVE-2026-84008.
- Fixed PST compression failure when the file contains items with matching name prefixes.

v8.1.1

Release date: 10/14/2026

- Preserved the original file type when checking for file type mismatches during extraction.
- Fixed a file type mismatch reported for files extracted from Inno Setup installers.
- Stopped reporting a corrupted archive error for files extracted from ZLIB streams.
- Fixed L01 extraction failing with "failed to process next entry".

v8.1.0

Release date: 9/23/2026

- Added support for K5M extraction.
- Added support for MSIP decryption.
- Added the index of each extracted entry for formats.
- Improved memory usage by tracking ongoing requests and applying more aggressive garbage collection.
- Upgraded the 7-Zip library to 26.02 to resolve reported CVEs.
- Fixed an engine crash when processing EML files.

v8.0.3

Release date: 8/13/2026

- Added support for AFF4 extraction.

v8.0.2

Release date: 8/5/2026

- Patched the 7z library to address CVE-2026-14266.

v8.0.1

Release date: 7/22/2026

- Added support for extracting split EnCase (E01) images via multipart upload.

- Enhanced CLI extraction feature to handle XFS and LVM extraction correctly.
- Improved extraction performance for large X12 EDI files by more than 2×.
- Upgraded several third party libs to resolve CVEs (CVE-2026-41511, CVE-2026-45785).
- Fixed corruption of non-ASCII attachment filenames when extracting from MSG files.
- Fixed a compression engine hang when creating RAR archives with many entries or long filenames.
- Fixed a race condition that could return inconsistent archive information.

v8.0.0

Release date: 7/2/2026

- Supported extraction binary fields in Parquet files.
- Supported extraction of SquashFS files with appended RAUC data.
- Improved TAR extraction to accept files with base-256 header fields.
- Improved handling of DMG files with missing trailing metadata.
- Improved handling of invalid stream-optimized VMDK files.
- Fixed memory leaks during archive task and ASAR file processing.
- Fixed missing file path when extract large archive file.
- Upgrade Windows Compiler from VS2019 to VS2022.

v7.8.2

Release date: 5/20/2026

- Supported X3P extraction and compression.
- Supported GPKG and TAZ compression.
- Supported VMDK files containing ISO data.
- Improved extraction for virtual disks with an empty main subfile by honoring the "Handle unsupported file systems" setting.
- Suppressed split archive errors for ZIP, CAB, CHM, and EXE_SFX files extracted from ZLIB streams.
- Fixed PST compression failure caused by duplicated input paths when "Reuse processing result" is enabled.

v7.8.1

Release date: 4/15/2026

- New format support: SquashFS, Web Archive (WARC), BLZ.
- Enhanced EXE_SFX extractor to handle failure in case of extract-on-error is enabled.
- Added support for extracting data appended after the end of payload in XZ archives.

- Loosened header validation rules for CPIO extraction to improve compatibility.
- Loosened Linux data corruption rules in virtual disk extraction.
- Fixed EML extractor to use correct line endings (CRLF) for generated HTML content.
- Implemented extract-on-error support for PST extraction.
- Improved X12 EDI extraction performance.

v7.8.0

Release date: 3/25/2026

- Added support for 3MF, Delta ESD, and BLZ file formats.
- Enhanced PST sanitization to better handle corrupted files.
- Optimized X12EDI processing by skipping unnecessary DATA file extraction.
- Preserved entry names during recursive compression.
- Resolved extraction issues by correctly distinguishing NSISBI from NSIS formats.
- Added “Extract on Error” support for 3DML extraction.
- Improved handling of GPKG files with empty or missing tables.
- Upgraded OpenSSL to version 3.6.1.
- Added support for Debian 13.

v7.7.2

Release date: 2/5/2026

- Added support for YENC decoding.
- Improved handling of long file paths on Windows when extracting Inno Setup installers.
- Upgraded 7-Zip library to version 25.01.
- Better way to handle big OST extraction.

v7.7.1

Release date: 1/14/2026

- Added support for UFDR, LZ4, Mozilla Archive (MAR).
- Added CIS Level 2 support for Windows Server 2022.

v7.7

Release date: 12/16/2025

- Added support for combining split extracted files (requires MetaDefender Core 5.17.0 or later) for 7z, ZIP, RAR,

TAR, WinZip, XZ, Clonezilla and GZ formats.

- Added the ability to enable or disable CLI extraction within workflows.
- Improved TAR extraction reliability, preventing failures when end-of-archive blocks are missing.
- Enhanced VMDK extraction to properly extract disk partitions.
- Fixed cases where EXE SFX extraction could fail.
- Improved 3DML, MSG compression.
- Reduced Linux package size by up to 35%.

v7.6.2

Release date: 11/12/2025

- Added support for QCOW extraction.
- Refined the logic for handling "**Other extraction errors**" to exclude "**Unsupported format**" errors in some cases.
- Improved the Zlib extraction failure message.

v7.6.1

Release date: 10/14/2025

- Added support for MBR and EXT file formats.
- Introduced a configuration option to control the UTF-8 name field in ZIP file format.
- Added support for extracting encrypted ALZ/EGG archives using a provided password.
- External command-line extraction improvements:
  - Allow users to define an “encrypted archive” pattern in the json configuration file.
  - A configuration to block files if command-line extraction fails.
- Migrated the Skip File Type setting from Global to Workflow.
- Enhanced extraction for AppImage and Inno Setup files.

v7.6.0

Release date: 9/24/2025

- Support extracting data after the end of payload for zlib.
- Enhance error message structure in case of extraction failure.
- Support extracting on GUID Partition Table (GPT).
- A configuration option to handle unsupported file system for VMDK.
- Handle legacy time format in MBX to avoid extraction loops.

v7.5.2

Release date: 8/20/2025

- Added support for new file types: AppImage, LZIP.
- Support extracting disk images without partitions in VMDK.
- Support extracting data after the end of the payload in Gzip and Tar.
- New configuration option to handle extraction failures caused by "data after the end of payload."
- Improved logic for detecting invalid file structures.
- Improved logic for detecting timeout.
- Enhance B3DM extraction.
- Added support for Windows Server 2025 and Ubuntu 24.04.

v7.5.1

Release date: 7/23/2025

- Support new file types: ASAR.
- Enhance zip-based extraction to respect the encoding setting.
- Upgrade WinRAR module to 7.11.
- Support running an external command line to extract files.

v7.5

Release date: 6/25/2025

- Support new file types: PYZ.
- Improve Inno installer extraction:
  - Support versions newer than 6.4.0.
  - Fix failure extraction in some cases.
- Upgrade .NET to version 8.
- Drop support CentOS 7/RH 7/Debian 10/Ubuntu 20.

v7.4.2

Release date: 5/27/2025

- Support new filetype extractions: VHDX, Inno Installer.
- Support Amazon Linux 2023.

v7.4.1

Release date: 4/23/2025

- Classify the archive failure categories be more meaningful:
  - Separate "Others" with "Other extraction errors".
  - Rename "General unextractable errors" to "Not extractable".
- Support MDZIP extraction.
- Enhance PEM extraction.

v7.4

Release date: 3/27/2025

- Support configurations in workflow (requires MD Core 5.14.0)
- Support extraction of new file types:
  - PCAP with http, ftp, imap protocol.
  - Multipart SMIME
  - SCDOC
- Stabilize PST compression.
- Improve EML compression:
  - Preserve entry names, attachment properties for TNEF attachments.
  - Handle extraction of empty files.
- Add a new configuration allowing content-type mismatching with EML email body.
- Improve compression of archive files containing many child files.
- Enhance MSG extraction.
- Fix inconsistent behaviors for handling password protected files.

v7.3.2

Release date: 2/19/2025

- Support YZ1 extraction.
- Support extracting object attachments in TNEF extraction.
- Support re-encoding Base64 files in compression.
- Handle EML files with special characters in attachment filenames.
- Fix incomplete MSG extraction and prevent adding random bytes to files.
- Enhance extraction of P7M file with PEM format.

v7.3.1

Release date: 1/15/2025

- Handle "object attachment" in TNEF file.
- Resource optimization in case of extracting large ZLIB file.
- Improve MSG compression method.
- Improve MBX extraction.
- Enhance "extract on error" with encrypted archive files.

v7.3

Release date: 12/18/2024

- Support extracting concatenated ZIP, ZILB.
- Support ISO-8859-8 charset attachment file name in EML.
- Support Rocky Linux 9.4, Debian 12.
- Enhance extracting password-protected files.
- Improve for PST, MSG:
  - Add tombstone files for blocked files.
  - Improve compression feature.

v7.2.2

Release date: 11/21/2024

- Support new file types: 3DML, NAR, Git Pack, NUPKG, SXC, SXD, SXF, SXI, SXM, SXW, STC, STD, STI, STW, HPWX, PBIT, PBIX, SHOW, SLDM, SLDX, VVSX, VSIX, PPAM.
- Support UTF-8 password for ZIP.
- Return more details about extraction errors.
- Enhance PST sanitization.
- Digitally sign all binary files.

v7.2.1

Release date: 10/8/2024

- Support decoding base64 file.
- Handle encoding in CAB extraction and compression.
- Enhance CPIO extraction.

v7.2

Release date: 9/18/2024

- Support new file types: TNEF, MBX, MHT, GPKG, B3DM, DWF and DWFX.
- Support the APFS file system in DMG extraction.
- Enhance canceling timed out extraction and cleaning up temp files in high load situation
- Improve VMDK extraction by using less disk space and increasing speed.

v7.1.2

Release date: 8/19/2024

- Fix high memory consumption when extracting MS Office files.

v7.1.1

Release date: 7/24/2024

- Support VMDK and ZSTD extraction.
- Enhance cancelation of extraction in high-load situations.
- Fix EXE_SFX hang issue.

v7.1

Release date: 6/25/2024

- Support new file types: RAW with HFS format and MS OLE.
- Support GPG asymmetric keypair extraction (users must provide the GPG Home directory with all keys imported).
- Support skipping extraction for selected file types.
- Add a new configuration to allow extract on error.
- Support Ubuntu 22.04.

v7.0.1

Release date: 4/10/2024

- Fix SMINE extraction failures in some cases.

v7.0

Release date: 4/3/2024

- Support RAW (with NTFS Format), git-bundle, RIBC, and SMINE extraction.

- Enhance UTIB, TAR, and split Encase extraction.
- Upgrade 7z to a newer version.
- Upgrade the engine to .NET 6.

v6.5.3

Release date: 3/6/2024

- Enhance EXE SFX file extraction.
- Auto-handling entry name encoding (supported UTF-8 and SJIS).
- Upgrade third-party libraries to resolve a vulnerability.

v6.5.2

Release date: 2/5/2024

- Fix the temp file issue when extracting PST files.
- Fix failures when extracting .DOSSZIE file.

v6.5.1

Release date: 1/16/2024

- Enhance VHD extraction.

v6.5

Release date: 12/18/2023

- Support new file types: e-dossier (ES3), System Center Configuration Manager Package Files (PCK)
- Better handling big archive files.
- Password protected PST file is handled properly.
- Enhance temp file generation.

v6.4.1

Release date: 11/9/2023

- Better handling EML file encoding in base64

v6.4

Release date: 9/27/2023

- New configuration to preserve children filenames inside an archive file when creating a sanitized file
- Support UUencoding (UU, UUE), Encase (E01) extraction

v6.3.2

Release date: 8/9/2023

- Fixed the compression issue when creating a RAR file with Chinese file path

v6.3.1

Release date: 7/19/2023

- Support extraction for non-standard VIB files

v6.3

Release date: 6/20/2023

- Support CRX extraction
- Return an attribute to identify whether a file is encrypted or not after decrypted (requires MetaDenfender Core 5.5.1+)
- Improve logging
- Enhance PST extraction and compression
- Support hiding internal data for Archive sanitization

v6.2.3

Release date: 5/22/2023

- Remove an unnecessary file (MSG.data) when extracting MSG files

v6.2.2

Release date: 4/25/2023

- Improve SFX extraction performance
- Support EXE_SFX decryption

v6.2.1

Release date: 3/29/2023

- Fix performance degradation in MetaDefender Core 5.4.1+

- Remove the UNC path in the PidLidReminderFileParameter when sanitizing a MSG file (CVE-2023-23397)

v6.2

Release date: 3/15/2023

- Support extraction for new file types:
  - XPI (Firefox extension)
  - VTAR (VM Tar)
  - GPT-styled VHD
  - Split CAB
- Preserve filename/folder name encoded with special encodings like Shift-JIS
- Improve HFS extraction

v6.1.1

Release date: 2/13/2023

- Support extraction for SPF
- Fix engine failed to decrypt/encrypt MS Office, PST dues to long socket path

v6.1.0

Release date:1/11/2023

- Support extraction for UTIB
- Enhance extraction and compression of EML
- Enhance extraction of Self-Extracting Executable files
- Preserve symlinks in TAR compression

v6.0.2

Release date: 12/2/2022

- Preserve MS Word/Excel format during decryption
- Support retaining password for DOT, DOCM, DOTX, DOTM, XLT, XLSM, XLTX, XLTM, XLSB

v6.0.1

Release date: 10/27/2022

- Enhance EML extraction

- Support UTF-8 password when decryption PDF files
- Support log level configuration
- Support APK with Signature Scheme v2

v6.0

Release date: 10/5/2022

- VC++ Redist 2017 is required
- Dependency check when intializing the engine

v5.3.28

Release date: 9/7/2022

- Add line breaks when writing base64 encoded data in EML

v5.3.27

Release date: 8/22/2022

- Enhance password-protected decryption for Excel file

v5.3.26

Release date: 8/1/2022

- Enhance PST, HFS extraction

v5.3.25

Release date: 6/29/2022

- Handle password with special characters in PGP decryption

v5.3.24

Release date: 5/30/2022

- Support non-ASCII password in PDF decryption
- Support archive extraction with Copy method
- Improve archive extraction on Linux

v5.3.23

Release date: 4/27/2022

- Fix a bug that causes a failure when extracting a big TAR file
- Support non-ASCII password in PDF decryption

v5.3.22

Release date: 3/28/2022

- Support extracting password protected workbook/worksheet in Excel
- Support extracting password protected POT, PPS, XLT

v5.3.21

Release date: 2/10/2022

- Support ALZ, EGG encrypted extraction
- Enhance service behavior when missing .NET Core
- Support re-encrypt MS Office files on Linux

v5.3.20

Release date: 1/5/2022

- Better timeout handling when recombine EML, MSG, PST
- Fixed temp files left over after scanning
- Enhance Power Point password decryption
- Support Email Template (OFT) extraction

v5.3.19

Release date: 11/22/2021

- Support EGG extraction
- Improve VHD extraction
- Support to extract PGP/GPG file with password (need to have preinstalled GnuPG)

v5.3.18

Release date: 9/29/2021

- Support MS Office 2003 decryption
- Support JMOD, MSIX, APPX file extraction

- Improve temp file handling

v5.3.17

Release date: 8/17/2021

- Support IPA, OST file format extraction
- Fixed crashed issues when processing PST file.
- Fixed EML issues: broken attachments after extracting, attachment file names are changed
- Fixed MSG sanitization failure when attachment filenames contain special symbols

v5.3.16

Release date: 7/8/2021

- Support X12EDI, TBWX file format extraction

v5.3.15

Release date: 6/3/2021

- Improve the logic counting files before extracting
- Improve cleanup mechanism

v5.3.14

Release date 4/27/2021

- Better handling in case of exceeded archive file size
  - A better error message
  - Crash issue

v5.3.13

Release date: 4/8/2021

- Fixed high CPU usage on Ubuntu

v5.3.12

Release date: 4/1/2021

- Support password extraction for PST and MS Office files on Linux (require .NET Core to be installed)
- Retain password protection for RAR

- Improve DMG extraction on Linux

v5.3.11

Release date: 3/3/2021

- Update the module to use VS C++ 2017
- Improve password-protected PDF, .DMG extraction

v5.3.10

Release date: 1/27/2021

- Support PST extraction (on Windows only)
- Return an error message when decrypting a PDF file with a wrong password

v5.3.9

Release date: 12/29/2020

- Enhance NSIS extraction

v5.3.8

Release date: 12/1/2020

- Fixed a hang issue when canceling an extraction request

v5.3.7

Release date: 10/26/2020

- Better MSG extraction and compression

v5.3.6

Release date: 10/1/2020

- Support CHM extraction
- Support detect partial extraction for ALZ, ACE

v5.3.5

Release date: 9/1/2020

- Improve ACE, CAB extraction

v5.3.4

Release date: 8/8/2020

- Support ACE extraction

v5.3.3

Release date: 7/6/2020

- Better split encrypted archive handling
- Improve 7z self-extracting archive processing

v5.3.2

Release date: 5/20/2020

- Support ALZip (ALZ) extraction
- Better EML, MSG extraction

SBOM release notes

v5.0.1

Release date: 10/28/2026

- Detect the TLS versions enabled by AWS Elastic Load Balancing security policies in Terraform and AWS CloudFormation files, including the hybrid post-quantum key exchange of PQ policies, and report them in the CBOM.
- Add quantum-readiness rating for TLS protocols and cipher suites.
- Allow blocking AI models by model type and base model in AIBOM.
- Keep the declared dependencies of CycloneDX reports, and the notes and annotations of CycloneDX and SPDX reports, in the scan result and in exported reports.
- Reduce database size (requires the matching SCA 5.0.1 engine).

v5.0.0

Release date: 10/05/2026

- Introduce CBOM (Cryptographic Bill of Materials):
  - Report cryptographic algorithms, libraries, certificates, protocols, and related material alongside software components.

  - Classify cryptographic assets as quantum-resistant or quantum-vulnerable for post-quantum migration planning.
  - Detect certificates embedded in binary files.
- Introduce AIBOM (AI Bill of Materials):
  - Report AI components and models, including their licenses, in the same unified report.
- Add dedicated SBOM, CBOM, and AIBOM views, and export them together or separately, in JSON or PDF.
- Restructure the scan result into per-module blocks with an engine-level summary, verdict, and risk count (requires MetaDefender Core V5.23+); earlier Core versions keep the previous format, without CBOM or AIBOM.
- Enhance SBOM detection of CycloneDX report with all types.

v4.9.1

Release date: 8/27/2026

- Report non-vulnerable dependency packages too, when Return all dependency packages is enabled.
- SPDX packages without a purl reference are now detected.
- Validation failures now report a reason.

v4.9.0

Release date: 8/6/2026

- Add criticality and end-of-life (EOL) detection when scanning CycloneDX and SPDX reports.
- Improve component detection accuracy for CycloneDX reports missing a bom-ref or purl field.
- Add package description for PHP, Ruby, JavaScript, Python, Java, and NuGet packages.

v4.8.1

Release date: 7/9/2026

- Fixed an issue where SBOM failed when scanning files with dependency detection enabled.

v4.8.0

Release date: 6/30/2026

- Add support for Julia and R programming languages in source code scanning.
- Improve scanning speed by up to 20% for container mode.
- Improve accuracy of OS package version reporting.
- Provide more complete vulnerability results when scanning SPDX 3.0.1 reports.

v4.7.2

Release date: 4/23/2026

- Support scanning of SPDX 3.0.1 reports.
- Support scanning of CycloneDX 1.4 reports.
- Add support for Kotlin.
- Add support for frequent update
- Add support for structured property

v4.7.1

Release date: 3/12/2026

- Support scanning of CycloneDX 1.7 reports.
- Add support for TypeScript, CoffeeScript, Scala, Groovy, and Clojure.

v4.7.0

Release date: 2/10/2026

- Support European Union Vulnerability Database (EUVD).

v4.6.0

Release date: 1/29/2026

- Add support for EPSS scoring.
- Support multiple severity sources.
- Preserve report metadata when scanning CycloneDX and SPDX reports.

v4.5.0

Release date: 12/22/2025

- Add support for scanning .csproj files.
- Integrate GHSA as a source for malicious package detection.
- Enhance CVSS scoring logic to accept scores from multiple sources.
- Introduce Database v6 with a reduced size and improved efficiency.

v4.4.0

Release date: 11/4/2025

- Support CIS Level 2 for Windows Server 2022.
- Enhance dependency detection of package.json (npm).
- Return CVSS Score.
- Enhance detection for application files.
- Add support recent malicious NPM packages.
- Reduce database file size.

v4.3.0

Release date: 9/2/2025

- A faster way to detect package information for Python.
- Improve the UI to display more details.
- Return the latest version information.
- Add support for several CVEs.
- Support Ubuntu 24.04 and Windows Server 2025.

v4.2.0

Release date: 7/8/2025

- Provide more license information: license URL, category, permissions, conditions, limitations.
- Return unique identifier, checksum, property (archive, executable) for a package.
- Add support for composer.json (PHP).
- Support Amazon Linux 2023, Ubuntu 22.04 CIS level 2.

v4.1.0

Release date: 5/22/2025

- Return "Authors" and "Release date" info for packages.
- Support processing SPDX file format.
- Support parsing composer.lock file (for PHP).

v4.0.1

Release date 3/27/2025

- Fixed an initialization issue on RH8 and Debian 11.

v4.0

Release date 3/20/2025

- Improvements for C/C++ libraries
  - Utilize PE information to detect DLL/EXE libraries.
  - Add more signatures for C/C++ libraries.
- Support processing CycloneDX file format.
- Improve Python detection methods.

v3.1.0

Release date 1/14/2025

- Improve license detection.
- Better way to handle internal databases.

v3.0

Release date: 11/4/2024

- Detect vulnerability for binary C# files.
- Support more file types for the other programing languages.
- Improve Maven license detection.
- Improve engine initialization time.
- Improve JAR file detection.

v2.1

Release date: 8/28/2024

- Support SPDX and CycloneDX reports (requires MetaDefender Core V5.11.1+).
- Fix several issues when running multiple threads.

v2.0

Release date: 7/8/2024

- Support license risk detection (requires MetaDefender Core V5.10.1+).
- Improve the method to compare package versions.

v1.2.1

Release date: 4/15/2024

- Fix initialization failure in non-English environments.

v1.2

Release date: 4/1/2024

- Support recursive dependency checks for source code.
- Return CWE info.
- Detect vulnerabilities in version ranges.
- Fix detection issues in case of CRLF line endings.

v1.1.1

Release date: 1/9/2024

- Return fixed versions for CVEs.
- Support more file names: C/C++ package manager file (`conan.lock`)
- Add package detection for`pom.properties` and `[package_name]-[package-version].jar`.

v1.1

Release date: 12/06/2023

- Release Linux version
- Support log configuration
- Improve Node, Python detection

V1.0.6

Release date: 10/25/2023

- Improve processing speed.
- Enhance tar extraction to get layer ids properly.

V1.0

Release date: 10/9/2023

- SBOM engine for MetaDefender Core Windows
- Support source code and container scan
- Support 11 programming languages

YARA engine release notes

V5.0.3

Release date: 7/23/2026

- **Bugfix**: Fixed an issue with the Engine crashing when shutting down
- **Upgraded to YARA-X v1.19.0**: The engine now uses the latest version of YARA-X, offering improved rule validation, stricter syntax enforcement, and enhanced performance.

v5.0.2

Release date 9/17/2025

- **Bugfix:** File under “C:/Program” is deleted

v5.0.1

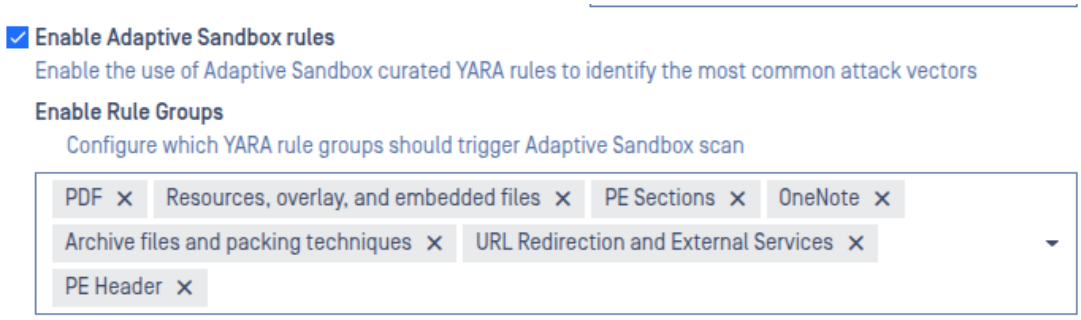
Release date 9/11/2025

- **Bugfix:** Fixed an issue with dependency check creating a file under "C:\Program"

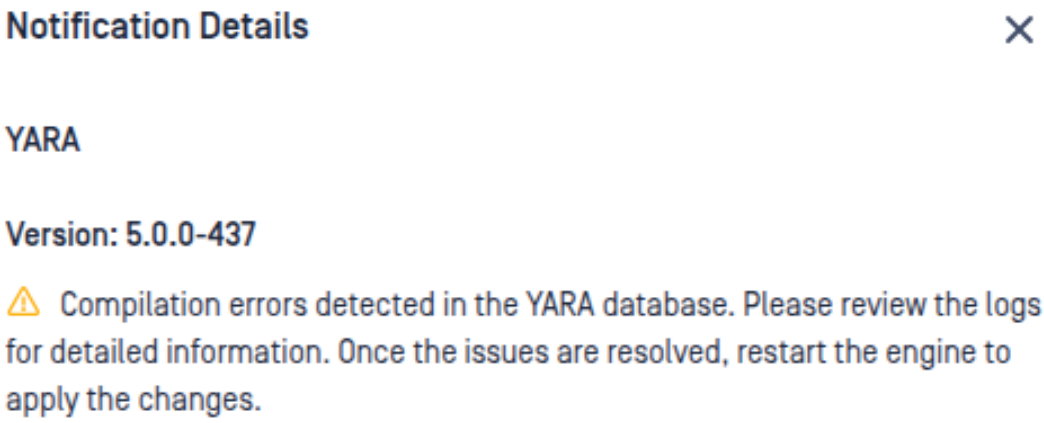
v5.0.0

Release date 9/8/2025

- **Upgraded to YARA-X v1.5.0** The engine now uses the latest version of YARA-X, offering improved rule validation, stricter syntax enforcement, and enhanced performance.
- **Customizable Adaptive Sandbox Rules** You can now enable or disable specific rule groups within the Adaptive Sandbox Ruleset, allowing for more flexible and targeted threat detection.



- **UI Feedback for YARA Rule Errors** Any issues with uploaded YARA rules are now clearly indicated in the **Core UI**, making troubleshooting faster and more intuitive.



v4.3.0

Release date 4/9/2025

- Added Adaptive Sandbox curated rules to identify the most common attack vectors
- Improved application security

v4.2.0

Release date 12/1/2024

- Digitally sign all binary files.

v4.1.0

Release date: 10/8/2024

- Upgrade Yara to version 4.5.1

v4.0.9

Release date: 8/19/2024

- Upgrade YARA to version 4.3.1.
- Upgrade OpenSSL to version 3.1.3.
- Change the module name from "Yara" to "YARA".

v4.0.8

Release date: 10/24/2022

- Expose matched info in the scan result

- Support multiple hits

v4.0.7

Release date: 9/7/2022

- Support Debian 11

v4.0.6

Release date: 7/12/2022

- Fix memory leak issue

v4.0.5

Release date: 3/29/2021

- Upgrade Yara to 4.0.5

v4.0.2

Release date: 1/25/2021

- Upgrade Yara to 4.0.2

v4.0.1

Release date: 6/20/2020

- Upgrade Yara to 4.0.1
- A new method to load database

Adaptive Sandbox release notes

Embedded Engine

Remote Engine

Embedded Engine

Known issues

Issue	Solution
MSI certificate validation is not supported on Windows OS	•

Issue	Solution

V3.3.1

Release date: 10/13/2026

New Features and Improvements

- Fixed configuration issues with No Threat Detected verdict
- Removed Java deprecation warning

v3.3.0

Release date: 10/10/2026

New Features and Improvements

- **LLM Scanning**Submit a model to MetaDefender Aether in the same way you submit any other file. Aether runs two analyses: static analysis of the model file and dynamic analysis of the running model. If Aether finds a threat, Core blocks the file.**Static analysis**Aether never loads or runs the model. It reads only the parts of the file that can carry code or configuration and skips over the weights, so memory use stays low even for very large models.Any payload Aether finds inside the file, including data hidden between tensors, is extracted and analysed on its own.**Dynamic analysis**Aether connects to your inference server through its OpenAI-compatible API and runs 38 attack probes against the live model, covering 9 of the OWASP Top 10 for LLM Applications categories. Probes run in order of severity. Each scan plants single-use canary tokens, so if one appears in a response, the leak is confirmed. Every finding shows the prompt, the model's response, the severity, the OWASP category and the rule that flagged it.Static and dynamic analysis are separate options, so you can run either one or both.
- **More Evidence from emulation**
- **HTTP traffic**: Emulation now records the full request and response for each HTTP call, including headers and body.
- **Delayed macros:** VBA emulation now supports Application.OnTime, which malware uses to delay its payload.
- **Deeper unpacking**
- **Installers**: Better extraction from nested installers and more installer types, including PyInstaller, py2exe, ps2exe and Nuitka files.
- **Verdict adjustment**Escalating and Disabling Threat Indicator is now supported
- **Offline Reputation Service**Supplying your own reputation database is now supported
- **Extended YARA triggers**Adaptive Sandbox scan can be triggered for custom YARA rules

New Features and Improvements

v3.2.0

Release date: 31/07/2026

New Features and Improvements
• **Detection for Emerging Evasion Techniques:** Threat actors keep getting more creative — Sandbox keeps pace. This release adds detection for *etherhiding* (concealing threat actor infrastructure inside blockchain transactions) and intentionally malformed .NET assemblies built to break automated analysis tools. • **Smarter Remote Template Detection:** Remote template injection analysis got a full pass this release, aimed at keeping the false-positive rate flat while closing real detection gaps. A rebuilt link-detection and URL-classification pipeline now tells genuine injection attempts apart from ordinary embedded links — fixing two indicators that had been flagging legitimate URLs as malicious, and closing a gap where obfuscated, octal-encoded malicious IP links were going undetected. Fewer false alarms, no loss of coverage. • **Broader Detection & Coverage Improvements:** This release also expands detection coverage on several fronts: new detection engineering for wiper malware, CAPTCHA identification via OCR (including obfuscated but rendered fake CAPTCHAs used in ClickFix attacks), new config extractors for PrivateLoader and Quasar RAT, and expanded IOC coverage in the local reputation database.

v3.1.0

Release date: 31/01/2026

New Features and Improvements
What's new • **Updated ChatGPT summary prompt:** Improved the default value of ChatGPT summary prompt to provide better context for analysis. (Existing prompts are unaffected.) • **E xpanded archive support:** Multiple archive types are now supported with Core 18.1. • **Deep CDR hyperlink trigger:** The hyperlink Deep CDR type can now be used to trigger Sandbox analysis. • **C2 URL construction from malware configuration** — For supported malware families, C2 server syntax is now parsed and final URLs are constructed when possible. This construction enables the next-stage download attempt. • **Enable emulation for base64 strings in documents** — Aether now emulates base64 decoded files, even in

New Features and Improvements

cases where string chunks appear separated. Also, it detects invalid Base64 characters within otherwise valid strings.

- **Rust binary metadata detection** — Aether can now detect metadata specific to Rust-compiled binaries, improving classification accuracy for this increasingly common language target
- **INI file parser and heuristic detection** — Aether now detects and parses INI configuration files, correctly recognizing the media type and extracting embedded artifacts including URLs and threat indicators.
- **Extended script emulation coverage** for VBScript, JScript and HTML-embedded scripts.
- **Added new detection "clues"** for encoded / decrypted content, such as `DecodeData` in JScript (`TextDecoder.decode`), improving visibility into unpacking and de-obfuscation behavior.
- **Introduced richer environment and DOM support** (e.g. `NodeList.forEach()`, `Document.importNode()` for SVG, WshShell environment getters/setters and additional environment types), enabling more realistic script execution paths.

Improvements

- **Security hardening of dynamic analysis and emulation components** Reduced security vulnerabilities across the dynamic analysis and emulation stack to lower risk and increase trust in analysis outputs and integration points.
- **More efficient extraction and prioritization of deeply nested content** Refined how embedded archives and extracted files are handled across layers, including updated extraction-layer semantics and smarter file-prioritization on the whole layer. This reduces over-processing of deeply nested archives while still prioritizing the most relevant content for analysis.
- **Cleaner, more secure dependency and configuration baseline** Resolved multiple dependency-related security findings, removed unused or obsolete configuration options and helper utilities, and improved test structure for specific emulators. Together these changes reduce attack surface, simplify configuration, and make automated test coverage more targeted.
- **Higher-quality IOC and URL processing for real-world content** Improved decoded-string and URL handling in scripts (including encoded PowerShell), and tightened email and URL-IOC extraction rules. These updates reduce false positives and make extracted IOCs and URLs more accurate and actionable.
- **Enhanced handling of password-protected and obfuscated samples** (including reflection by name and password-protected macros), so more real-world malicious workflows can be exercised during emulation.
- **Upgraded underlying packages** in internal components to keep the emulation stack more robust and maintainable.
- **IOC extraction moved to direct MWCFG parsing** — IOCs are now extracted directly from parsed malware configurations instead of relying on a generic method, which was less accurate. This removes duplicate URLs and ensures IOCs correctly reflect their malware config origin.

Bug Fixes

- **Aligned malware configuration data model with network indicators** Updated the malware configuration data model to match the latest network-indicator schema so that network-related artifacts are represented

New Features and Improvements

correctly and consistently across the platform.

- **More resilient handling of large and problematic documents** Improved system behavior when processing oversized or malformed content so that “document too large” conditions and related failures are handled gracefully, with clearer reporting and fewer noisy errors in both logs and UI.
- **More reliable archive extraction and file-type handling** Fixed failures when unzipping certain ZIP files and addressed scan issues for specific file types, so that previously problematic content is now extracted and analyzed as expected.
- **More accurate detection for encrypted Office files and emails** Fixed cases where encrypted Office attachments were not scanned even when the password was guessed correctly, and corrected email detection behavior in the filesystem transform, ensuring these data sources are consistently inspected.
- **Corrected URL and IOC handling in reports and offline scenarios** Fixed problems where URLs derived from redirects were missing from JSON reports, and corrected IOC processing behavior for allowlisted IOCs, so that reported URLs and indicators better match real analysis conditions.
- **Fixed hashing edge-case errors**: Corrected hash calculation for rendered OOXML images, removing edge-case failures and making verdicts and hashes more reliable for downstream systems.
- **Fixed multiple emulation failures**, such as VBA emulation issues on specific customer malware samples and long delays in samples due to improper `WScript.Quit()` handling.
- **UPX not detected as packer for some samples** — Certain UPX-packed binaries were not being flagged as packed. Detection logic has been corrected.
- **Ineligible file types sent to PE unpacking and disassembly** — Files without a valid structure were reaching the PE unpack and disassembly stages, producing errors. These files are now filtered out before reaching those stages
- **False positives in BTC wallet IOC detection** — The BTC wallet YARA rule was matching Base64 content and producing false positives.
- **Network indicator model mismatch in malware config DTO** — The malware config data transfer object was using an outdated network indicator model. Updated to align with the current model definition.
- **Password extraction regex missing matches** — Two bugs fixed in password extraction from file content process, making extraction more reliable.

v3.0.0

Release date: 26/01/2026

New Features and Improvements

What's new

We are excited to announce the release of **MetaDefender Aether for Core v3.0.0.** With this update, users can now enable dynamic file analysis and behavioral threat detection directly within their Core environment—no

New Features and Improvements

additional infrastructure required.

- **Simplified and Enhanced OSINT Lookups** - OSINT lookups were simplified and local reputation results are now visible in the OSINT results as OFFLINE_REPUTATION
- **Verdict Renaming** – Updated verdicts to provide clearer guidance on how to interpret analysis results and take appropriate actions based on potential threat levels
- **E-PUB support** - Added support for EPUB files, to ensure broader compatibility with widely used e-reading formats.
- **IOC Skip List Migration** - Moved the IOC skip list (allow/blocklist) from the local application database to the separated detection database package to enable frequent updates
- **Core Component Hardening** - Hardened core components with additional security safeguards to strengthen the overall defense posture.
- **QR Code Detection Improvements** - Improved to allow the scanning of rendered images in documents and emails
- **Batch File-Type Detection** - Enhanced to improve accuracy across complex payloads.

Bug Fixes

- **BAT Script Classification Fix** - Corrected file-type misclassification for obfuscated BAT scripts, eliminating related false positives.

Threat Detection 2.0.0 Release Notes

MetaDefender Aether for Core supports independent updates to detection logic and threat intelligence, enabling faster deployment of new protections and a more rapid response to emerging threats. The following updates have been delivered over the past several months.

What's new

- **Significant Enhancements to PE Installer Analysis** - Added deep static extraction and analysis for Windows installers: NSIS, Inno, InstallShield, Advanced Installer, Wise, WiX, InstallAnywhere, and Actual Installer. It now extracts prioritized embedded files, analyzes installer scripts, and scores custom installers heuristically.
- **AI / ML Model Security Scanning** - Introduced security analysis for Machine Learning models, including multi-serialization parsing and deep static inspection to detect hidden malicious payloads before they impact AI workflows.
- **Zero-Day Exploit Detection** - Added detection for recent Windows Explorer LNK vulnerabilities (CVE-2025-50154, CVE-2025-59214) that leak NTLM credentials without user interaction. Also, introduced detection for critical XXE exploitation in Apache Tika (CVE-2025-66516).
- **Phishing Campaign Intelligence** - Introduced indicators for seasonal / opportunistic lures (holidays, global events). Improves campaign clustering and early phishing detection.
- **Invisible String Obfuscation Detection** - It now detects *GlassWorm*-style obfuscation techniques by identifying invisible and deceptive code constructs used to evade analysis and enable silent execution. This includes detection of homoglyph characters, PUA (Private Use Area) Unicode characters, and hidden or non-printable

New Features and Improvements

whitespace that visually alters code without changing execution.

- **Advanced Evasion Technique Detection** - Expanded detection for techniques used to evade automated analysis, including meaningless infinite loops, abuse of password-protected in-house macros, .NET PE constant protection and control-flow obfuscation, and deobfuscation of hexadecimal-encoded JavaScript.
- **Expanded File Type & Archive Coverage** - Added support for VSIX files, CRX archives, and improved archive file filtering and prioritization.
- **HTML Threats & Web Payload Extraction** - It now extracts hidden downloadable payloads from HTML href data blobs and improves detection for abused HTML capabilities (e.g., disabled context menu).
- **Detection of IP logging and tracking services** - Added indicators to flag the presence of IP-logging URLs that may be abused by attackers to identify victim location, apply geofencing, or support other reconnaissance and targeting activities, improving visibility into early-stage attack workflows.

Improvements

- **Encrypted Documents** - Enhanced decryption for protected Office and PDF documents by introducing a multi-step password recovery with fallback logic for phishing-delivered encrypted files.
- **Base64-Encoded Payload Extraction** - Added base64 decoding for dynamically created files during emulation. This improves visibility into malware that reconstructs payloads at runtime to bypass static analysis and hide secondary stages until execution.
- **ClickFix Variant Detection** - Improved detection for new ClickFix variants that abuse hex-encoded URLs and msiexec execution.
- **Email & Phishing Analysis Accuracy** - Improved EML parsing to correctly associate images with embedded URLs, support text-based attachments, and enhance call-to-action detection across multiple languages.
- **Multi-Stage Infection Chain Visibility** - Improved emulation triggering for downloaded scripts, enabling full infection chain analysis when payloads are fetched from an external resource.
- **Better Detection Precision & URL Noise Extraction Reduction** - Reduced false positives by refining encryption-related indicators shared by malware and legitimate installers, adapting string-detection heuristics for PE files.

Bug Fixes

- **MSI Extraction Reliability** - Fixed multiple issues causing missing or incomplete payload extraction from MSI installers.
- **URL Redirection & Download Chain Handling** - Fixed missing redirected URLs, deduplicated download tasks, and added detection for delayed redirections.
- **Indicator Stability & Engine Robustness** - Fixed unintended indicator triggers (including PDF-related), skipped unnecessary PE disassembly when packers are detected, and resolved multiple internal indicator bugs.
- **Duplicate Tags** - Tags originating from multiple sources were being shown repeatedly in the UI. The fix ensures that tags are properly deduplicated so each one appears only once.

v2.5.1

Release date: 11/20/2025

New Features and Improvements

Improvements

- **Stability & Security -** Updated core dependencies to enhance performance and security.

Bug Fixes

- **Fixed zlib support** – zlib files are now correctly processed by the Archive Engine instead of the Adaptive Sandbox, ensuring proper handling and improved efficiency.

v2.5.0

Release date: 10/30/2025

New Features and Improvements

What's New

- **Periodic Threat Detection Updates -** MetaDefender Sandbox now supports independent updates to detection logic and threat indicators, enabling faster deployment of new protections and quicker responses to emerging threats.
- **Offline Mode Certificate Validation** - Added a transform configuration to whitelist signed files without revocation checks; auto-enabled in air-gapped environments, but can be disabled for maximum security.
- **MP3 Filetype Support** – Expanded filetype coverage with MP3 parsing and analysis.

Bug Fixes

- **Incorrect URL Extraction** – Resolved parsing issues causing incomplete or inaccurate URL extraction.
- **Reporting & Metadata** – Corrected metadata keys to ensure consistency.

Threat Detection 1.0.0 Release Notes

Threat Detection 1.0.0 detects AI-based evasion techniques, advanced installer and filetype support, and improved zero-day defense, empowering organizations with proactive, agile protection against modern threats.

What's new

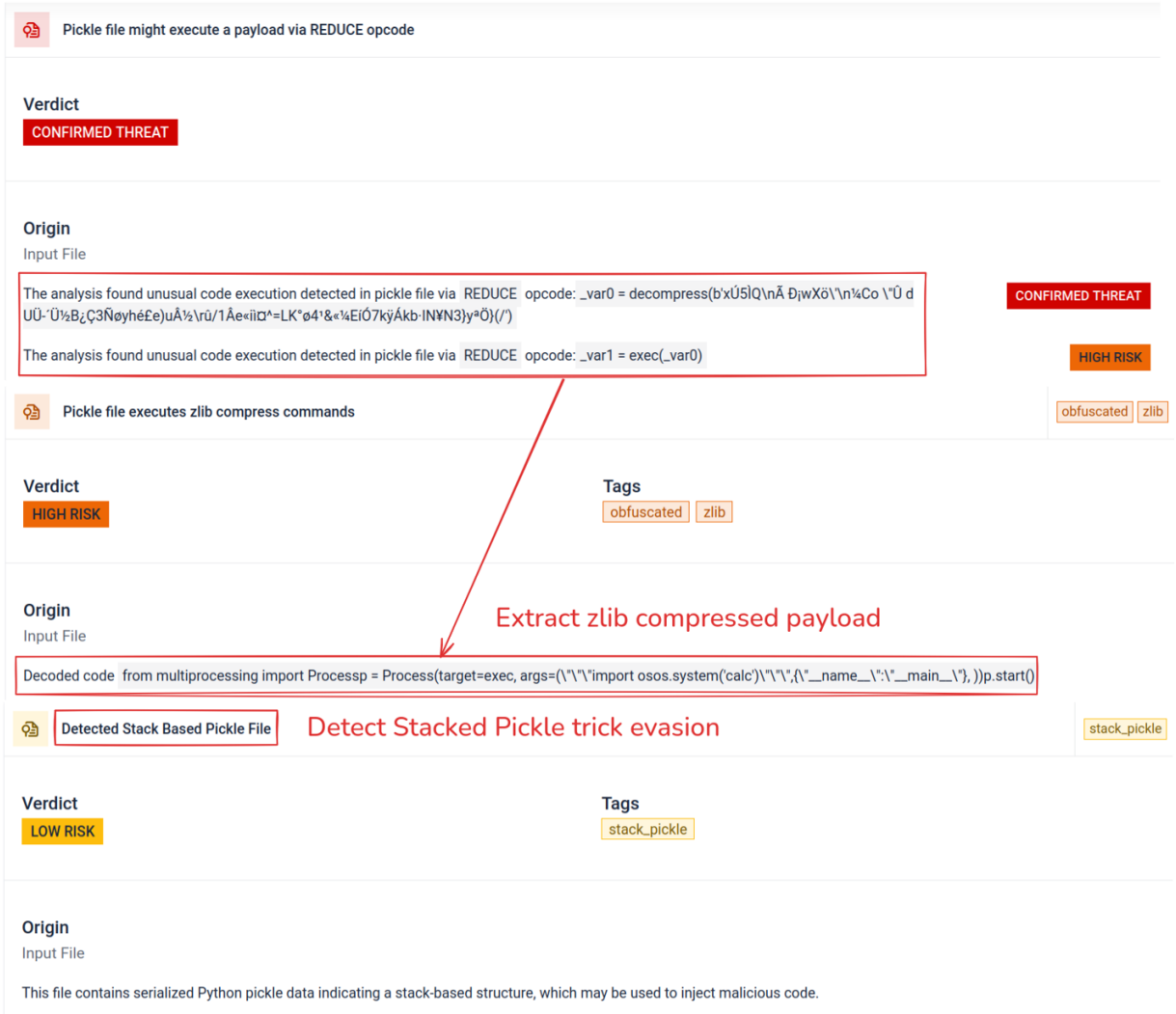
- **Double Base64 Decoding** – Detects payloads hidden in multiple layers of Base64 encoding, commonly used by advanced malware to evade security controls.
- **Extended Threat Indicators for Pickle & PyTorch** – Detects weaponized Python serialization and machine learning model files often used for supply chain and AI-related attacks.

New Features and Improvements



Detection of Pickle file capabilities

- **Improved AI Evasion Detection** – Enhanced identification of the **nullifAI evasion technique** and **stack pickle manipulations**, strengthening AI/ML malware defense.

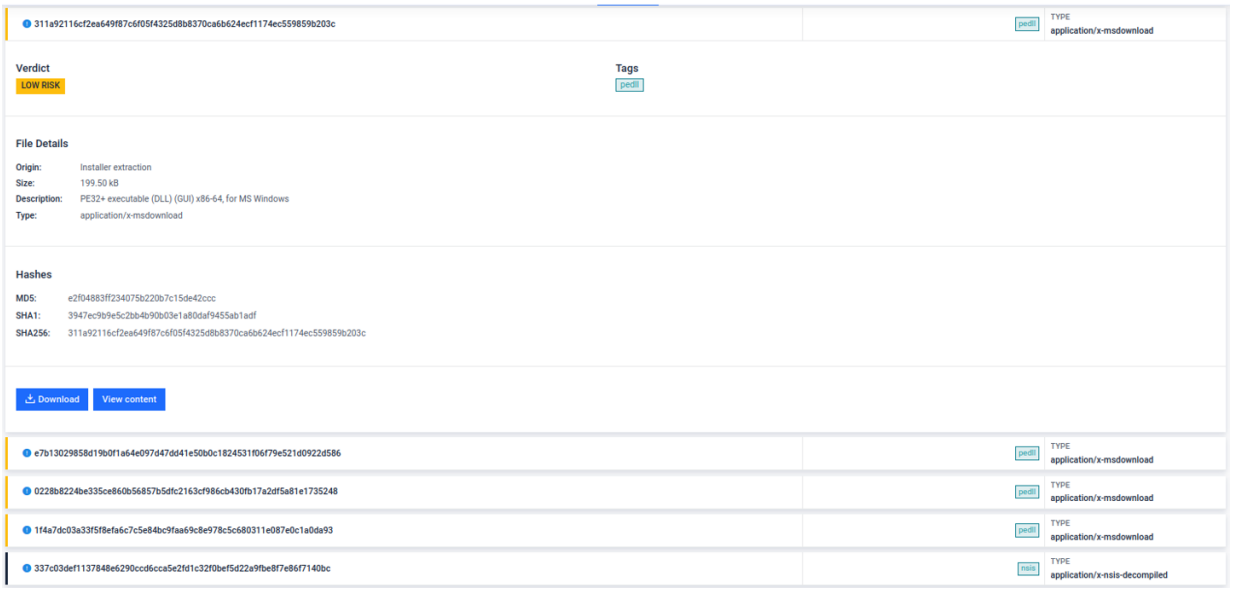


Stacked Pickle trick evasion

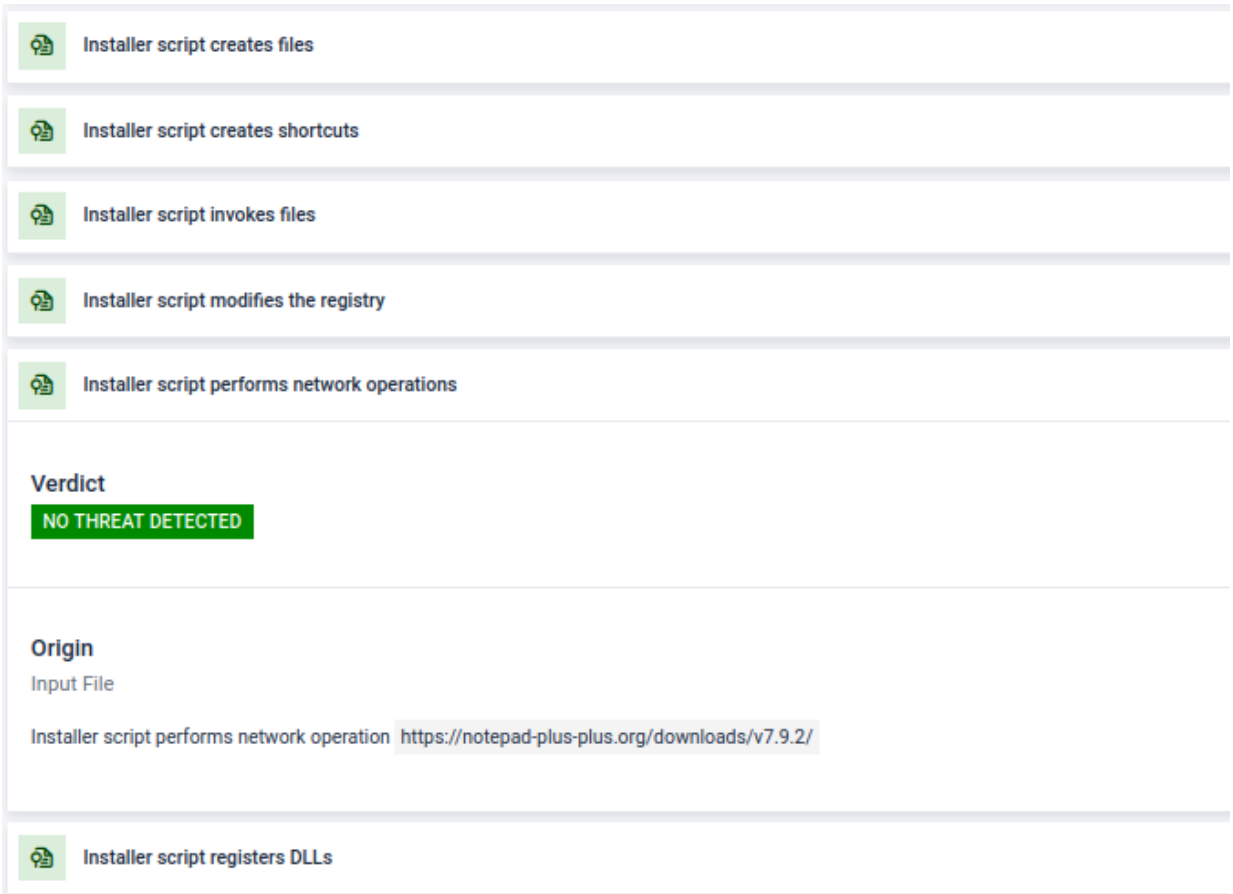
- **New Installer Package Support** – Added extraction and analysis for:

New Features and Improvements

- **Advanced Installer packages**
- **NSIS (Nullsoft Scriptable Install System) packages**
- **Wise Installer packages** - This expands coverage for malware distributed via custom installer frameworks.



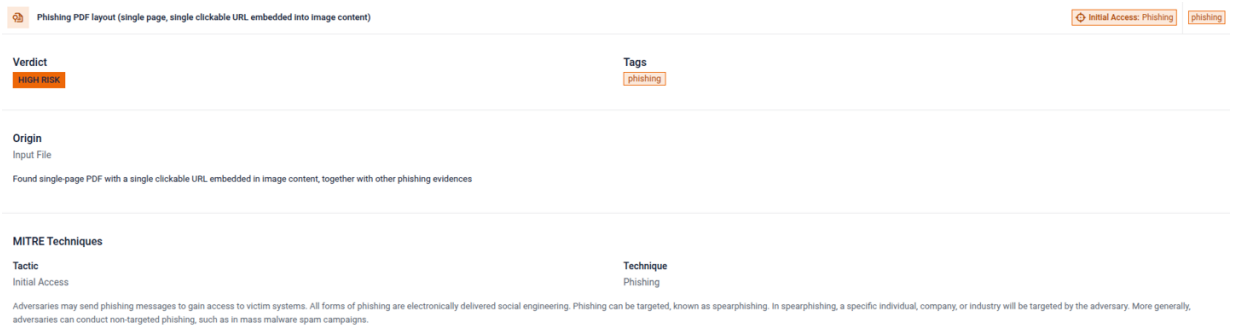
Static file extraction for PE installer



New Features and Improvements

Threat indicators detecting actions defined in installer scripts

- **CVE-2018-15982 Detection** – Identifies exploitation of a critical Adobe Flash vulnerability.
- **Equation Editor Exploit Detection** – Detects obfuscated versions of this long-abused Microsoft Office exploit.



Signal group and threat indicator detecting obfuscated and malformed exploit documents

- **Extended PDF Threat Indicators** – Better phishing detection in PDF documents, with new heuristics for malicious links and embedded content.

v2.4.0

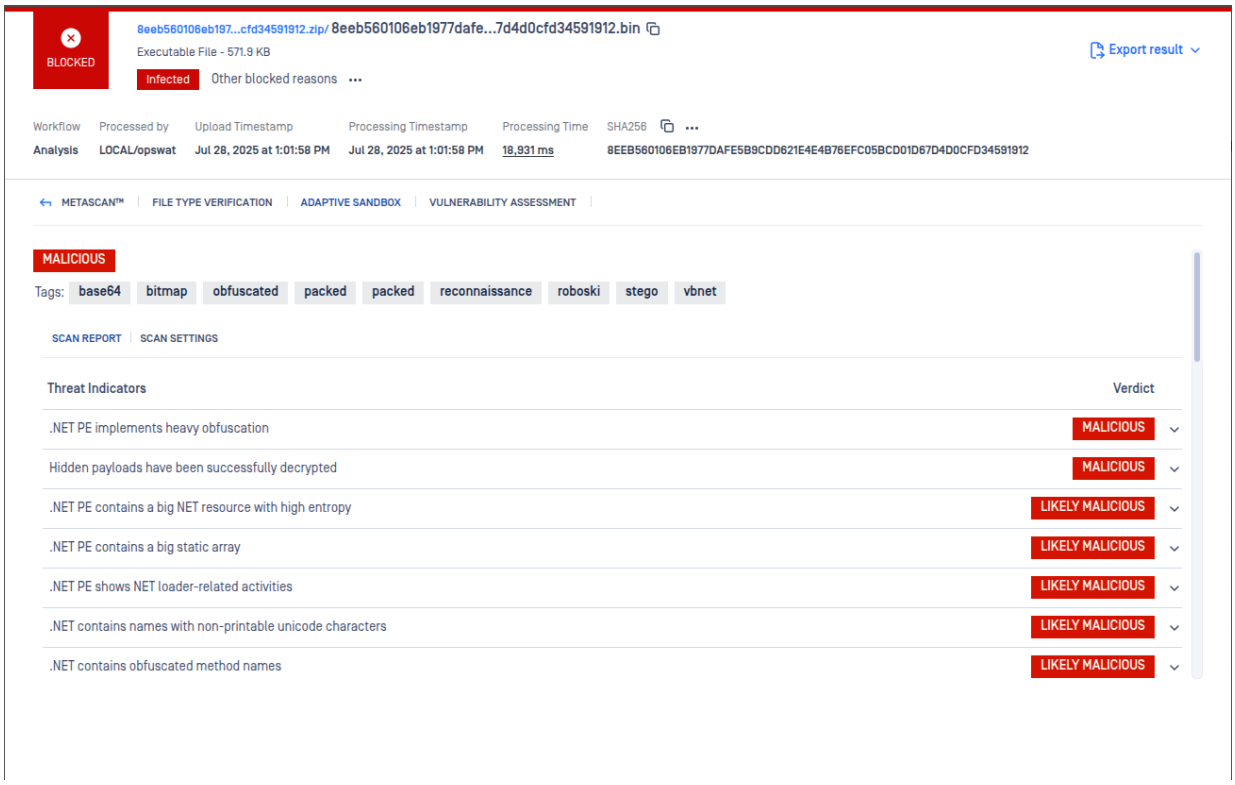
Release date: 07/31/2025

New Features and Improvements

What's New

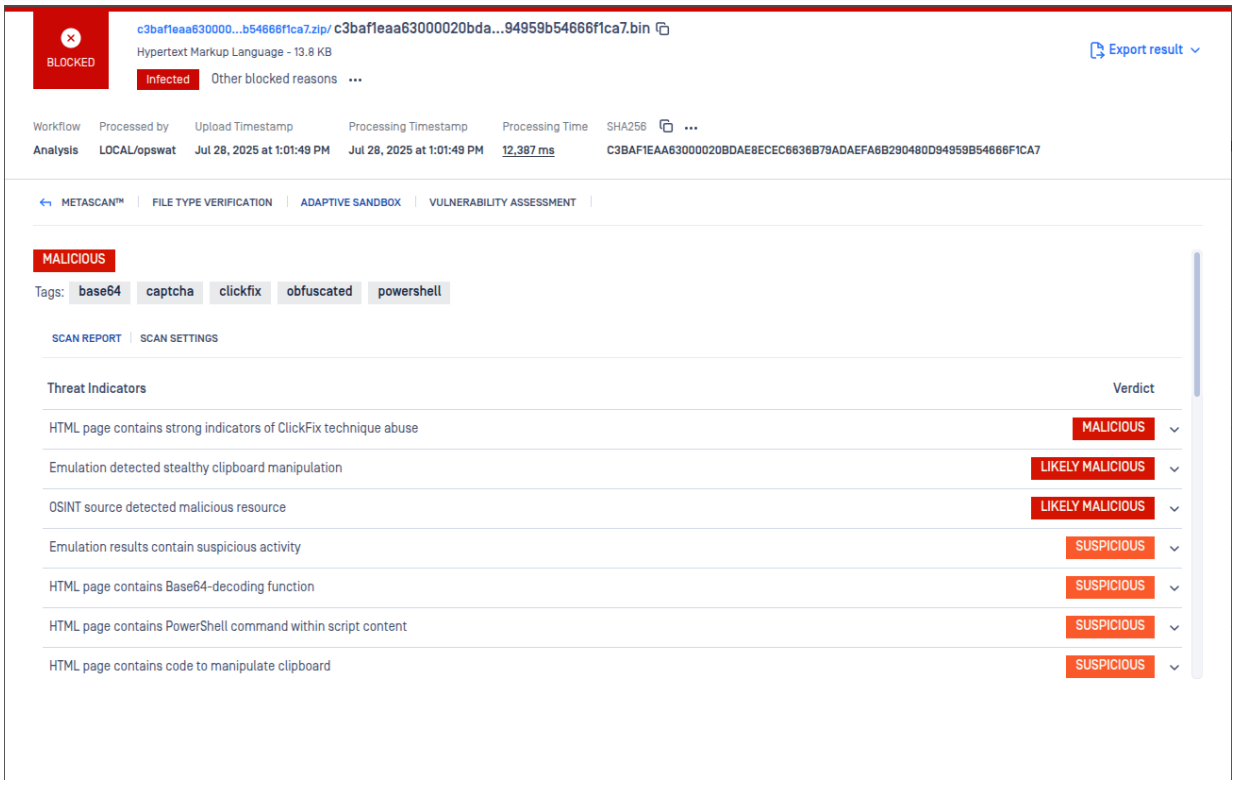
- **Updated YARA & Malware Config Extraction Logic**
  - Updated YARA & Malware Config Extraction Logic for Lumma Stealer (ChaCha), MetaStealer, and Snake Keylogger.
- **.NET Loader Unpacking Enhancements**
  - **.**NET Loader Unpacking for Roboski and ReZer0 loaders.

New Features and Improvements

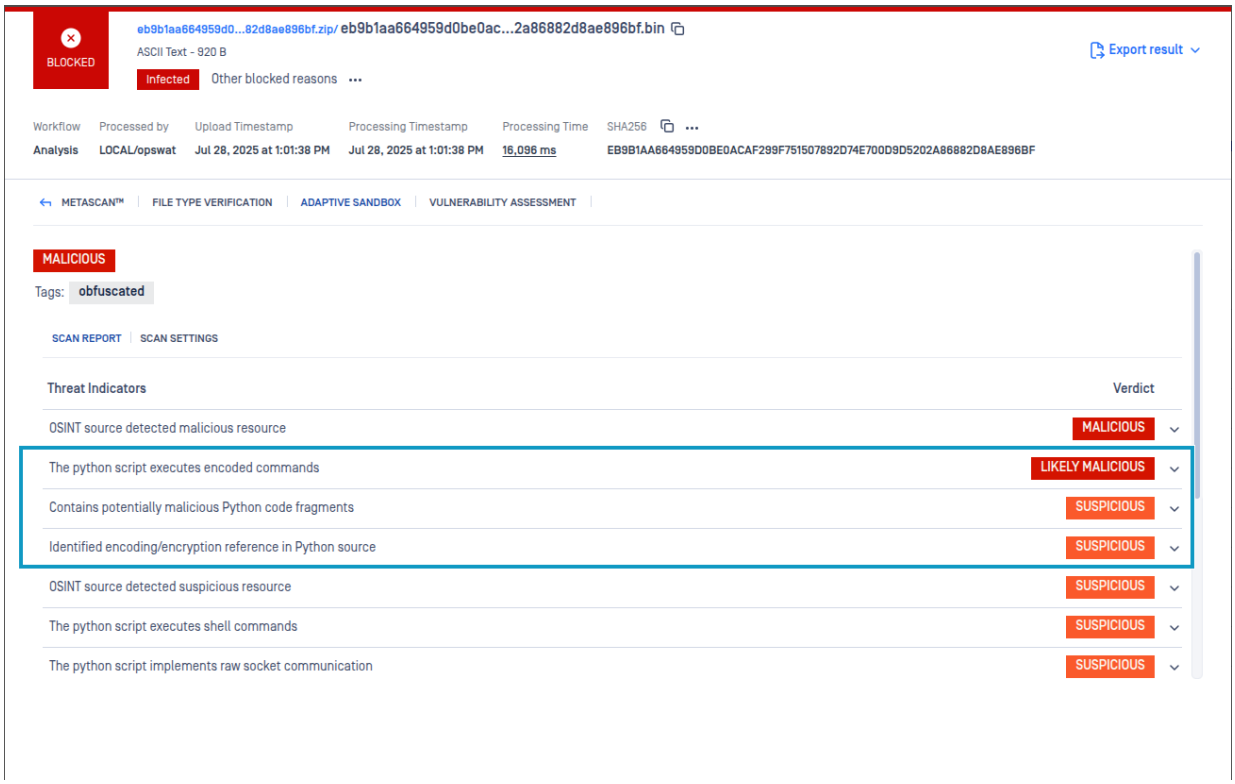


- **Control Flow Deobfuscation in .NET Files**
  - Added control flow deobfuscation in .NET files to enhance payload unpacking performance.
- **Early Detection for ClickFix Variants**
  - Introduced early detection capabilities for ClickFix and its variants, a trending social engineering technique.

New Features and Improvements



- **Automated Decoding of Base64 Commands**
  - Implemented automated decoding of base64-encoded commands for Python and Bash scripts, uncovering deeper layers of obfuscation and enhancing threat detection accuracy.



New Features and Improvements

- **Support for ACCDE File Analysis**
  - Now supporting analysis of ACCDE files—expanding coverage to include Microsoft Access Applications and strengthening visibility into embedded threats.

Improvements

- **Improved .NET Binary Detection**
  - Improved detection of suspicious .NET binaries by analysing low-level anomalies - such as oversized static arrays, high-entropy resources, and abnormal call patterns - enabling earlier identification of obfuscated threats.
- **Expanded Threat Coverage**
  - Boosted IOC and MITRE ATT&CK coverage with deeper parsing of LNK files, smarter enrichment of indicators, and more flexible tagging using YARA-derived metadata-sharpening threat correlation and context across reports.
- **Smarter File Detection**
  - Updated the DIE (Detect It Easy) Database to improve the identification of file types and packed binaries, boosting classification accuracy across diverse samples.
- **Heuristic Lookup Optimization**
  - Disabled OSINT lookups for heuristically extracted domains - reducing false positives and improving the precision of threat verdicts.

Bug Fixes

- **Improved Malware Unpacking**
  - Resolved an unpacking issue with downloaded files, improving analysis of second-stage malware.
- **Corrected Hash Validation**
  - Corrected a hash mismatch during PE resource extraction - ensuring accurate file integrity checks.
- **Sharper Verdict Accuracy**
  - Refined detection logic for Office documents, emails, and URLs to minimize false positives, in turn sharpening verdict confidence.
- **Fixed URL Construction**
  - Corrected URL construction logic for absolute paths in Open Directory Scans.

New Features and Improvements

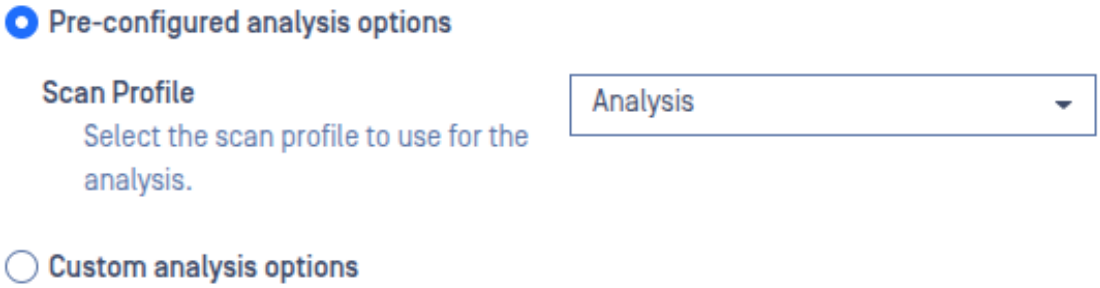
- **Enhanced Extractor Stability**
  - Improved stability of the base64-encoded file extractor, which now handles edge cases that previously caused failures.
- **Smali Parsing Fix**
  - Fixed an error in parsing Smali files within APK packages - improving visibility into APK Packages.

v2.3.0

Release date: 06/05/2025

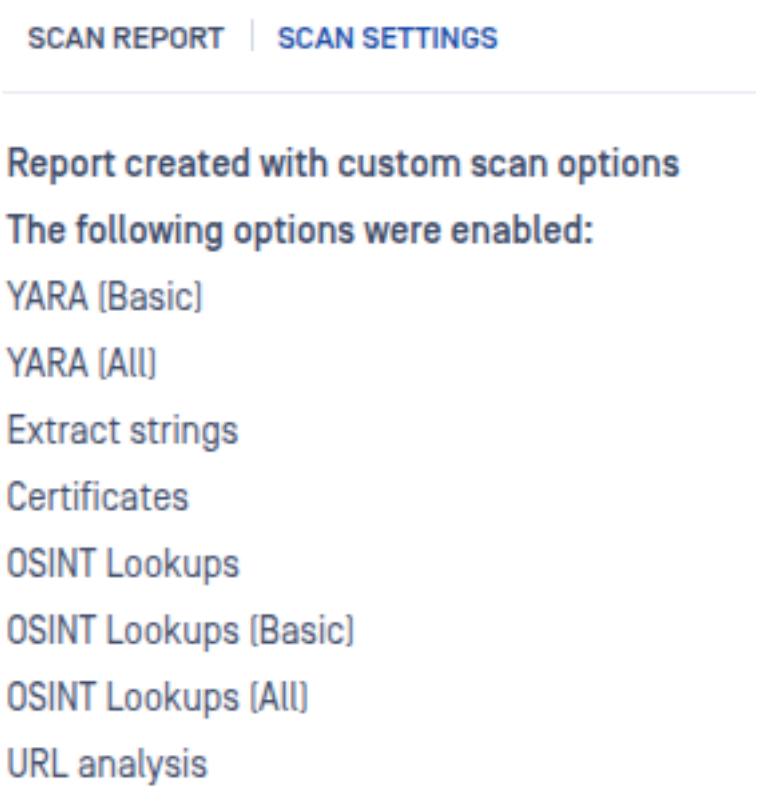
New Features and Improvements

- **Scan Execution Profiles**: Users can now select from predefined scan profiles or create custom profiles, enabling a more streamlined and flexible scanning experience tailored to specific needs.



- **YARA result triggering:** Sandbox scans can now be automatically initiated based on curated rules embedded within the Adaptive Sandbox's YARA engine, allowing for intelligent, rule-driven threat escalation. YARA Trigger
- **Enhanced file type support:** The file type selector has been updated to more accurately reflect the full range of formats supported by the Adaptive Sandbox engine, improving usability and clarity.
- **Result updated with scan preferences:** Selected scan settings are now displayed in the Result tab, making it easier to troubleshoot and understand how each scan was configured and executed.

New Features and Improvements



- **Classify Threat Indicators based on sandbox capabilities**: Improved classification and visibility of threat indicators derived from sandbox behavior. Enhancements include better translation, risk categorization, and filtering to support faster triage.
- **HTML URL Parsing:** Integrated a new parser for analyzing URLs embedded within HTML content in emails and web threats, improving detection precision in phishing and redirection scenarios.
- **Offline URL Model Update:** The model has been retrained using a newly compiled set of URLs to enhance detection accuracy.
- **Performance Optimizations:** Improved threat detection performance by enhancing domain resolution speed, scan performance, and reducing memory usage.
- **String Extraction:** Improved string extraction logic, including better UTF-8 handling and efficiency.
- **URL Handling:** Fixed issues with multi-redirected URLs, reputation lookups, and file type misdetection.
- **New Malware Family Detection:** Added config extractors for XWorm, Stealc, and new Lumma Stealer variants, boosting coverage of active campaigns. Also extended detection rules for emerging loaders and RAT such as PrivateLoader and Millenium RAT.
- **Short-Lived Certificate Whitelist Bypass Detection:** Prevents whitelisting of malware signed with short-lived certificates - a tactic used to bypass detection mechanisms.
- **API Hashing Detection:** Flags obfuscated API calls using hashing, being critical for catching stealthy malware.
- **Phishing DKIM Replay Abuse Detection:** Flags spoofed DKIM attempts, enhancing phishing resilience against

New Features and Improvements

recent campaigns that abuse Google Sites and DKIM replay to send valid-signed emails, bypassing filters and stealing credentials.

- **Enhanced PE Anomaly Detection:** Added 40+ signatures to detect PE structure anomalies, such as significant discrepancies between virtual size and raw size in PE headers.
- **APK Certificate Signer Validation:** Added validation and whitelisting of APK certificate signers to improve trust assessment and reduce false positives in mobile threat detection.
- **Improved File Extraction From Base64 Artifacts:** Now supports extraction of embedded Base64-encoded files, including reverse cases seen in malware.
- **Script-Based Threat Detection Enhancements:** Improved detection of malicious JavaScript, VBA, and PowerShell behavior - including decoding, decryption, and suspicious cmdlet abuse. Emulated PowerShell code is now extracted as readable strings, enhancing visibility and analysis.
- **Disassembly Code Extraction Improvements:** Fixed issues with disassembly functions on RET instructions, implemented string annotation for immediate strings, and enhanced filtering to avoid disassembling junk code, improving the accuracy of extracted disassembly data.
- **Adaptive Indicator Tuning for Signed PEs:** Fine-tuned context-aware indicators for signed PE files in offline environments, reducing false positives by improving detection accuracy when certificate validation is unavailable.
- **Certificate Validation Process Fix:** Resolved the issue where edge-case certificates, which are technically valid, were incorrectly detected as invalid.

v2.2.0

Release date: 02/17/2025

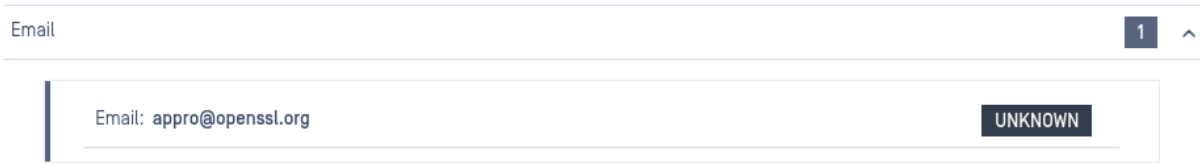
New Features and Improvements

- **Daily Scan limit**: Daily scan limit is enforced based on User license, Licensing
- **Threat Indicator details**: Report is extended with Threat Indicators details, providing better and easier assessment of the file.



- **Updated IOC details**: Improved IOC details UI

New Features and Improvements



- **Dependency handling**: Visual C++ Redistributable is no longer required, making it easier to install and update the engine on the fly.
- **URL Parsing**: Embedded engine will no longer try to parse Remote URL, fixing false Error messages in Engine logs
- **Property File**: Property files are no longer overridden
- **Embedded File extraction**: Embedded file extraction is now correctly working on Windows instances, improving the amount of data available for scans.
- **Embedded File Yara**: Yara rules are now correctly applied to Embedded files on Windows, ensuring better static analysis of files.
- **Zero-Day Office Document Support**: We now support the analysis of broken Office documents, including those used in zero-day attacks. This helps detect previously evasive threats targeting Office file vulnerabilities.
- **Mitigation for Bloated Executables**: A new mitigation feature identifies intentionally bloated executables that attempt to bypass sandbox environments. This enhances the platform's ability to analyze suspicious files that may try to evade detection.
- **New Malware Family Detection**: Detection capabilities have been extended to include notable malware families such as WezRat, Remcos, Lumma Stealer, among others. This expands our database and improves detection across a wider array of threats. Supported malware families via YARA
- **JavaScript-Compiled PE File Support**: We've added unpacking and decompilation support for JavaScript-compiled PE files, enabling better analysis of malicious payloads that may be delivered through JavaScript.
- **Malicious Techniques Detection**: MetaDefender now identifies and flags malicious techniques such as disabling Data Execution Prevention (DEP), Authenticode evasion (SigFlip), and misleading script encoding. This enhances the platform's ability to spot sophisticated attack vectors.
- **WebDAV Communications Identification**: MetaDefender can now identify WebDAV communications, improving the detection of network-based attacks that use this protocol.
- **Support for Password-Protected Office Documents**: MetaDefender now supports the analysis of password-protected Office documents, improving its ability to handle encrypted files during analysis.
- **UNC Paths on IOCs Page**: The IOCs (Indicators of Compromise) page now supports UNC path display, providing easier access to file locations in network environments.
- **Enhanced File Type Analysis**: Analysis for APK, RDP, Shell Script, and CHM files has been expanded, improving detection capabilities across these file formats.
- **Heuristic Enhancements for Packed Executables**: The heuristic engine has been upgraded to better detect packed executables, which are commonly used to obfuscate malware.
- **Extended .NET Obfuscation Detection**: We've improved detection for obfuscated .NET assemblies and added

New Features and Improvements

better handling for unmanaged .NET functions, making it easier to detect threats hiding in .NET-based applications.

- **XOR Decryption Improvements**: XOR decryption capabilities have been enhanced for PE files and URLs, increasing the platform's ability to analyze encrypted malware.
- **Extended Adaptive Context for Executables and Emails**: The adaptive context feature has been expanded to include deeper analysis of executables and email files, improving detection of hidden threats.
- **Extended Base64 Decoding for Downloaded Files**: The platform now supports extended base64 decoding for downloaded files, enhancing its ability to analyze encrypted or obfuscated content.
- **Blacklist for Abused Certificates**: We've extended the blacklist for abused certificates and bootloaders, improving protection against threats using known malicious certificates.
- **Whitelist Accuracy**: The file whitelisting process has been fine-tuned for greater accuracy, ensuring only trusted files are excluded from scans.
- **Improved Emulation Support**: Emulation for JavaScript coming from email attachments has been extended, improving detection of email-borne JavaScript threats.
- **Polyglot Detection**: The ability to detect polyglot files has been improved, increasing the platform's ability to identify files that contain multiple formats.
- **Enhanced Emulation for Common Threats**: Emulation for VBA, JavaScript, PowerShell, Microsoft Equation, and shellcode has been improved, providing deeper analysis for common attack techniques.
- **Improved PE Signature Validation**: PE file signature validation has been enhanced, improving detection of malicious files and ensuring more accurate assessments.
- **Improved QR Code Analysis**: Enhancements to QR code analysis allow for better identification of hidden threats and malicious links within QR codes.
- **VBA Macro Signature Detection**: A bug in the VBA macro signature detection has been resolved, ensuring more accurate detection of malicious macros in Office files.
- **Oledump Tool Update**: The oledump tool has been updated to reduce the incorrect extraction of strings, improving the analysis of OLE files.

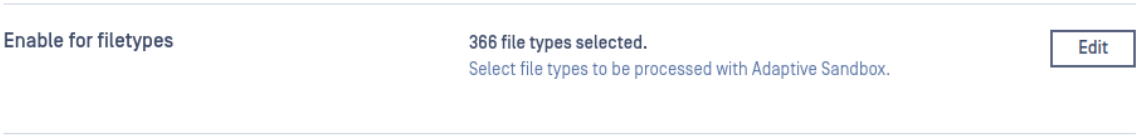
v2.1.0

Release date: 12/02/2024

New Features and Improvements

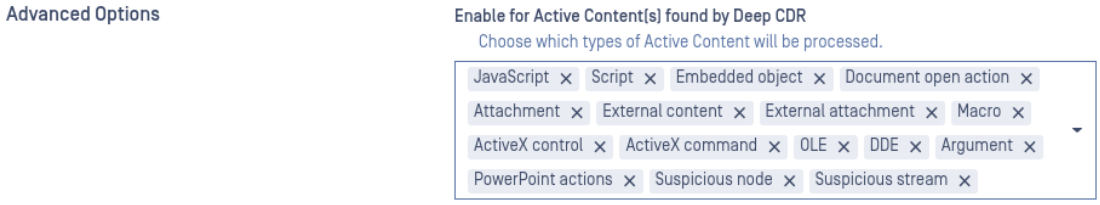
- **Min-Max Analysis timeout changed:** Minimum and Maximum Analysis Timeout changed to 60-86400, for better handling of Large Files.
- **.Dll files are signed:** Windows .Dll files are now signed for better security
- **Enhanced Archive handling:** Archive File handling improved to reduce Scan times.

New Features and Improvements



- **Java Dependency checking is fixed:** Fixed Java dependency checking to no longer mismatch some Java versions
- **Deep CDR Triggers options added:** New Deep CDR Trigger option is added to Workflow configs for a more granular setup option.
- **Removed Scan modes from configuration:** Scan mode workflow config is no longer supported.
- **Context-Aware Threat Indicators:** Improved threat indicators by factoring in the context of the analysis, leading to more accurate threat assessments.
- **Reduced False Positives**: Lowered false positive rates for heuristically detected or non-clickable IP addresses and URLs, improving the accuracy of threat analysis.
- **Malicious Document Detection**: Improved the detection of malicious documents, adding new indicators and reducing the risk of document-based attacks.
- **Enhanced Emulation**: Increased emulation success rates, particularly through better recognition of file content types eligible for emulation.
- **Python Script Detection**: Improved detection of malicious Python scripts, a growing vector for attacks.
- **Better XOR Decryption**: Extended XOR decryption capabilities, improving analysis of encrypted malware.
- **Improved IOC Extraction**: Enhanced the extraction of indicators of compromise (IOC) from emulation for a more comprehensive report.
- **YARA Rule Updates**: Reviewed and vetted third-party YARA rules. By default, YARA rules are loaded with priority from the OPSWAT repository.
- **Symantec Quarantine Repair**: Implemented a repair function for files restored from Symantec quarantine, ensuring files can be analysed post-restoration.
- **MSC File Support**: Added the ability to identify and parse Microsoft Management Console (MSC) files, further broadening threat detection capabilities.
- **JPHP Support**: Enhanced malware detection with the ability to parse and decompile JPHP files, expanding the range of supported file types and languages.
- **.NET API Call Detection**: Added detection of unmanaged .NET API references, improving analysis of .NET-based malware.
- **OT Malware Detection**: Introduced a YARA ruleset specifically for OT (Operational Technology) malware, expanding protection to critical infrastructure systems.
- **LNK File Threat Indicators**: Strengthened detection for LNK icon smuggling and LNK-MOTW (Mark of the Web) bypass attacks, both common techniques in modern malware.
- **Ransomware Detection Enhancement**: Added severity Yara rule matches related to ransomware, helping to prioritize and respond to ransomware threats more effectively.

New Features and Improvements



v2.0.0

Release date: 08/28/2024

New Features and Improvements

- Upgraded to Java 17 and Python 3.10 for all relevant Sandbox components
- Support for AutoIT script files, including compiled AutoIT Portable Executables
- Parsing of MSI metadata and actions, including implementation for filtered file extraction
- Parsing of ODF files and macro extraction
- Parsing of Python pickle files, including implementation for malicious Threat Indicators
- Capability to identify potential obfuscation for extracted macro code
- New Threat Indicator for deceptive filenames commonly used for phishing files
- New Threat Indicator for undetected Equation Editor RTF exploit
- Enhanced parsing of LNK metadata and actions, including new Threat Indicators
- Improved Python-specific Threat Indicators
- Include proper tags for Golang, Rust and compiled-Python Portable Executables
- Improved processing for nested extracted files
- Enhanced Threat Indicators for imported APIs and emulation respectively

Fixes and improvements:

- Fixed minor bugs and misdetections
- Improved emulation efficacy
- Improved application security

v1.7.1

Release date: 05/24/2024

New Features and Improvements

- Ensured support for Ubuntu 22.04
- Added new threat indicators
- Disabled IP address OSINT lookups to avoid false positive findings
- Added verdict to IOCs on the UI
- Reduced false positive / false negative detection
- Updated YARA rule-set
- Fixed office file emulation errors



v1.7.0

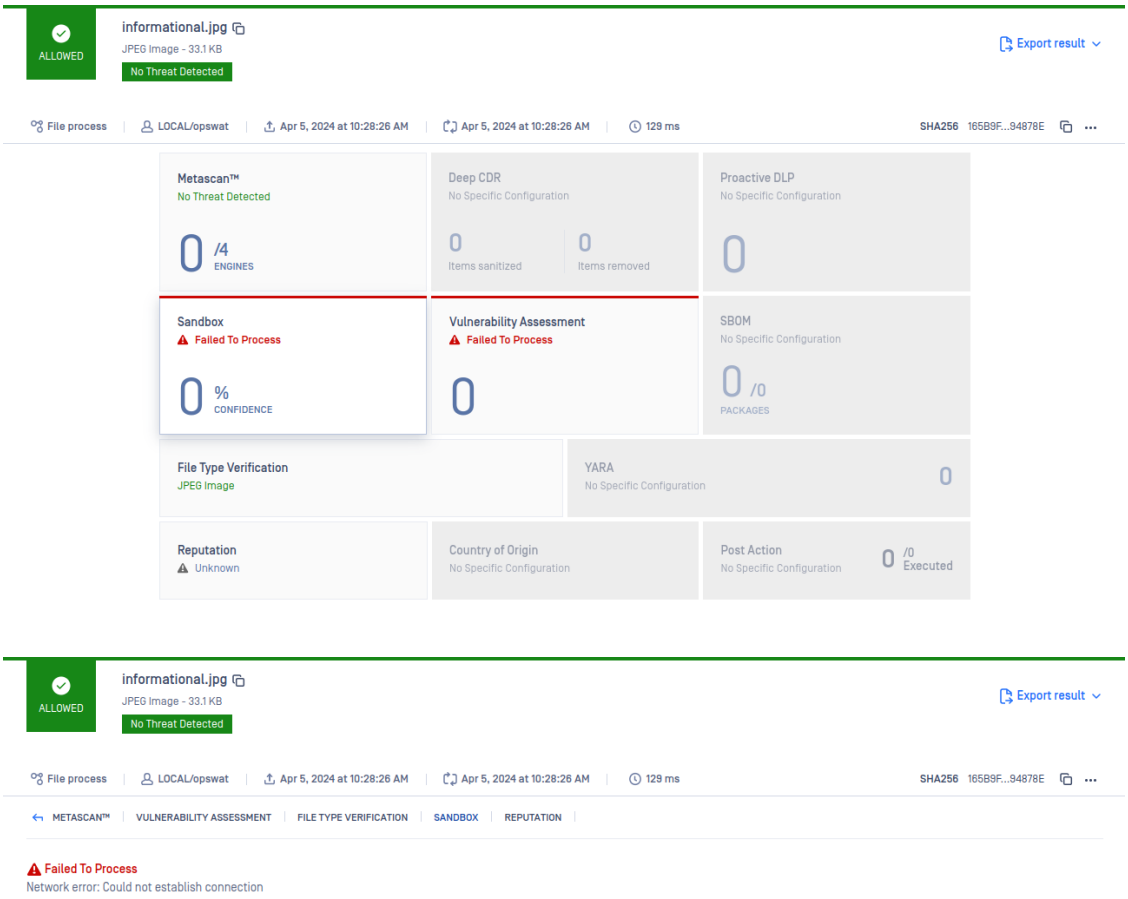
Release date: 04/26/2024

New Features and Improvements

- Malware config extraction support
- Python Unpacking & Decompilation for PyInstaller, Nuitka, and py2exe
- Improved error reporting
- Added long path support on Windows
- Added HTTP redirection support
- Included disassembly of exported functions for Windows binaries
- Threat indicator to flag when executable files have two different sections with the same section name
- Extraction of VBA macro code from DWG files (shown as OLE Stream in File Details section)
- Enhanced script language detection using the guesslang library
- Fine-tuned several threat indicators to reduce false positive ratio

New Features and Improvements

- Improved detection for phishing calendar invites
- Enhanced recursive analysis of active content containers (email, Office documents, PDF, etc.)
- Improved scan process for corrupt OLE2 documents
- Fixed several issues with existing threat indicators (ELF binaries, URL extraction, EML)



v1.6.0

Release date: 01/29/2024

New Features and Improvements

- Improved engine performance and stability
- Implemented configurable OPSWAT Reputation secret in engine global config
- New indicators for Windows APIs related to specific activities
- Implemented flagging for LSASS dump using minidump
- Extracted remote templates inside xTable struct in MS Office documents

New Features and Improvements

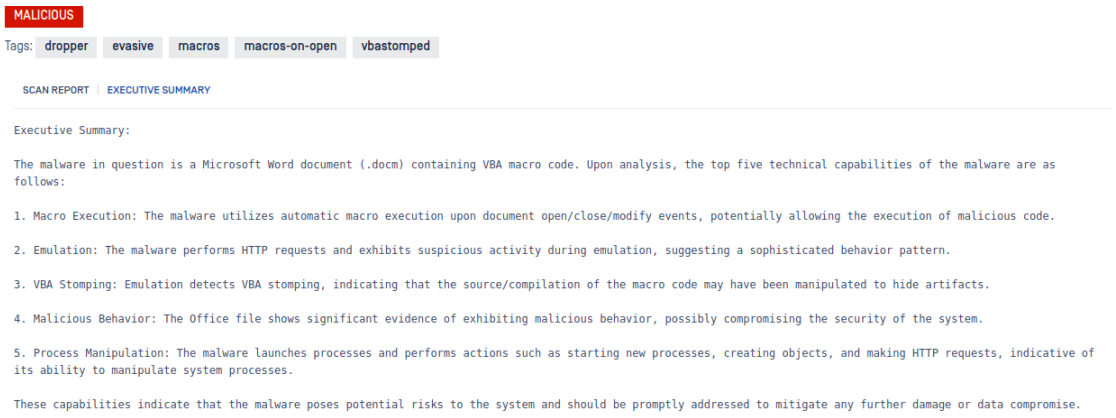
- Implemented parser for Debian packages
- Expanded malware configuration extractors to encompass the latest and most pertinent threats
- Improved detection of dynamic syscalls using the HellsGate bypass technique
- Enhanced Quishing and Phishing email detection
- Improved the capabilities of Batch, CSV, HTA, JavaScript, LNK, PowerShell, VBA, and VBScript emulation and fine-tuned timeout handling
- Fixed several UTF-8 parsing issues in content parsers (related to HTML & OLE files)
- Ensured that all whitelisted submissions get the Benign verdict
- Improved the stability of concurrent OSINT lookup tasks

v1.5.0

Release date: 11/06/2023

New Features and Improvements

- Updated Threat Indicators
- Improved office file emulation
- Improved PE file analysis
- Updated YARA rule-set
- Improved disassembly for x64 architecture
- Improved file type detection
- New IOC types for Crypto wallets
- New Executive Summary (ChatGPT report)



ETH Wallet	[redacted]
XMR Wallet	[redacted]

Known issues

- Crypto Wallets IOCs sometimes parsed and displayed incorrectly on the UI

v1.4.0

Release date: 09/22/2023

New Features and Improvements

- Support filenames with various Unicode characters
- Support unpacking of 64-bit executables
- Support malicious documents embedded in PDF files hidden as ActiveMime objects in MHTML format
- New threat indicators to detect the WikiLoader malware family (Microsoft Office files)
- Detection and extraction of embedded RTF files in Office documents, as described in CVE-2023-36884
- Enhance Threat Indicator for Mavinject
- Improved office file emulation
- Improved application security
- Improved large file processing

v1.3.4

Release date: 08/02/2023

New Features and Improvements

- Updated Threat Indicators
- Improved office file emulation
- Improved verdict calculation

v1.3.3

Release date: 07/07/2023

New Features and Improvements
• Fixed global config “reset to defaults” feature • Improved office file emulation • Updated YARA ruleset • Updated Threat Indicators • Updated verdict calculation

v1.3.2

Release date: 06/14/2023

New Features and Improvements
• Updated logging for MetaDefender Core support package • Improved handling of embedded JavaScript files

v1.3.1

Release date: 06/05/2023

New Features and Improvements
• Enabled XML file support by default • Updated reputation sources • Improved verdict calculation • Fixed global config reset to default values feature • Fixed report generation for files including Email IOCs

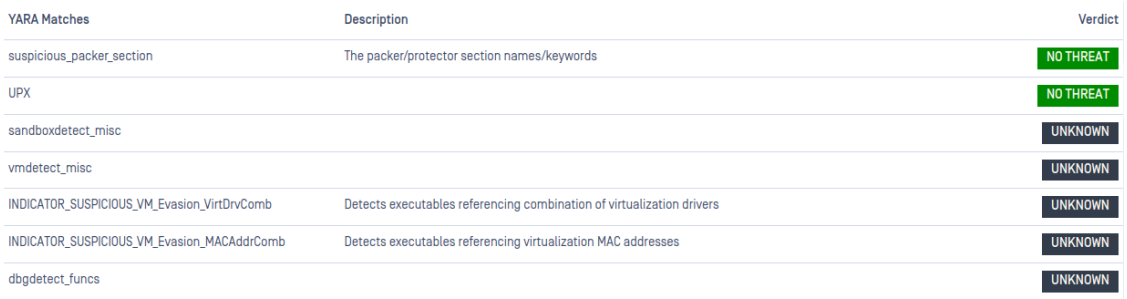
v1.3.0

Release date: 05/26/2023

New Features and Improvements
• Updated YARA rule database

New Features and Improvements

- YARA matches displayed on MDCore UI
- Dependency check on startup



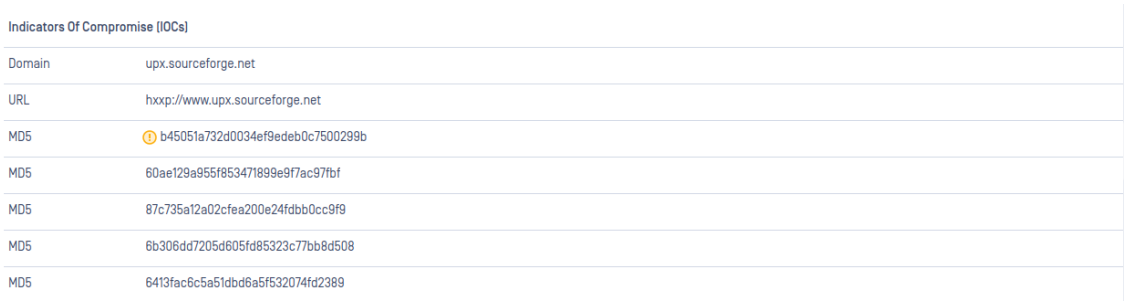
YARA Matches	Description	Verdict
suspicious_packer_section	The packer/protector section names/keywords	NO THREAT
UPX		NO THREAT
sandboxdetect_misc		UNKNOWN
vmdetect_misc		UNKNOWN
INDICATOR_SUSPICIOUS_VM_Evasion_VirtDrvComb	Detects executables referencing combination of virtualization drivers	UNKNOWN
INDICATOR_SUSPICIOUS_VM_Evasion_MACAddrComb	Detects executables referencing virtualization MAC addresses	UNKNOWN
dbgdetect_funcs		UNKNOWN

v1.2.0

Release date: 05/17/2023

New Features and Improvements

- Scan results are extended with the list of IOCs
- Rapid mode support added and enabled by default
- Reputation lookup support added and enabled by default
- Reputation lookup verdict improvements
- Improved embedded engine performance



Indicators Of Compromise (IOCs)	
Domain	upx.sourceforge.net
URL	hxxp://www.upx.sourceforge.net
MD5	b45051a732d0034ef9edeb0c7500299b
MD5	60ae129a955f853471899e9f7ac97fbf
MD5	87c735a12a02cfea200e24fdbb0cc9f9
MD5	6b306dd7205d605fd85323c77bb8d508
MD5	6413fac6c5a51dbd6a5f532074fd2389

v1.1.0

Release date: 05/08/2023

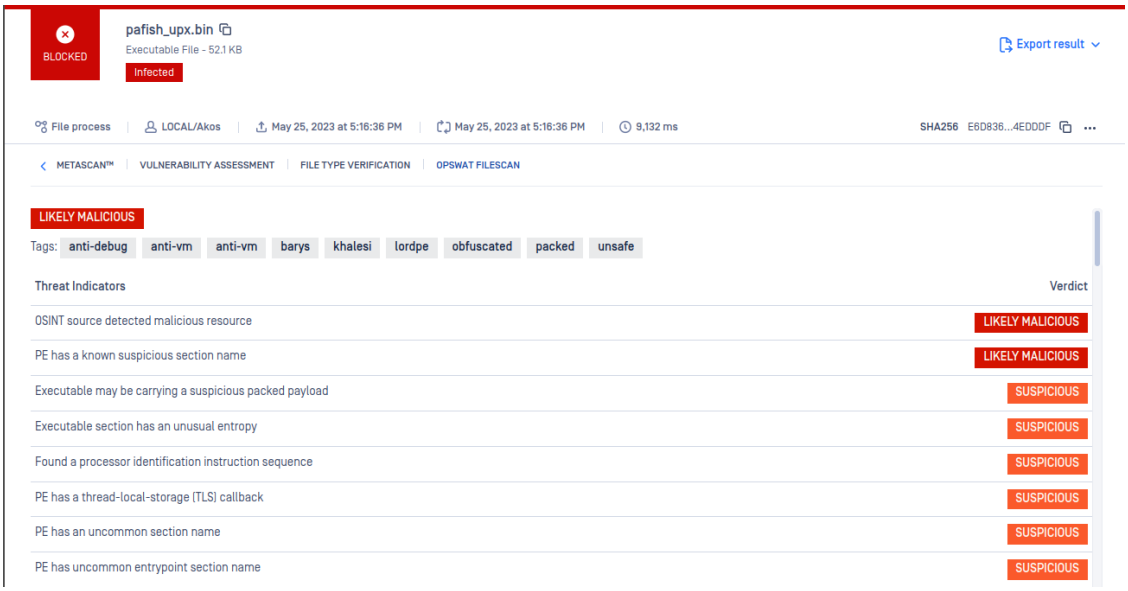
New Features and Improvements	
• Improved Microsoft Office file handling	

New Features and Improvements	
• Security and performance improvements	

v1.0.0

Release date: 04/06/2023

New Features and Improvements
• First versions of Embedded and Remote engines for MetaDefender Core customers



Remote Engine

Pro Tip

Keep your MetaDefender Aether (previously know MetaDefender Sandbox) standalone instance up to date for the latest features. See the release notes for more information.

V3.3.1

Release date: 13/10/2026

New Features and Improvements
• Fixed configuration issues with No Threat Detected verdict

New Features and Improvements

v3.3.0

Release date: 10/10/2026

New Features and Improvements
• **LLM Scanning**Submit a model to MetaDefender Aether in the same way you submit any other file. Aether runs two analyses: static analysis of the model file and dynamic analysis of the running model. If Aether finds a threat, Core blocks the file.**Static analysis**Aether never loads or runs the model. It reads only the parts of the file that can carry code or configuration and skips over the weights, so memory use stays low even for very large models.Any payload Aether finds inside the file, including data hidden between tensors, is extracted and analysed on its own.**Dynamic analysis**Aether connects to your inference server through its OpenAI-compatible API and runs 38 attack probes against the live model, covering 9 of the OWASP Top 10 for LLM Applications categories. Probes run in order of severity. Each scan plants single-use canary tokens, so if one appears in a response, the leak is confirmed. Every finding shows the prompt, the model's response, the severity, the OWASP category and the rule that flagged it.Static and dynamic analysis are separate options, so you can run either one or both. • **More Evidence from emulation** • **HTTP traffic**: Emulation now records the full request and response for each HTTP call, including headers and body. • **Delayed macros:** VBA emulation now supports Application.OnTime, which malware uses to delay its payload. • **Deeper unpacking** • **Installers**: Better extraction from nested installers and more installer types, including PyInstaller, py2exe, ps2exe and Nuitka files. • **Extended YARA triggers**Adaptive Sandbox scan can be triggered for custom YARA rules

v3.2.0

Release date: 31/07/2026

New Features and Improvements
• **Implemented instrumented file system within the PE emulator:** The PE emulator now instanciates an instrumented filesystem in which the emulated sample resides. A great step towards the first official version of Filescan's PE emulator. This feature enables proper emulation of any file operation and interaction which would have previously terminated the emulation. This feature brings the flexibility of manually configuring the fake files that "exist" within, so that the emulated sample actually locates any file defined in the configuration

New Features and Improvements

file. Hence, this milestone not only enables a new feature, but extends configuration and flexibility of the emulation environment.

- **Detection for Emerging Evasion Techniques:** Threat actors keep getting more creative — Sandbox keeps pace. This release adds detection for *etherhiding* (concealing threat actor infrastructure inside blockchain transactions) and intentionally malformed .NET assemblies built to break automated analysis tools.
- **Detection for Emerging Evasion Techniques:** Threat actors keep getting more creative — Sandbox keeps pace. This release adds detection for *etherhiding* (concealing threat actor infrastructure inside blockchain transactions) and intentionally malformed .NET assemblies built to break automated analysis tools.
- **Smarter Remote Template Detection:** Remote template injection analysis got a full pass this release, aimed at keeping the false-positive rate flat while closing real detection gaps. A rebuilt link-detection and URL-classification pipeline now tells genuine injection attempts apart from ordinary embedded links — fixing two indicators that had been flagging legitimate URLs as malicious, and closing a gap where obfuscated, octal-encoded malicious IP links were going undetected. Fewer false alarms, no loss of coverage.
- **Broader Detection & Coverage Improvements:** This release also expands detection coverage on several fronts: new detection engineering for wiper malware, CAPTCHA identification via OCR (including obfuscated but rendered fake CAPTCHAs used in ClickFix attacks), new config extractors for PrivateLoader and Quasar RAT, and expanded IOC coverage in the local reputation database.

v3.1.0

Release date: 31/04/2026

New Features and Improvements

What's new

- **Expanded archive support:** Multiple archive types are now supported with Core 18.1.
- **Deep CDR hyperlink trigger:** The hyperlink Deep CDR type can now be used to trigger Sandbox analysis.
- **TLS certificate verification:** Added an option to enable TLS certificate validation using the operating system's certificate store.

Bug Fixes

- **Improved URL handling:** Fixed URL parsing when the Load Balancer feature is enabled.

v3.0.0

Release date 26/01/2026

New Features and Improvements

We are excited to announce **MetaDefender Aether 3.0.0**, OPSWAT's next-generation unified zero-day detection solution. With this release, the proven **MetaDefender Sandbox dynamic analysis engine is now fully integrated into MetaDefender Aether as Layer 2**, strengthening Aether's multi-layered detection architecture and delivering deeper, faster, and more scalable threat analysis at the perimeter.

Improvements

- **E-PUB support** - Added support for EPUB files, to ensure broader compatibility with widely used e-reading formats.
- **Verdict Renaming** – Updated verdicts to provide clearer guidance on how to interpret analysis results and take appropriate actions based on potential threat levels

Bug Fixes

- **Duplicate Tags** - Tags originating from multiple sources were being shown repeatedly in the UI. The fix ensures that tags are properly deduplicated so each one appears only once.

v2.5.1

Release date 11/20/2025

New Features and Improvements

Improvements

- **Stability & Security -** Updated core dependencies to enhance performance and security.

Bug Fixes

- **Fixed zlib support** – zlib files are now correctly processed by the Archive Engine instead of the Adaptive Sandbox, ensuring proper handling and improved efficiency.

v2.5.0

Release date: 10/30/2025

New Features and Improvements

What's New

- **Full Report Downloading** – Introduced the ability to download generated reports from the Remote Engine directly via the Core UI.

v2.4.0

Release date: 07/31/2025

New Features and Improvements

What's New

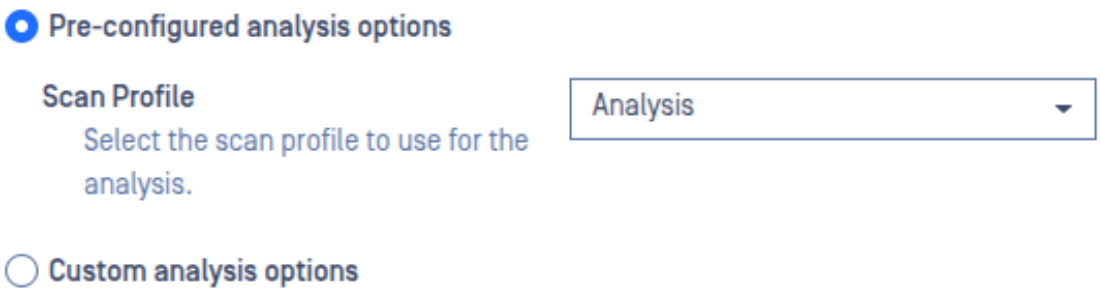
- **Load Balancing / Failover**: Implemented Round Robin load balancing and failover to ensure even traffic distribution and high availability during node failures.

v2.3.0

Release date: 06/05/2025

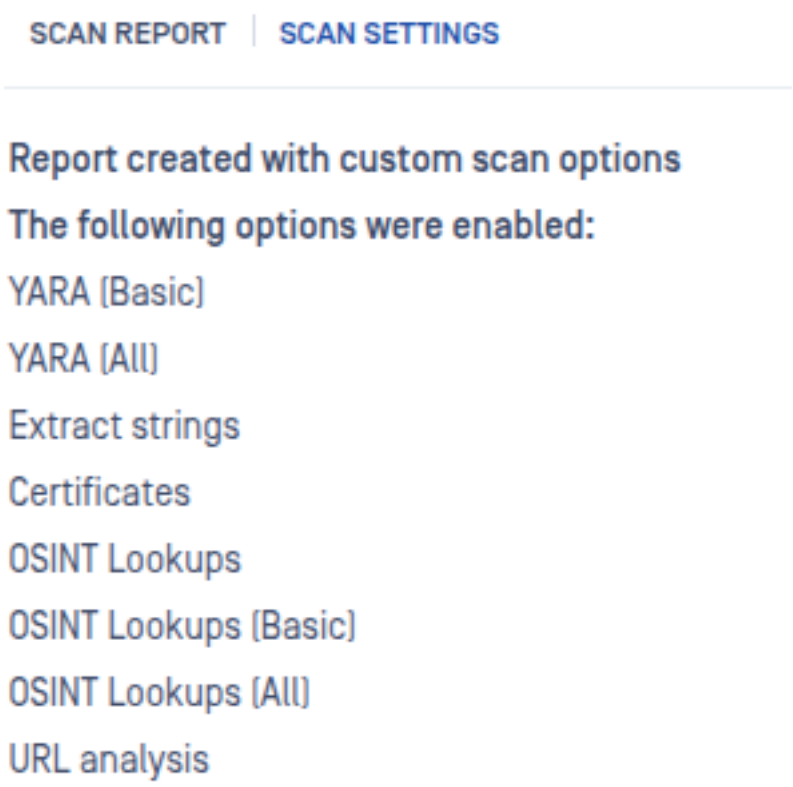
New Features and Improvements

- **Scan Execution Profiles**: Users can now select from predefined scan profiles or create custom profiles, enabling a more streamlined and flexible scanning experience tailored to specific needs.



- **YARA result triggering:** Sandbox scans can now be automatically initiated based on curated rules embedded within the Adaptive Sandbox's YARA engine, allowing for intelligent, rule-driven threat escalation. YARA Trigger
- **Enhanced file type support:** The file type selector has been updated to more accurately reflect the full range of formats supported by the Adaptive Sandbox engine, improving usability and clarity.
- **Result updated with scan preferences:** Selected scan settings are now displayed in the Result tab, making it easier to troubleshoot and understand how each scan was configured and executed.

New Features and Improvements



v2.2.0

Release date: 12/02/2024

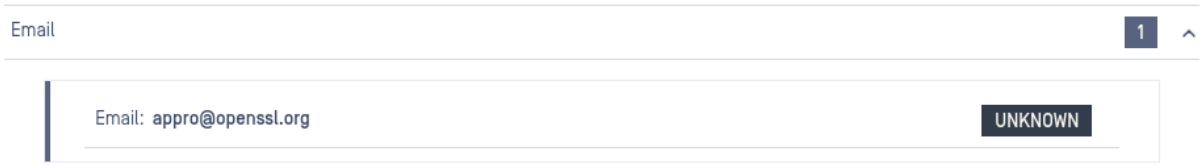
New Features and Improvements

- **Threat Indicator details**: Report is extended with Threat Indicators details, providing better and easier assessment of the file.



- **Updated IOC details**: Improved IOC details UI

New Features and Improvements



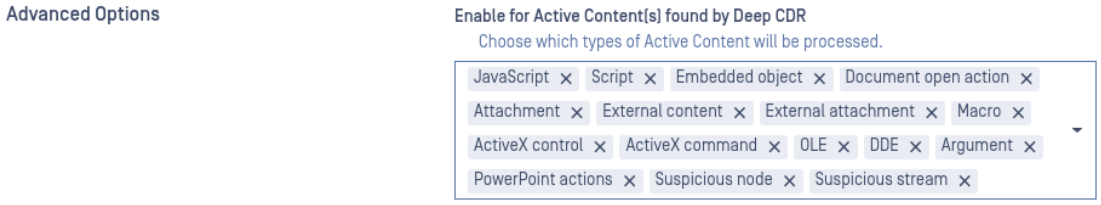
- **Dependency handling**: Visual C++ Redistributable is no longer required, making it easier to install and update the engine on the fly.

v2.1.0

Release date: 12/02/2024

New Features and Improvements

- **Min-Max Analysis timeout changed:** Minimum and Maximum Analysis Timeout changed to 60-86400, for better handling of Large Files.
- **.Dll files are signed:** Windows .Dll files are now signed for better security
- **Enhanced Archive handling:** Archive File handling improved to reduce Scan times.
- **Deep CDR Triggers options added:** New Deep CDR Trigger option is added to Workflow configs for a more granular setup option.
- **Improved Error Handling:** More Errors messages are available for easier problem solving.



v2.0.1 - Windows

Release date: 09/27/2024

New Features and Improvements

- Fixed dependency check for Visual Studio Redistributable on Windows

v2.0.0

Release date: 07/23/2024

Backward compatibility

Sandbox standalone version 2.0.0 is not backward compatible with older engine versions

New Features and Improvements

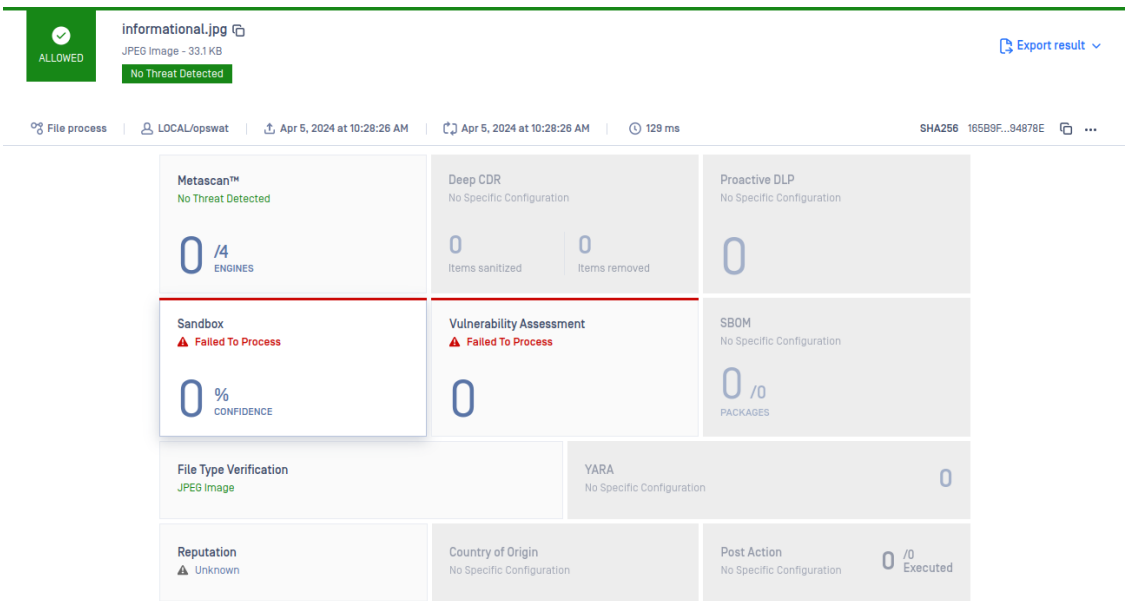
- Ensured support for Ubuntu 22.04
- Added verdict to IOCs on the UI
- Sandbox standalone version 2.0.0 compatibility

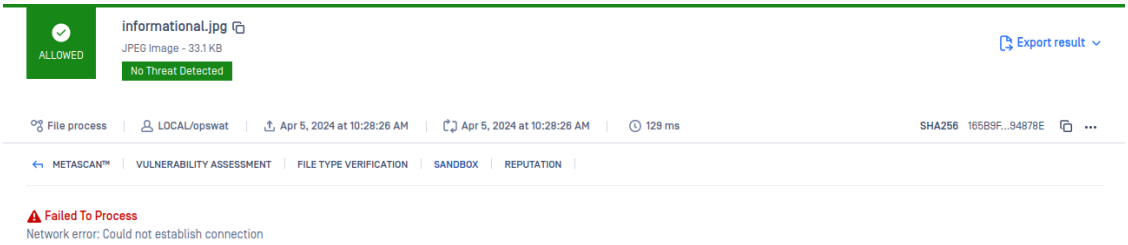
v1.7.0

Release date: 04/26/2024

New Features and Improvements

- Improved error reporting
- Added HTTP redirection support





v1.6.0

Release date: 01/29/2024

New Features and Improvements

- Improved engine performance and stability

v1.5.0

Release date: 11/06/2023

New Features and Improvements

- New IOC types for Crypto wallets



Known issues

- Crypto Wallet IOCs will not be displayed on the UI until a new Filescan Standalone version is released

v1.3.3

Release date: 07/07/2023

New Features and Improvements

- Fixed global config “reset to defaults” feature

v1.3.2

Release date: 06/14/2023

New Features and Improvements
• Updated logging for MetaDefender Core support package

v1.3.1

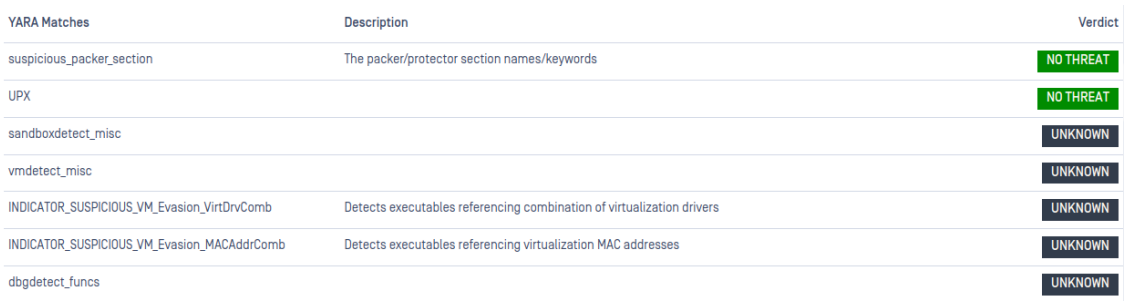
Release date: 06/05/2023

New Features and Improvements
• Fixed global config reset to default values feature • Fixed report generation for files including Email IOCs

v1.3.0

Release date: 05/26/2023

New Features and Improvements
• YARA matches displayed on MDCore UI



YARA Matches	Description	Verdict
suspicious_packer_section	The packer/protector section names/keywords	NO THREAT
UPX		NO THREAT
sandboxdetect_misc		UNKNOWN
vmdetect_misc		UNKNOWN
INDICATOR_SUSPICIOUS_VM_Evasion_VirtDrvComb	Detects executables referencing combination of virtualization drivers	UNKNOWN
INDICATOR_SUSPICIOUS_VM_Evasion_MACAddrComb	Detects executables referencing virtualization MAC addresses	UNKNOWN
dbgdetect_funcs		UNKNOWN

v1.2.0

Release date: 05/17/2023

New Features and Improvements
• Scan results are extended with the list of IOCs

New Features and Improvements

- Private scanning option added, enabled by default
- Reputation lookup support added and enabled by default
- Reputation lookup verdict improvements



Indicators Of Compromise (IOCs)	
Domain	upx.sourceforge.net
URL	hxxp://www.upx.sourceforge.net
MD5	b45051a732d0034ef9edeb0c7500299b
MD5	60ae129a955f853471899e9f7ac97fbf
MD5	87c735a12a02cfea200e24fdbb0cc9f9
MD5	6b306dd7205d605fd85323c77bb8d508
MD5	6413fac6c5a51dbd6a5f532074fd2389

v1.1.0

Release date: 05/08/2023

New Features and Improvements

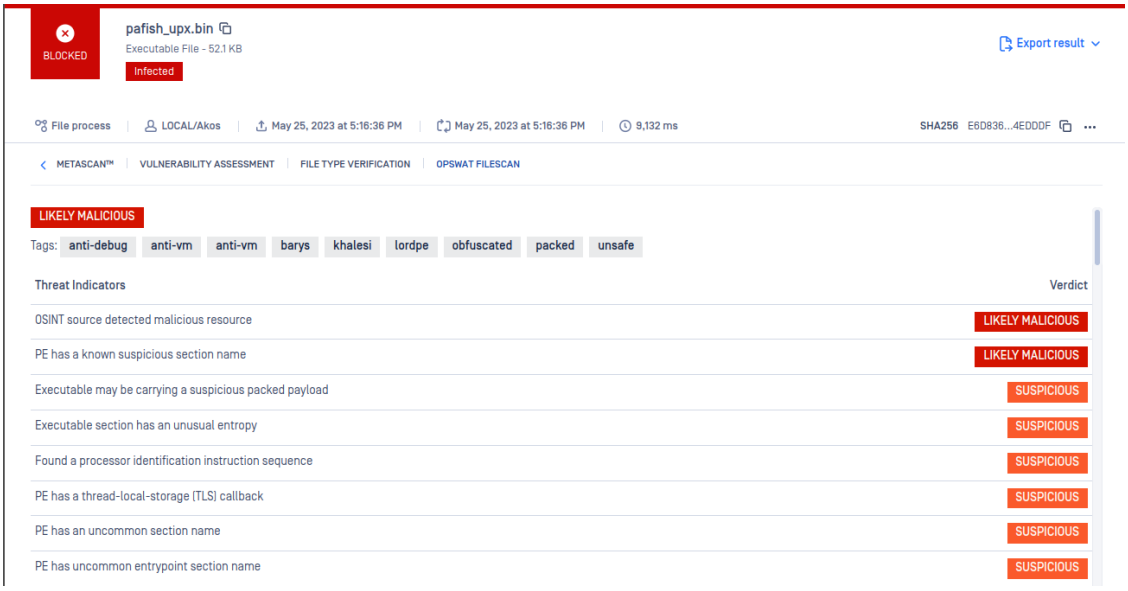
- Improved Microsoft Office file handling
- Security and performance improvements

v1.0.0

Release date: 04/06/2023

New Features and Improvements

- First versions of Embedded and Remote engines for MetaDefender Core customers



Reputation Engine release notes

2.2.0

Release date: 12/1/2025

New Features and Improvements
What's New • **NSRL Allowlist Support:** The Reputation Engine now includes an allowlist of hashes sourced from the National Software Reference Library (NSRL). These hashes have been carefully filtered to ensure accuracy and reliability. • **Third-Party Allowlist Support:** The Reputation Engine now incorporates whitelisted hashes from trusted third-party vendors. All entries are curated in-house to maintain integrity and security.

2.1.2

Release date: 11/10/2024

New Features and Improvements
.DLL files are signed
Improved dependency checking

2.1.1

Release date: 04/09/2024

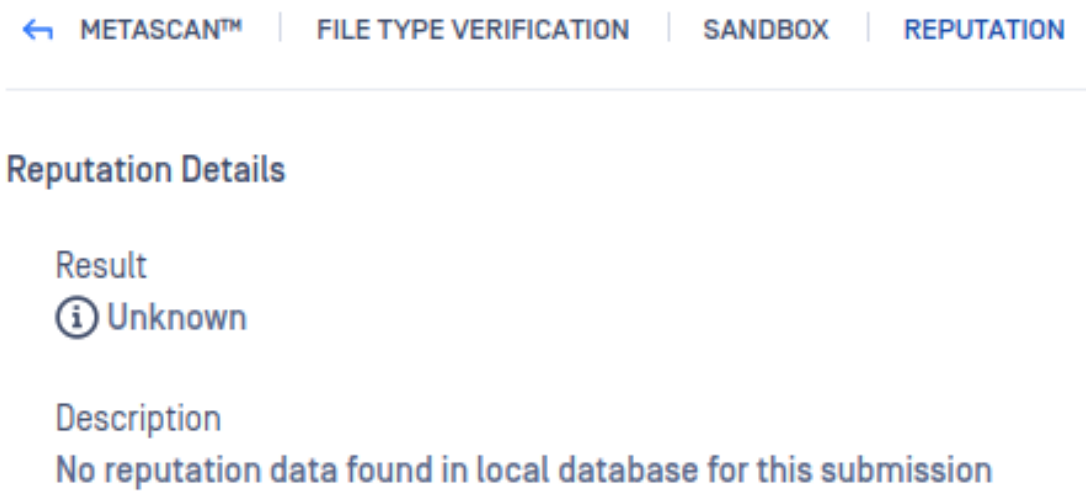
New Features and Improvements
Fixed an issue which caused the engine to crash on some Window devices
Improved application security
Added dependency checking for Visual Studio Redistributable package



2.1.0

Release date: 05/06/2024

Initial version
Added SHA256 support
Updated verdict with description
Fixed global configuration appearance on the UI



2.0.2

Release date: 19/03/2024

Initial version
For detailed product information please see the Overview product page.

Country of Origin release notes

v2.4.0

Release date: 9/23/2026

- Supported signed APK, VSIX, JAR, and DEB file formats.

v2.3.1

Release date: 7/22/2026

- Upgrade OpenSSL to 3.6.3.

v2.3.0

Release date: 7/2/2026

- Fixed an issue where the "Verify Digital Certificate" setting could revert to its default value under certain conditions.
- Upgrade Windows Compiler from VS2019 to VS2022.

v2.2.1

Release date: 5/20/2026

- Added support Windows 11 24H2, 25H2, Rocky Linux 10.

v2.2.0

Release date: 3/25/2026

- Supported Inno Setup Installer.
- Upgraded OpenSSL to version 3.6.1.
- Added support for Debian 13.

v2.1.0

Release date: 9/24/2025

- Do not return "Allow" for Country filter.

v2.0.1

Release date: 8/6/2025

- Support CAB file type.
- Support digital signature validation for CAB, EFI, EXE SFX file format.
- Block files in case of failing integrity check.

v2.0

Release date: 7/7/2025

- Support Ubuntu 22.04 CIS level 2.
- Validate the file integrity against the digital signature.

v1.2.0

Release date: 6/25/2025

- Allow user to input Trusted root certificates to validate against a digital signature.
- Drop support CentOS 7/RH 7/Debian 10/Ubuntu 20.

v1.2.1

Release date: 12/1/2024

- Digitally sign all binary files.

v1.0

Release date: 3/20/2024

- Support block/allow by countries.
- Support block/allow by vendor.
- Support file types: DLL, EXE, EXE_SFX, EFI, and MSI.

Threat Intelligence engine release notes

v1.0.0

Release date: 26/01/2026

New Features and Improvements
First versions of Embedded and Remote engines for MetaDefender Core customers

Predictive Alin AI release notes

V 3.6.0

Release date: 30 September 2026

- Added concurrent scanning, significantly increasing throughput on multi-core systems with no change to verdicts, with the largest gains on Android APK files.
- Added a configurable limit on simultaneous resource-intensive scans to keep memory usage stable under heavy load.
- Added a unique rule identifier to rule-based, signature and reputation detections, shown in the reported threat name.
- Updated the detection database format. This engine version requires a matching database version, and incompatible databases are now rejected cleanly at startup.
- Improved startup performance, with lower memory usage, faster startup and a significantly smaller detection database.
- Improved scan time on very large files while maintaining full detection coverage.
- Improved resilience against malformed files, eliminating hangs and crashes and reducing memory usage when

processing them.

- Improved PDF detection accuracy.
- Fixed a crash when scanning certain malformed Windows executables, which left the file without a verdict.
- Fixed a detection gap on certain Windows executables carrying multiple digital signatures.
- Fixed validly signed large Windows executables being treated as unsigned.
- Fixed a lost verdict when certain files containing malicious embedded content were rescanned.
- Fixed false positives on legitimate legacy Windows applications, certain PDF files and other files identified in the field.

V 3.5.0

Release date: 31 August 2026

- Added analysis and detection for additional image formats (JP2, EXR, PSD, PCX, TGA, DDS), covering common image-based attack techniques, initially in monitoring mode.
- Added recognition of archive and container formats so they are handled correctly:
  - 7Z, RAR, GZ, BZ2, XZ, CAB, PKG/XAR, RPM, DEB, TAR, DMG and ZIP
  - ODT, ODS, ODP, ODG, OTT, OTS, OTP, EPUB, APPX, MSIX, ASICE, XPS, OXPS, HWPX, PBIX, PBIVIZ, THMX, VSDX, VSTX and TDSX
- Added recognition of legacy Visio (VSD, VST, VSS), Publisher (PUB) and Project (MPP) files, and distinguished Word templates (DOT) from documents.
- Added malicious macro detection for Excel binary workbooks (XLSB), matching existing coverage for other macro-enabled Excel formats.
- Added deeper analysis of .NET applications, including detection of obfuscated or packed files.
- Added HTML file detection, including analysis of embedded content.
- Added optional detection of hidden data appended to PNG and JPEG images, disabled by default.
- Enabled machine-learning detection for Office documents (DOCX, XLSX, PPTX).
- Updated the script detection model.
- Improved Android APK analysis, including more reliable handling of malformed or obfuscated packages.
- Improved performance when running multiple scans in parallel.
- Improved XML file detection, including support for additional encodings and formats.
- Improved performance of repeated scans of the same file.
- Fixed scan failures on certain PDF files, such as those produced by print or mail systems.
- Fixed false positives on plain-text emails containing PDF-like text.
- Fixed Java class files being misidentified as macOS executables.
- Fixed inconsistent threat names on some detections.

- Fixed false positives and slow scans on certain large Microsoft installers and redistributables.
- Fixed false positives on large animated GIF images.
- Fixed an issue where scanned files could remain locked after scanning, which could prevent drives from being unmounted.

V 3.4.3

Release date: 17 August 2026

- Added detection of malicious macros in Office documents (DOCM, XLSM, PPTM) and legacy Office files.
- Added detection of malicious scripts embedded in PDF documents.
- Added scanning of Outlook MSG email bodies for malicious content.
- Added detection of malicious scripts embedded in SVG files.
- Added support for macro-enabled and template PowerPoint files:
  - PPTM, POTM, PPSM, PPAM
- Added reputation-based detection of known malicious and phishing URLs, IP addresses and domains.
- Added new detection rules for malicious HTML content.
- Added configurable control over which files extracted from containers are deep-scanned.
- Updated detection models for Android APK and PDF files with improved accuracy.
- Improved engine startup time and memory usage.

V 3.4.2

Release date: 13 August 2026

- Added support for scanning Android APK files, alongside existing support for Windows, Linux, PDF and Office files.
- Added detection of malicious content in EML email bodies, along with support for scanning standalone script and text files.
- Improved protection of the detection database, which is now encrypted at rest.
- Updated the detection database.
- Fixed an issue where macro-based detection did not apply to macro-enabled Word documents (DOCM and DOTM).

V 3.4.1

Release date: 06 August 2026

- Added detection of malicious content hidden inside image files, targeting image-poisoning campaigns.

- Added malicious macro detection for Excel files:
  - XLSM, XLTX, XLTM, XLAM
- Added new behavioral detection rules for Windows executables.
- Improved accuracy on Office documents by reducing false positives on legitimate files.
- Fixed a crash that could occur when the engine shut down.
- Fixed an issue in version 3.4.0 where macOS executables received no verdict.
- Fixed an issue where detection database packages for versions 3.4.0 and 3.4.1 could not be uploaded to MetaDefender Core v5.19.0 and v5.20.0.

V 3.4.0

Release date: 03 August 2026

- Enabled signature, heuristic and rule-based detection for Windows, Linux and macOS executables, PDF, Office and image files.
- Added optional deep inspection of Windows, Linux and macOS executables, disabled by default with no performance impact when off.
- Added scanning support for macOS archive containers that were previously not scanned.
- Added detection of malicious scripts hidden in image metadata.
- Fixed certain valid BMP images being rejected as an unknown file type.
- Fixed Java archive (.jar) files that cannot be extracted, including encrypted ones, being reported as malicious; they are now reported as unsupported.

V 3.3.3

Release date: 28 June 2026

- Added content-based video file recognition.
  - Supported formats: MP4, AVI, MKV, MOV, WebM, FLV, WMV

V 3.3.2

Release date: 21 June 2026

- Added recognition of audio files.
- Fixed an engine startup failure on MetaDefender Core v5.18.0 in certain virtualized environments.

V 3.3.1

Release date: 16 June 2026

- Improved file type recognition for Adobe Illustrator (.ai) files.
- Improved scanning performance for Office documents.
- Fixed an issue where reputation-based detection failed to start on Linux.

V 3.3.0

Release date: 9 June 2026

- Added multi-layered detection for image files used as attack vectors.
  - Machine-learning detection for images is enabled in Deflection mode and disabled in Detection mode.
- Added detection of suspicious, malformed or malicious Office documents (DOCX, XLSX, PPTX).
- Added analysis of Android APK files to support machine-learning detection.

V 3.2.3

Release date: 27 May 2026

- Added support for legacy Office files (.doc, .xls, .ppt) and Outlook messages (.msg), including those containing macros, with safe handling of malformed files.

V 3.2.2

Release date: 20 May 2026

- Added analysis of links found in PDF, Office and EML files.
- Fixed file type recognition failures on certain macOS executables.
- Fixed an issue where scanned files remained locked for several minutes after scanning, preventing drives from being unmounted.

V 3.2.1

Release date: 11 May 2026

- Added more precise recognition of executable types, so installers, firmware and similar files are routed to the appropriate engines instead of being scanned as standard executables.
- Added support for additional Linux file types.
- Updated detection capabilities for Windows, Linux, macOS and PDF files.
- Removed Office and image files from detection.
- Fixed an issue where scanning certain Linux files could exhaust system memory.
- Fixed a scan-time slowdown when scanning SD cards over USB on MetaDefender Smart Touch.

V 3.2.0

Release date: 04 May 2026

- Added recognition of email (EML) files so they are analyzed instead of skipped.
- Added recognition of legacy Office documents (DOC, XLS, PPT) and Outlook messages (.msg).
- Added recognition of Android APK files.
- Added image scanning with settings configurable per scanning profile.
  - Supported formats: JPG/JPEG, PNG, WEBP, GIF
- Added per-document-type settings for DOCX, XLSX and PPTX files.
- Added analysis of links embedded in files.
- Added support for older CPUs on Linux and Windows, selected automatically at startup.
- Added support for 32-bit deployments.
- Improved compatibility with older Linux distributions.
- Improved performance under high parallel scan load.
- Fixed a false positive where a third-party antivirus flagged a file in the engine package.

V 3.1.4

Release date: 27 April 2026

- Fixed a false positive on a legitimate Windows utility bundled with third-party industrial software.

V 3.1.3

Release date: 14 April 2026

- Fixed a scan hang and high memory usage on certain malformed PDF files.
- Fixed false positives on certain legitimate Windows and macOS files, including installers and developer tools.

V 3.0.1

Release date: 31 March 2026

- Fixed files not being scanned when their paths contained forward slashes.
- Fixed an engine crash during batch scanning.
- Improved detection reliability for Windows executables.

V 3.0.0

Release date: 23 March 2026

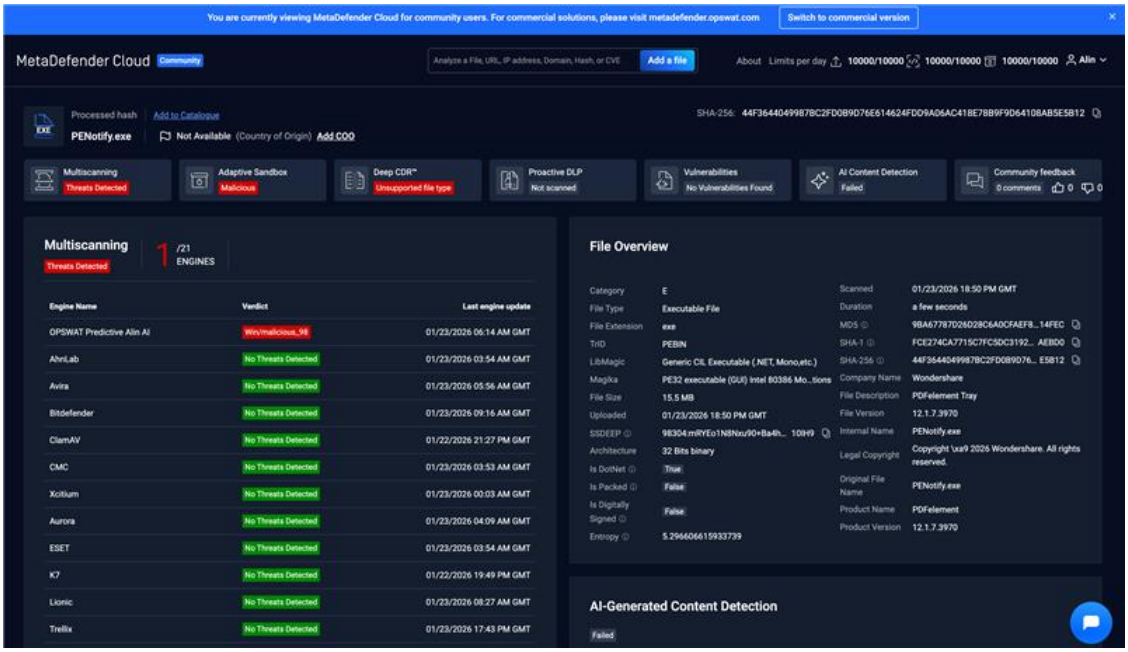
Release type: MAJOR

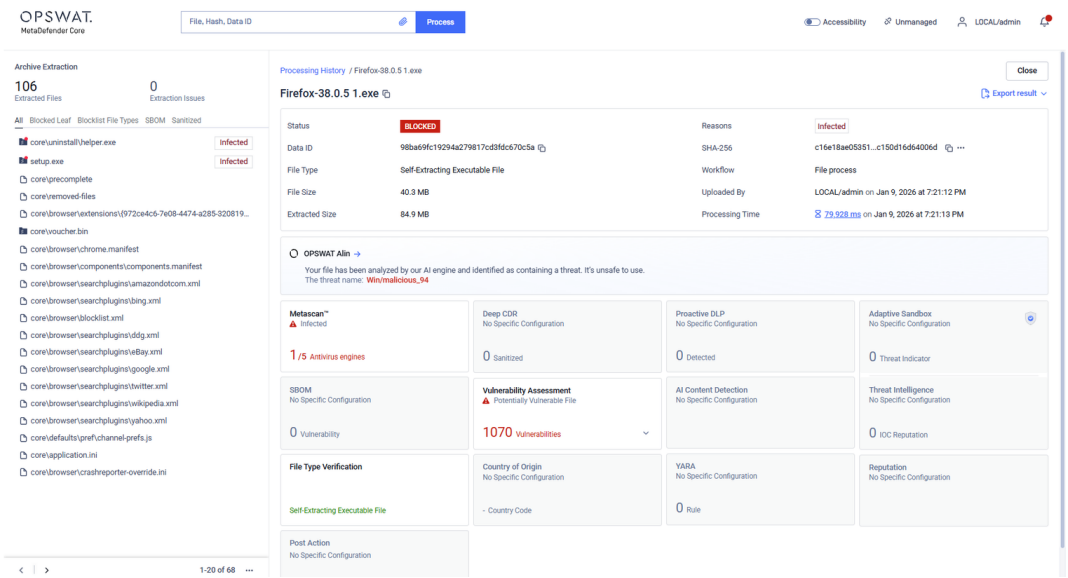
- General Access release of Predictive Alin AI for customer-facing enterprise use.
- Reputation integration to improve deflection decisions for known good and known bad files.
- File type detection optimized for malware screening workflows across:
  - Executables: PE, ELF, Mach-O
  - Documents: PDF
- File parsing for PE, ELF, Mach-O, and PDF to extract structural signals used as context for ML inference.
- Integration availability:
  - **MetaDefender Cloud** - Predictive Alin AI runs inline as part of the MetaScan package.
  - MetaDefender Core - Predictive Alin AI runs inline as part of the MetaScan package on Linux and Windows.
- Standardized engine verdict codes: Reference: https://www.opswat.com/docs/mdcloud/integrations/description-on-scan-result-codes
  - 0 - No Threat Detected
  - 1 - Infected: The file was analyzed and determined to be malicious.
  - 10 - Not Scanned: The file was not analyzed (for example, due to size limits or an internal processing error).
  - 23 - Unsupported File Format: The file type is not currently supported by the engine.
- Threat detection naming convention for policy integration:
  - Format: <platform>/malicious_<threat_confidence>
  - Platform mapping:
    - Windows - PE-based executables
    - Linux - ELF-based binaries
    - Darwin - Mach-O-based binaries
    - Generic - non-platform-specific formats such as PDF
  - Examples:
    - PE: Windows/malicious_99
    - ELF: Unix/malicious_99
    - Mach-O: MacOS/malicious_99
    - PDF: Generic/malicious_99

Info

Known considerations

- This is the first General Access release. Establish an initial operational baseline before tightening policy.
- Malicious naming outputs are standardized for policy integration and should be interpreted alongside the broader security context.





Introduction

Predictive Alin AI 3.0.0 marks the **General Access release** of OPSWAT's proprietary machine learning-driven malware detection engine, purpose-built for **high-speed, pre-execution protection** in modern, high-throughput environments.

What This Means for MetaDefender Customers?

- **Predictive Alin AI** is fully integrated into the MetaDefender platform and accessible across all products and

environments, available in both **cloud and on-premises environments**, supporting **Windows and Linux** operating systems

- **Predictive Alin AI** operates in parallel to MetaScan to enhance detection coverage via machine learning algorithms enhanced by MetaDefender Aether's dynamic analysis, for reduced false positives and zero-day detection
- **Predictive Alin AI** targets the primary malware entry points, with extensive support for executable formats and PDF files, where most modern threats originate.

How Is Predictive Alin AI Continuously Improving?

Predictive Alin AI has a documented Machine Learning pipeline lifecycle concept for transparency, including:

- Data ingestion and normalization controls (hashing, deduplication, validation)
- Labeling and enrichment through MetaDefender Aether using multi-source evidence (static characteristics, dynamic analysis context, consensus-oriented verdict context, and threat intelligence enrichment)
- Training, evaluation, and quality gating against false-positive benchmarks before promotion
- Continuous refresh and monitoring to adapt to malware evolution

Key Benefits

- Fast, pre-execution malware detection (P99 under 100ms) to enable allow, block, and quarantine decisions at scale.
- Faster scan times for high-volume workflows while improving accuracy and reducing false positives (0.1% False Positive Rate and over 90% accurcacy across the board)
- Extensive support for executable formats (PE, ELF, Mach-O) plus PDF for common document-borne threats.
- Available as a custom engine in MetaDefender Cloud and MetaDefender Core.

Next Steps to Upgrade:

Customers currently using MetaDefender products can add Predictive Alin AI to unlock high-speed, machine learning–based detection for pre-execution threats while retaining familiar multiscanning capabilities. To discuss upgrade options or transition planning, please contact your OPSWAT representative.

Predictive Alin AI reflects where we're heading with pre-execution malware detection: faster decisions, better accuracy, and the ability to handle high-volume environments without added complexity.

Compatibility

- Platforms: Windows, Linux

OPSWAT AI Content Inspector release notes

V 2.4.0

Release date: 11 October 2026Release type: MINOR

AI Content Inspector 2.4.0 helps teams make more confident decisions about AI-generated content by distinguishing verified content credentials from unreliable labels and detecting AI use recorded in earlier file editing steps. Offline validation of C2PA Content Credentials supports provenance checks without internet access, including for images inside Office documents and PDFs. HTML support extends detection coverage, faster scanning helps teams handle higher volumes, and consistent text scores across CPU and GPU environments make results easier to compare. Stability fixes help keep scanning reliable when processing problematic files.

New Features

- Added support for HTML files (.html, .htm). Page text and embedded images are checked for AI-generated content.
- Added C2PA Content Credentials validation against the official C2PA trust lists. The trust lists ship in the database, so validation works offline.
- Added a C2PA trust level to the scan result for every labelled file: `verified`, `unverified_signer`, `signature_expired`, `not_verifiable` or `no_credentials`.
- Added reading of the C2PA edit history. AI use recorded in an earlier editing step is detected, including for images embedded in Office documents and PDFs.

Changes and Improvements

- Only Content Credentials from trusted signers now count as evidence that a file was AI-generated, and only those identify the AI model behind it. Credentials from test or revoked sources are ignored, so spoofed or unofficial labels no longer skew results.
- Files are processed faster, especially when many are scanned at once, with the same detection results as before.
- Text detection now gives the same results whether the engine runs on standard servers or GPU-accelerated hardware.
- Damaged or deliberately malformed files can no longer disrupt the engine or affect other scans.
- When a file's Content Credentials name an AI provider, the result now shows whether our own detection agrees. This helps analysts judge whether the label can be trusted.
- Each result now records which version of the trusted signer lists was used to validate the file's credentials.

Bugfixes

- Fixed an issue where some presentations could crash the content extractor while loading fonts.
- Fixed an issue where some malformed C2PA labels could crash the C2PA reader.

V 2.3.1

Release date: 29 September 2026

Release type: PATCH

AI Content Inspector 2.3.1 adds support for WMF/EMF images and OpenDocument spreadsheets and presentations,

introduces a Linux NVIDIA GPU engine package, and delivers updated image and text detection models, revised default confidence thresholds, and stability fixes for e-mail, Office, and text processing.

New Features

- Added support for WMF and EMF images (.wmf, .emf)
- Added support for OpenDocument spreadsheets and presentations (.ods, .ots, .odp, .otp)
- Added a Linux NVIDIA GPU engine package (requires CUDA 13.0 and NVIDIA driver R580 or newer)
- Added engine and database versions to every scan result
- Added metadata forensic analysis for images attached to .eml and .msg messages

Changes and Improvements

- Updated the ML SDK from 1.7.0 to 1.8.0, model weights from 1.7.0 to 1.8.0, and the DB API from 10 to 11
- Replaced OriginLens v0.5.1 with v1.4.2 in both FAST and FORENSIC profiles. The CNN and CommFor detectors and image heat maps have been removed; `output_images` remains available in the Core interface
- Texts shorter than 45 words are no longer scored and are reported as Uncertain
- Python engine modules are now compiled with Nuitka

Bugfixes

- Fixed an issue where scanning .msg files with certain code pages failed with error code 1
- Fixed an issue where correctly typed .msg files were rejected as unsupported
- Fixed an issue where capture date, camera, and GPS values stored in nested EXIF IFDs were not read
- Fixed an issue where the fraud result was reported as `not detected` when the fraud classifier did not run; it is now reported as `not scanned`
- Fixed an issue where unreadable Office documents were reported as an unsupported file type; they now fail with `err_code` 1 and a descriptive error message
- Fixed an issue where files with no extractable text returned a model score; they now report text as Not Detected
- Fixed an issue where text scans did not fail when the required text classifier could not be loaded

V 2.3.0

Release date: 14 September 2026

Release type: MINOR

AI Content Inspector 2.3.0 introduces support for Microsoft Office document formats, adds engine lifecycle notifications, and includes improvements to document text processing, an ML model update, and fixes for result consistency.

New Features

- Added support for Microsoft Word documents (.docx, .docm, .doc, .rtf, .odt)
- Added support for Microsoft PowerPoint presentations (.pptx, .pptm, .potx, .potm, .ppsx, .ppsm, .ppt, .pot, .pps)
- Added support for Microsoft Excel spreadsheets (.xlsx, .xlsm, .xltx, .xltm, .xlam, .xlsb, .xls, .xlt)
- Added out-of-support lifecycle notifications, informing users when the engine version reaches end of support

Changes and Improvements

- Reworked document text cleaning to improve the quality and consistency of text extracted for analysis
- Improved the performance of markdown text cleanup

Bugfixes

- Fixed an issue where the result JSON could be inconsistent when preferences were misconfigured
- Fixed an issue where sub-verdicts could be inconsistent when scanning with some features disabled

V 2.2.0

Release date: 14 August 2026

Release type: MINOR

AI Content Inspector 2.2.0 introduces support for additional email file formats, expands forensic analysis capabilities for embedded images, and includes improvements to provenance validation, image processing, and overall engine stability.

New Features

- Added support for **E-mail Message (.eml)** files
- Added support for **Microsoft Outlook Message (.msg)** files
- Added **metadata forensic analysis for images embedded within PDF documents**, enabling the engine to inspect image-level metadata and forensic signals extracted from PDF content

Changes and Improvements

- Strengthened **C2PA provenance validation and handling** to improve reliability when analyzing Content Credentials and provenance information
- Improved **image input normalization** to provide more consistent preprocessing across supported image formats and improve downstream analysis reliability

Bug Fixes

- Fixed a dependency issue affecting the **C2PA library**
- Fixed processing issues affecting **EPS files**
- Fixed an issue where **grayscale and bilevel EPS images** could fail during processing

- Resolved a stability issue where processing **FPX files** could cause the scan to terminate unexpectedly. **FPX is no longer supported** by AI Content Inspector
- Fixed an issue affecting **LA-mode images**, including certain `.cur` cursor files, which could fail with a `"not enough image data"` error

V 2.1.1

Release date: 10 August 2026

Release type: PATCH

- Detect AI Model Family from Image Metadata
- New OriginLens image detector
- Improved speed for image scanning (fast & forensic mode)
- Google SynthID watermark detection for images is now reported under its proper name, consistent with the text watermark detector
- Text detector display names aligned with the detection SDK
- Fixed legitimate high-resolution images failing with "Image size exceeds limit … could be decompression bomb". The limit is now 250 megapixels by default and configurable via AIGCD_MAX_IMAGE_PIXELS
- Fixed image fingerprint detection failing on files with malformed XMP metadata
- Fixed scans aborting when an embedded image inside a document could not be decoded
- Fixed Stable Diffusion 3 provenance being reported under an inconsistent model name

V 2.1.0

Release date: 13 July 2026

Release type: MINOR

- Introduce Fast mode and Forensic mode
  - Fast mode: Optimized for high-speed analysis with a fast response
  - Forensic mode: Optimized for deep analysis and investigative workflows
- Added an option to disable output image generation
- Improved PDF extraction performance
- Improved scan performance with faster processing times

V 2.0.1

Release date: 28 May 2026

Release type: PATCH

- Upgraded the core image and text detection components for improved accuracy and performance
- Resolved issues affecting text scanning functionality
- Enhanced support for offline and packaged deployments

V 2.0.0

Release date: 21 May 2026

Release type: MAJOR

- Added 10 new image detectors to analyze AI-generated footprints in visual content
- Introduced 9 new text detectors to identify AI-generated footprint signals in textual content
- Enhanced results page with deeper insights, providing more detailed outputs from individual detectors, including images and analysis breakdowns
- Introduced a new Uncertain Hits Threshold setting to improve control over uncertain detection results
- Added support for PDF file type including both AI text detection and AI image detection workflows
- Introduced fraud detection support for the following content types:
  - Invoice fraud
  - Car accident insurance fraud
  - Home insurance fraud

Introduction

OPSWAT AI Content Inspector 2.0.0 marks the General Availability release of OPSWAT's proprietary AI-powered content authenticity and document fraud detection engine. Purpose-built for modern, high-throughput environments, AI Content Inspector helps organizations identify AI-generated content and detect AI fraudulent documents with speed and accuracy.

Fully Integrated with the MetaDefender Platform

AI Content Inspector is fully integrated into the MetaDefender platform and is available across MetaDefender products and deployment environments, including both cloud and on-premises installations on Windows and Linux.

Working alongside Proactive DLP and Deep CDR, AI Content Inspector extends content authenticity and fraud detection coverage through advanced AI technologies. The engine analyzes both textual and visual artifacts to deliver AI-generation and fraud-indicator verdicts, enabling organizations to make informed security decisions before content enters critical workflows.

Designed to Combat Modern Fraud

AI Content Inspector focuses on the primary entry points for content-based fraud across enterprise workflows. It supports a wide range of image formats, text-bearing documents, and PDF files where AI-driven deception and document fraud most commonly occur, including:

- Invoice and accounts payable fraud
- AI-generated business and financial documents
- AI-generated or manipulated home damage images and insurance claims
- Detection of AI-generated insurance claim evidence

Key Benefits

Accelerated Content Review

- Faster review cycles for high-volume workflows such as claims processing and invoice verification
- Improved operational efficiency while reducing false positives

Intelligent Pre-Decision Analysis

- Rapid content inspection enables organizations to automatically allow, flag, block, or route content for further review at scale

Comprehensive Fraud Detection Coverage

Detects both AI-generated content and document fraud indicators across common enterprise fraud scenarios, including:

- AI-generated invoices and financial documents
- AI-generated vehicle and accident imagery fraud
- AI-generated home and property insurance fraud
- AI-generated synthetic claim documentation

Broad File Format Support

- Extensive support for image formats commonly used in fraud attempts
- Coverage for text-bearing documents and PDF files across enterprise workflows

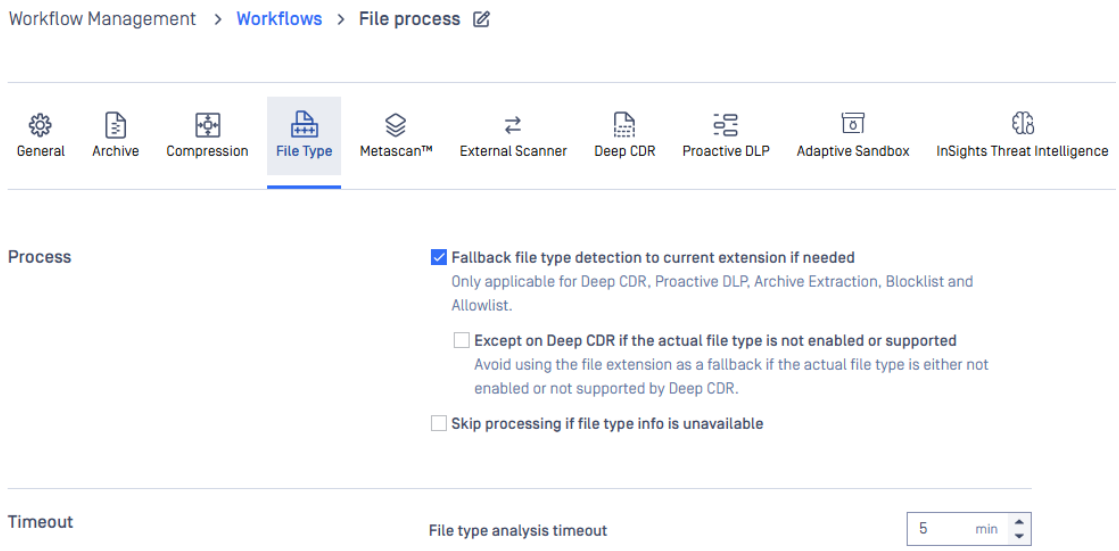
Availability

AI Content Inspector 2.0.0 is available as a standalone engine within MetaDefender Core and MetaDefender Cloud, providing organizations with advanced AI-driven content authenticity verification and document fraud detection capabilities across their security ecosystem.

File Type Verification Engine

With MetaDefender's file type verification technology, users can process files based on their true file types, so the system can take more precautions with risky file types like EXEs, perhaps setting different policies or rules based on file type. Spoofed file types indicate potentially malicious intent, so to mitigate this risk, MetaDefender offers functionality to block files with incorrect extensions, for example, prevents an instance of EXE file which is posed as TXT file from entering the organization. Also, more strict rules, like remediation steps can be applied. For example, Deep CDR is triggered as a post-action for the file type verification. This is highly configurable so that Deep CDR or any other secure processes can happen based on the target file name, file format and any other recognized file property data as well.

Configuration:



Fallback file type detection to current extension if needed: if this configuration is enabled, Deep CDR, Proactive DLP, Archive Extraction modules and Blocklist/Allowlist features will use the true file type from the file type engine to perform its job first; if it is failed, these modules and features will use the file extension as a backup.

Except on Deep CDR if the actual file type is not enabled or supported: when ticked, file extension will not be used as backup if actual file type is not enabled or supported by Deep CDR module.

Skip processing if actual file type could not be detected: if the File type engine is crashed/disabled/timeout/ cancelled, MetaDefender Core will return "Not scanned".

Supported file types

The File Type Verification engine enables organizations to evaluate and process files based on their true file type rather than relying on file extensions, which can be easily spoofed. This capability allows security teams to enforce granular policies for high-risk formats, such as executables (EXEs), and take proactive measures to block potentially malicious content. For example, files with mismatched types and extensions (e.g., an EXE file masquerading as a TXT) can be automatically flagged or blocked.

The engine supports accurate identification of thousands of known file formats. When a file's type cannot be conclusively determined, it is classified as binary or text and scanned accordingly to maintain inspection depth and

reduce blind spots. This approach strengthens defenses against file-based threats and ensures consistent enforcement of security policies.

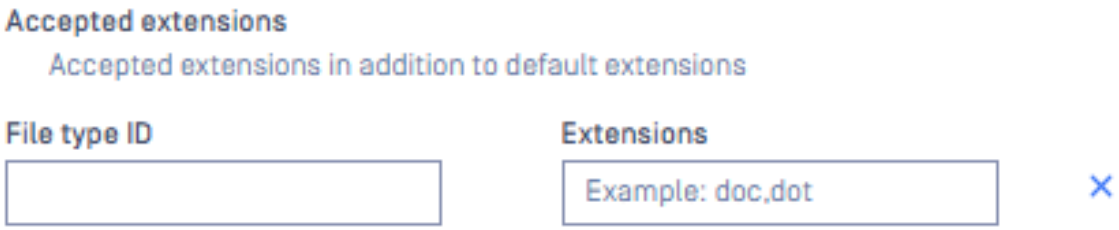
Configuration

This guide provides detailed configuration options under **Workflow Management > Workflows > [Workflow name] > File Type**, enabling users to customize detection settings effectively.

Detect file type mismatch

Accepted extensions

Accepted extensions is a feature that allows customers to add extensions for specific file type to avoid mismatching.



Section	Mandatory	Note
File type ID	Yes	
Extensions	No	Default value: empty. Can be * to accept any extension.

This feature also supports regular expressions in the value field. To indicate that a value is a regular expression, use the designated prefix. For example:

regex:^[^.]+$|.(?![A-Za-z]{3}$)[^.]+$|.(PNG|png)$

Include file details

See details of additional info returned by the engine at File info details.

Detect encrypted archive

Perform detection on the encryption attribute of archive files, such as ZIP and RAR.

If the option is turned on, encrypted archive files will have the returned output with value True in field encrypted.

Note: For *Encrypted ZIP* file, this feature only works when Core version is greater than 5.1.0.

Output for Encrypted ZIP · json

```
"file_info": {
```

```
    "description": "Encrypted ZIP Archive",
    "encrypted": true,
    "extensions": "zip,zipx",
    "groupID": "A",
    "groupIDs": [
        "A"
    ],
    "group_description": "Archive Files",
    "likely_type_ids": [
        {
            "score": 1,
            "typeID": "ZIP"
        }
    ],
    "type": "application/zip",
    "typeID": "ZIP",
    "type_ids": [
        "ZIP"
    ]
}
```


Overwrite a file type by another file type

This option is to alter the detected result (type ID) to a similar one.

Filetype	Sub filetype
Encrypted-GPG	GPG
Restricted-PDF	Encrypted-PDF

Classify with Machine Learning

Classify files using Machine Learning. See details in here.

Distinguish zero-byte files

If enabled, zero-byte files will be detected as ZERO instead of DATA.

Detect Base64 encoded text

Returns Base64 if the file is TXT and Base64 encoded.

Detect ASCII text

Enable ASCII detector to cover wide variety of ASCII text, e.g., non-printable or control characters. We can configure

the threshold for allowable non-printable characters via spin-box *Non-printable threshold*.

Non-printable characters

ASCII code	Character
1 -> 8	Control characters
11	[Vertical tab]
12	[Form feed]
14 -> 31	Control characters
127	[DEL]

Detect Shift-JIS text

Enable Shift-JIS text detector to cover wide variety of Shift-JIS text.

Supported engines-metadata header

Include more file info details

Beside the basic file type info such as true file type, file type group, etc... the File type engine is able to return more details about the file. To retrieve more file info please follow the steps

1. Use MetaDefender Core API to process a file
2. Send the **engines-metadata** header with this value

Json · javascript

```
{
  "file_type": {
    "include_file_details.file_metadata": false,
    "include_file_details.other_details": true
  }
}
```

A **file_info_details** attribute will be added to the output, e.g.:

Code · javascript

```
{
  ...
```

```
  "file_info_details": {
    "properties": {
      "portfolio": true
    }
  },
  "file_size": 3600732,
  "file_type": "application/pdf",
  "file_type_description": "Adobe Portable Document Format",
  "file_type_id": "PDF",
  ...
}
```

Note: Enabling this feature may impact the performance.

Overwrite file type detection

The File Type Verification Engine analyzes the file's structure and content to verify the true file type for given files and minimize the risk of file type spoofing. However, in some specific cases, the File Type Verification Engine returns unexpected results from the client's perspective. For example, if a file contains "Hello, **John**", it can be either an HTML file, or a text file, or even a CSV file with a comma separator.

Therefore, to handle these similar cases, MetaDefender Core now supports users to overwrite the true file type with 2 simple steps:

1. Use MetaDefender Core API to process a file
2. Send the **engines-metadata** header with this value

Json · javascript

```
{
  "file_type": {
    "overwritten": {
      "file_type_id": "HTML"
    }
  }
}
```

Other settings

- Detect encrypted archive: Perform detection on encryption attribute of archive files.
- Classify with Machine Learning: Classify text-based files with Machine learning engine. Module setting 'Enable Machine Learning module' needs to be turned on.
- Detect Base64 encoded text: Returns Base64 if the file is TXT and Base64 encoded.
- Detect file type mismatch: Detects when the extension of the file does not match with the allowed extensions.
  - Accepted extensions: Accepted extensions in addition to default extensions.

- File type ID: File type to apply
- Extensions: List of extensions separated by commas

- Detect ASCII text: Enable ASCII detector to cover wide variety of ASCII text, e.g., non-printable or control characters.
- Non-printable threshold: Percent of non-printable characters over total characters allowed in text.

Json · javascript

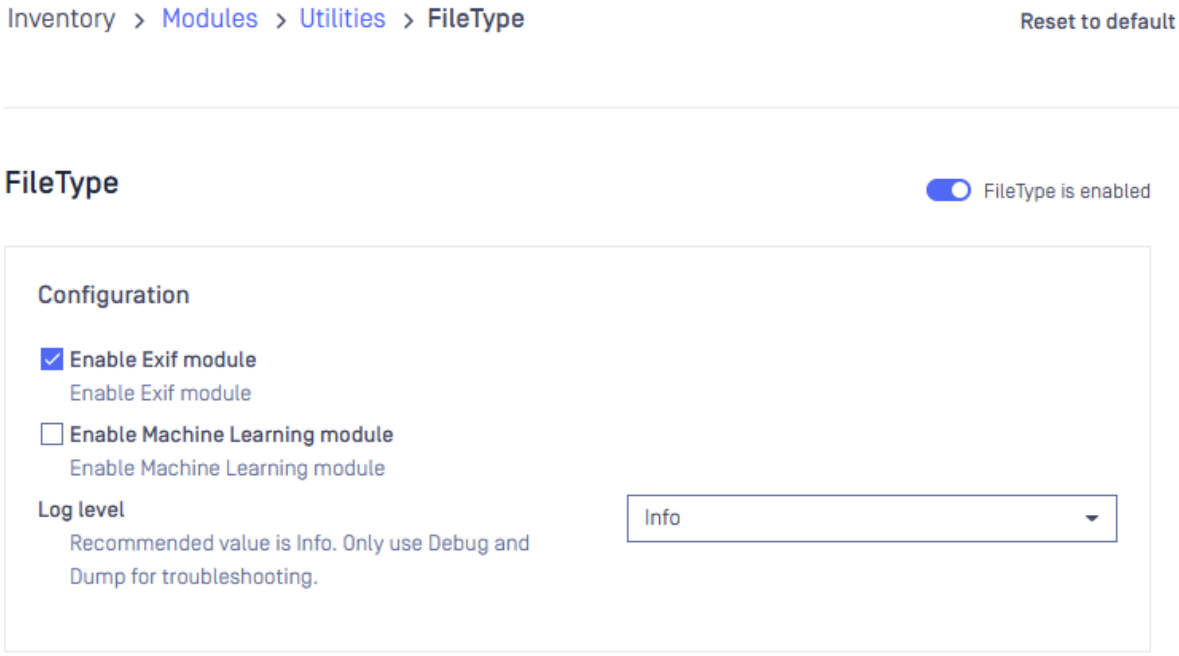
```
{
  "file_type": {
    "detect_encrypted_archive": true,
    "classify_with_ml": false,
    "detect_base64": false,
    "filetype_mismatch": true,
    "filetype_mismatch.accepted_extensions": [
      {
        "filetype_to_apply": "doc",
        "extensions_to_apply": "txt"
      }
    ],
    "enable_ascii": false,
    "enable_ascii.non_printable_threshold": 5
  }
}
```

Block files based on file info details

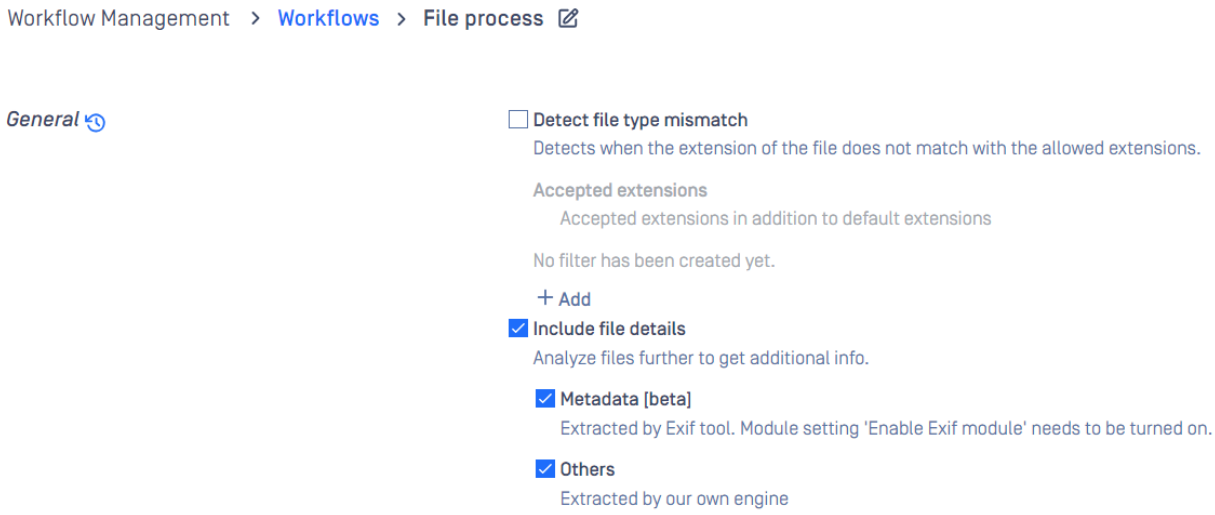
Configure to return file info details

Beside the basic file info such as true file type format, the file type also provides advance file details, to enable this feature, follow these steps:

- At **Inventory > Modules > Utilities > FileType**, Tick **Enable Exif module**



- At **Workflow Management > Workflows > [Workflow name] > File Type > General**, Enable **Include file details > Metadata**

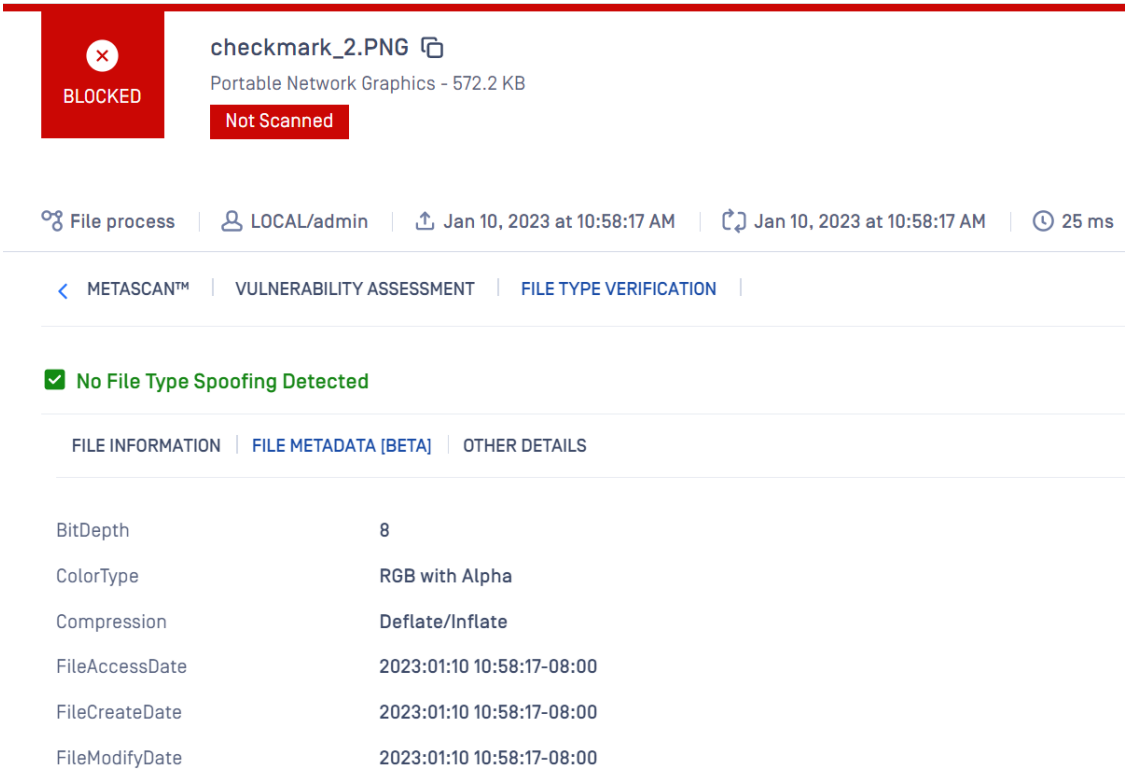


JSON output:

Code · json

```
"filetype_info": {
  ...
    "file_metadata": {
      "BitDepth": "8",
      "ColorType": "RGB with Alpha",
      "Compression": "Deflate/Inflate",
      "FileAccessDate": "2023:01:10 10:58:17-08:00",
      "FileCreateDate": "2023:01:10 10:58:17-08:00",
```

```
        "FileModifyDate": "2023:01:10 10:58:17-08:00",
        "FilePermissions": "-rw-rw-rw-",
        "Filter": "Adaptive",
        "Gamma": "2.2000000000000002",
        "ImageHeight": "598",
        "ImageSize": "1405x598",
        "ImageWidth": "1405",
        "Interlace": "Noninterlaced",
        "Megapixels": "0.83999999999999997",
        "PixelUnits": "meters",
        "PixelsPerUnitX": "5669",
        "PixelsPerUnitY": "5669",
        "SRGBRendering": "Perceptual"
    },
  ..
}
```

The feature is in BETA, please use the data for information only. The JSON structure and attribute names can be changed without notice.

Configure to block files based on file details

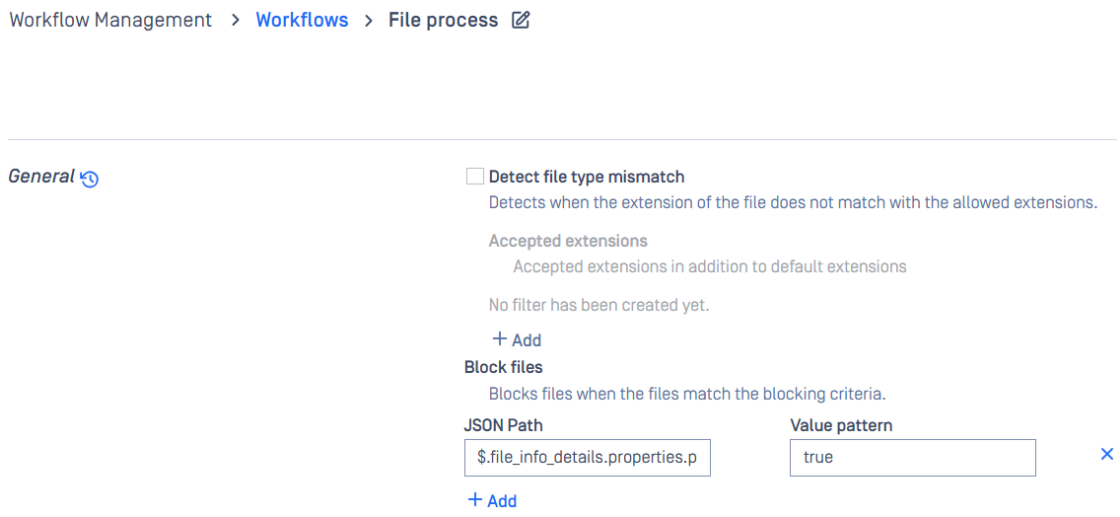
The file type engine allows you to block a file based on its file info details.

`JSON Path`: Specifies the path to the attribute in the JSON. See the "filetype_info" section in the exported JSON, currently supports the paths under "file_info_details" and "file_metadata".

Example: `$.file_metadata.producer`

`Value pattern`: Defines the pattern to match against the attribute value. Supports regular expressions.

Example: `GemBox.*?`

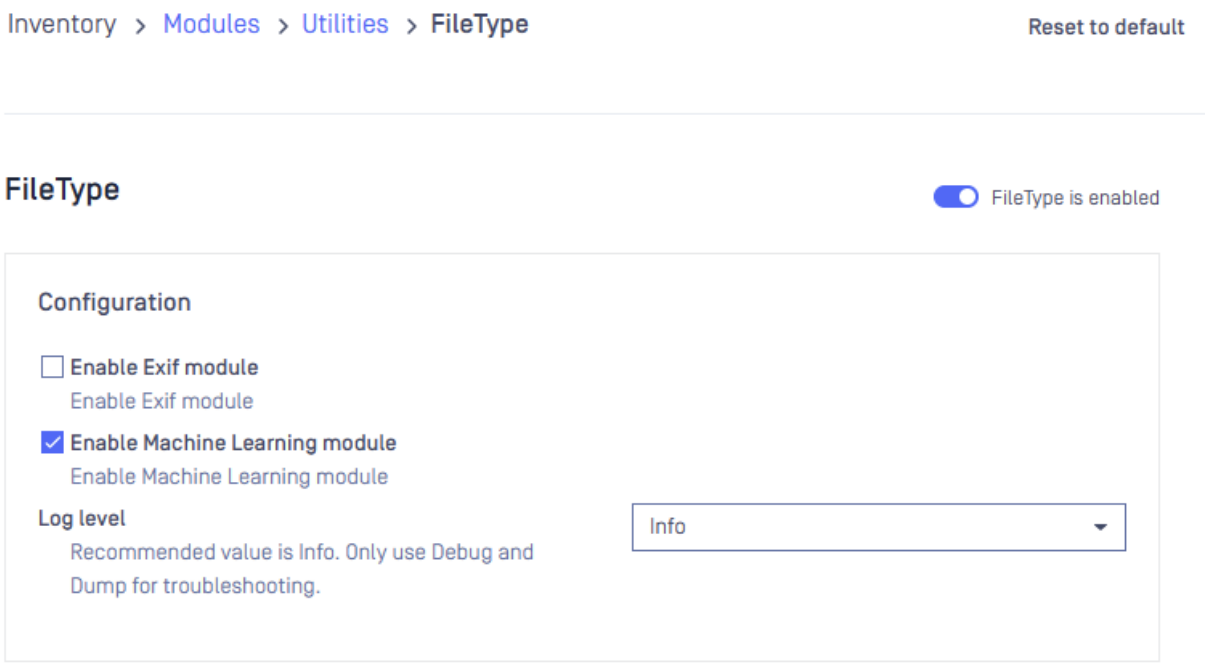


Text detection with AI

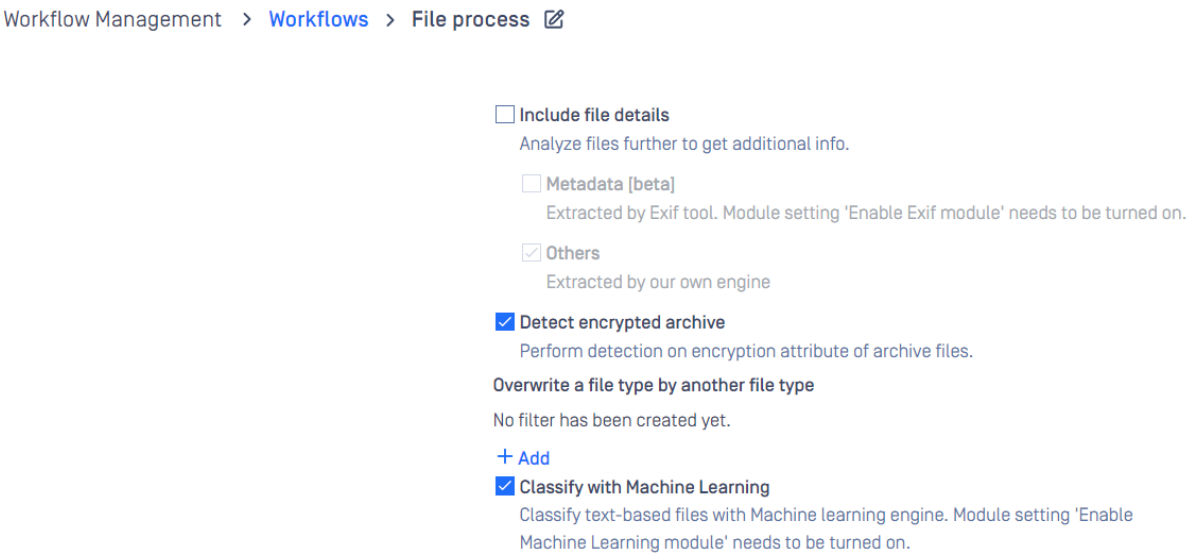
This feature is designed to address the challenge of text-based file detections. These files, such as log files, script files, and readme files, are all simply "text" and lack distinct characteristics revealed by other methods. Analyzing the contents is crucial for accurate classification.

To enable this AI feature:

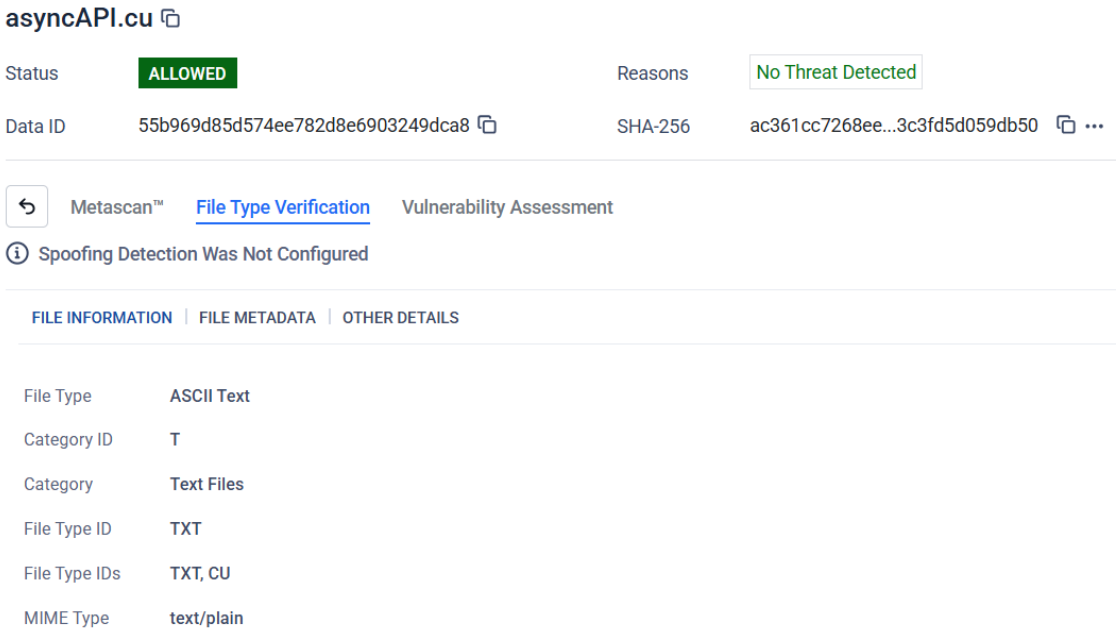
- At **Inventory > Modules > Utilities > FileType**, Tick **Enable Machine Learning module**



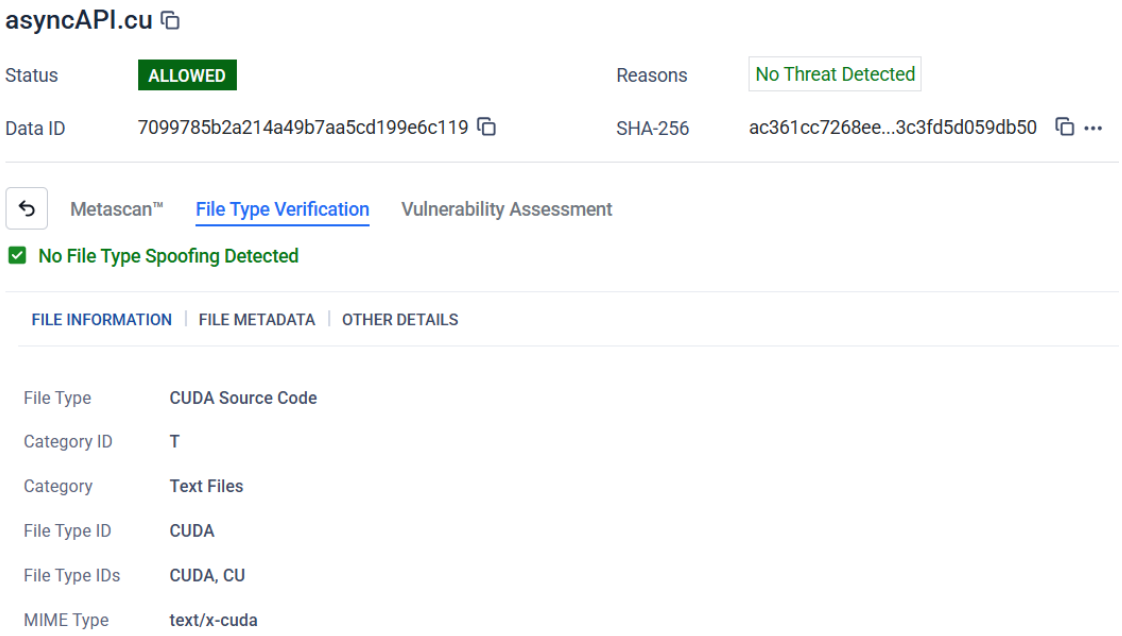
- At **Workflow Management > Workflows > [Workflow name] > File Type > General**, Enable **Classify with Machine Learning**



Without AI enabled, CUDA source code is detected as a text file.



With AI enhancement, CUDA source code is detected as a CUDA file



Info

This feature is disabled by default. If you enable the AI-powered feature, OPSWAT will not use your content to train or fine tune its services. Unlike file signature matching, AI-based relies on heuristics and is not guaranteed to be precise. Experiment with your own dataset to determine if the AI engine fits your need.

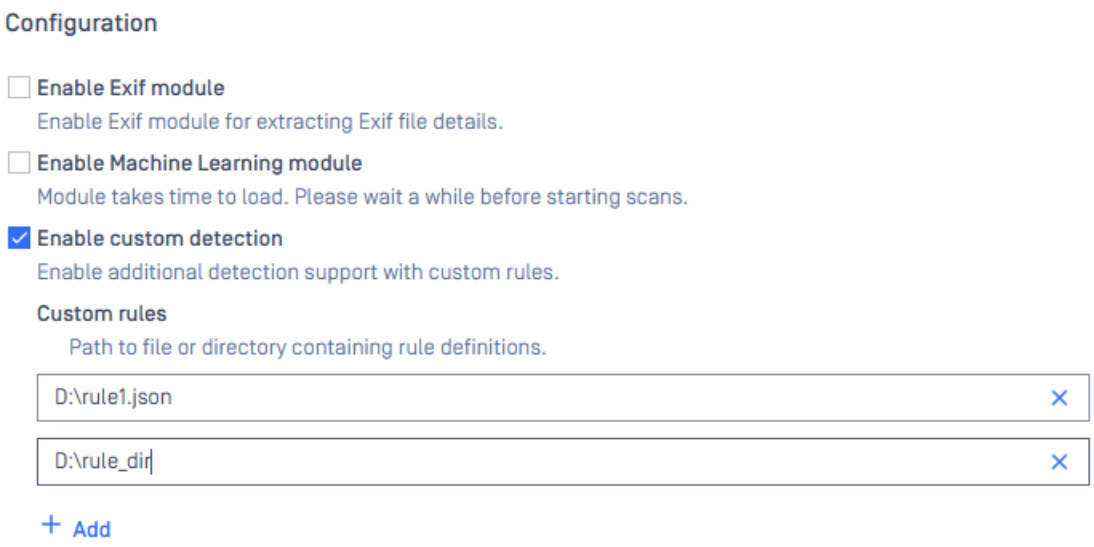
Custom detection

The custom detection feature is an advanced feature that allows users to define their own rules for identifying specific patterns within files. This capability enables users to support their own file types for detection quickly, without needing to wait for official support from the FileType engine.

Enable custom detection

This feature is disabled by default. To enable this feature:

- At **Inventory > Modules > Utilities > FileType**, Tick **Enable custom detection**
- At **Inventory > Modules > Utilities > FileType**, section **Enable custom detection**, specify paths to rule files and/or rule directories that contain JSON rule files.



Info

When there are updates on the rule files or the rule directories, the engine needs to be restarted in order for the rules to take effective.

When new items of the configuration are added, the rules are loaded automatically along with the changes insides the existing files or directories if available.

JSON custom rules

Rule definitions

Info of file types detected with custom rules and the rules are defined in JSON format with fields described as in the table below. The rule file must have `.json` extension.

File type Info

Field	Mandatory	Meaning
rule_type	Required	Can be either *xml-based* or *zip-based* or *binary*.
file_type	Required	File type description to be used to output.
file_type_id	Required	File type ID.
mime_type	Optional	Mime type to be used to output. Default value: **application/octet-stream**.

Field	Mandatory	Meaning
encrypted	Optional	Encrypt property to be used to output. Default value: **false**.
group	Optional	Group ID to be used to output. Default value: **0**. See the list of group IDs below.
extensions	Optional	Extension(s) for the file format. This value will be used to check mismatching. Default value: empty.

Binary detection rules

Field	Mandatory	Meaning
min_score	Optional	Minimum confidence score for this file type. Default value: **0.0**.
priority	Optional	Priority of the rule. Default value: **1**.
variables	Optional	Input variables array. Default value: empty.
[variable].name	Required	Name of the variable.
[variable].type	Optional	Type of the variable. Currently, only "built-in" variables are supported.
[variable].data_type	Optional	Data type of the variable. This can be *i8*, *i16*, *i32*, *i64*, *u8*, *u16*, *u32*, *u64*, *float*, *double* or a string.
[variable].length	Optional	Length of the variable. Applicable for string-type variable. Default value: **0**.
[variable].endianness	Optional	Endianness of the variable. This

Field	Mandatory	Meaning
		can be *host*, *little_endian*, *big_endian*. Default value: **little_endian**.
[variable].offset	Optional	(Similar to offset described in ***Binary detection rule***).
rules	Required	An array contains binary detection rules.
Binary detection rule		
[rule].score	Optional	Score of the rule. Default value: **0.0**.
[rule].description	Optional	The description of the rule. Default value: empty.
[rule].match	Required	Object contains the detection match.
offset	Optional	Offset to check for the match. Can be either integer or object. Applicable for type *exact*, *search*, *regex*, and *oneof*.
base	Optional	(For object-type *offset*) Base for the offset, can be one of the following values: • beginning (Default): Beginning of file. • end_of_data: End of file. • variable: the offset is an input variable.
variable	Optional	(For object-type *offset*) Variable for the offset. Required for variable-type base.
relative_offset	Optional	(For object-type *offset*) Relative

Field	Mandatory	Meaning
		value for the offset. Default value: **0**.
type	Required	Can only be one of the following values: *exact*, *compare*, *search*, *regex*, *oneof*.
For exact-type match		
offset	Optional	(mentioned above)
For compare-type match		
operator	Required	Can only be one of the following values: *equal*, *greater*, *greater_or_equal*, *less_than*, *less_than_or_equal*.
value	Required	The value to compare. This can be integer, string, or object.
variable	Optional	The input variable for the value. It's required for object-type value.
For search-type match		
offset	Optional	(mentioned above)
search_range	Optional	Searching range. 0 is for searching the whole file. Default value **0**.
occurrences	Optional	Number of occurrences for the match. Default value **1**. This can be either integer or object.
min	Optional	(For object-type *occurrences*) Minimum number of occurrences required for the match. Default value **0**.
max	Optional	(For object-type *occurrences*)

Field	Mandatory	Meaning
		Maximum number of occurrences required for the match. Default value **0**.
algorithm	Optional	Searching strategy. Can be either *linear* or *KMP*. Default value: **linear**.
For regex-type match		
offset	Optional	(mentioned above)
range	Optional	Searching range. 0 is for searching the whole file. Default value **0**.
For oneof-type match		
offset	Optional	(mentioned above)
[rule].pattern	Required	Object contains the detection pattern to be matched.
type	Required	Can only be one of the following values: *hex*, *text*, *encoded*, *variable*.
For hex-type pattern		
data	Required	Hex data. This can be a hex value or an array of hex values.
mask	Optional	Mask bytes.
For variable-type pattern		
name	Required	Name of variable.
For text-type pattern		
data	Required	The text data.

Field	Mandatory	Meaning
For encoded-type pattern		Not supported for now.

Zip-based detection rules

Field	Mandatory	Meaning
entry_rules	Required	An array contains all rules for entry checking.
[entry_rule].pattern	Required	Name pattern of the entry.
[entry_rule].type	Required	Can only be *exact* for now.
[entry_rule].score	Optional	Score for the entry. Default value: **0.0**.
[entry_rule].data	Optional	Rule to detect binary data of the entry (refer to section **Binary detection rules**).

Xml-based detection rules

Field	Mandatory	Meaning
root_node	Required	Object contains root node information.
name	Required	Name of the root node.
score	Optional	Root node contribution score. Default value: **0.0**.
attributes	Optional	An array contains attributes.
[attribute].name	Required	Attribute name.
[attribute].value	Required	Attribute value.
attributes_contribution	Optional	Attributes contribution score.

Field	Mandatory	Meaning
		Default value: **0.5**.
tree_nodes	Optional	An array contains tree nodes information.
score	Optional	Tree nodes contribution score. Default value: **0.0**.
nodes	Optional	An array contains tree nodes.
[node].name	Required	Node name.
[node].level	Required	Node level.
[node].parent	Required	Node parent.

Group ID and name

Group	Group	Group
A: Archive Files	G: Image Files	T: Text Files
AP: Application Files	I: Disk Image Files	Z: Email Files
D: Office Documents	M: Media Files	O: Other
D_ENC: Encrypted Documents	OPENSSL_ENC: OpenSSL Encrypted Files	
E: Executable Files	P: Adobe Files	

Example rules

Below are some JSON example rules.

Binary - Exact Match · json

```
{
  "rule_type": "binary",
  "file_type": "Seclore PDF",
  "file_type_id": "SeclorePDF1",
```

```
  "mime_type": "application/pdf-seclore",
  "extensions": [
    "pdf"
  ],
  "min_score": 1.0,
  "priority": 1,
  "rules": [
    {
      "pattern": {
        "type": "hex",
        "data": "255044462D"
      },
      "match": {
        "type": "exact",
        "offset": 0
      },
      "score": 0.35
    },
    {
      "pattern": {
        "type": "hex",
        "data":
"4658494D4C484445534141434149444E55494941424D55524D452E4454444C2E455456505947534F4B544C4F4F50434E48434C4945
      },
      "match": {
        "type": "exact",
        "offset": 61440
      },
      "score": 0.65
    }
  ]
}
```


Binary - OneOf Match · json


```
{
  "rule_type": "binary",
  "file_type": "GNU ustar TAR",
  "file_type_id": "GNUTAR_B256_80",
  "mime_type": "application/x-tar",
  "encrypted": false,
  "group": "A",
  "extensions": [
    "tar"
  ],
  "min_score": 1.0,
  "priority": 1,
  "rules": [
    {
      "pattern": {
```

```
        "type": "hex",
        "data": [
          "80",
          "FF"
        ]
      },
      "match": {
        "type": "oneof",
        "offset": 124
      },
      "score": 0.35
    },
    {
      "pattern": {
        "type": "hex",
        "data": [
          "7573746172202000",
          "7573746172003030"
        ]
      },
      "match": {
        "type": "oneof",
        "offset": 257
      },
      "score": 0.75
    }
  ]
}
```

Zip-based · json

```
{
  "rule_type": "zip-based",
  "file_type": "CUSTOM PBIX",
  "file_type_id": "CUS_PBIX",
  "mime_type": "application/custom-pbix-0.6",
  "extensions": [
    "cus_pbix"
  ],
  "entry_rules": [
    {
      "pattern": "DataModel",
      "type": "exact",
      "score": 0.3
    },
    {
      "pattern": "Connections",
      "type": "exact",
      "score": 0.3
    }
```

```
  ]
}
```

XML-based · json

```
{
  "rule_type":"xml-based",
  "file_type": "CUSTOM XML",
  "file_type_id": "CUSXML",
  "mime_type": "application/custom",
  "extensions": [
    "cus",
    "xml"
  ],
  "root_node": {
    "name": "cRoot",
    "score": 0.2,
    "attributes": [
      {
        "name": "attr",
        "value": "attrValue"
      }
    ]
  },
  "tree_nodes": {
    "score": 0.8,
    "nodes": [
      {
        "name": "cA",
        "level": 1,
        "parent": "cRoot"
      },
      {
        "name": "cB",
        "level": 1,
        "parent": "cRoot"
      },
      {
        "name": "cC",
        "level": 1,
        "parent": "cRoot"
      },
      {
        "name": "Pattern",
        "level": 2,
        "parent": "cB"
      }
    ]
  }
}
```

XML custom rules (deprecated)

Rule definitions

Info of file types detected with custom rules and the rules are defined in XML format with fields described as in the table below. The rule file must have `.xml` extension.

Field	Mandatory	Meaning
File type info		
description	Required	File type description to be used to output.
id	Required	File type ID.
mime	Optional	Mime type to be used to output. Default value: **application/octet-stream**.
group	Optional	Group ID to be used to output. Default value: **0**. See the list of group IDs below.
extension	Optional	Extension(s) for the file format. This value will be used to check mismatching. Default value: empty.
score	Optional	Confidence score for the custom file type. Value range [0, 1]. Default value: **0.25**.
Patterns for detection		
FrontBlock	Required	Define patterns at specific offsets
FrontBlock.Pattern	Required	Define offset (stored in *Pos*) and hex pattern to be compared (stored in *Bytes*).
GlobalStrings	Optional	Define patterns at random offsets.

Field	Mandatory	Meaning
GlobalStrings.String	Optional	Define string pattern to be matched.

Group ID and name

Group	Group	Group
A: Archive Files	G: Image Files	T: Text Files
AP: Application Files	I: Disk Image Files	Z: Email Files
D: Office Documents	M: Media Files	O: Other
D_ENC: Encrypted Documents	OPENSSL_ENC: OpenSSL Encrypted Files	
E: Executable Files	P: Adobe Files	

Notes

The current use case is to turn a unknown (DATA) or not surely (non-DATA with score < 1.0) (detected by native rules of the engine) file type into a user-custom one with higher score.

There can be cases in which a file matches both a custom and a built-in rule. In order to prioritize the detection result from the custom rule, the custom rule should be defined with a high confidence score, e.g., 1.1.

The detection "score" can be found in the JSON scan result: *filetype_info.file_info.likely_type_ids.score*

Example rules

Below are some XML example rules.

Rule 1 - Patterb at offset 0 · xml

```
<CustomRule ver="1.0">
  <Info>
    <description>FTA1 - an OPSWAT-defined file format</description>
    <id>FTA1</id>
    <mime>application/fta1-opswat</mime>
    <group>A</group>
    <extension>fta,fta1,opswat</extension>
    <score>0.9</score>
```

```
  </Info>
  <FrontBlock>
    <Pattern>
      <Bytes>AABBCCDDEE</Bytes>
      <Pos>0</Pos>
    </Pattern>
  </FrontBlock>
</CustomRule>
```

Rule 2 - Pattern at offset 16 · xml

```
<CustomRule ver="1.0">
  <Info>
    <description>FTA2 - an OPSWAT-defined file format</description>
    <id>FTA2</id>
    <mime>application/fta2-opswat</mime>
    <group>0</group>
    <extension>fta,fta2,opswat</extension>
    <score>0.9</score>
  </Info>
  <FrontBlock>
    <Pattern>
      <Bytes>AABBCCDDFF</Bytes>
      <Pos>0</Pos>
    </Pattern>
    <Pattern>
      <Bytes>FAFAFAFAFAFAFAFAFAFAFAFAFAFA</Bytes>
      <Pos>16</Pos>
    </Pattern>
  </FrontBlock>
</CustomRule>
```

Rule 3 - Have global string · xml

```
<CustomRule ver="1.0">
  <Info>
    <description>FTA3 - an OPSWAT-defined file format</description>
    <id>FTA3</id>
    <mime>application/fta3-opswat</mime>
    <group>0</group>
    <extension>fta,fta3,opswat</extension>
    <score>0.9</score>
  </Info>
  <FrontBlock>
    <Pattern>
      <Bytes>FF112233</Bytes>
      <Pos>0</Pos>
    </Pattern>
  </FrontBlock>
```

```
  <GlobalStrings>
    <String>OPSWAT</String>
  </GlobalStrings>
</CustomRule>
```

Rule 4 - Multiple global strings · xml

```
<CustomRule ver="1.0">
  <Info>
    <description>FTA4 - an OPSWAT-defined file format</description>
    <id>FTA4</id>
    <mime>application/fta4-opswat</mime>
    <group>O</group>
    <extension>fta,fta4,opswat</extension>
    <score>0.9</score>
  </Info>
  <FrontBlock>
    <Pattern>
      <Bytes>FF112244</Bytes>
      <Pos>1</Pos>
    </Pattern>
  </FrontBlock>
  <GlobalStrings>
    <String>OPSWAT</String>
    <String>FTA</String>
  </GlobalStrings>
</CustomRule>
```

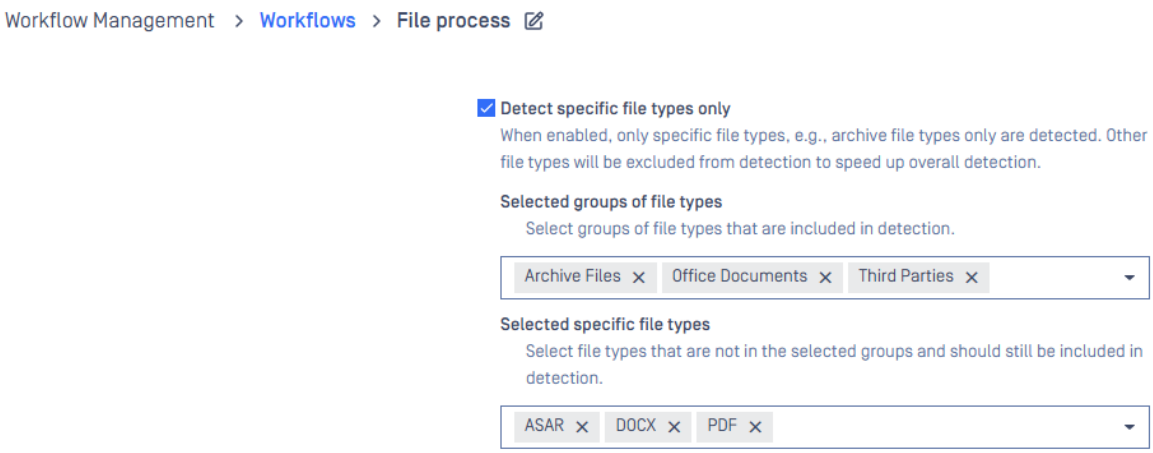
Detect specific file types only

This feature allows users to run detections for a chosen set of file types. By narrowing the scope, the engine concentrates solely on the selected targets, helping speed up the overall detection time.

Enable the feature

This feature is disabled by default. To enable this feature:

- At **Workflow Management > Workflows > [Workflow name] > File Type > General**, Enable **Detect specific file types only**
- Select file type groups at the sub-setting **Selected groups of file types**.
- Select file types at the sub-setting **Selected specific file types**.



Based on the selected groups and file types, the engine identifies which file types should be detected and selects the appropriate analyzers or detectors to run.

Special cases

- Additional file types may be processed beyond those explicitly selected. For example, if users choose file type A under Selected specific file types, and analyzer X supports both A and B, analyzer X will be invoked, and files of type B will also be detected..
- File types defined in user-created custom rules will not appear in the selection list of the `Detect specific file types` feature. This means these custom file types are excluded from the feature and will be detected as `DATA`.

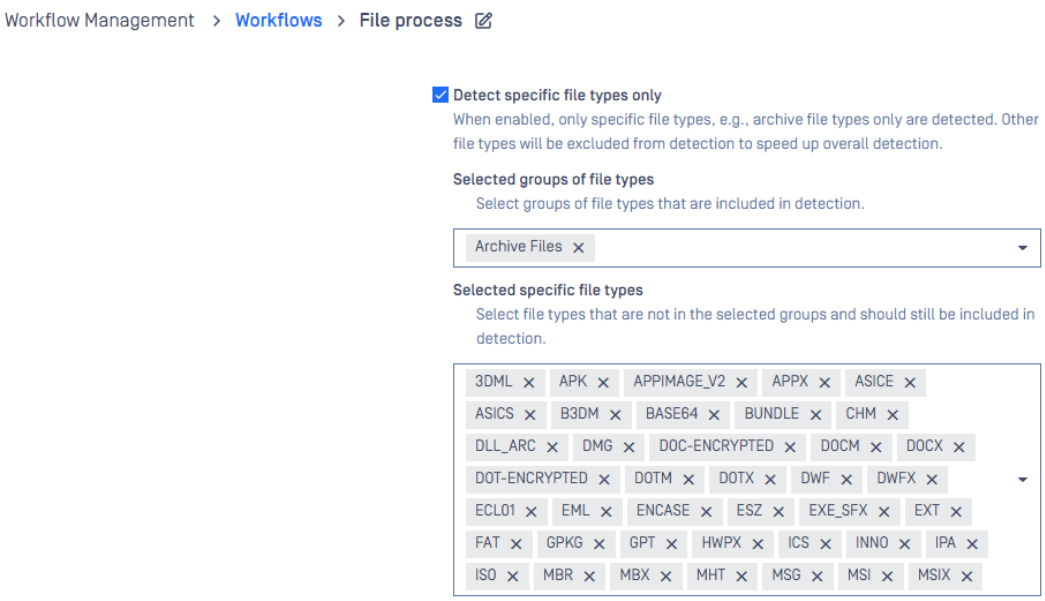
Use case: Run detections on extractable files only

The following is the way to configure detections to run only on extractable files.

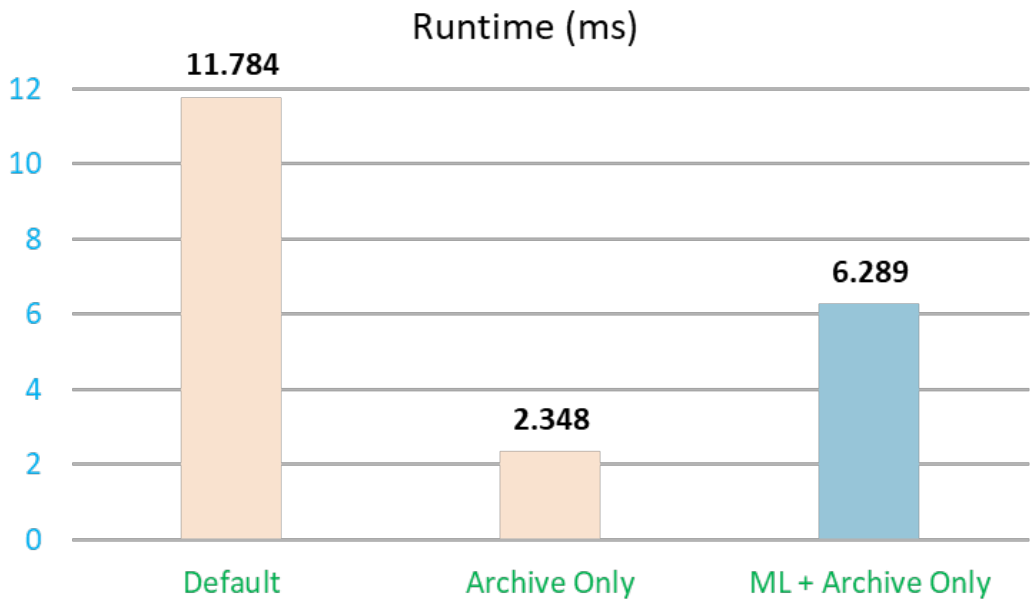
`Selected groups of file types` should be added with `Archive Files`which covers archive formats such as ZIP, 7z, RAR, and others.

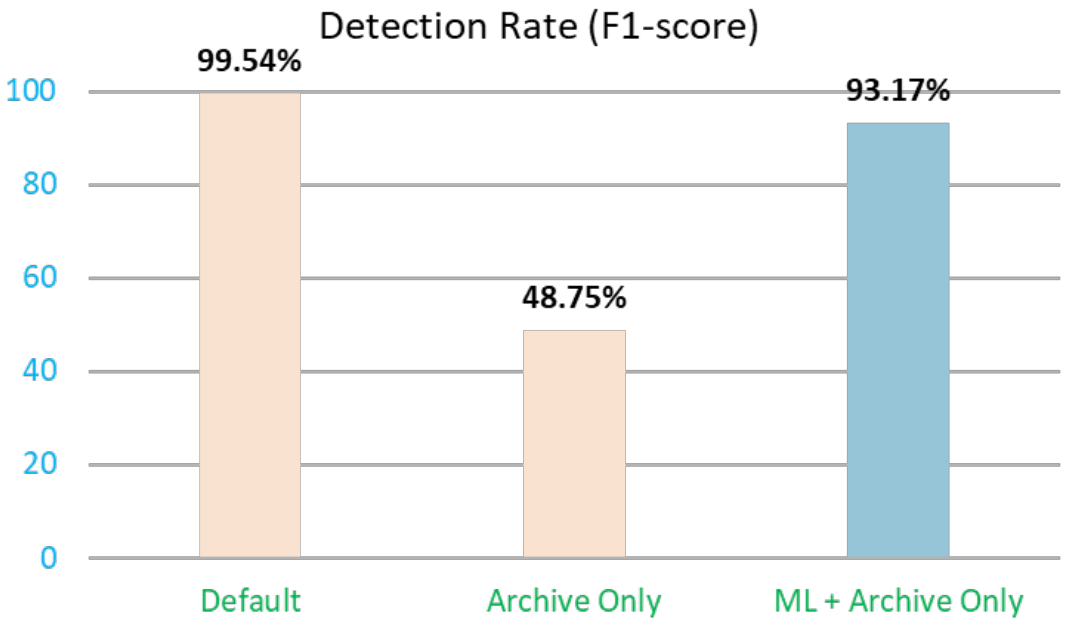
There are also non-archive files that are still extractable, for example, MS Office, disk images... If these are applicable to your use case, they should be added in the *Selected specific file types* section. You do not need to enable all options; only select the file types relevant to your use case.

The supported extractable non-archive file types: `3DML, APK, APPIMAGE_V2, APPX, ASICE, ASICS, B3DM, BASE64, BUNDLE, CHM, DLL_ARC, DMG, DOC-ENCRYPTED, DOCM, DOCX, DOT-ENCRYPTED, DOTM, DOTX, DWF, DWFX, ECL01, EML, ENCASE, ESZ, EXE_SFX, EXT, FAT, GPKG, GPT, HWPX, ICS, INNO, IPA, ISO, MBR, MBX, MHT, MSG, MSI, MSIX, MSM, MSO-ENCRYPTED, NTFS, NUPKG, ODP, ODS, ODT, OFT, OLE, OPENPGP-ENCRYPTED, OST, OTP, OTS, OTT, OXPS, P7M, PACK, PBIT, PBIX, PDF-ENCRYPTED, PKG, POT-ENCRYPTED, POTM, POTX, PPAM, PPS-ENCRYPTED, PPSM, PPSX, PPT-ENCRYPTED, PPTM, PPTX, PST, QCOW, RIBC, SHOW, SLDM, SLDX, STC, STD, STI, STW, SXC, SXD, SXG, SXI, SXM, SXW, TNEF, TWBX, UDF_ISO, UTIB, VDI, VHD, VHDX, VMDK, VSDM, VSDX, VSIX, VSSM, VSSX, VSTM, VSTX, WIM, X12EDI, XLAM, XLS-ENCRYPTED, XLSB, XLSM, XLSX, XLT-ENCRYPTED, XLTM, XLTX, XPS, ZLIB`.



Below are comparisons of run time and detection rate across three modes: Default, Archive Only, and ML + Archive Only. ML stands for Machine Learning which requires feature **Classify with Machine Learning** to be enabled. The test results are for reference only and were obtained using a dataset in which approximately 55% of the files were archive files. Actual results may vary depending on file types and nature of files.





Archive Extraction Engine

Detecting threats in compressed files, such as .ZIP or .RAR, can be difficult due to their large file size and ability to mask hidden threats such as archive bombs (malicious files designed to crash anti-malware programs). MetaDefender offers fast processing of archives by allowing administrators to perform archive handling once for each file type, instead of requiring each individual anti-malware engine to use its own archive handling methods. Additionally, administrators can customize the way archive scanning is performed to avoid threats like zip bombs.

System requirement

- On Linux: To extract password protected MS Office files, you need to install .NET 8 dependencies

Supported file types

The archive engine can extract 171 file types:

File type	Note
7z	a password protected file can be extracted with a given password
ACE	
ALZ	
AR	

File type	Note
ARJ	
BZ2	
CAB	all split CAB files can be extracted
CPIO	
CRX	
DEB	
UDEB	
DSD	
EGG	
GEM	
GZ	
JAR	
JARCS	
JMOD	
LHA	
LZH	
LZMA	
NAR	
PCK	
RAR	all split RAR files can be extracted. A password protected file can be extracted with a given password

File type	Note
RPM	
SPF	
STORY	
TAR	if a TAR is a split archive, only the first file can be extracted
TAZ	
TBZ	
TGZ	
TSE	
TSEC	
TSEZ	
VIB	
VTAR	
WAR	
WSP	
XAR	
XZ	
YZ1	
ZIP	if a ZIP is a split archive, only the first file can be extracted. A password protected file can be extracted with a given password
MDZIP	

File type	Note
XPI	
UUE	
ZSTD	
ASICE	
ASICS	
DOCM	
DOCX	
DOTM	
DOTX	
HPWX	
ODP	
ODS	
ODT	
OTP	
OTS	
OTT	
OXPS	
PBIX	
PBIT	
POTM	

File type	Note
POTX	
PPAM	
PPSM	
PPSX	
PPTM	
PPTX	
RIBC	
SHOW	
SLDM	
SLDX	
SCDOC	
STC	
STD	
STI	
STW	
SXC	
SXD	
SXI	
SXM	
SXW	

File type	Note
SXG	
TWBX	
UTIB	
VSDM	
VSDX	
VSSM	
VSSX	
VSTM	
VSTX	
XLAM	
XLSB	
XLSM	
XLSX	
XLTM	
XLTX	
XPS	
GPG	please see the details here
APK	
APPX	
DLL	

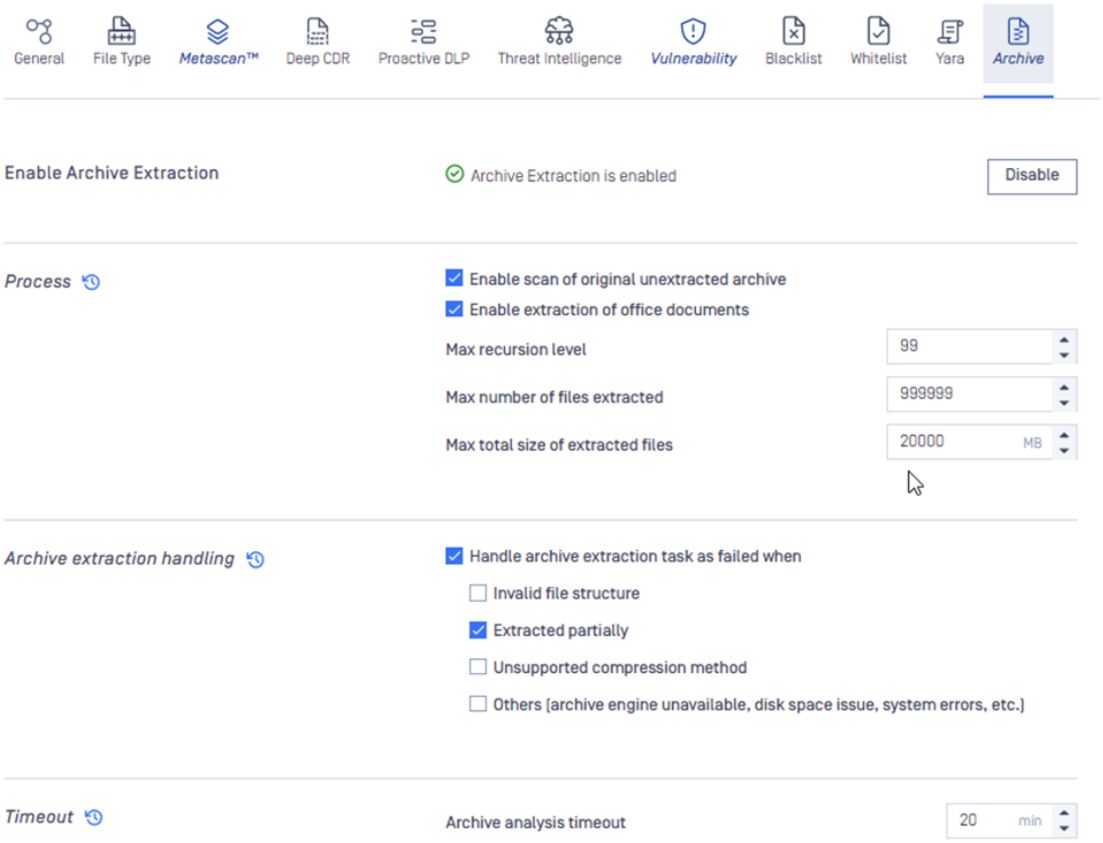
File type	Note
SFX	includes 7z SFX (created by 7-Zip), CAB SFX (created by IExpress), RAR SFX (created by WinRAR), ZIP SFX (created by PKZip)
IPA	
MSI	
MSIX	
PKG	
DMG	supported file systems: APFS, HFS, FAT
ISO	
UDF	
VDI	
VHD	partition Styles: GPT, MBR; File Systems: FAT16, FAT32, NTFS, EXT32, HFS
VHDX	partition Styles: GPT, MBR; File Systems: FAT16, FAT32, NTFS, EXT32, HFS
VMDK	partition Styles: GPT, MBR; File Systems: NTFS, HFS, FAT12, FAT16, FAT32, EXT2, EXT3, EXT4.
WIM	
E01	
NTFS	
FAT	
HFS	
EXT	

File type	Note
MBR	
OVA	
RAW	NTFS file system only
3DML	
CHM	
MSM	
P7M	
SMIME	Support multipart SMIME
GIT BUNDLE	
MS OLE	
B3DM	
GPKG	
ZLIB	
GIT PACK	
VSIX	
NUPKG	
NPZ	detected and extracted as ZIP file
OEX	detected and extracted as ZIP file
WHL	detected and extracted as ZIP file
ICS	

File type	Note
X12EDI	
ESZ	
MHT	
EML	extracted to header, body, attachments
MSG	extracted to header, body, attachments
OFT	
OST	
PST	
TNEF	
MBX	
DWF	
DWFX	
PCAP	with http, ftp, imap protocol
INNO	support Inno version 6.4.0 or older only
PYZ	
ASAR	
AppImg	
QCOW	support QCOW1, QCOW2, QCOW3
LZ4	
MAR	

File type	Note
UFDR	
YENC	
EDS	
3MF	
BLZ	
SquashFS	
WARC	
BLZ	
X3P	
Parquet	
KM5	

- Encrypted PDF, MS Word (DOC, DOCX, DOCM, DOTX, DOTM, DOT)/Excel (XLS, XLSX, XLSM, XLSB, XLTM)/ PowerPoint (PPT, PPTX, PPTM, PPSM, POTX, POTM), ALZ, EGG.
- An encoded base64 file can be decoded by the archive extraction engine.
- The Parquet file format supports extracting only binary columns.
- Configure the archive extraction in the Workflow:



Supported engines-metadata header

Decode and encode children file names with Shift-JIS encoding

The archive engine uses UTF-8 as a default encoding to create the children file names, if an archive file uses Shift-JIS encoding, please specify this info in the engine-metadata to preserve file names

1. Use MetaDefender Core API to process a file

2. Send the **engines-metadata** header with this value

Json · javascript

```
{
  "extraction": { "charset": "SJIS" },
  "compression": { "charset": "SJIS" }
}
```

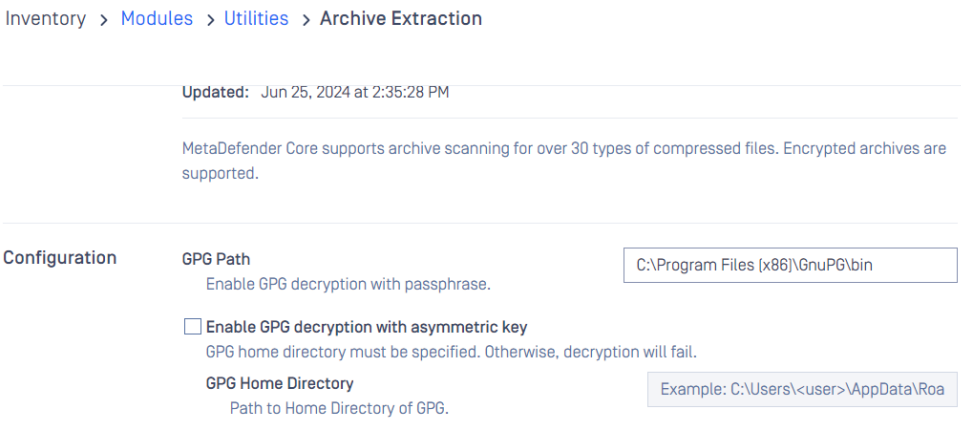
Decrypt GPG files

GPG files are created by applying GPG encryption to a file or message, ensuring that only authorized individuals with the corresponding private key can decrypt and access the original content. This is commonly used for securing sensitive information, such as emails, files, or communications.

Prerequisite: Install the GPG tool on the MetaDefender Core server.

Configure the GPG command line path: Inventory > Modules > Utilities > Archive extraction module. set GPG Path to the installed GPG bin folder

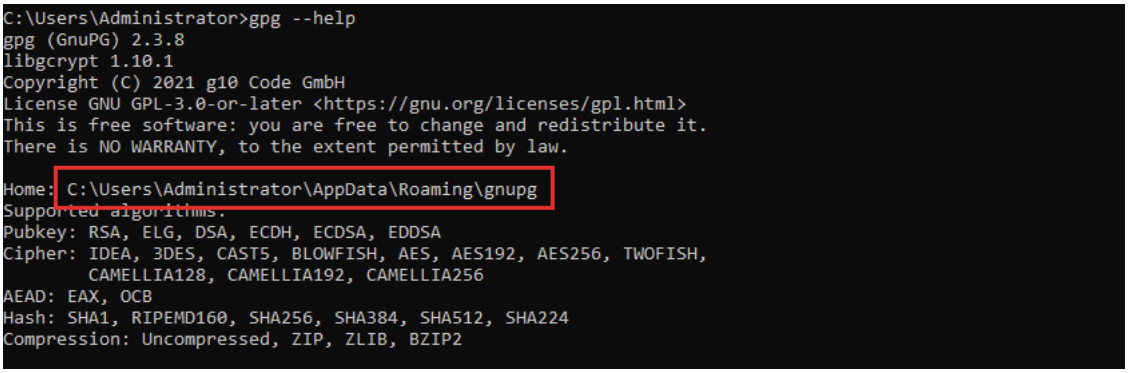
Symmetric encryption (encrypted with password)



With this setting, users can scan a GPG file with a provided password as a regular password-protected archive file.

Asymmetric encryption (encrypted with private/public key)

- Users need to import the private key to the local GPG tool first.
- Enable the "Enable GPG decryption with asymmetric key" option.
- Run **gpg --help** command to get the GPG Home directory, e.g.:



In Linux, users have to give "metascan" user permission to this home directory, including subfolders and files.

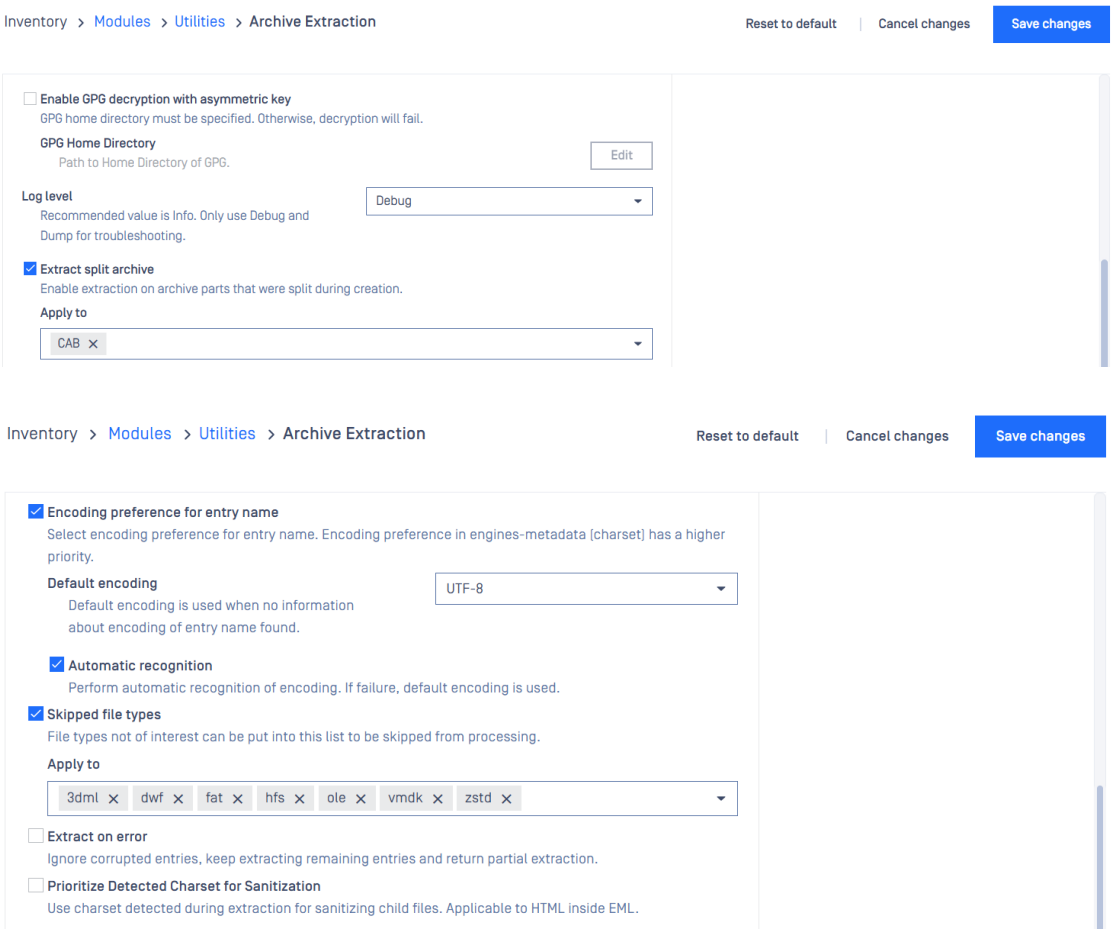
- Set the GPG Home directory in the UI.



With this setting, users can scan a GPG file with a provided passphrase as a regular password-protected archive file.

Configuration

This guide provides detailed configuration options under **Inventory > Modules > Utilities > Archive Extraction**, enabling users to customize extraction settings effectively.



GPG decryption

See details of **GPG Path** and **Enable GPG decryption with asymmetric key** at Decrypt GPG files.

Log level

- Allow users to configure log level.
- Recommended value is Info. Only use Debug and Dump for troubleshooting.

> **Log Rotation**
>
> Log Rotation is performed by compressing all the log files into an archive file when the current log size is greater than or equal to 512 MB and at least 30 minutes elapsed since the last rotation. Only 10 archived log files are kept.

Enforce HTML body content type matched

Enforce HTML body content to be matched as defined by content-type. Email body in HTML format mismatched with defined content-type will be replaced by simple HTML.

Extract split archive

Archive parts that were split during creation are distinguished from normal archive files and are not handled extraction by default due to possibly resulting in incomplete extracted entries.

- This setting allows users to enable extraction on archive parts that were split during creation.

Encoding preference for entry name

Archive files like ZIP on users' systems may have diverse character encodings of child entry names. For example, some archive files have child entry name with character encoding UTF-8 while some others have those with Shift-JIS.

- This setting allows users to select default character encoding and enable automatic recognition of character encoding for names of extracted entries.
- **Supported file types**: ZIP, TAR, and CAB.

> **Note**
>
> Character encoding preference in engines-metadata has a higher priority.

Default encoding

Select default character encoding, either UTF-8 or Shift-JIS.

Automatic recognition

Enable automatic recognition of character encoding to handle diverse character encodings of entry names.

Note

The recognition is based on testing the possibility of child entry names against target encodings. It is not guaranteed always correct or determined. In case of undetermined, default encoding will be used.

Skipped file types

Some uncommon file types may be supported extraction per some special use cases. Extraction of these file types can be unnecessary or even cost extra processing time for common workflows.

- File types not of interest can be put into this list to be skipped from processing. Customizing this setting can help balance efficiency with the flexibility to handle special requirements.
- Newly supported file types are usually considered to be put in this list by default.

Extract on error

There are some archive files that are slightly corrupted. Default behavior returns failure for those immediately and may result in Blocked for the file.

- In some workflows, users may expect to continue extraction, ignoring corrupted entries. This behavior helps maintain inspection of extractable entries.
- This setting allows users to customize the engine to ignore corrupted entries, keep extracting remaining entries and return Partial extraction.

Info

This setting will not change extraction result from failure to success. However, failure category could be changed. For details:

- `Corrupted Archive` changed to `Partially Extracted Archive`
- `Extraction Error` changed to `Partially Extracted Archive`

Prioritize Detected Charset for Sanitization

HTML email body that is extracted from an EML file may have charset incorrectly defined and differing from that defined in the EML structure which is a correct one. Charset extracted from the EML structure and specified in the HTML content as special hidden data can help direct the sanitization on the HTML content produce a valid output.

- Use charset detected during extraction for sanitizing child files. Applicable to HTML inside EML.

Differences between OPSWAT Archive engine and 7-zip

7-Zip is a widely-used utility for archive file extraction, renowned for its SDK-ready capabilities, extensive support for various file formats, and open-source architecture. This makes it a favored option for developers seeking to

incorporate robust extraction functionality into their applications. For example, OPSWAT integrates 7-Zip into its Archive Engine to facilitate efficient archive extraction. However, there are notable differences between the OPSWAT Archive Engine and 7-Zip:

- The OPSWAT Archive Engine extends beyond 7-Zip by leveraging multiple tools—including other open-source and commercial solutions—to achieve broader file format coverage. For instance, the Archive Engine supports the extraction of ALZ files (an ESTsoft proprietary format), which 7-Zip does not handle.
- OPSWAT's parsers are designed to process "container" file types, which are not conventional archive files such as MSG, EML, and PST. These file types can encapsulate other files within them (e.g., a PST file may contain multiple emails and contact files, while an MSG email file can include attachments). This capability ensures comprehensive extraction and analysis functionalities.
- 7-Zip is primarily designed for business extraction use cases and does not effectively address security scenarios where attackers attempt to deeply conceal data. Handling concatenated zip is an example, the 7-Zip extracts the first zip only, the OPSWAT Archive Engine can fully extract in this case.

Integrate with external command line tools

This feature enables users to integrate their own tools with the Archive Extraction engine to extract or decrypt files, allowing them to be further processed by other engines.

Preparation:

A JSON file is required to define which tools to use and how to execute them. The root of this JSON should be an array, where each element is a JSON object containing the following mandatory fields:

- `"file_type_id"`: Specifies the target file type.
- `"executable_path"`: The absolute path to the executable tool.
- `"arguments"`: Command-line arguments to run the tool. This must include the `$input` and `$output` placeholders. The `$password` placeholder is optional.
- `"patterns_indicating_encrypted"`: A list of substring in the command-line output that signifies an extraction failure caused by an incorrect password.

Example:

Code · json

```
[
  {
    "file_type_id": "OPENSSL-SALT",
    "executable_path": "/usr/bin/openssl",
    "arguments": [
      "enc",
      "-d",
      "-des-ebc",
      "-in",
      "$input",
      "-out",
      "$output",
```

```
      "-k",
      "$password"
    ],
    "patterns_indicating_encrypted": ["bad decrypt","bad password read"]
  },
  {
    "file_type_id": "OPENSSL-SALT",
    "executable_path": "/usr/bin/openssl",
    "arguments": [
      "enc",
      "-d",
      "-des3",
      "-in",
      "$input",
      "-out",
      "$output",
      "-k",
      "$password"
    ],
    "patterns_indicating_encrypted": ["bad decrypt","bad password read"]
  }
]
```

Enable the feature in Inventory > Modules > Archive extraction > Enable CLI Processing and configure the file path to point to the JSON file.



To apply the changes, disable and then re-enable the Archive Extraction module. Modifying the JSON file also requires restarting the engine.

Decrypt Microsoft Purview Information Protection (MPIP) files

Overview

Microsoft Purview Information Protection (MPIP, previously Microsoft Information Protection / MSIP and Azure Information Protection / AIP) lets organizations apply sensitivity labels to files. When a label carries an **encryption** setting, the file is wrapped with Azure Rights Management (Azure RMS) protection: the content is AES-encrypted, and the key needed to open it is held by the Azure RMS service for the tenant that applied the label. Only identities that RMS authorizes can obtain that key.

MetaDefender Core has detected these files, the File Type engine reports them as `MSO-ENCRYPTED-MSIP` (Office documents), but could not look inside them, so the payload was treated as opaque and could only be allowed or blocked by policy.

Starting with Archive Extraction engine 8.1.0, the Archive Extraction module can decrypt MPIP-protected files in

place and hand the original data to the rest of the workflow (Multiscanning, Deep CDR, Proactive DLP, Sandbox, and so on). Decryption uses the Microsoft Information Protection (MIP) File SDK and a service principal that your Microsoft 365 tenant administrator authorizes as an Azure RMS *super user*.

> ***Scope:*** *the integration decrypts files protected by* ***your own tenant*** *only. Azure RMS scopes super-user rights per tenant by design; files labeled by a partner or customer tenant cannot be opened with your credentials. Multi-tenant organizations must repeat the tenant setup for each tenant and configure a MetaDefender Core instance per tenant.*

How MPIP protection and decryption work

The decryption handshake

When the Archive Extraction module receives an `MSO-ENCRYPTED-MSIP` file with MPIP decryption enabled, the built-in MIP service performs the following sequence. Steps 1–3 happen once at startup and are cached; steps 4–6 run per file.

Step	Action
1	Authenticate as the service principal (client-credentials flow with Tenant ID, Client ID, Client Secret)
2	Acquire a token for the policy sync service and download the tenant's sensitivity-label policy
3	Acquire a token for Azure RMS
4	Parse the file, extract the publishing license
5	Send the publishing license to RMS and request an **end-user (use) license**. RMS checks the token's object ID against the tenant's super-user group and, if it matches, returns the unwrapped content key.
6	Decrypt the payload locally with the content key and pass the plaintext file to the workflow as an extracted child

Security boundary: the document body never leaves the MetaDefender Core host. RMS sees only the publishing license, the content ID, and the authentication token. This is relevant for data-residency and compliance conversations — the integration does not upload customer content to Microsoft.

Why a super user

A normal user can only open files whose publishing license grants them rights. A security gateway must open *every* file regardless of who it was addressed to. Azure RMS provides the **super-user** feature for exactly this case: members of the designated super-user group are always granted full use licenses for any content protected by the tenant. The MetaDefender Core service principal is added to that group. Removing it from the group is an instant kill-switch, no change is required on the MetaDefender Core side.

Performance characteristics

Metric	Typical value
First decryption after engine start	~3–5 s (token acquisition, policy sync, first RMS round trip)
Subsequent files	~200–800 ms each (only the use-license request hits the network)
Token lifetime	~1 hour, refreshed automatically
Policy cache	Persisted on disk, refreshed when label policy changes
Practical concurrency	~8 parallel RMS requests before service-side rate limiting

Prerequisites

On the Microsoft 365 side

- Global Administrator or Compliance Administrator role on the tenant that protects the files.
- Windows PowerShell 5.1.
- Access to the Microsoft Entra admin center.
- Azure Rights Management activated on the tenant (it is on by default for most Microsoft 365 subscriptions that include Purview Information Protection).

On the MetaDefender Core side

- MetaDefender Core with Archive Extraction engine **8.1.0 or later**.
- Outbound HTTPS (443) from the MetaDefender Core server to:
  - `login.microsoftonline.com` and `login.windows.net`
  - `api.aadrm.com` and `*.aadrm.com`

- `syncservice.o365syncservice.com`

Part 1 — Set up the Microsoft 365 tenant

Programmatic decryption requires **all four** of the following to be true. Skipping any one of them produces an `Access Denied` error at scan time that does not say which condition failed, so verify each before troubleshooting MetaDefender Core.

1. An Entra ID application with the right API permissions
2. The Azure RMS super-user feature enabled on the tenant
3. A super-user group designated in RMS
4. The application's service principal added as a member of that group

Step 1 — Register the Entra ID application

1. Sign in to the **Entra admin center** → **Identity** → **App registrations** → **New registration**.
2. Fill in:
   - **Name:** `MetaDefender-MPIP-Decryptor`
   - **Supported account types:** Accounts in this organizational directory only (single tenant)
   - **Redirect URI:** leave blank
3. Click **Register**.
4. From the **Overview** page, record the **Application (client) ID** and **Directory (tenant) ID**. You will enter both in MetaDefender Core.

Step 1a — Create a client secret

1. In the registered app → **Certificates & secrets** → **Client secrets** → **New client secret**.
2. Description: `metadefender-core`; expiry: 24 months (the maximum).
3. Click **Add**, then immediately copy the **Value** column (not the Secret ID).

Step 2 — Grant API permissions

In the app registration → **API permissions** → **Add a permission**.

2a. Azure Rights Management Services

1. Open the **APIs my organization uses** tab and search for `Azure Rights Management Services`.
2. Choose **Application permissions**.
3. Add `Content.SuperUser` and `Content.Writer`.

If `Azure Rights Management Services` does not appear in the search, the RMS service principal has not been provisioned in your tenant. Run the following in Windows PowerShell, wait 5–10 minutes, and retry:

2b. Microsoft Information Protection Sync Service

This API is hidden from the default picker. Use one of these methods.

Method A — search by application ID. In **APIs my organization uses**, paste this GUID into the search box:

```
Code

870c4f2e-85b6-4d43-bdda-6ed9a579b725
```

Select the **Microsoft Information Protection Sync Service** tile → **Application permissions** → add `UnifiedPolicy.Tenant.Read.All`.

Method B — edit the manifest (more reliable). App registration → **Manifest** → add this object to the `requiredResourceAccess` array, then save:

```
Code · json

{
  "resourceAppId": "870c4f2e-85b6-4d43-bdda-6ed9a579b725",
  "resourceAccess": [
    {
      "id": "23e15f4a-67fa-4f72-b1da-219ce72be8d0",
      "type": "Role"
    }
  ]
}
```

2c. Grant admin consent

On the **API permissions** page click **Grant admin consent for [tenant]**. All three permissions must show **Granted**:

API	Permission	Type
Azure Rights Management Services	`Content.SuperUser`	Application
Azure Rights Management Services	`Content.Writer`	Application
Microsoft Information Protection Sync Service	`UnifiedPolicy.Tenant.Read.All`	Application

Step 3 — Create the super-user group

1. **Entra admin center** → **Identity** → **Groups** → **All groups** → **New group**.
2. Fill in:
   - **Group type:** Microsoft 365
   - **Group name:** `mpip-superusers`
   - **Group email address:** `mpip-superusers` (the tenant domain is appended automatically)
   - **Owner:** your admin account
   - **Members:** leave empty for now
3. Click **Create** and wait about a minute for provisioning. Note the group's full email address from its Overview page.

Step 4 — Enable the RMS super-user feature

In **Windows PowerShell 5.1** or **Azure Cloud Shell (PowerShell)**:

Code · powershell

```
# One-time per machine
Install-Module -Name AIPService -Force

# Browser sign-in as Global Admin or Compliance Admin
Connect-AipService

# Turn on the super-user feature for the tenant
Enable-AipServiceSuperUserFeature

# Designate the group from Step 3
Set-AipServiceSuperUserGroup -GroupEmailAddress "mpip-superusers@yourtenant.onmicrosoft.com"
```

Replace `yourtenant.onmicrosoft.com` with your tenant's primary domain.

Step 5 — Add the service principal to the super-user group

1. **Entra admin center** → **Identity** → **Groups** → open `mpip-superusers`.
2. **Members** → **Add members**.
3. Search for `MetaDefender-MPIP-Decryptor`. The service principal shows a gear icon rather than a person icon. Select it → **Select** → **Add**.

If the service principal does not appear, force its creation and retry:

Code · powershell

```
Connect-MgGraph -Scopes "Application.ReadWrite.All"
New-MgServicePrincipal -AppId "<client-id-from-step-1>"
```

Step 6 — Verify the tenant configuration

Code · powershell

```
Get-AipServiceSuperUserFeature
# Expected: Enabled

Get-AipServiceSuperUserGroup
# Expected: mpip-superusers@yourtenant.onmicrosoft.com
```

If either result is `Disabled` or empty, re-run the corresponding cmdlet from Step 4.

Step 7 — Hand off credentials

The person configuring MetaDefender Core needs:

Value	Where to find it
Tenant ID	App registration → Overview → Directory (tenant) ID
Client ID	App registration → Overview → Application (client) ID
Client secret	The value copied in Step 1a

Part 2 — Configure MetaDefender Core

Enable MPIP decryption in the Archive Extraction module

1. In the MetaDefender Core Management Console go to **Inventory → Modules → Utilities → Archive Extraction** and open its settings.
2. Enable **Decrypt MSIP-protected documents**. The option is disabled by default.
3. Enter the values from Step 7:
   - **Tenant ID**
   - **Client ID**
   - **Client Secret**
4. Save. The Archive Extraction engine restarts and launches its MIP processing service. The service authenticates to the tenant and downloads the label policy at startup, so the first scan after enabling the feature takes a few seconds longer.

Workflow configuration

MPIP-protected files are handled as an archive-like container: the decrypted document becomes the single extracted child of the original file and flows through every engine enabled in the workflow.

Files whose decryption fails (wrong tenant, service principal not authorized, network error) are reported the same way as failed Office or PDF password decryption: the file remains detected as `MSO-ENCRYPTED-MSIP` / `PDF-ENCRYPTED-MSIP` with an extraction failure, and the workflow's existing handling for encrypted archives applies.

Verify

1. Using Microsoft Word signed in to the configured tenant, create a document and apply a sensitivity label whose settings include **Encrypt files**. Save it.
2. Submit the file to MetaDefender Core (Management Console Process, or the REST API).
3. In the scan result:
   - The top-level file is detected as `MSO-ENCRYPTED-MSIP`.
   - **Extracted files** shows one child — the decrypted document, detected as the ordinary Office type (for example `Microsoft Word Document`).

Troubleshooting

Symptom	Likely cause	Fix
`AccessDenied` on every file, tenant setup just completed	Super-user group membership has not propagated to RMS	Wait up to 30 minutes and retry. Confirm with `Get-AipServiceSuperUserGroup`.
`AccessDenied` persists after 30 minutes	Service principal not in the super-user group, or super-user feature disabled	Re-check Steps 4–6. Ensure the group is a **Microsoft 365** group, not a security group.
`The protection service is unavailable` / HTTP 503 from `api.aadrm.com/my/v2/enduserlicenses`	Azure RMS not activated on the *issuing* tenant, or credentials belong to a different tenant than the one that protected the file	Run `Get-AipServiceConfiguration` on the issuing tenant; `FunctionalState` must be `Enabled`. Confirm the Tenant ID in MetaDefender Core matches the tenant that applied the label.
Token acquired but only for `syncservice.o365syncservice.com`; RMS call fails	`Azure Rights Management Services` permissions missing or admin consent not granted	Re-check Step 2a and 2c. All three permissions must show **Granted**.
Feature stopped working after months of operation	Client secret expired	Create a new secret (Step 1a), update MetaDefender Core, delete the old one.

Limitations

- **Same-tenant only.** Files protected by other tenants cannot be decrypted.
- **Decryption only.** MetaDefender Core does not re-apply the label or encryption to the sanitized or extracted output.
- **Office and PDF payloads.** Supported inputs are the `MSO-ENCRYPTED-MSIP` file types.

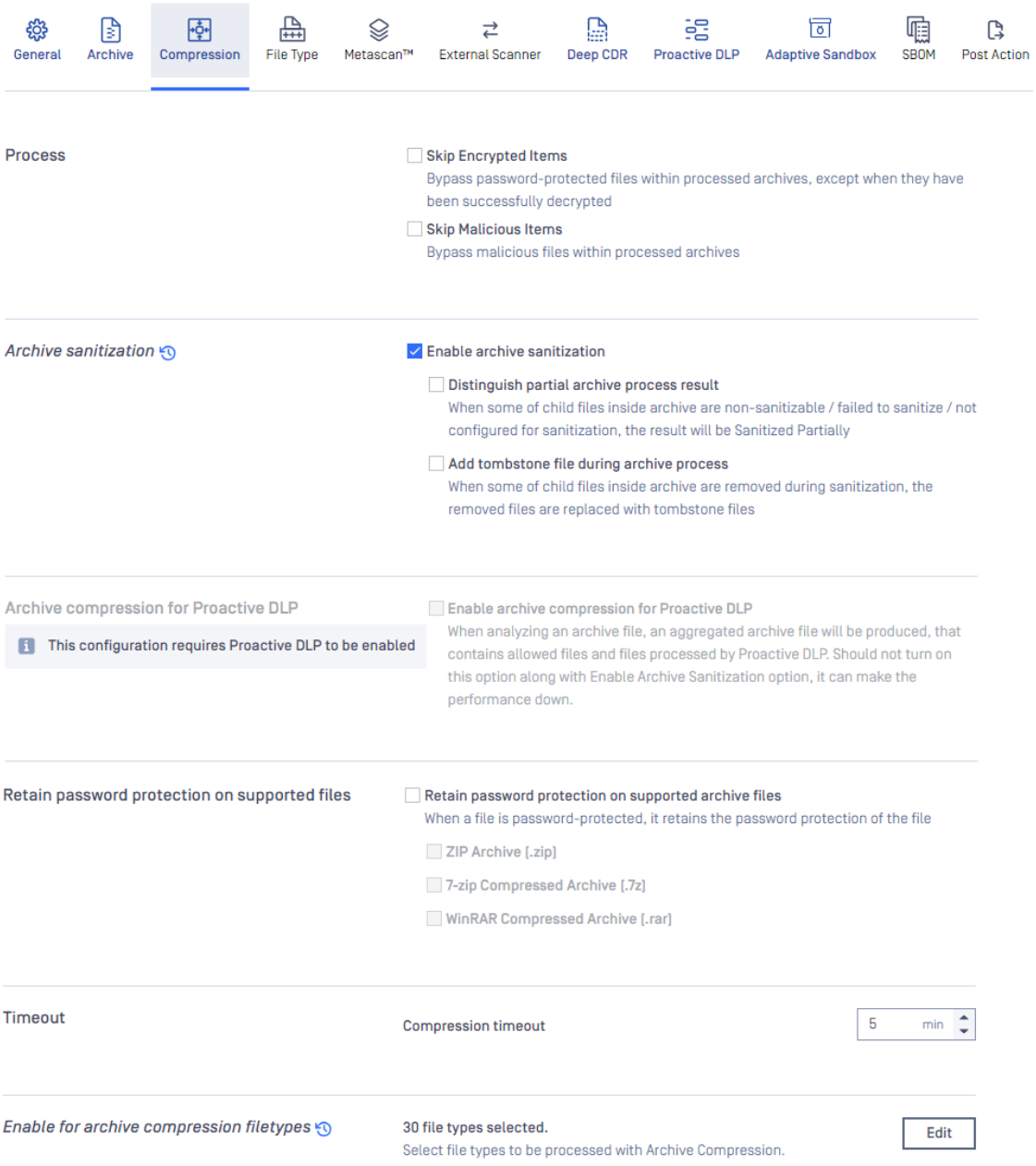
Archive Compression Engine

Archive Compression is commonly used in conjunction with the Deep CDR engine for sanitizing archive files. While the Deep CDR engine sanitizes the files within an archive, the Archive Compression engine compresses the sanitized files back into a new archive. For more detailed information, please refer to this page: Archive Sanitization.

Additionally, if you are using DLP without Deep CDR, you can configure Archive Compression to work with DLP. In this case, the compressed output file will contain both allowed files and files processed by Proactive DLP.

Archive Compression also includes the capability to retain the original password of the archive.

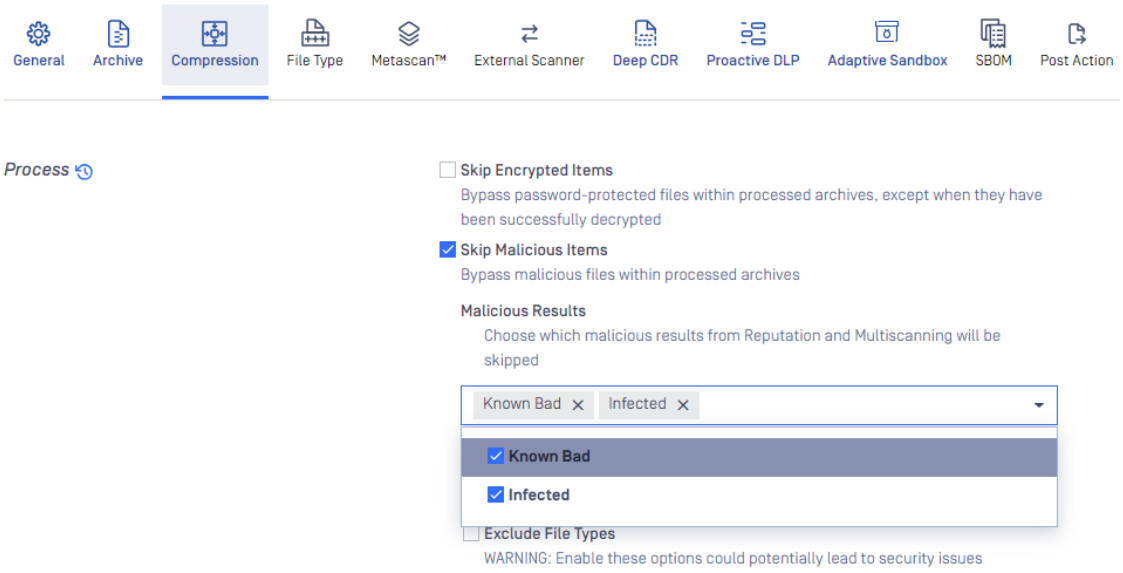
Configuration



Detailed explanations

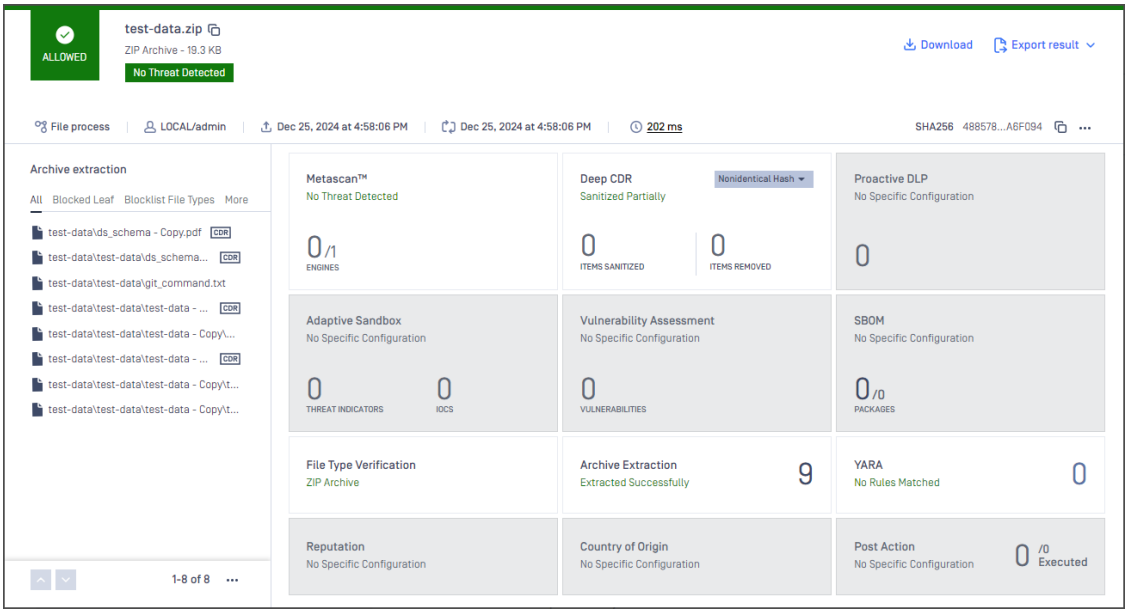
Process

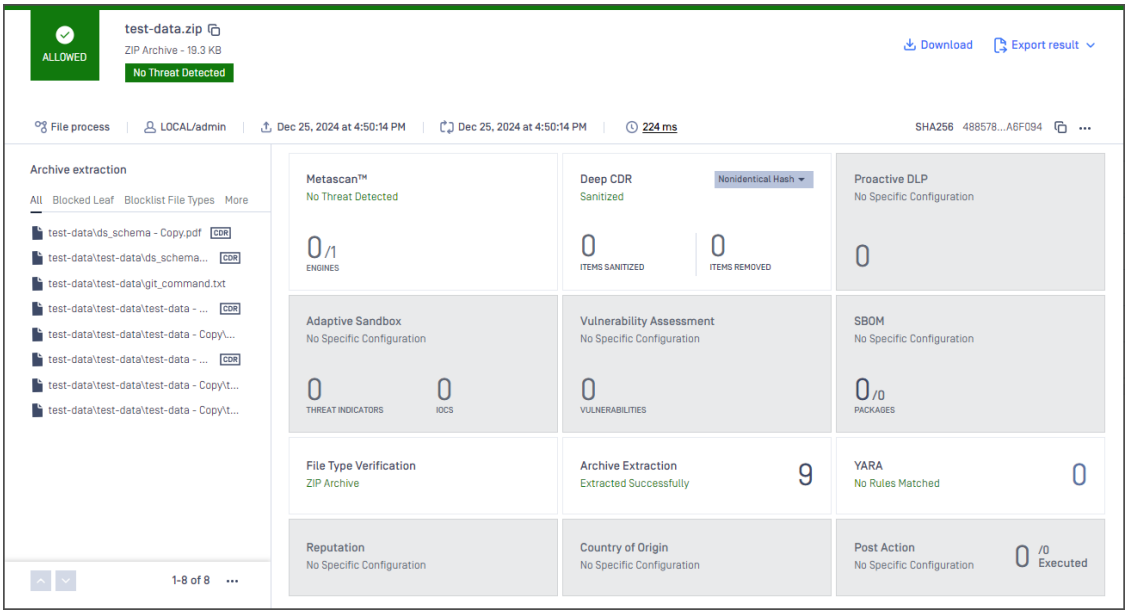
- **Skip Encrypted Items**: When enabled, encrypted files that cannot be decrypted are excluded from the output during the compression of the processed archive.
- **Skip Malicious Items**: When enabled, files identified as malicious are excluded from the output during the compression of the processed archive. You can configure which verdicts should be skipped as malicious:



Archive Sanitization

- **Enable archive sanitization**: When this option is enabled, child files inside archive will be sanitized. Otherwise they will not be processed.
- **Distinguish partial archive process result**: When this option is enabled, if any child file within an archive cannot be sanitized for any reason (e.g., unsupported file type, unselected file type for sanitization, etc.), the sanitization result will be marked as 'Sanitized Partially.' Conversely, if all child files within the archive are successfully sanitized without failures or timeouts, the sanitization result will be marked as 'Sanitized.'





- **Add tombstone file during archive process**: please refer to this page for more detailed information: Tombstone file

Archive Compression for DLP

When analyzing an archive file, an aggregated archive file will be produced, that contains allowed files and files processed by Proactive DLP.

> **Info**
>
> Should not turn on this option along with Enable Archive Sanitization option, it can make the performance down.

Retain password protection on supported files

Select the supported archive file types for which you want to retain the original password.

Timeout

The designated timeout duration for the compression operation.

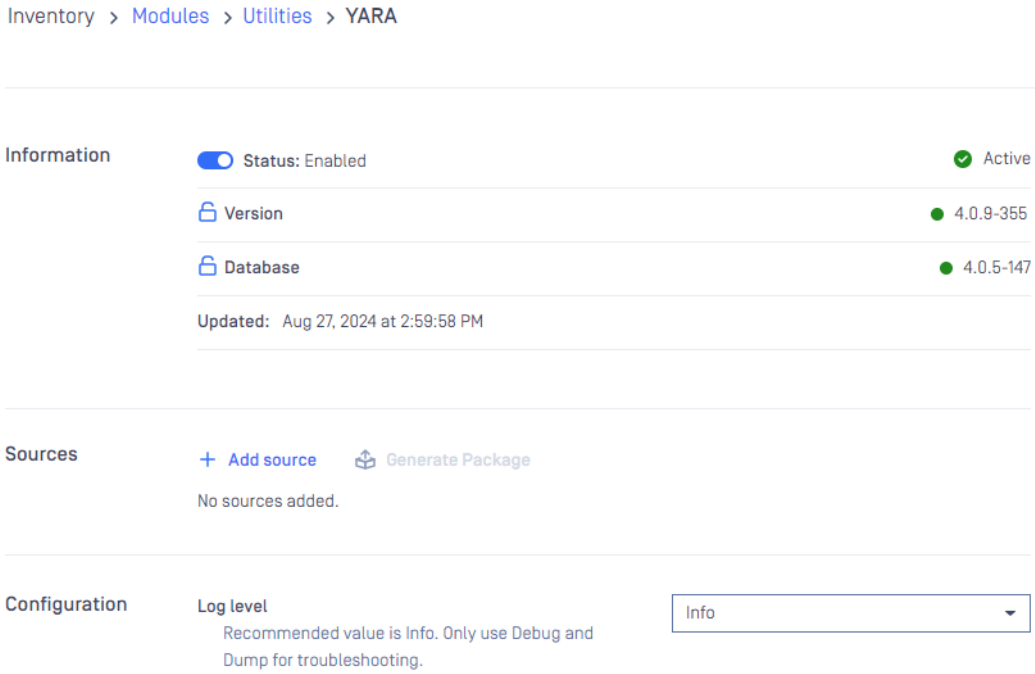
Enable for archive compression filetypes

Select the supported archive file types to apply compression, whether for archive sanitization or DLP processing.

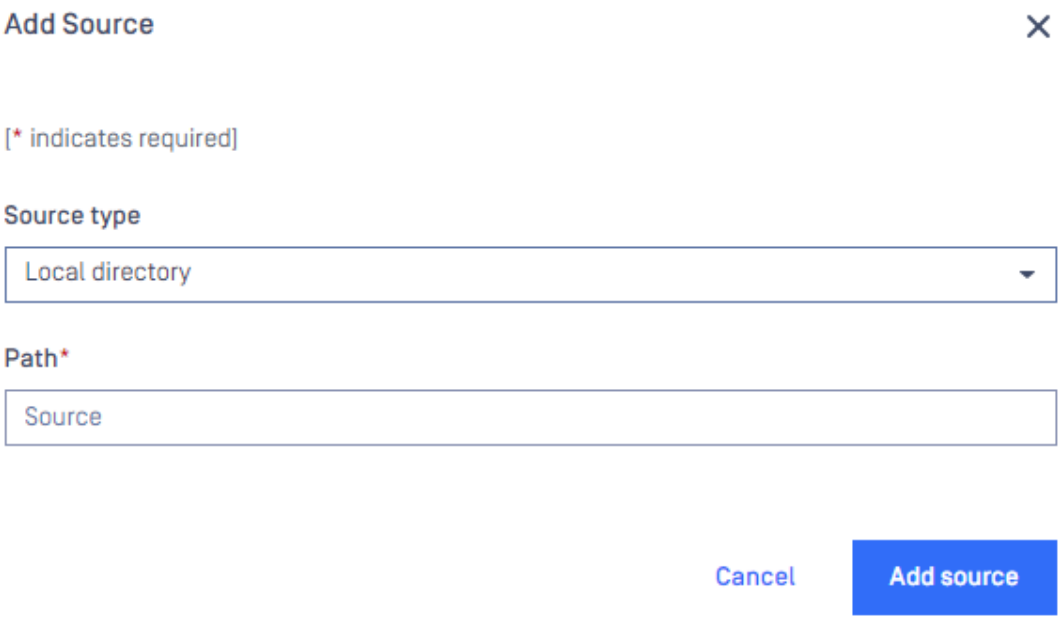
YARA Engine

Rule sources

Under **Inventory/Modules** menu it is possible to configure custom **YARA sources.**



To add new source, click on **Add new source** button. You can specify the type of the source, and the URL itself. The product supports 2 types of YARA sources: local directory and network source (HTTP/HTTPS). A network source must be a zip file. The content of the zip file will be used by the YARA engine. As a local directory, you can set a local path on the computer. This path must point to a directory. A copy of this directory will be used by the YARA engine.



To generate a package, click on the **Generate package** button. This will start the process, and the start time will be shown next to the buttons. Next to the sources, you can enable or disable the sources. Disabled sources will not be used when generating the next package.

Sources can be modified by clicking the row and removed by clicking the trash icon on the right side of the rows.

Please note that the included YARA modules are the following:

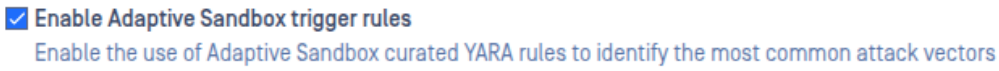
- Magic
- Hash
- Dotnet
- Macho
- Dex
- Cuckoo
- Androguard

For more details, check YARA modules documentation.

Configuration

Enable Adaptive Sandbox rules

The following option can be used to enable Adaptive Sandbox curated rules to identify the most common attack vectors. These rules are only used to find potentially malicious PE files, not meant to be used as a blocklist.



Result with Sandbox rule

Matched Adaptive Sandbox Trigger Rule	Description
Sandbox.FC_Microsoft_Windows_IsProcessorFeaturePresent_Function	-
Sandbox.pe_embedded_pe	Discover embedded PE files, without relying on easily stripped/modified header strings.
Sandbox.pe_unbalanced_virtual_physical_ratio	-

Info

Starting from Core version 5.14.0 and Adaptive Sandbox 2.3.0, these rules can be used to trigger Sandbox scans based on the results.

Upgrading from YARA to YARA-X

Differences between YARA and YARA-X

YARA-X introduces several changes and enhancements compared to traditional YARA. These differences can affect

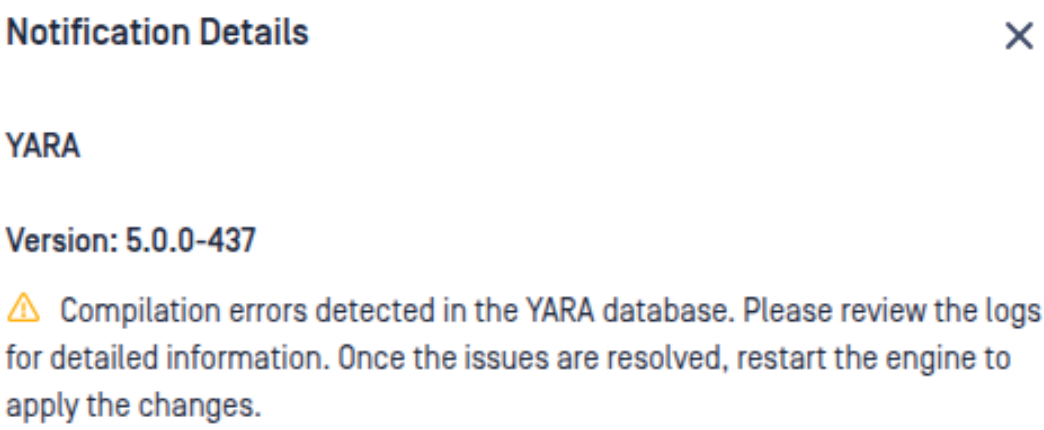
how rules are written, interpreted, and executed.

For a detailed comparison, refer to the official documentation: YARA-X vs YARA Rule Differences

This resource outlines key distinctions in syntax, behavior, and supported features.

Mitigating issues

If there are any problems with the uploaded YARA rules, they will be indicated in the **Core UI**.

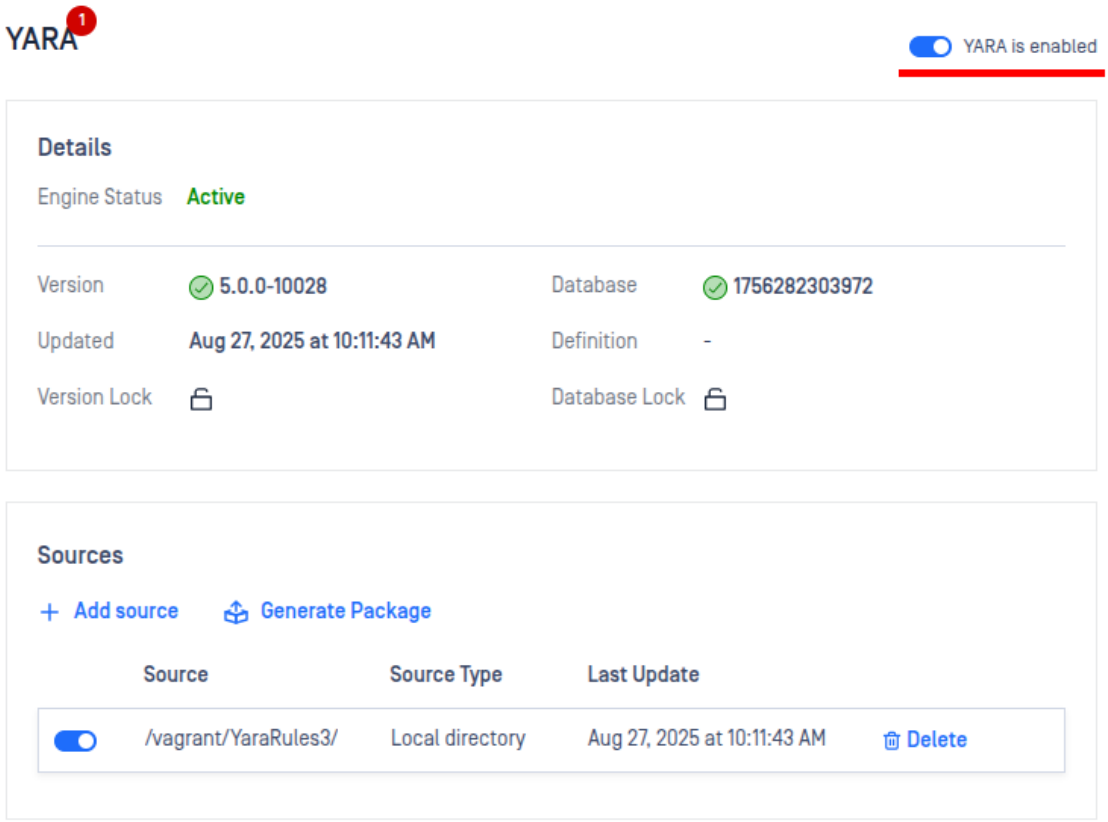


To investigate the issue further, you can:

- Check the **YARA engine log files** (accessible via the **Logs** section).
- Use the **YARA-X CLI tool** for detailed diagnostics.

Resolution Steps

1. **Fix the identified issues** in your YARA rules.
2. **Re-upload the corrected rules** to the system.
3. **Restart the engine** by disabling and then re-enabling it. This will trigger a new rule validation check.



Attack Vector Detection Coverage

Overview

The MetaDefender Core YARA engine provides comprehensive threat detection across the attack lifecycle, from initial delivery through execution and evasion. This analysis examines 80+ hand-picked and high efficiency detection rules across 10 major attack vector categories, focusing on the threats our engine identifies and the detection strategies employed.

Attack Vectors Covered:

1. **Document-Based Attack Vectors** - Malicious Office documents, PDFs, and RTF files
2. **Script and Macro-Based Execution** - PowerShell, JavaScript, VBA, and scriptlet attacks
3. **Encoding and Obfuscation Techniques** - Base64, XOR, hex encoding, and multi-layer obfuscation
4. **PE Structure Manipulation and Evasion** - Executable format modifications and packing
5. **Import Table and Dynamic Loading** - API hiding and dynamic resolution techniques
6. **Embedded Content and Payload Hiding** - Hidden executables in resources and overlays
7. **Network and Communication Vectors** - C2 channels and legitimate service abuse
8. **Shortcut and Link Abuse** - Malicious LNK files and hyperlink manipulation
9. **Cryptographic and Encoding Functions** - Encryption libraries and protocol obfuscation
10. **Anti-Analysis and Evasion** - VM detection and environment fingerprinting

1. Document-Based Attack Vectors

Description

Document-based attacks exploit legitimate file formats to deliver payloads or execute malicious code, leveraging user trust in common document types.

Detection Examples:

Key rules in this category include such as `FC_Encrypted_Office_Document` which identifies encrypted documents used to bypass security scanning, and `FC_PDF_Launch_Function` which catches PDFs with dangerous execution capabilities. `FC_Office_Document_with_VBA_Project` detects macro-enabled documents, among other document-based detection rules.

Required Rule Groups:

- Microsoft Office Documents
- PDF
- OLE (Object Linking and Embedding)
- Email Attachments
- OneNote
- Exploits

Key Associated MITRE ATT&CK Techniques:

- **T1204.002 - Malicious File (User Execution)**
- **T1137 - Office Application Startup**
- **T1566.001 - Spearphishing Attachment**

2. Script and Macro-Based Execution

Description

Script-based attacks leverage legitimate system interpreters to execute malicious code, using "living off the land" techniques with PowerShell, JavaScript, VBA and such script languages.

Detection Examples:

Notable examples include `FC_File_with_JavaScript_Script_Block` which detects JavaScript script blocks in files, and `FC_Scripting_VB` which identifies Visual Basic scripting content. `SC_JavaScript_Double_Indirect_Call` targets sophisticated JavaScript indirection methods, representing several of the script execution detection capabilities.

Required Rule Groups:

- JavaScript in HTML
- Script Execution
- Microsoft Shortcut (LNK)
- Exploits

Key Associated MITRE ATT&CK Techniques:

- **T1059.001 - PowerShell**
- **T1059.005 - Visual Basic**
- **T1059.007 - JavaScript/JScript**

---

3. Encoding and Obfuscation Techniques

Description

Obfuscation and encoding techniques hide malicious content from static analysis and security tools using various encoding schemes and data manipulation methods.

Detection Examples:

Among the obfuscation detection rules, `pe_double_base64d_pe` identifies executables encoded twice with Base64 for sophisticated evasion, and `pe_resource_single_byte_xor_PE` detects XOR-based obfuscation in PE resources. `FC_File_with_UTF8_and_Base64_Encoded_Strings` identifies multi-layer encoding schemes across different file formats, illustrating the breadth of encoding detection methods.

Required Rule Groups:

- Encoding
- Resources, overlay, and embedded files

Key Associated MITRE ATT&CK Techniques:

- **T1027 - Obfuscated Files or Information**
- **T1140 - Deobfuscate/Decode Files or Information**

---

4. PE Structure Manipulation and Evasion

Description

PE structure manipulation involves modifying executable file structures to evade analysis tools while maintaining functionality, using techniques like header anomalies and section manipulation.

Detection Examples:

Representative rules include `pe_section_rwx` which identifies dangerous read-write-execute sections used for dynamic code generation, and `pe_high_entropy_section` using entropy analysis to detect packed or encrypted content. `pe_overlapping_sections` catches deliberate PE manipulation designed to confuse analysis tools, showcasing the engine's comprehensive PE analysis capabilities.

Required Rule Groups:

- PE Header
- PE Sections
- PE Imports
- Signature

Key Associated MITRE ATT&CK Techniques:

- T1027.002 - Software Packing

---

5. Import Table and Dynamic Loading

Description

Import table manipulation hides malware functionality by using dynamic API resolution instead of static imports, making analysis more difficult.

Detection Examples:

Standout detection methods include `pe_dynamic_api_resolution_imports` which identifies the classic LoadLibrary/GetProcAddress pattern used to hide functionality, and `pe_zero_imports` which catches completely import-free executables (often indicating packing). `pe_imports_by_ordinal` targets obfuscated imports using ordinal numbers instead of function names, demonstrating various approaches to import analysis.

Required Rule Groups:

- PE Imports

Key Associated MITRE ATT&CK Techniques:

- T1055 - Process Injection
- T1129 - Shared Modules (Dynamic Loading)
- T1027.002 - Software Packing

6. Embedded Content and Payload Hiding

Description

Payload hiding techniques leverage file structures to conceal malicious content within legitimate containers, using resources, overlays, and encoding methods.

Detection Examples:

Prominent examples include `pe_embedded_pe` which identifies executables hiding other PE files using header-independent detection, and `pe_resource_base64d_pe` which catches Base64-encoded executables concealed in resource sections. `pe_large_overlay` flags suspiciously oversized overlays used for payload storage, reflecting the diverse payload hiding detection techniques available.

Required Rule Groups:

- Resources, overlay, and embedded files

Key Associated MITRE ATT&CK Techniques:

- T1105 - Ingress Tool Transfer
- T1027 - Obfuscated Files or Information

7. Network and Communication Vectors

Description

Network-based vectors encompass delivery mechanisms and communication channels, often abusing legitimate services to blend with normal traffic.

Detection Examples:

Selected examples include `FC_File_with_Amazon_SES_Redirect` which identifies abuse of legitimate Amazon email services for malware distribution, and `FC_WinHttpRequest_Object` which detects embedded HTTP objects used for C2 communication. `FC_FTP_Credentials` catches embedded authentication data suggesting exfiltration preparation, highlighting key network-based threat detection capabilities.

Required Rule Groups:

- Application Layer Protocols
- URL Redirection and External Services

Key Associated MITRE ATT&CK Techniques:

- T1041 - Exfiltration Over C2 Channel
- T1567 - Exfiltration Over Web Service
- T1071 - Application Layer Protocol

---

8. Shortcut and Link Abuse

Description

Shortcut and link abuse leverages LNK files and hyperlinks to execute commands or redirect users while appearing as harmless shortcuts.

Detection Examples:

The primary rules in this limited category are `SC_Microsoft_LNK_Commands` which identifies malicious shortcuts with complex command sequences used for script execution or payload downloading, and `FC_HTML_JavaScript_with_Hyperlink_Reference` which detects JavaScript-based hyperlink manipulation. These rules target user execution tricks and social engineering tactics that exploit trust in familiar link formats.

Required Rule Groups:

- Microsoft Shortcut (LNK)

Key Associated MITRE ATT&CK Techniques:

- T1547.009 - Shortcut Modification
- T1204.002 - Malicious File (User Execution)

---

9. Cryptographic and Encoding Functions

Description

Cryptographic and encoding functions detect the presence of encryption libraries and encoding schemes used for payload protection or communication encryption.

Detection Examples:

Core examples from this category include `FC_File_with_CryptoJS_Decrypt_Function` which identifies suspicious CryptoJS usage in documents where it's unexpected and often malicious, and `FC_File_with_CryptoJS_Encode_Function` which detects CryptoJS encoding functions. These rules address cryptographic library usage in unexpected contexts, among other cryptographic detection methods.

Required Rule Groups:

- Encoding

Key Associated MITRE ATT&CK Techniques:

- T1573 - Encrypted Channel
- T1027 - Obfuscated Files or Information

---

10. Anti-Analysis, Evasion and Privilege Elevation

Description

Anti-analysis techniques detect and evade security analysis environments, ensuring malware only executes in intended target systems. Such attacks include

Detection Examples:

Notable examples include `SC_User_Access_Control_Bypass` which identifies techniques to bypass Windows User Access Control for privilege escalation, and `pe_unusual_entrypoint_section` which detects executables with entry points in unexpected sections used to evade analysis. These complement our more sophisticated anti-analysis detection capabilities deployed across the engine.

Required Rule Groups:

- Archive files and packing techniques
- Exploits
- PE Header

Key Associated MITRE ATT&CK Techniques:

- T1497 - Virtualization/Sandbox Evasion

Rule Group Mapping Note

The rule group categorizations provided in the "Required Rule Groups" sections represent a best-effort mapping between the existing rule group structure and the attack vector categories. While these mappings provide comprehensive coverage for each attack vector, they are not perfect 1:1 matches due to the different organizational approaches between rule groups and attack vector classifications. Some rule groups may contain rules that extend beyond the specific attack vector category, and some attack vectors may benefit from rules in multiple groups. The suggested groupings are designed to provide optimal detection coverage for each attack vector category.

Why are Japanese TXT files detected as "application/octet-stream" instead of "text/plain"?

Check Your Article:

This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems using **FileType engine version 7.3 and later**.

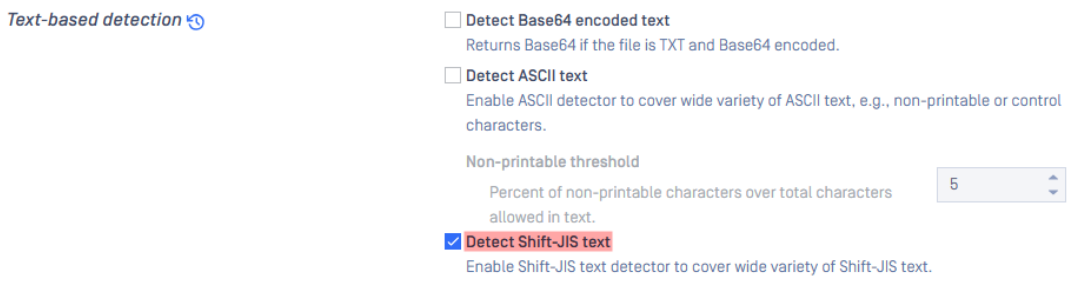
Issue: When processing plain text files containing Shift-JIS encoded characters (e.g., Japanese text), the files are detected as "**data**" **(application/octet-stream)** instead of "**ASCII text**" **(text/plain)**.

Root Cause: This occurs because **Shift-JIS recognition** is not enabled by default in the **FileType engine**.

Remediation Steps:

To ensure Shift-JIS encoded text files are properly detected as text, follow these steps:

1. Open **MetaDefender Core** configuration settings.
2. Navigate to:
   - **Workflow File Type Text-based detection**
3. Enable the option:
   - **Detect Shift-JIS text**
4. Save the configuration changes.



With this configuration, **Shift-JIS encoded files** will be accurately recognized as "**text/plain**" instead of "**application/octet-stream**".

Additional Information: This feature is available in **FileType engine version 7.3 and later**. If you are using an older version, please refer to the documentation for instructions on configuring the Engine Update.

Support:

If you have followed through this article but are still unable to remediate **Japanese TXT files detected as "application/octet-stream"**, please follow these instructions on How to Create Support Package With Bundle Tools?, before **creating a support case or chatting with our support engineer**.

How to Connect a MetaDefender Core Container to the same instance in Shared Database Mode?

Introduction:

This knowledge base article provides detailed instructions on deploying MetaDefender Core in a containerized environment using a shared database.

Details:

If you have a MetaDefender Core container deployment and wish to upgrade or make changes that involve removing and recreating one or more MetaDefender Core containers, but you want to retain all previous configurations and historical data, it is important to proceed carefully. To ensure continuity, the following environment variables are mandatory in order to connect to the same instance. Failing to use these variables will result in the creation of a new MetaDefender Core instance:

- *MD_INSTANCE_NAME*
- *MDCORE_DB_NAME*

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Can local AVs interrupt ongoing scans?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Yes, local AV's may interrupt ongoing scans.

Since MetaDefender Core does not include a real-time AV protection engine, many users may opt to run one or more additional real-time AV agents alongside MetaDefender Core.

Often, these anti-malware agents may already be installed on the system before the MetaDefender Core is deployed.

Please Note:

If you already have an anti-malware product installed on your system which also happens to be one of the anti-malware engines in your version of MetaDefender Core, it will interfere with the MetaDefender Core scanning process.

Similarly, these additional anti-malware agents may be responsible for blocking your MetaDefender Core files.

Pro Tip:

For this reason, OPSWAT recommends that you disable the real-time protection of your anti-malware product/s. If your corporate policy does not allow you to disable your real-time anti-virus product, you will need to add some exception rules.

To implement your exception rule, you must exclude the following from the relevant anti-malware product's real-time protection:

- Exclude full installation path of MetaDefender Core.
- Exclude the temporary upload path used by MetaDefender Core.
- Exclude specific processes:
  - On Windows: "engineprocess.exe", "engineprocess32.exe", "ometascan.exe", "postgres.exe" and "nginx.exe" processes.
  - On Linux: "engineprocess", "ometascan", "postgres" and "nginx" processes.

If you do not add this exception or if you do not disable real-time protection, MetaDefender Core files may be blocked, results returned by MetaDefender Core scanning will not be consistent, and the return value of the scans would be one of the following:

- **Clean**: If your existing anti-malware product was able to clean the threat
- **Failed** (or other errors): If your existing anti-malware product removed the file before MetaDefender Core could scan it.

Please Note:

If you are using Symantec Endpoint Protection as your local AV, please adjust the settings as instructed in How do I prevent Symantec Endpoint Protection from interfering with the optimal functioning of MetaDefender Core?

Pro Tip:

Should you require a real time protection agent to be installed in addition to MetaDefender Core, please ensure that it is not a product featured among the MetaDefender Core's included scanning engines.

Support:

If you need help learning how to add an exception rule (to exclude a given folder from an AV products scan area) please tell us what product you are using and we may be able to help you - but be sure to include the product version.

To contact us, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

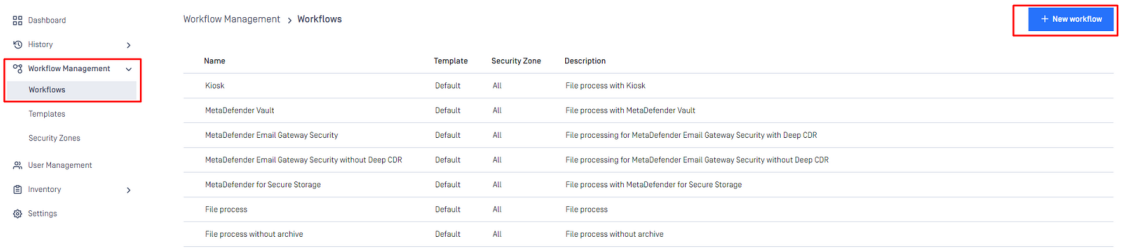
Can I create new workflows?

Check Your Version:

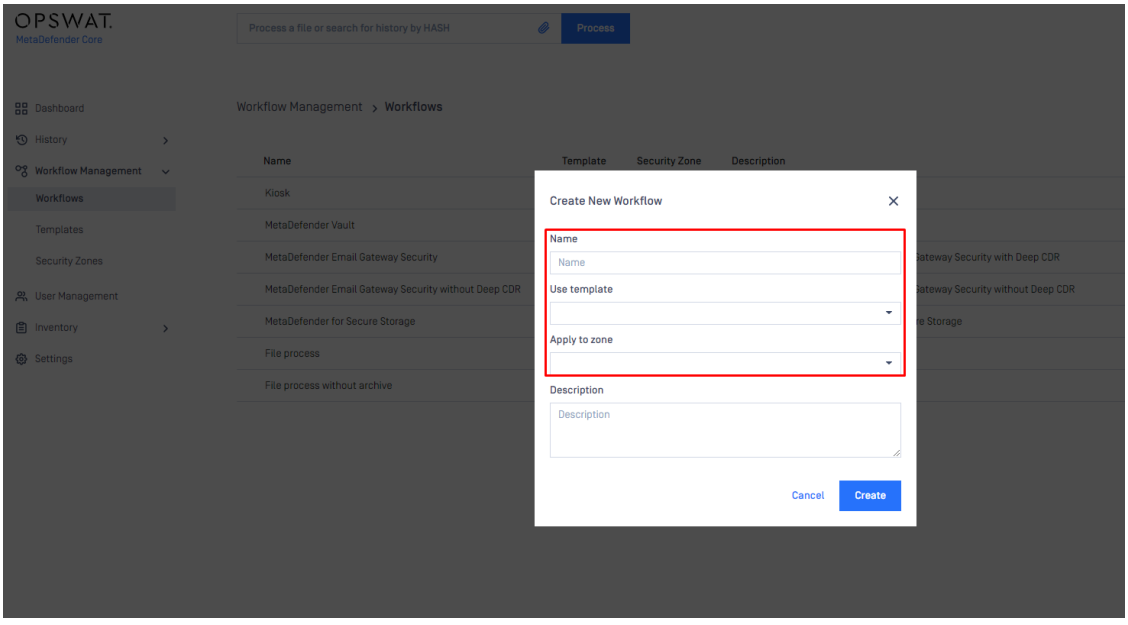
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

In order to create new custom workflow rules on the MetaDefender Core product, you will need to perform the following steps:

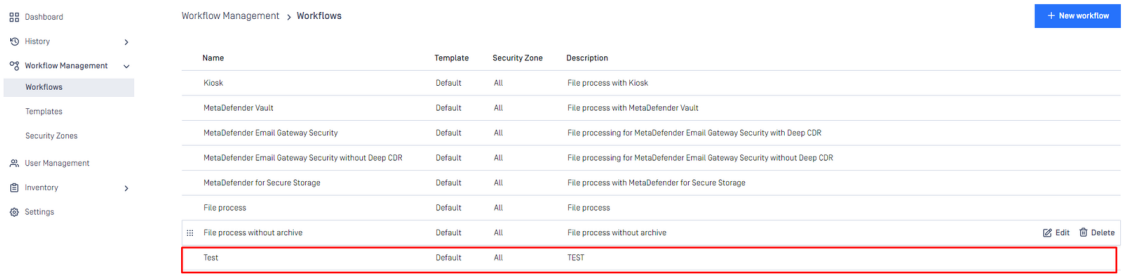
- Navigate to the Core console and access the Workflows tab.



- Click on the “New workflow” tab and provide the requested information.



- Once the workflow rule has been created it is visible at the bottom of the list.



Support:

If **Further Assistance** is required, please follow the instructions on How to Create Support Package With Bundle Tools?, then proceed to create a support case or chat with our support engineer.

How to enable crash dumps for MetaDefender Core processes?

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows systems.

Overview

Windows Error Reporting (WER) can automatically generate crash dump files when a process fails unexpectedly. Configuring WER through the Windows Registry allows administrators to capture `.dmp` files for MetaDefender Core components such as `engineprocess.exe` and `ometascan.exe`.

Prerequisites

- Administrative access to the Windows system.
- Adequate free disk space (full dumps can be several GB).
- The MetaDefender Core service account must have permission to write to the dump folder.

Steps to Enable LocalDumps for MetaDefender Core

Step 1: Create the dump folder

1. Create a directory, for example: C:\Dumps\MetaDefender

2. Grant **SYSTEM** and the MetaDefender Core service account write permissions.

Step 2: Configure the registry

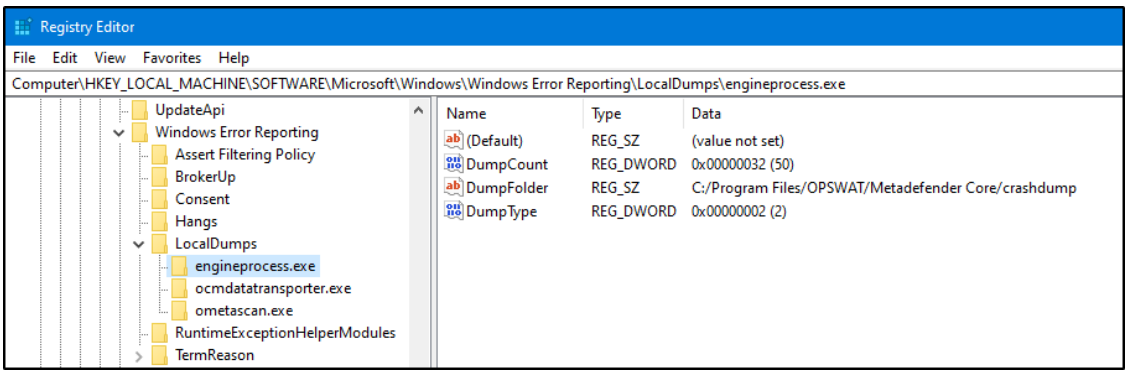
Create the following registry keys and values for each process if missing:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps\engineprocess.exe

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps\ometascan.exe

Under each key, add:

Value Name	Type	Data
DumpFolder	REG_EXPAND_SZ	C:\Program Files\OPSWAT\MetaDefender Core/crashdump
DumpCount	DWORD	32
DumpType	DWORD	2 (Full dump) or 1 (Mini dump)



You can automate this using a `.reg` file:

Code

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error
Reporting\LocalDumps\engineprocess.exe]
"DumpFolder"="C:\Program Files\OPSWAT\MetaDefender Core\crashdump"
"DumpCount"=dword:00000032
"DumpType"=dword:00000002
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting\LocalDumps\ometascan.exe]
"DumpFolder"="C:\Program Files\OPSWAT\MetaDefender Core\crashdump"
"DumpCount"=dword:00000032
"DumpType"=dword:00000002
```

Step 3: Restart services

After creating the registry entries, restart the MetaDefender Core services or reboot the system to apply changes.

---

4) Verification

When a crash occurs, WER should generate `.dmp` files in the specified folder (e.g., `C:\Program Files\OPSWAT\MetaDefender Core\crashdump`).

If dumps are not generated:

- Verify if WER is not disabled by Group Policy.
- Confirm the process has permission to write to the dump directory.
- Check **Event Viewer > Application** for WER-related errors.

---

5) Notes

- `DumpType=1` generates a small (mini) dump, suitable for basic debugging.
- `DumpType=2` generates a full dump, preferred for OPSWAT Support investigations.
- Only enable dumps temporarily to avoid excessive disk usage.

> **Support:**
>
> If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

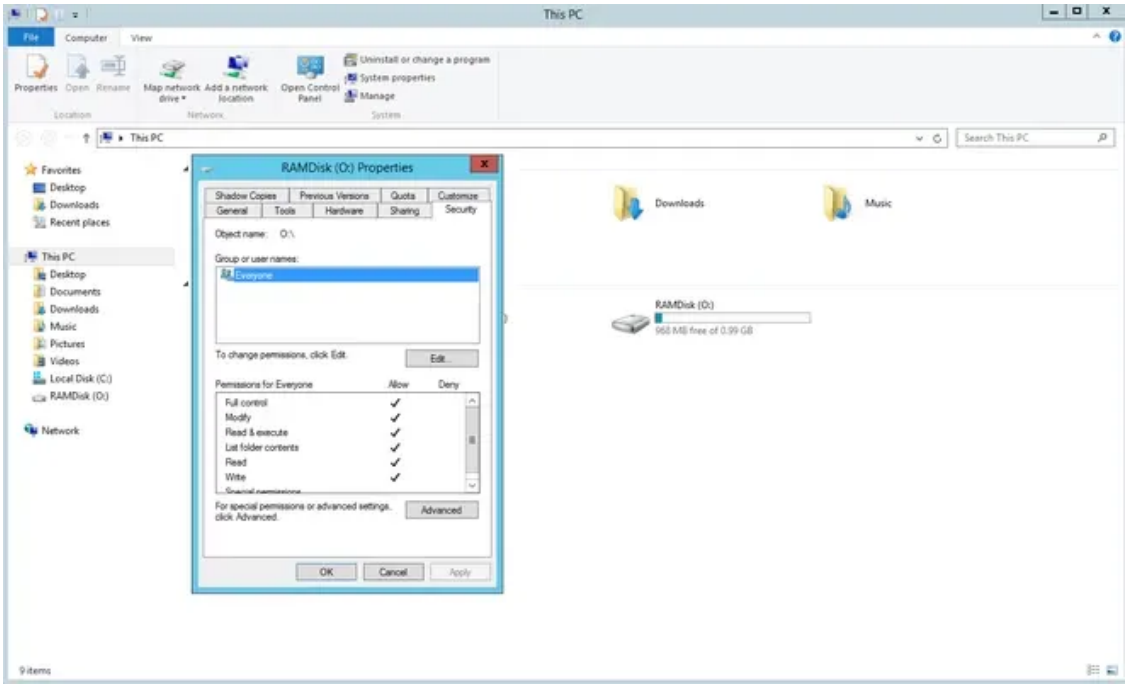
Can I control access to the RAM disk in MetaDefender Core v5?

> **Check Your Version:**
>
> This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Yes.

You can set security permissions for the RAM disk just like a regular hard disk.

To do so, right-click on the **RAM Drive**, select **Properties**, and then navigate to the **Security** tab to access security permissions for that drive.



Support:

If you have followed all of the steps above but are still unable to set the desired **RAM Drive** security permissions, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I prevent Symantec Endpoint Protection from interfering with the optimal functioning of MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Please Note:

If Symantec Endpoint Protection (SEP) is used as a local AV on the machine where MetaDefender Core resides, it can interfere with the correct functioning of the OPSWAT product - unless SEP settings are updated as follows.

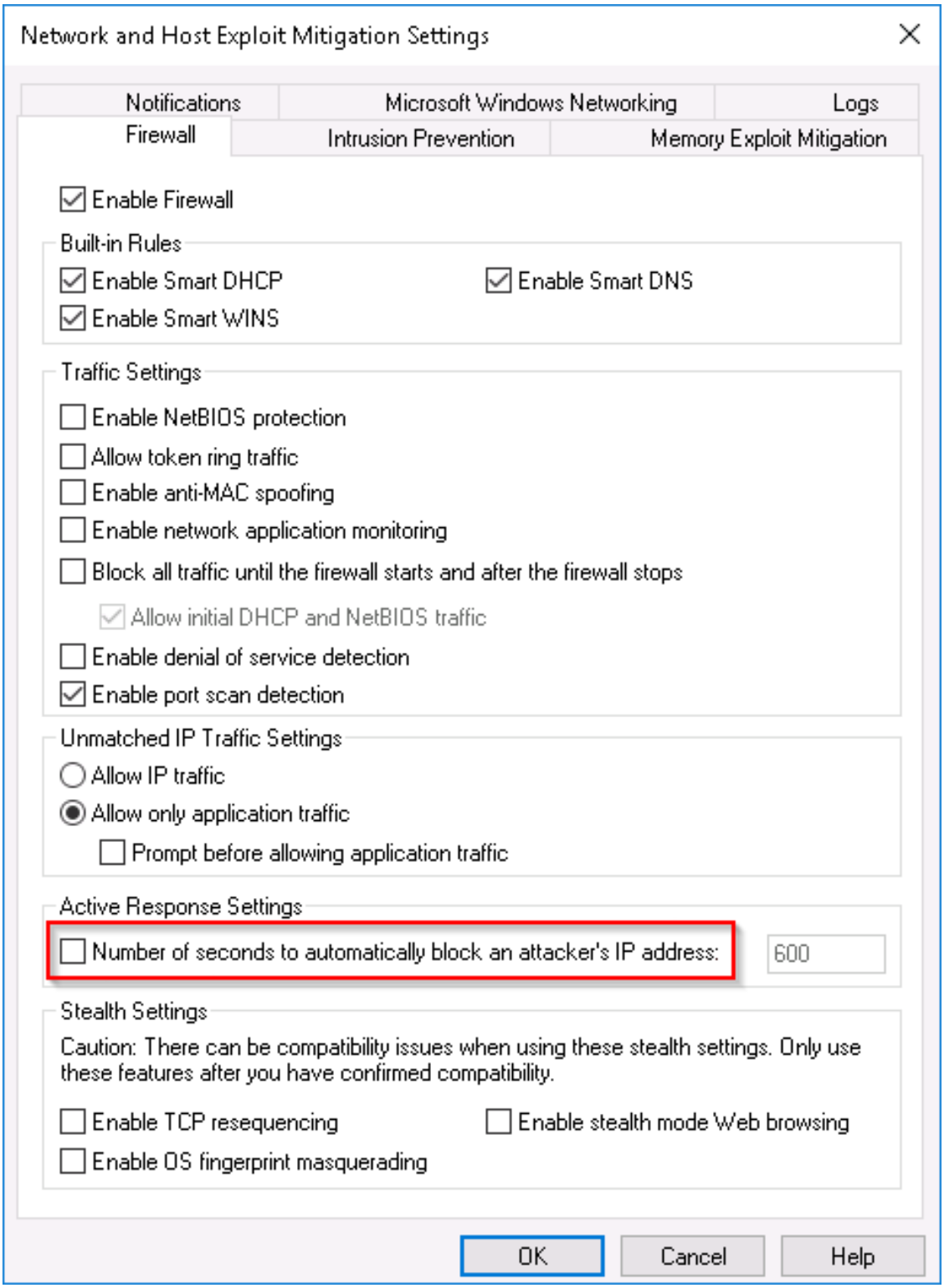
Update your SEP settings as follows to ensure optimal functioning of MetaDefender Core:

1. Exclude the folder where MetaDefender Core is installed from the SEP real-time protection:

- Add security risk exception *> *Folder > C:\Program Files\OPSWAT**

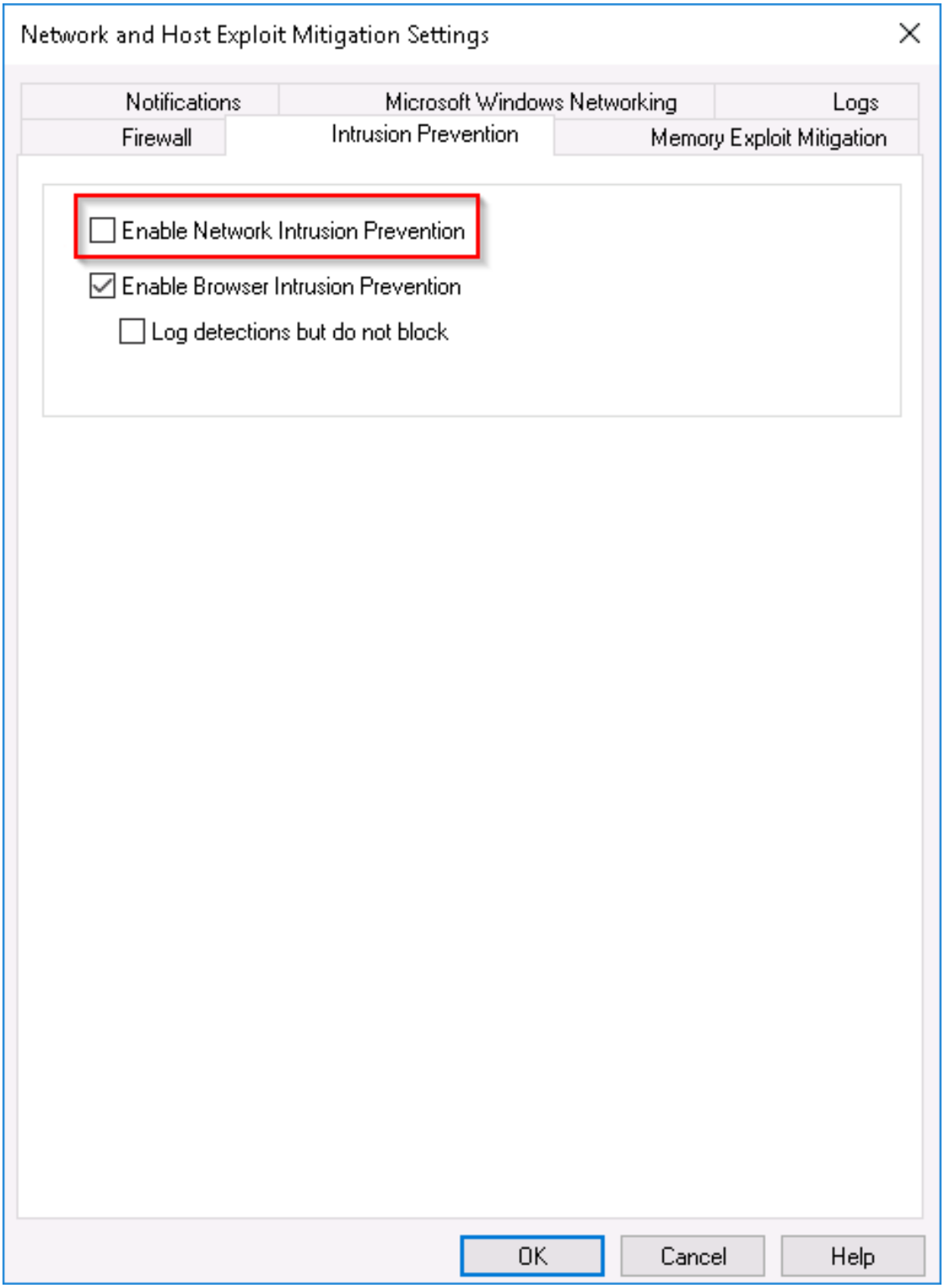
2. Go to **Network and Host Exploit Mitigation Settings > Firewall**:

- Uncheck the **Number of seconds to automatically block an attacker's IP address** option, as follows:



3. Go to **Network and Host Exploit Mitigation Settings > Intrusion Prevention**:

- Uncheck the **Enable Network Intrusion Prevention** option, as follows:



Support

If you have followed all of the steps above but are still experiencing issues, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What is the retry mechanism for MetaDefender Core connecting to the remote Postgres Database?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core will automatically attempt to reconnect to the remote PostgreSQL database server in the event of a failure. The retry mechanism is configured to initiate a new attempt every 500 milliseconds, continuing until either a 180,000 millisecond timeout is reached or a maximum of 120 retries is completed.

Support:

If **Further Assistance** is required then please proceed to create a support case or chat with our support engineer.

How to use MetaDefender Core V5 Blocklist/Allowlist feature?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

MetaDefender Core V5.x includes the option to block or allow files by creating a Blocklist or Allowlist. The user can select files to be blocked or allowed based on:

- Filetype group
- MIME-type
- Filename

Full documentation can be found here: Workflow Configuration Template

The conventional usage of this feature would be, to create a list of files to be blocked or allowed, by any of the three selectors mentioned above (filetype groups, mime-types, file names) or a combination of them.

Using filetype groups VS. MIME-types VS file extensions

When possible, It is better to use Filetype groups over MIME types, and MIME types over File names. It is shorter to define, leaves less space for human error and can also leverage OPSWAT's file detection mechanism, so that even if an imposter file has the extension .doc but in reality, it is a .exe it will be treated as .exe

Using Regular Expressions

The rules we create can consist of literal strings but can also include wild cards in the form of Regular Expressions.

For example, if we use the string : ^.*.docx$ in the "Blocklist by file names", it will test as True for all files whose name is terminating with docx. Each file processed by MetaDefender Core will be tested against the rules defined in the Blocklist. As soon as any of the rules tests as True the file will be blocked.

Advanced usage

Sometimes the business rule is something like "Block all files except...". Such a scenario is accommodated in the system by the usage of Regular Expressions. In Regular Expression, we can create an expression that will test as True when a certain string is NOT found (known as Negative Look Ahead) For example, if we use the string: ^..*((?!docx$).)*$ in the "Blocklist by file names", it will test as True for files whose names do NOT terminate with docx.

- To make the above Regular Expression case insensitive we can use: ^..*((?![dD][oO][cC][xX]$).)*$

In many cases, we will need to allow more than one file type. For example, if we use the string: ^..*((?!docx$)(?!xls$).)*$ in the "Blocklist by file names", it will test as True for files whose names do NOT terminate with either docx or xls. The Negative Look Ahead block (?!XYZ$) can be repeated as many times as required.

In the example given above "Block all files except docx" there is a hidden problem. .docx files are actually archive files, containing other files (such as xml, gif, jpeg etc...) This means that if the business rule is **block everything except docx** it most likely means **block everything except docx and all the files it contains.**

Pro Tip:

You can use a tool such as regex101 to create and test your regular expressions.

Support:

If **Further Assistance** is required then please proceed to create a support case or chat with our support engineer.

How can I configure the maximum queue size in MetaDefender Core v5 ?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

There are two ways of configuring maximum queue size in MetaDefender Core.

1. REST API:

The maximum queue size can be configured in MetaDefender Core v5 via REST API, as follows :

Set scan config

Request	Value
Method	PUT
URL	/admin/config/scan

Request HTTP header parameters:

Name	Type	Required	Value
apikey	string	true	Session id, can be acquired by POST - Login

Request body:

JSON path	Type	Required	Value
max_queue_per_agent	int	true	Max queue size allowed per agent

Example:

Code · json

```
{
   "max_queue_per_agent": 700
}
```

An example of a successful response can be found below :

HTTP status code: **200**

Response contains information about the modified scan configuration

Code · json

```
{
   "max_queue_per_agent": 700
}
```

In case the configuration change was not correct, an error response like the one below will be returned :

Internal error

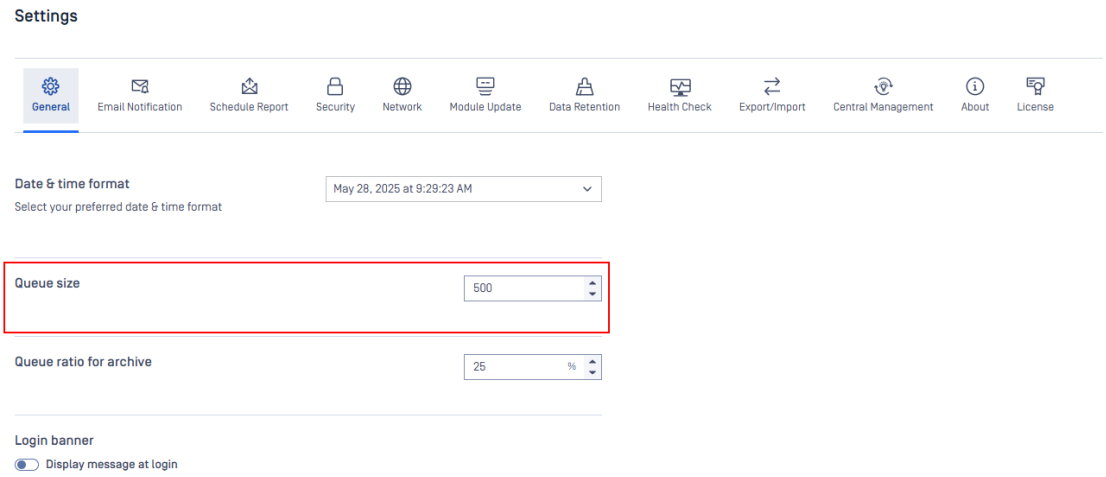
HTTP status code: **500**

Code · json

```
{
    "err": "Error while modifying configuration"
}
```

2. MetaDefender Core UI

You can go to Settings General and change the queue size to the number you want.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with our support engineer**.

How do I remove an engine from my MetaDefender instance?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Please Note:

MetaDefender Core downloads engines based on the license key activated on the system. If an engine is manually removed, MetaDefender Core will automatically download the engine again the next time it updates.

In order to completely remove an engine (so that it disappears from the engine list, with no engine files left on the system or to be downloaded again), you will need to get a new license key that does not include the engine.

To do this, you will need to contact the OPSWAT support team.

Support:

If you have need help removing an engine from your MetaDefender instance, please log a support case or chat with our support engineer.

How can the Temp folder be changed?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

Useful info and best practice recommendations:

- Temporary folder is used to store temporary files processed by MetaDefender Core and its engines.
- The temporary files will be cleaned up once the processing is done.
- The use of high speed IO disk for temp folder is recommended.
- Ensure sufficient space is allocated for the temp drive.

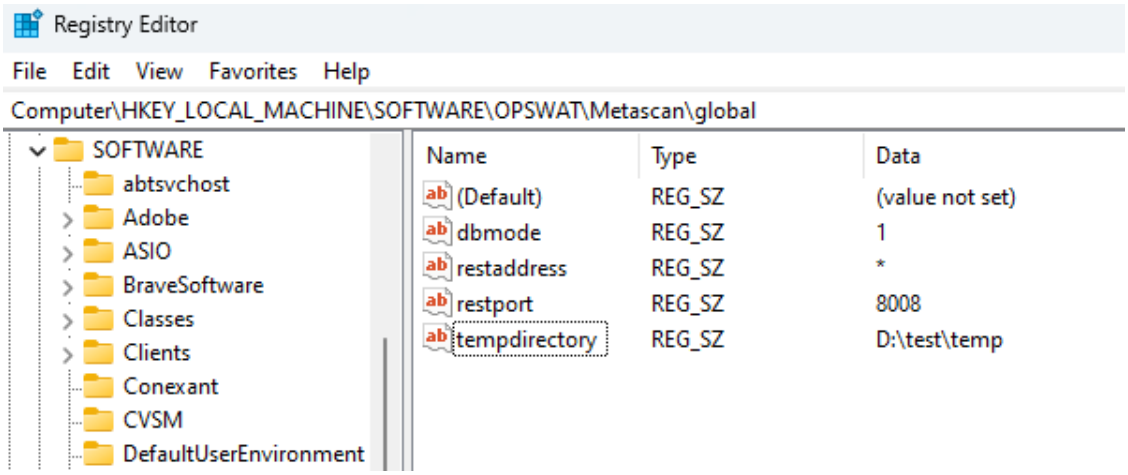
It can be changed by configuring your MetaDefender Core settings as below:

Linux

- Add *tempdirectory* parameter under `[global]` section of the configuration file for the server is located in `/etc/ometascan/ometascan.conf`

Windows

- Open Regedit
- Under `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global` add new string value named `tempdirectory`



- Restart MetaDefender Core service

Support

If you need help configuring TEMP folder, please create a support case or chat with our support engineer.

How can I increase parallel count and Max queue size?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

If you're running a high-performance environment with 32 to 48 CPU cores and need faster file processing speeds, MetaDefender Core allows you to scale performance through two key configurations: queue size and parallel scan count. This article outlines both approaches.

When Should You Scale Up?

You should consider increasing the queue size and parallel processing count **only if**:

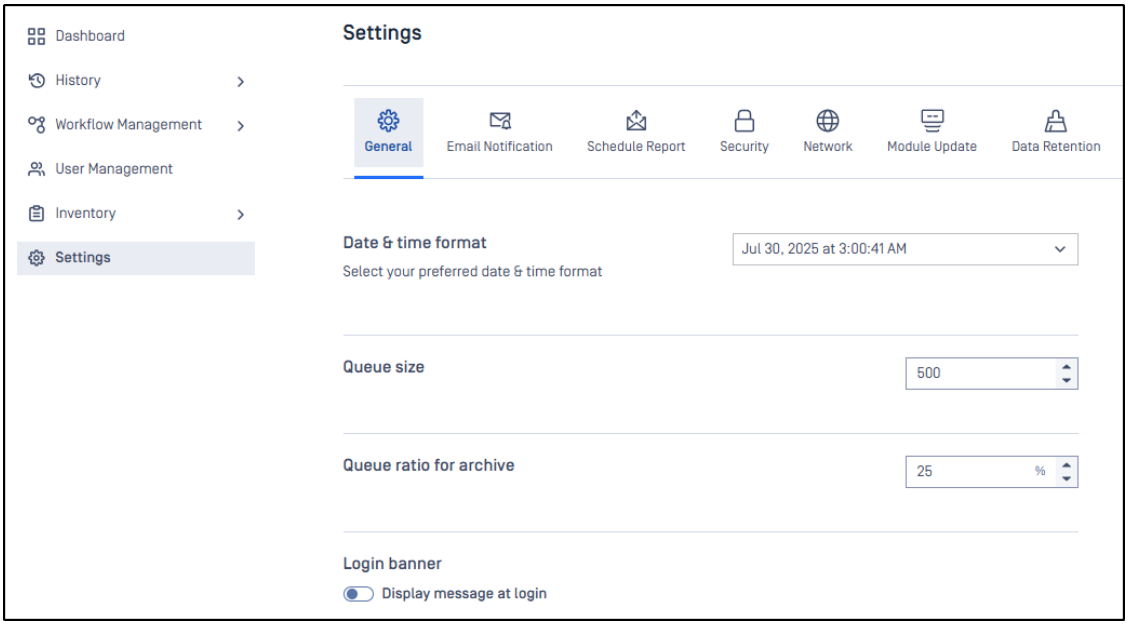
- You have a high number of CPU cores (32–48).
- Fast processing of large volumes of files is critical to your workflow.
- You're experiencing bottlenecks due to a full scan queue or limited parallelism.

Approach A: Increasing Maximum Queue Size

When the MetaDefender Core scan queue is full, **incoming scan requests are denied** until space becomes available. To avoid this and support a higher load, you can increase the **maximum queue size** from the MetaDefender Core UI

or by using the REST API.

The option can be found under the **Settings > General** tab.



If you want to configure the queue using the APIs, use the following request to update the queue size:

`PUT /admin/config/scan`

Request Body:

```
{   "max_queue_per_agent": 500 }
```

> **Warning:**
> An API key is required to authenticate this request. Make sure to retrieve the key from your MetaDefender Core deployment.

For a more detailed guide, refer to the article: How can I configure the maximum queue size in MetaDefender Core v5 ?

Approach B: Increasing Parallel Scan Count per Engine

Another method to maximize CPU utilization is by increasing the **number of parallel scans per engine**. This is especially effective with engines that support **multi-threaded scanning**.

Configuration Key

To configure this setting, use the `parallelcount` key in your MetaDefender Core configuration.

- This setting controls how many scan threads run **simultaneously per engine**.
- Proper tuning can significantly boost performance in multi-core environments.

For details on modifying this setting, see: MetaDefender Configuration

Support:

If you have any issues **Increasing the Scaling Up Performance** on MetaDefender Core, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How can I change MetaDefender Core Deep CDR's timeout settings?

Check Your Version:

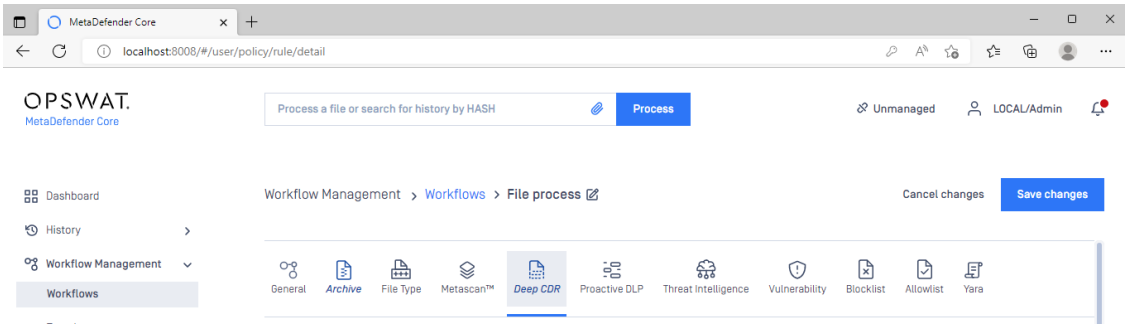
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Pro Tip:

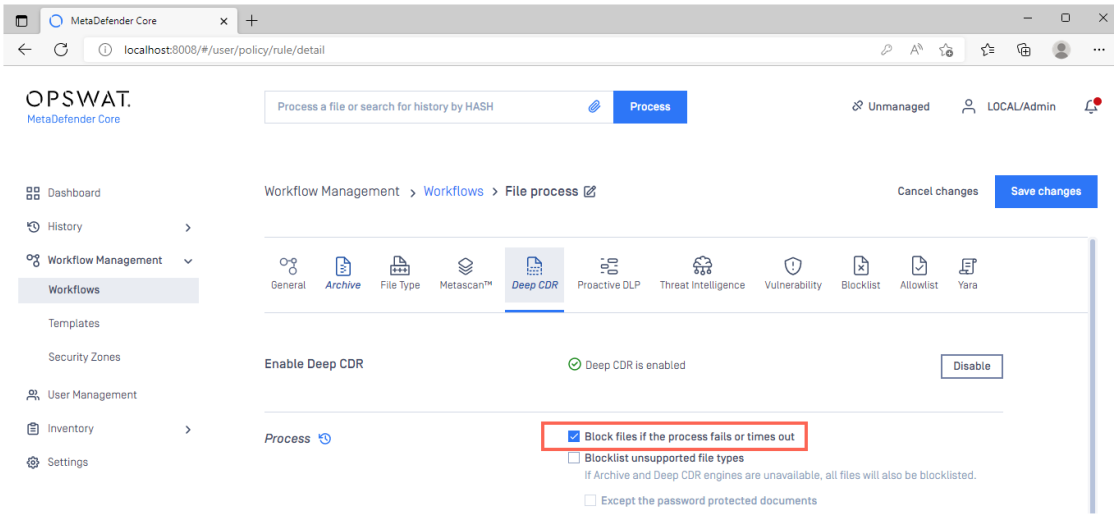
When processing complex files with MetaDefender Core's Deep CDR engine, it's not uncommon for the sanitization to fail due to exceeded timeout settings, particularly in the case of larger, encrypted archive files that take time to extract, process and sanitize.

However, Deep CDR's timeout setting can be easily configured to allow time for processing your anticipated datasets, so that no such failures occur.

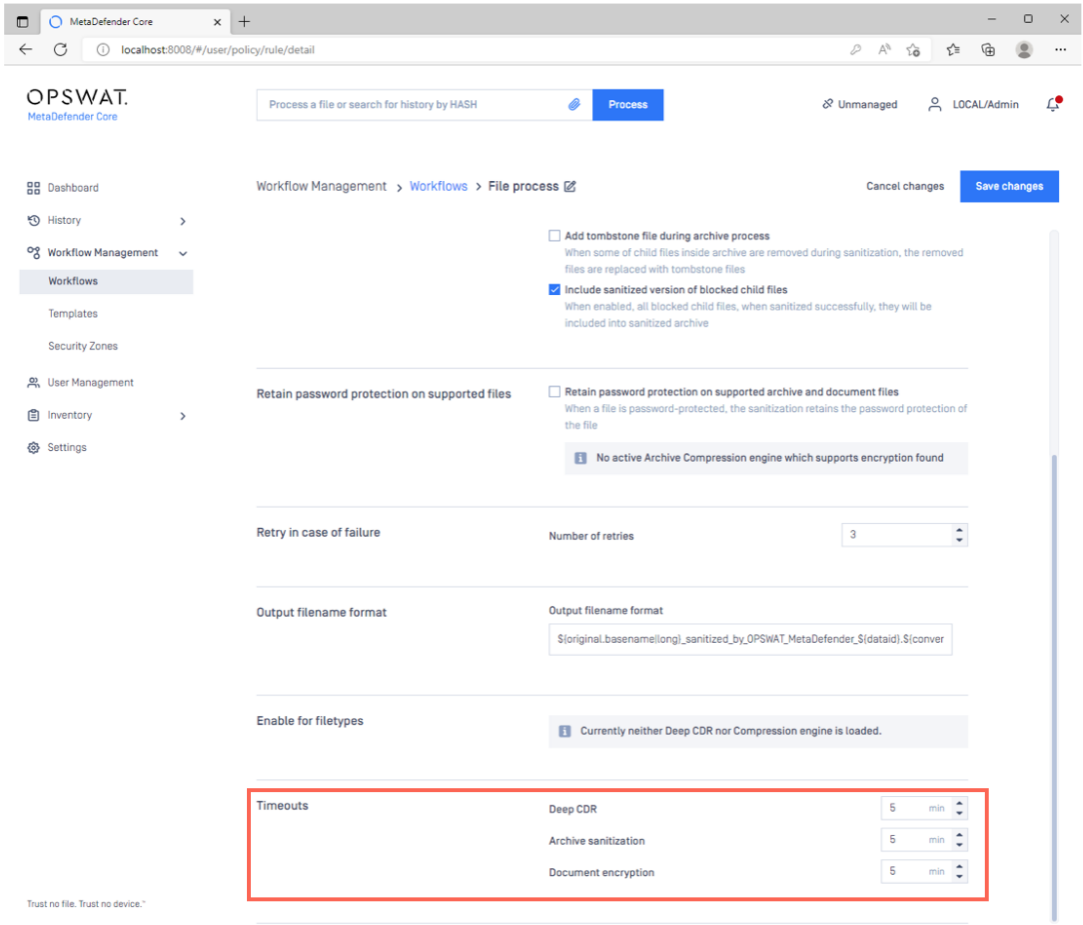
1. To access these settings, please go to your **MetaDefender Core Management Console>Workflow Management>{Your Chosen Workflow)>Deep CDR**, as illustrated in the screenshot below.



2. Here you will find numerous options, including the option to block files if the Deep CDR process fails or times out, as highlighted in the screenshot below.



3. To alter timeout settings, scroll down to the bottom of the page (illustrated below), where you can configure maximum times for all aspects of file processing, including Deep CDR, archive sanitization and document encryption.



Support:

If you have followed the instructions above but are still **Seeing Deep CDR Timeouts**, please follow these

instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

My scans keep failing due to an exceeded archive file number, how do I determine the number of files in an archive and then configure my process settings accordingly?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core's built-in Archive Extraction Engine supports a wide range of archive, archive-like, and encrypted file formats, many of which can contain hundreds or even thousands of child-files that will need to be extracted for processing.

Pro Tip:

Our supported file types include the following.

Pure archive formats	*Zip, 7z, JAR, RAR, RAR5, TAR, ISO, CAB, ARJ, LHA, LZH, RPM, DEB, LZMA, WIM, DMG, XAR, MSM, SFX, XZ, VDI, VHD, CPIO, HFS, APK, GZ, MSI, TAZ, TGZ, TBZ, BZ2, VIB, AR, ALZ, TSE, TSEC, TSEZ, ACE, PST, OST, OFT, IPA, JMOD, MSIX, APPX, EGG, DLL,* and *PGP* (with password only)
Self-extracting archive formats	which are created by both *7zip, WinRAR, PKZIP* and *IExpress*
Microsoft Office Documents	(e.g. *DOCX* files) which are detected as archive files by default
Email/Calendar files	(e.g. *EML, MSG* and *ICS*) which are extracted to scan header, body, attachments
Encrypted files	*7z, Zip, RAR, PDF*, MS Word (*DOC, DOCX, DOCM, DOTX* and *DOTM*) / *Excel (XLS, XLSX, XLSM, XLSB, XLTM)* / Powerpoint (*PPT, PPTX, PPTM, PPSM, POTX* and

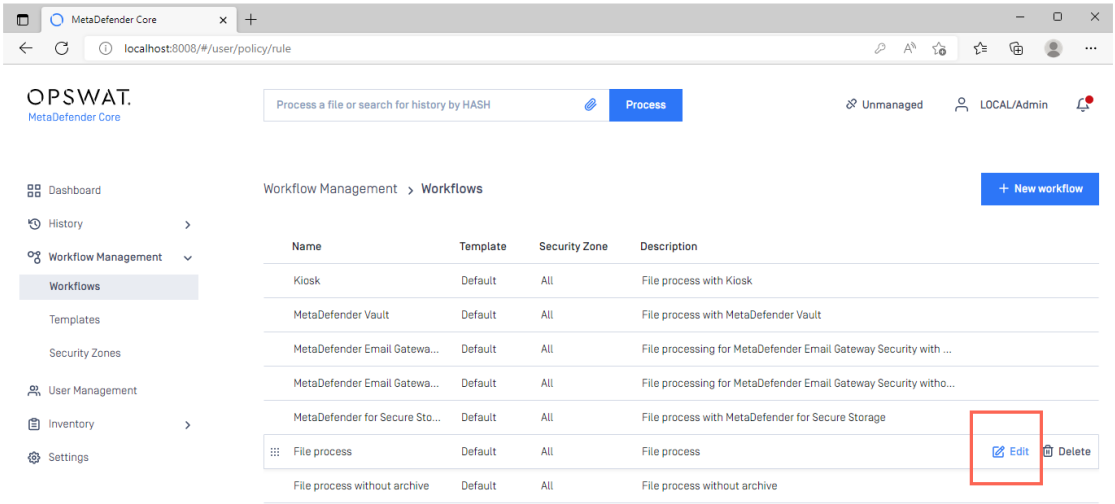
	POTM) which can be decrypted by the archive module with a given password.

To avoid seeing your scans fail due to **Exceeded Archive File Number** and other archive related issues, users attempting to scan large volume archives, large image files and other archive-like formats should adjust their settings accordingly.

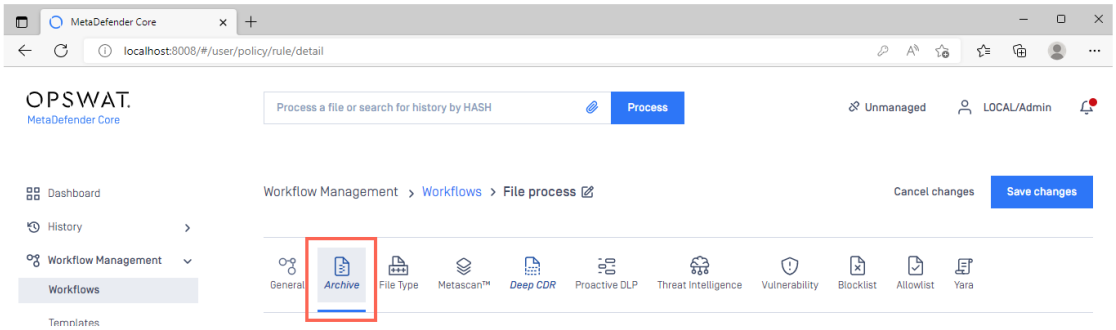
Unfortunately, there's no quick-fix in terms of estimating the number of child-files or folders within a given archive, so it is recommended that you test and adjust your settings against a file that is a fair, general representation of the larger files you're expecting to scan.

To access these settings, follow the instructions below:

1. Go to your **MetaDefender Core Management Console>Workflow Management>Workflows**, to select your chosen workflow, as illustrated in the screenshot below.



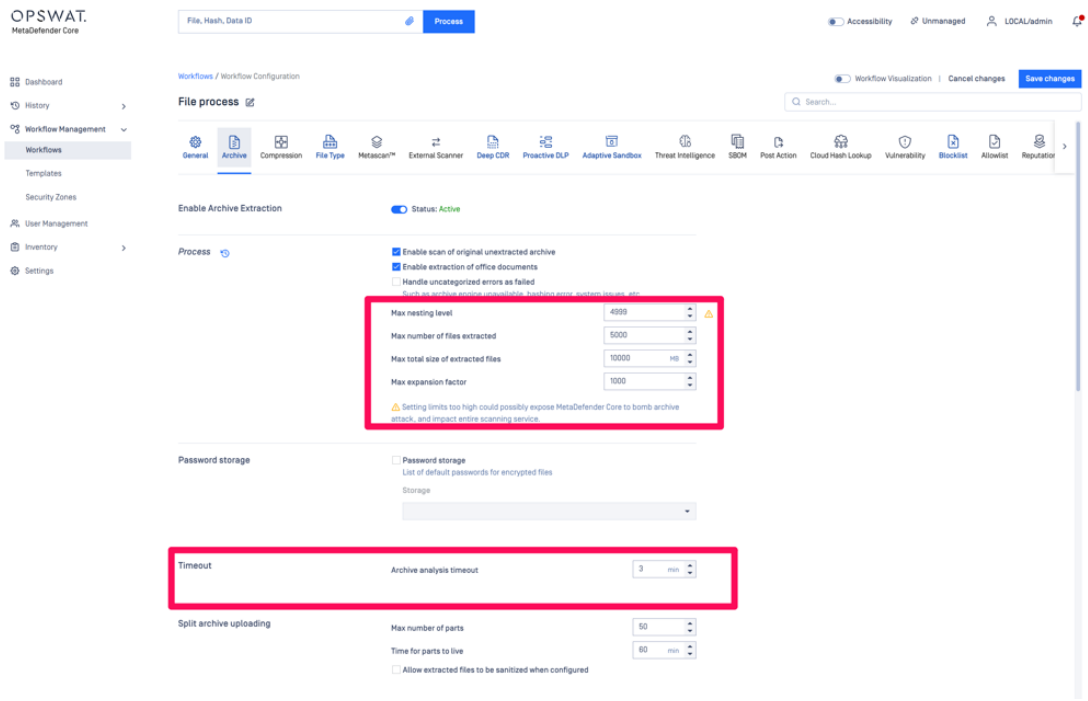
2. Inside your chosen workflow, click on the **Archive** tab, as shown below, where you can configure a range of archive-related rules.



3. Now, adjust the following values to a level that lets your test-file scan to completion. Then allow some room for error:

1. **Max nesting level**
2. **Max number of files extracted**
3. **Max expansion factor**
4. **Archive analysis timeout**
    1. Large or complex archives may require a longer timeout (for example, 60~120 minutes) to complete analysis.

So, if your test archive is allowed through at 4500, set the levels to 5000, etc.



4. Finally, click **Save** to apply your settings before processing your files as needed.

By adjusting this set of rules on any workflow in which you're expecting a throughput of large volume archive, image and archive-like files, you can minimize or even eliminate the number of scans that fail due to archive-related factors, including exceeded archive file numbers.

Support:

If you have followed the instructions above but are still **Experiencing Failed Scans Due To Exceeded Archive Numbers**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why are password protected archives blocked and how do I unblock them?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Pro Tip:

MetaDefender Core supports a wide range of encrypted (password-protected) archive file formats, including:

- *7z, Zip, RAR, PDF*, MS Word (*DOC, DOCX, DOCM, DOTX* and *DOTM) / Excel (XLS, XLSX, XLSM, XLSB, XLTM*) and Powerpoint (*PPT, PPTX, PPTM, PPSM, POTX* and *POTM*).

All of the formats above can be decrypted by MetaDefender Core's archive module for scanning, on provision of the associated passwords.

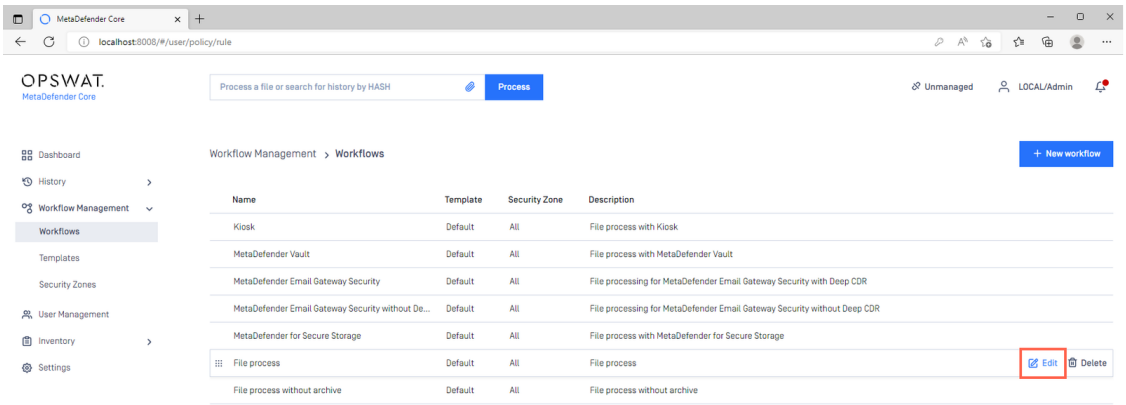
We advise customers to rescan the password protected file via both Core UI and REST API

Seeing a **Blocked** encrypted/password-protected archive could mean that you have failed to enter the associated password or that the file format you are attempting to scan is not supported by MetaDefender Core.

In addition, failing to enable archive handling in your settings may also lead to **Blocked** results.

To enable archive handling:

1. Go to **MetaDefender Core Management Console>Workflow Management>Workflow**s, and click **Edit** to adjust the relevant workflow rules, as illustrated below.

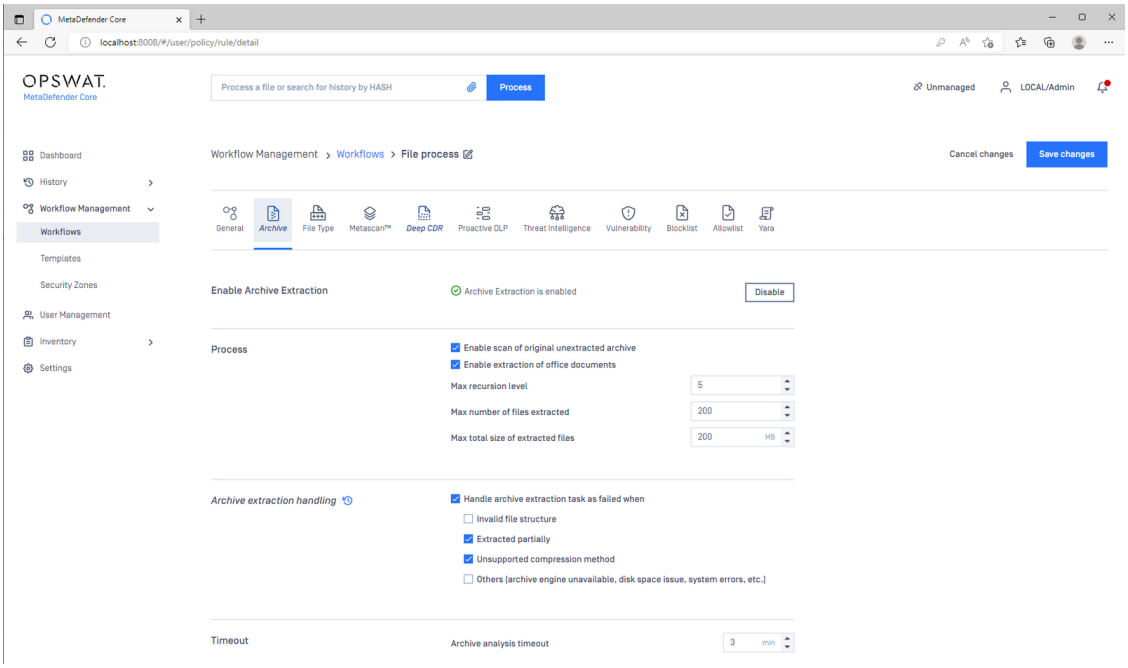


2. Once in your chosen workflow, go to the **Archive** tab.
3. Here, you should ensure that the following actions are enabled:
    1. Archive extraction
    2. Scan of original unextracted archive
    3. Extraction of office documents

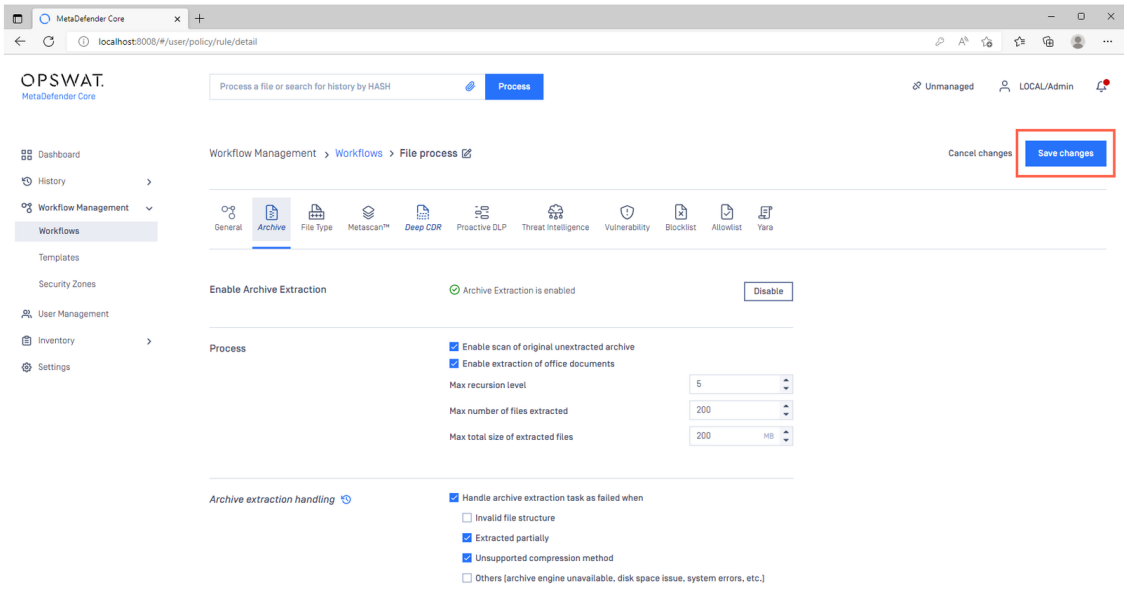
You can also ensure that tasks are handled as **Failed** when they are/in the case of:

- Extracted partially

- Unsupported compression method.

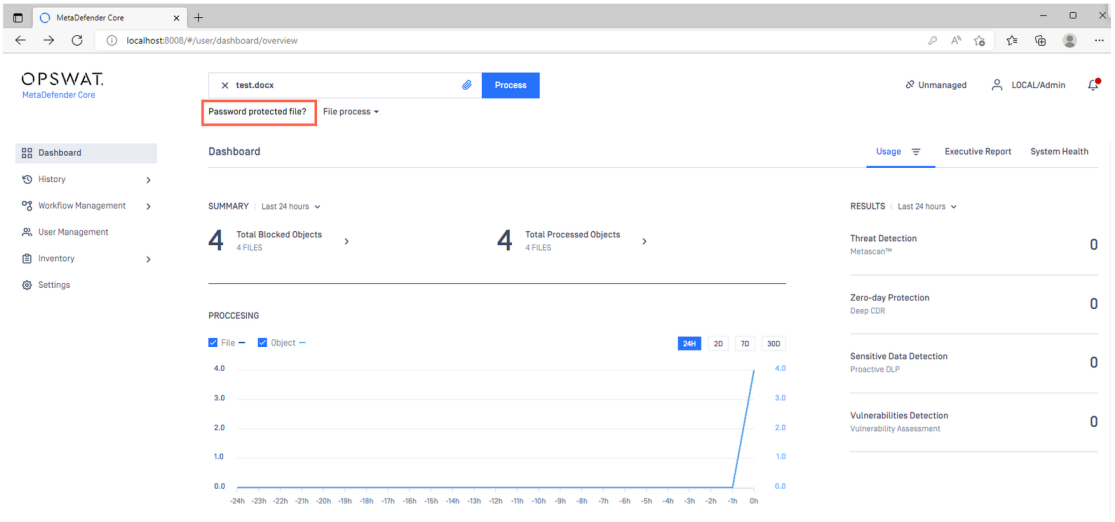


4. Finally, click **Save changes** to apply your settings.

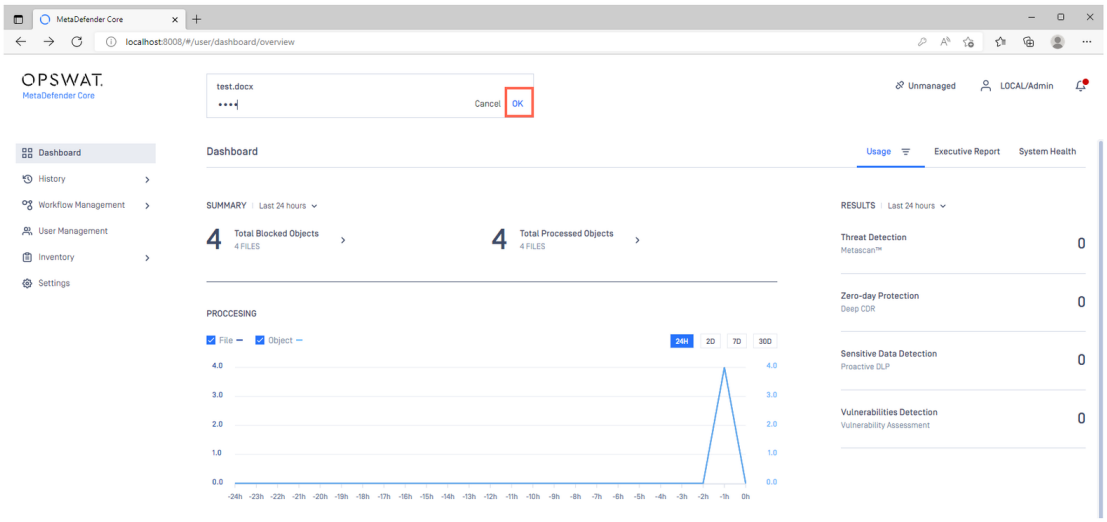


To unlock supported password-protected archives for scanning (once you have configured your chosen workflow), please do the following:

1. Go to your **MetaDefender Core Management Console>Dashboard**, and drop a password encrypted file into the field for processing.
2. You will now be asked whether the file is password protected, as highlighted in the screenshot below, click to confirm.



3. Enter the file's password in the field provided, then click **OK**.



4. Your file is now ready for processing and, provided the encrypted file type is supported and your archive settings are configured, the file should no longer return a **Blocked** result. (if no other restriction settings are in place on the workflow)

Support:

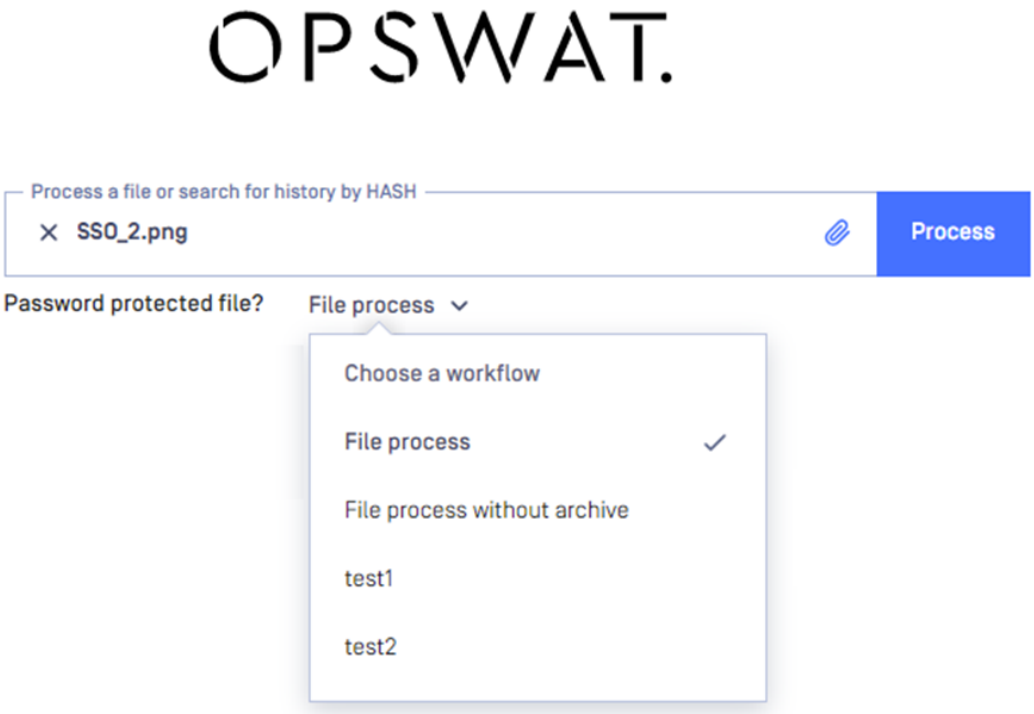
If you have followed the instructions above but are still **Unable To Unblock Password Protected Archives**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How to disable WEB UI file scan without user authentication?

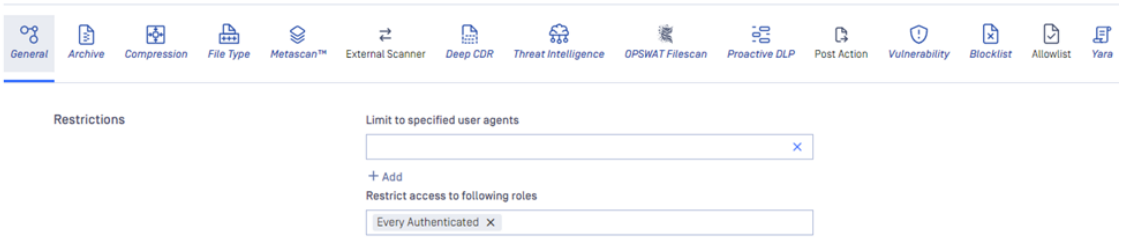
Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems.

To disable the functionality of the WEB UI file scan without user authentication, you will need to change the access for all the Workflows accessible in this scan mode:



To do so, please login to the MetaDefender Core Web interface (default http://localhost:8008), navigate to Workflow management -> Workflows -> Select the workflows which are available at the first screen and then select the restrictions -> Restrict access to the following roles : Every Authenticated



This will limit the access to scanning to Authenticated users only.



Support:

If you need help learning how to disable WEB UI file scan without user authentication, please log a support case or chat with our support engineer.

How To Allow Only Certain Files to be Scanned with MetaDefender Core?

Check Your Version

The steps provided in this article apply to MetaDefender Core versions 4.x and 5.x. The user interface and options may vary slightly depending on your version.

MetaDefender Core offers extensive customization options, empowering administrators to filter files based on their extensions. This feature allows you to define rules that permit scanning only specific file types, enhancing security and efficiency. This article explains how to configure file extension filtering in MetaDefender Core using the Web UI and provides a sample regular expression to allow only certain file extensions, effectively restricting the scanning process.

Prerequisites:

- Access to the MetaDefender Core Web UI
- Appropriate user permissions to modify workflows

Steps:

1. Launch a web browser and navigate to the MetaDefender Core Web UI.
2. Log in using your administrator credentials.
3. Go to the "Workflow Management" section.
4. Click on "Workflows" in the left-hand menu.
5. Select the workflow to which you want to apply file extension filtering.
6. In the workflow details, navigate to the "Blocklist" tab.
7. Enable the "Blocklist by filename" option.
8. Click the "Add" button to create a new filename rule.
9. In the "Filename" field, enter the following sample regular expression: Sample regular expression: `^(?!.*\.(?:extension1|extension2|extension3)$).*$`

Replace `extension1`, `extension2`, `extension3`, etc., with the desired file extensions. For example, to allow only ".txt" and ".pdf" files, use `^(?!.*\.(?:txt|pdf)$).*$`.

Note: Make sure the regular expression accurately represents the desired file extensions. If you need assistance with

regular expressions, consult the appropriate documentation or reach out to your organization's IT team.

10. To apply filtering based on file size, enable the "If bigger than" option.

11. Specify the minimum file size in the corresponding field.

12. Click "Save" or "Apply" to save your changes.

Additional Resources:

- For more information on using the MetaDefender Core Blocklist/Allowlist feature, refer to the following documentation: How to use MetaDefender Core V5 Blocklist/Allowlist feature?
- To learn about other workflow configuration options in MetaDefender Core, including the Blocklist and Allowlist, consult the Workflow Configuration Template documentation: Workflow Configuration Template

Support

If you have followed the instructions above but are still **Unable to Allow Only Certain Files to be Scanned with MetaDefender Core**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How to generate an API key on the Core deployment?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

In order to be able to generate an API key on the Core console the following steps are necessary:

1. Navigate to the core console (Console\User Management\Users and Groups)

2. Select an available administrator type user.

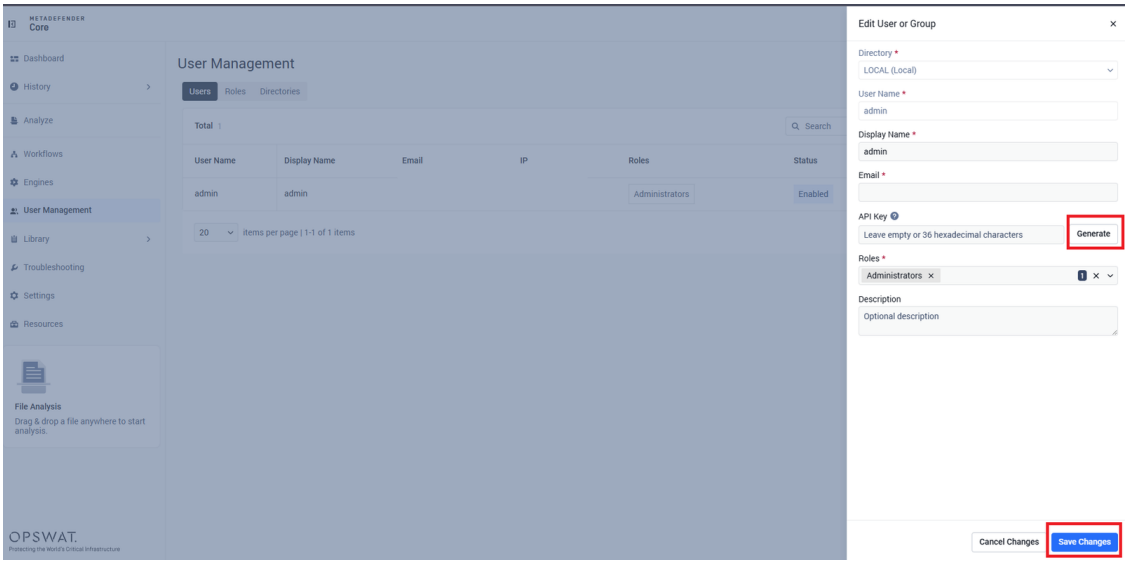
3. Navigate to the API key tab and click on the “generate” tab.

4. Users can also input an API key as long as it has an accepted format.

Please note

Must have exact 36 characters. Must contain numeric and lowercase letter characters only [0-9 a-f]. Must contain at least 10 lower case letter characters. Must contain at least 10 numeric characters. Not allowed to contain more than 3 consecutive numeric characters. Not allowed to contain more than 3 consecutive lower case letter characters.

5. Once the key is generated you will need to click on the “Save changes” tab for the changes to take effect.



Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

How to Modify the Hostname of Your MetaDefender Core Server and Potential Impact?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Issue: After updating the hostname of the MetaDefender Core server, the system may experience issues such as Deployment ID changes or difficulties with service operations, certificate validation, and endpoint agent communication.

Cause: Changes to the hostname can lead to mismatches in the system certificates and configurations, which might affect the normal functioning of the MetaDefender Core server.

Solution:

1. **Disable HTTPS and Remove Current Certificates** : Launch the MetaDefender Core UI, navigate to 'Settings' -> 'Security Tab', click 'Details' under 'Secure Connections', and select 'Do not use certificate'. Save the changes and go to the 'Inventory' -> 'Certificates Tab' to remove the existing certificate.
2. **Deactivate the License** : Go to the 'Settings' -> 'License Tab' in the MetaDefender Core UI. Click on the 'Deactivate License' button. This ensures that your license can be safely transferred to the server once the hostname is changed.
3. **Stop MetaDefender Core Services** :

- For Windows: Open 'services.msc' from the Start menu, locate 'MetaDefender Core Service', and select 'Stop'.

- For Linux: In a terminal window, input 'sudo systemctl stop ometascan' to stop the services.

4. **Rename the Server and Reboot the System** :

- For Windows: Navigate to 'Control Panel' -> 'System and Security' -> 'System' -> 'Advanced system settings'. Under the 'Computer Name' tab, input the new hostname and save changes. Then, restart the server.
- For Linux: In a terminal window, use 'sudo hostnamectl set-hostname [new hostname]' to rename the server, replacing '[new hostname]' with your new hostname. Reboot the server using 'sudo reboot'. Confirm the new hostname with the 'hostname' command.

5. **Reactivate the License** : After changing the hostname and restarting the services, reactivate your license. Go to 'Settings' -> 'License Tab' in the MetaDefender Core UI, and follow the on-screen instructions for license activation. For detailed steps on how to activate the license, please visit MetaDefender Core Licensing.
6. **Import New Certificate with the New Hostname** : Create a new certificate (PEM or PKCS12 format) with the new hostname. In the MetaDefender Core UI, go to 'Inventory' -> 'Certificates Tab' to add the new certificate. In 'Settings' -> 'Security Tab' -> 'Secure Connections', choose the newly added certificate and save changes. Ensure to import the new certificate into the trusted root certificate store for any endpoint agents or security systems that communicate with MetaDefender Core.

If other systems depend on the MetaDefender Core server's hostname, update their configurations to reflect the new hostname. This might include DNS records, configuration files, or other settings.

Best Practice: Prior to renaming the hostname, it's recommended to test these changes in a non-production environment, ensuring that they conform to your organization's security policies and regulations.

Support:

If you require further assistance, please follow these instructions on How to Create Support Package With Bundle Tools?, before logging a support case or chat with our support engineer.

What are the permissions on the shared folder for the temp directory?

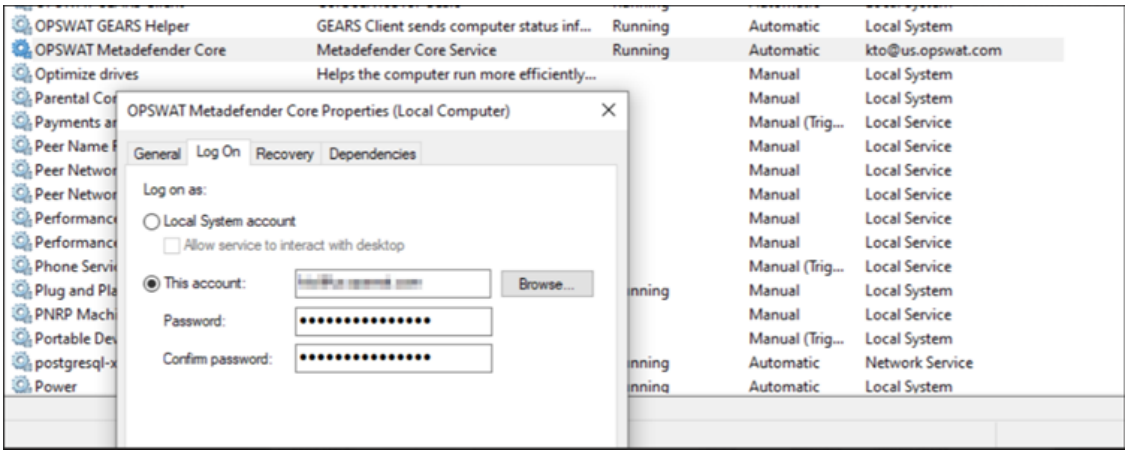
Check Your Version

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

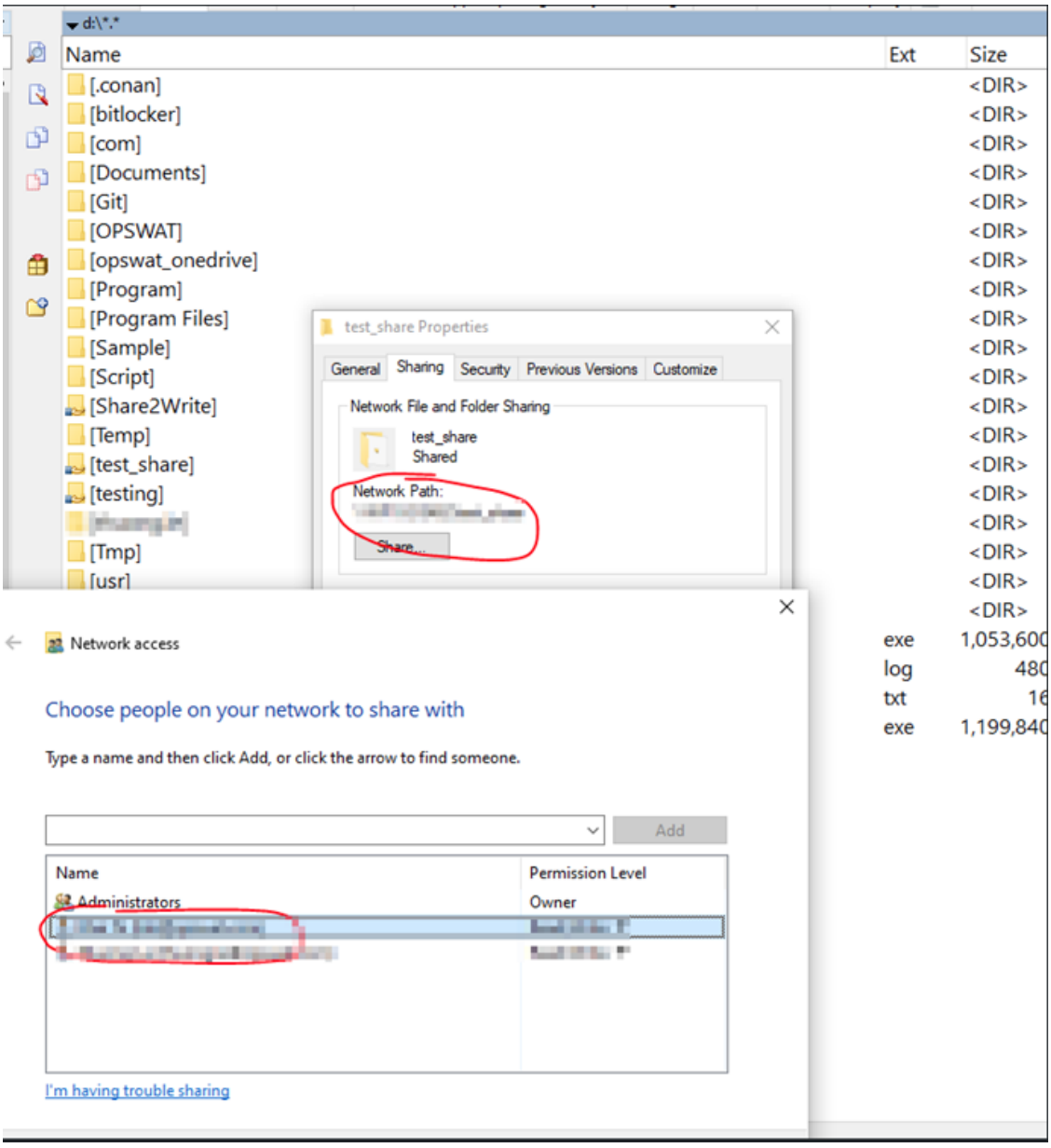
Kindly adhere to the following steps to ensure the appropriate permissions are granted for the Shared folder:

- Initially, the customer should modify the Log on credentials, transitioning from SYSTEM admin to a user possessing the necessary permissions for reading and writing data in Core's folders and their related components.
- Subsequently, when sharing a folder across a network, it is recommended to share the folder using the same account.

For instance, if you have a Core operating on a machine, let's refer to it as A, and you alter the Log on credentials from SYSTEM to a user associated with the AD server (also linked to machine A)



On the other machine that you want to share the folder, add this user to the folder that you want to share



Support

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to implement a numerical escape function for HTML?

This knowledge base article provides step-by-step instructions on how to fix an issue related to Outlook and its HTML rendering engine. It seems that certain escaped characters are not rendered correctly in Outlook when specified using their HMTL entity name, for example:

`’`

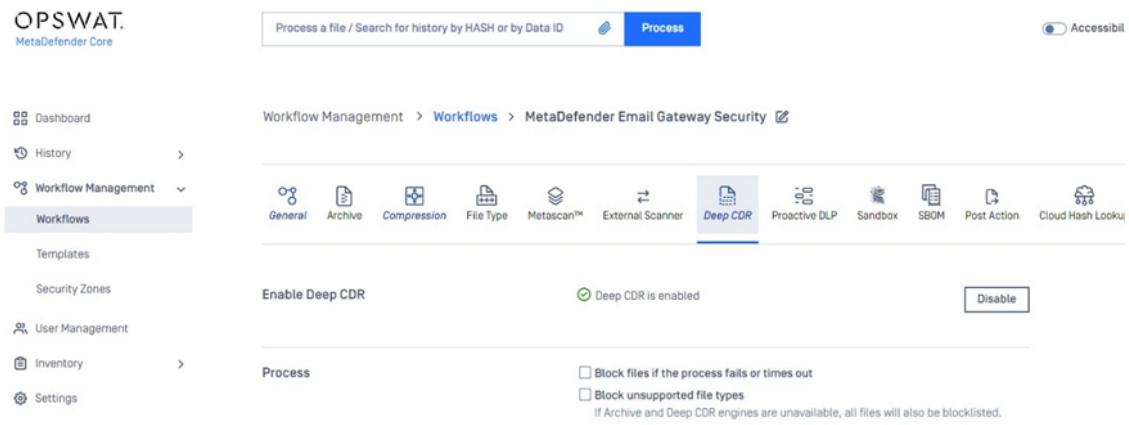
Original Email: Please find the summary of today's meeting

Processed email: Please find the summary of today`’`s meeting

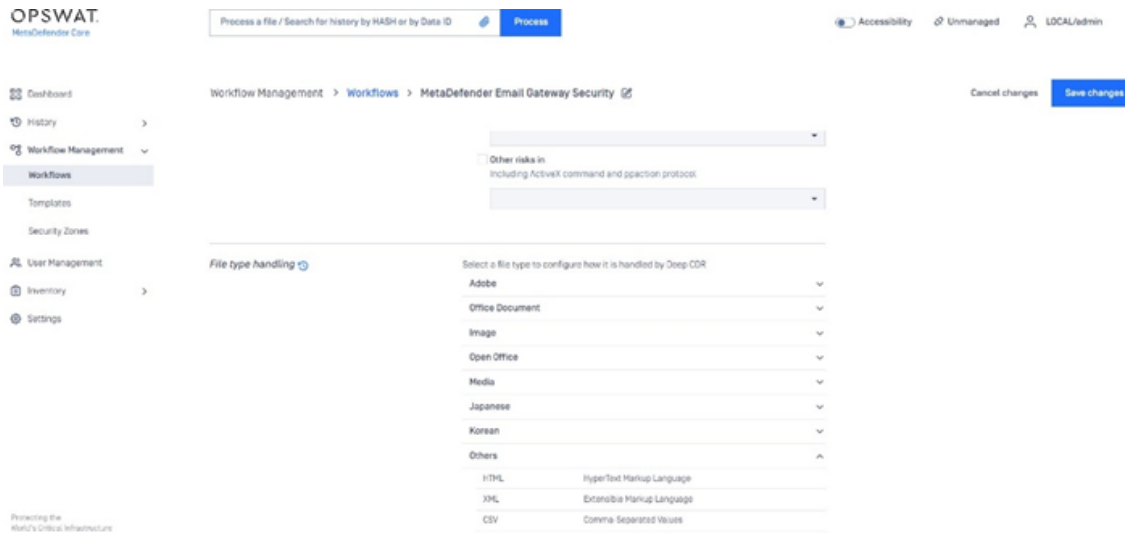
Note

The fix is available starting with CDR engine version 6.8.0

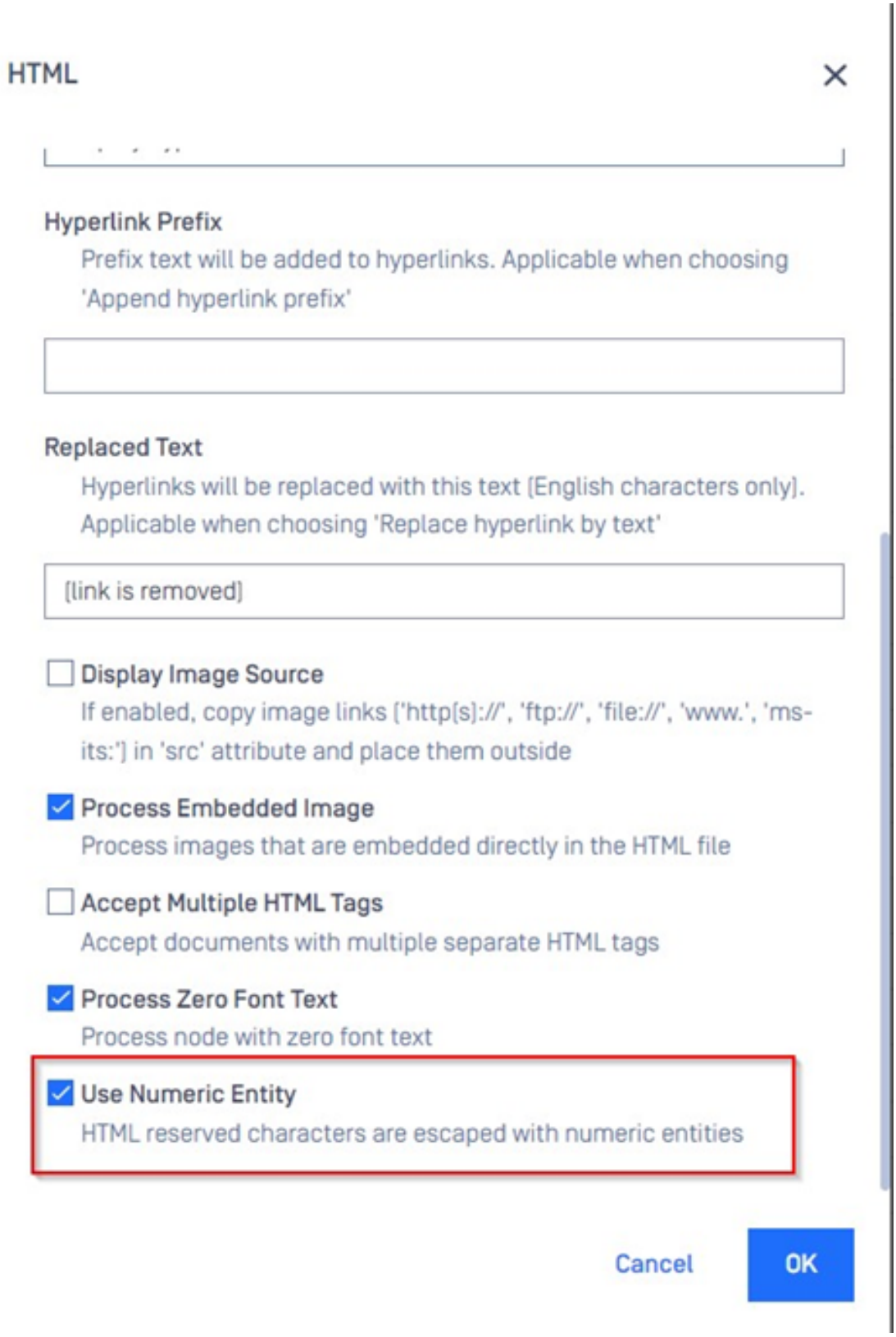
1. Navigate to the MD Core Web UI.
2. Locate and choose the workflow associated with the EGS instance.
3. Access the DeepCDR tab and proceed to the Advanced section.



4. Under File Type Handling, select "Others" and specifically choose "HTML."



5. Scroll down to find the option labeled "Use Numeric Entity" and activate it.



6. Confirm your changes by clicking on the "OK" button, followed by "Save Changes."

Support

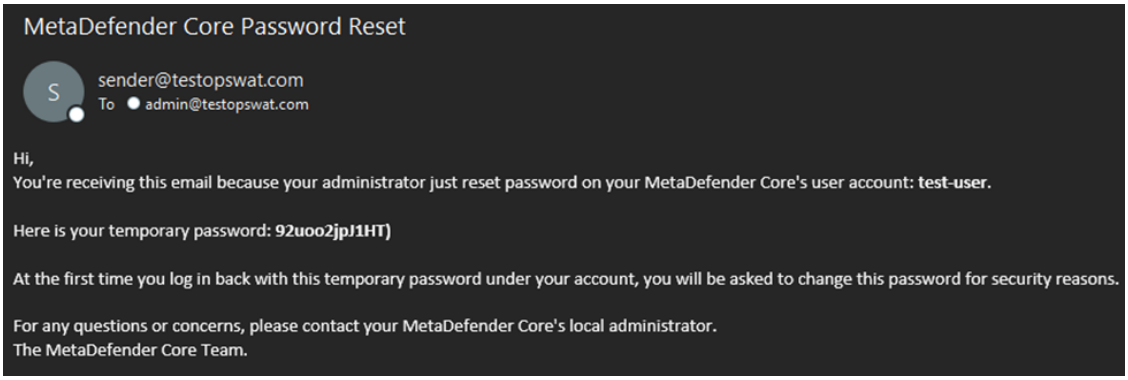
If you encounter any problems with this guide or if the steps provided do not work, proceed to log a support case or chat with our support engineer.

How to reset password for users?

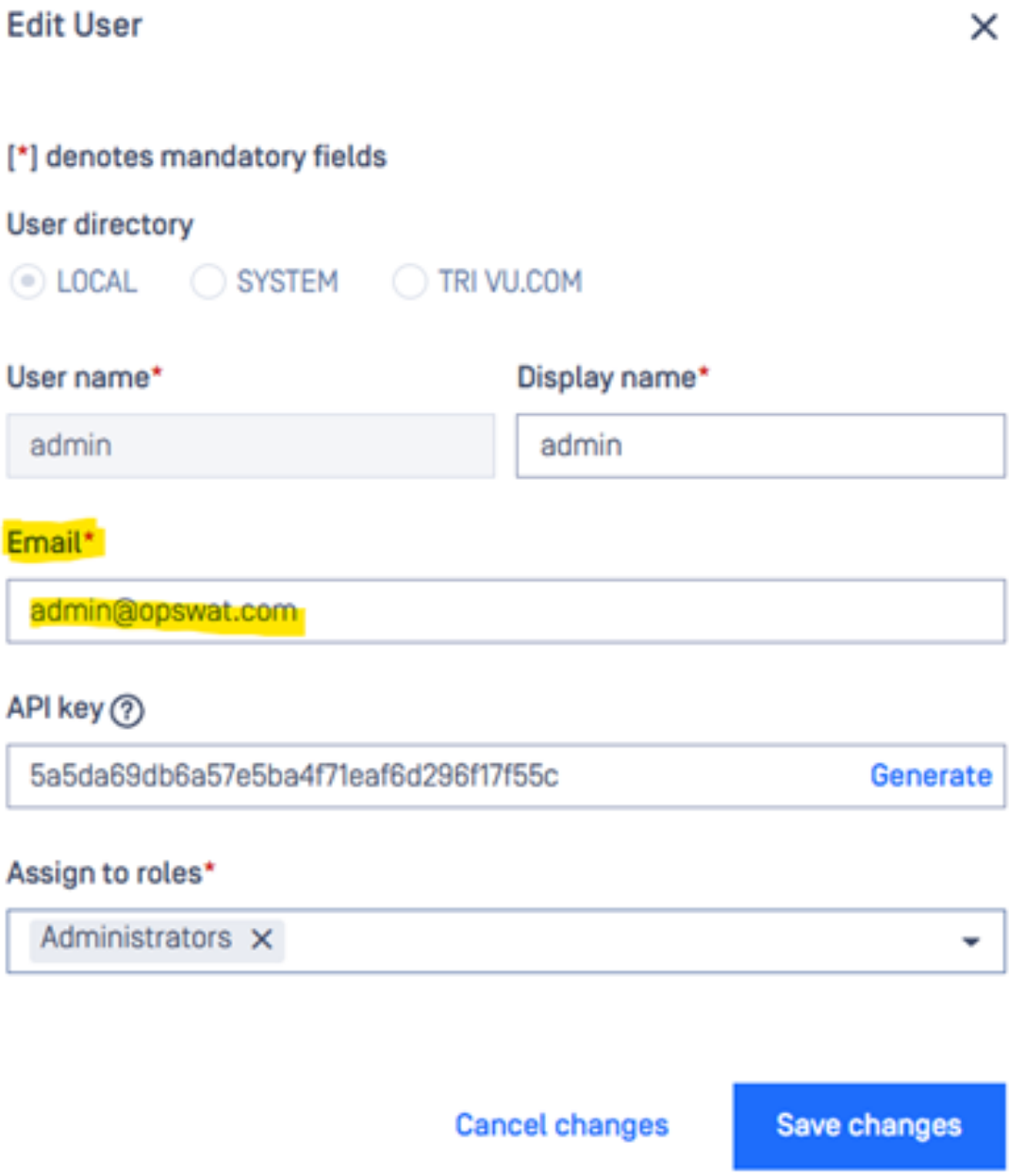
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

Resetting a User's Password User Management in MetaDefender Core requires an active SMTP Server Configuration upfront, to provide a way to inform the users by Email of their Credentials Change, so they can acknowledge and act immediately, changing their temporary password:

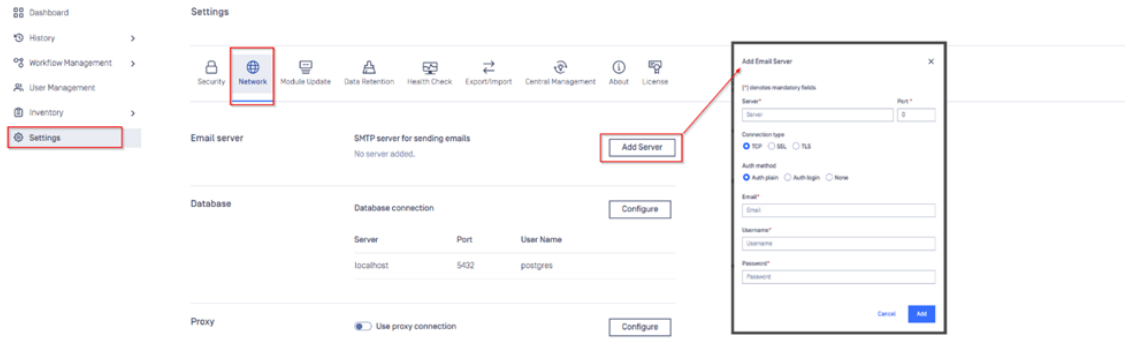


The user's Email Address should also be configured correctly for the SMTP Server to send the email to the recipient successfully.



This represents an enforcement action to prevent security breaches like someone impersonating the actual user and taking advantage of their account.

SMTP Server can be configured accessing **Settings** > **Network** > **Email Server** > **Add Server**:



For cases when MetaDefender Core is deployed in an isolated environment and SMTP cannot be configured, the alternative way to bypass this is by removing the User and Re-Adding it with a new password.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to Block/Allow Custom Extensions using RegEx?

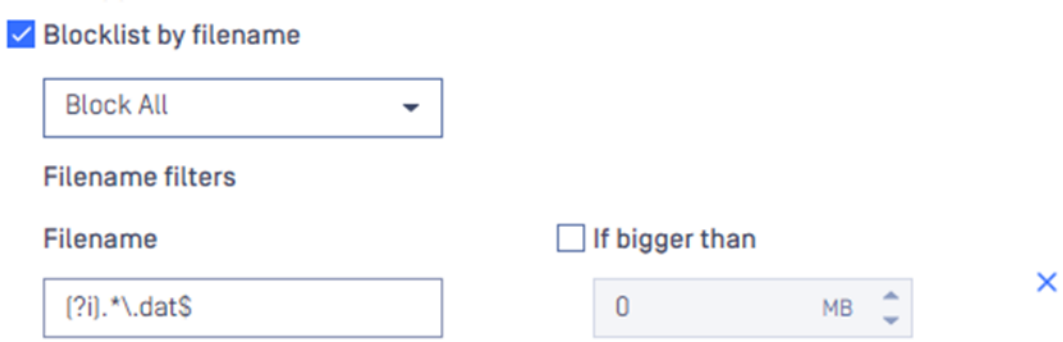
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core allows blocklisting and allowlisting based on both the FileType and Filename.

However, for custom extensions, it may be possible these are not listed within the FileType category, in which case you can create a rule using RegEx to achieve the desired result.

For example, for a .dat extension, which is not a standard one, but rather depends on the software that generates it, you will not find it under FileType. You can then use the filename option and create a rule as in the example below:



This is a case-insensitive pattern that will block all the files ending in **.dat** .

Metacharacter	Matches
(?i)	Flag that enables case-insensitive matching
.	Matches any single character
*	Matches zero or more occurrences of the previous pattern
\	Escape character to remove the special meanings of metacharacters. In the case above it is used to match a literal dot

Metacharacter	Matches
$	Ensures the pattern matches the end of the string

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to migrate MetaDefender Core from PostgreSQL local to PostgreSQL remote?

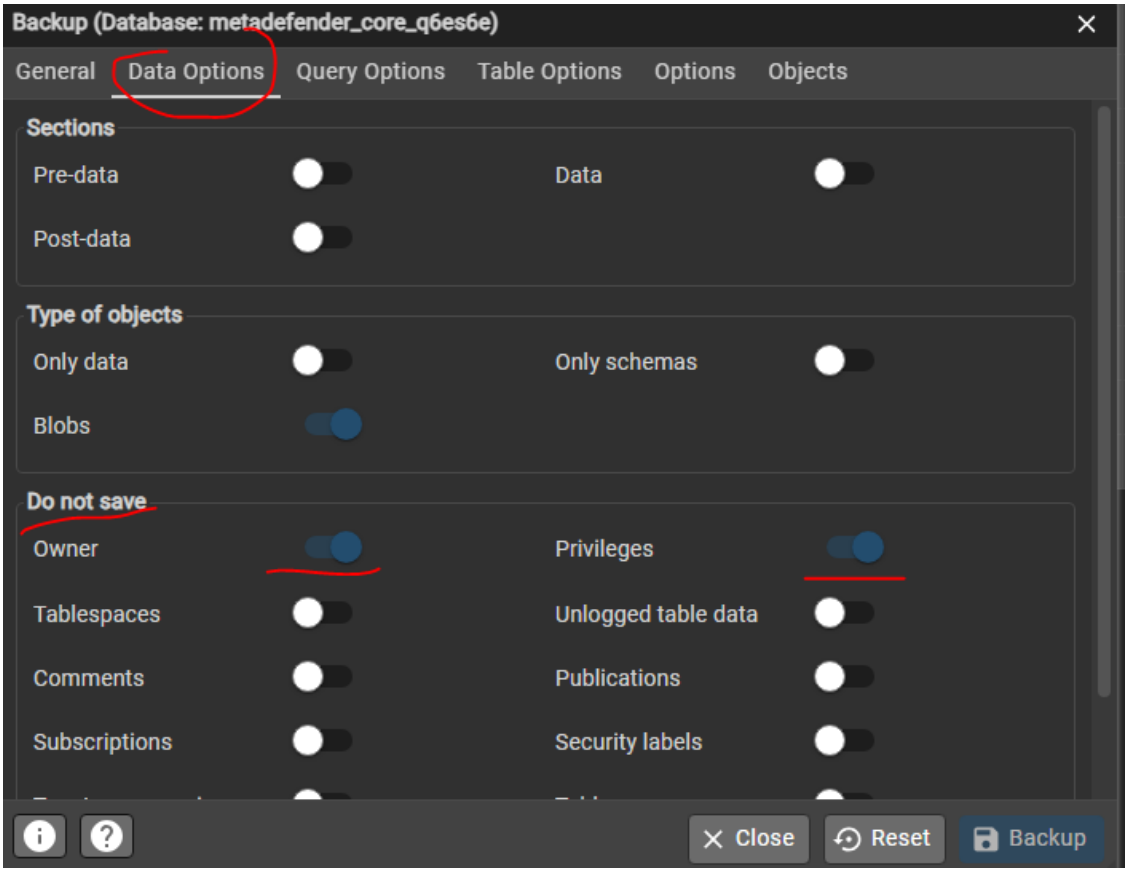
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Migrating MetaDefender Core from a local PostgreSQL database to a remote PostgreSQL setup is a critical task for organizations seeking enhanced scalability, performance, and security. This transition allows MetaDefender Core to leverage the robustness of remote database management while maintaining data integrity and ensuring smooth operational continuity. By decoupling the application from the local database, businesses can optimize their infrastructure, manage resources more effectively, and better support growing data demands. In this guide, we will explore the key steps and considerations for successfully migrating your MetaDefender Core environment to a remote PostgreSQL database.

Here are the steps:

- First, we should export the settings: if anything is wrong, we can still have a backup for local users and settings.
  - Goto UI, Settings Export/Import export the configuration to a zip file
- Export the whole database to a file:
  - We can use pgAdmin or `pg_dump` to export all the databases:
    - SharedDB: `metadefender_core` or Standalone: `metadefender_core_xxxx`
    - NOTE: **When backup we need to check this option so the exported DB will not have the restriction on privileges when importing back**



- Example with pg_dump: --no-owner --no-privileges

Windows · bash

```
pg_dump.exe --file "d:\\opswat\\backup_db" --host "127.0.0.1" --port "5432" --username
"postgres"
--format=c --blobs --no-owner --no-privileges --verbose "metadefender_core_q6es6e"
```

Linux · bash

```
pg_dump --file "/opswat/backup_db" --host "127.0.0.1" --port "5432" --username "postgres"
--format=c --blobs --no-owner --no-privileges --verbose "metadefender_core_q6es6e"
```

- Remove the current MetaDefender Core - Reinstall MetaDefender Core with the **same version** in the **same mode (standalone/shareddb)** This time we will config this MetaDefender Core to connect to **remote PG** - Stop MetaDefender Core - Delete current db of MD Core in remote server: `metadefender_core_xxx` / `metadefender_core` - Create a whole brand new db with the same name: `metadefender_core_xxx` / `metadefender_core` - Using pgadmin or `pg_restore` to restore the backup file into the remote PG

- Example with `pg_restore`:

Windows · bash

```
pg_restore.exe --host "remote_pg_ip" --port "5432" --username "postgres"
 --dbname "metadefender_core_q6es6e" --verbose "d:\\opswat\\backup_db"
```

Linux · bash

```
pg_restore --host "remote_pg_ip" --port "5432" --username "postgres"
 --dbname "metadefender_core_q6es6e" --verbose "/opswat/backup_db"
```

Support:

If you have followed all of the steps above but are still unable to **migrate MetaDefender Core from PostgreSQL local to PostgreSQL remote**, please follow these instructions on How to Create Support Package With Bundle Tools?, before logging a support case or chatting with our support engineer.

How to change Core Web Management Interface Address Port?

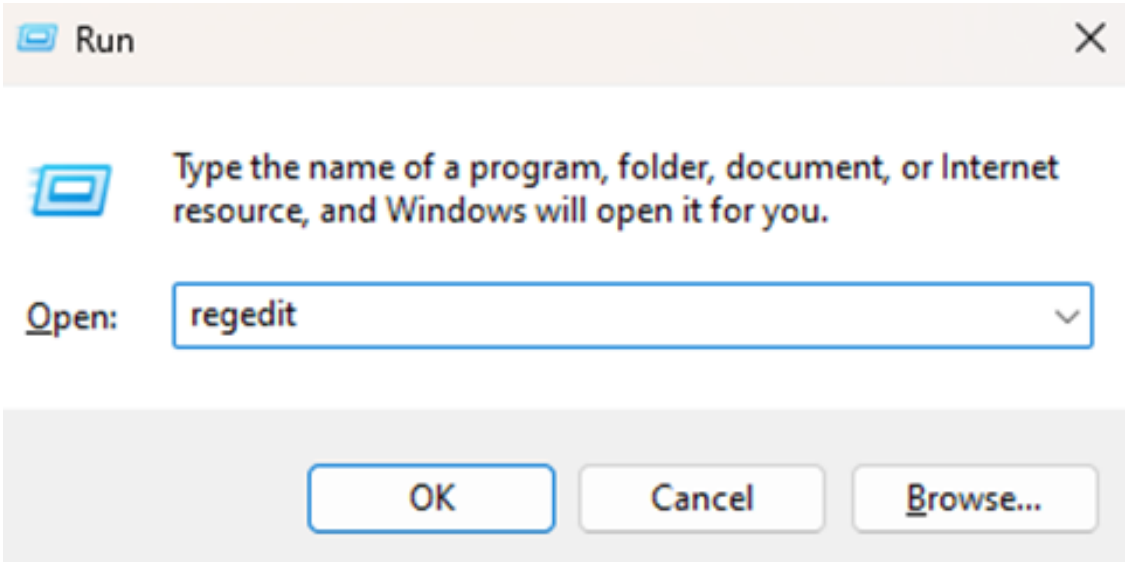
By default, MetaDefender Core's Web Management Interface is available via Port 8008.

This is customizable via Registry Editor in Windows and /etc/ometascan/ometascan.conf file in Linux.

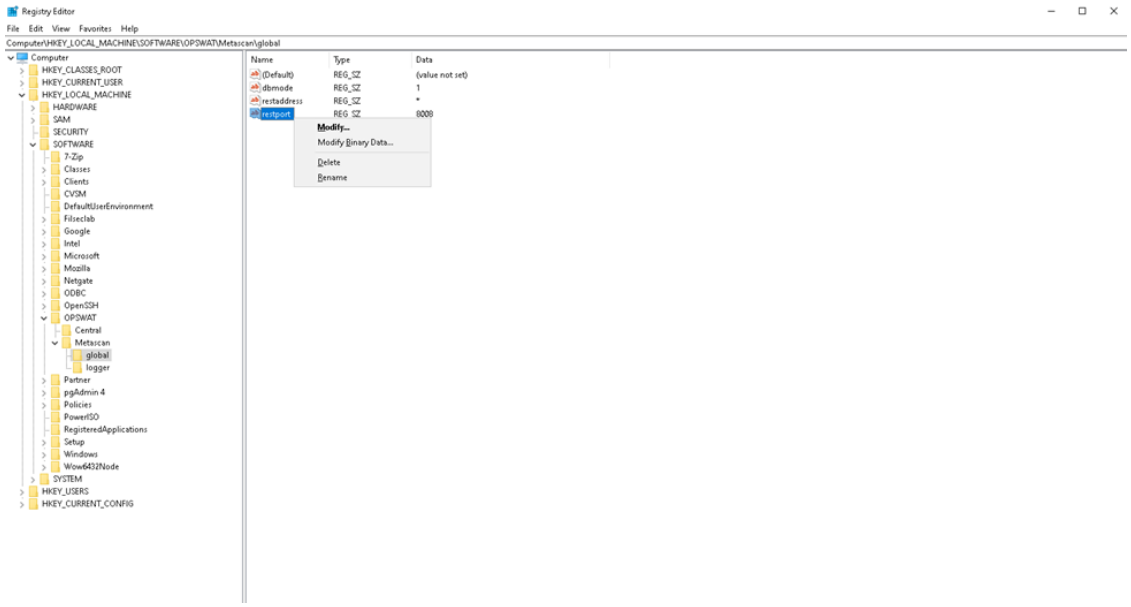
To change the port, you can follow the steps below:

Windows:

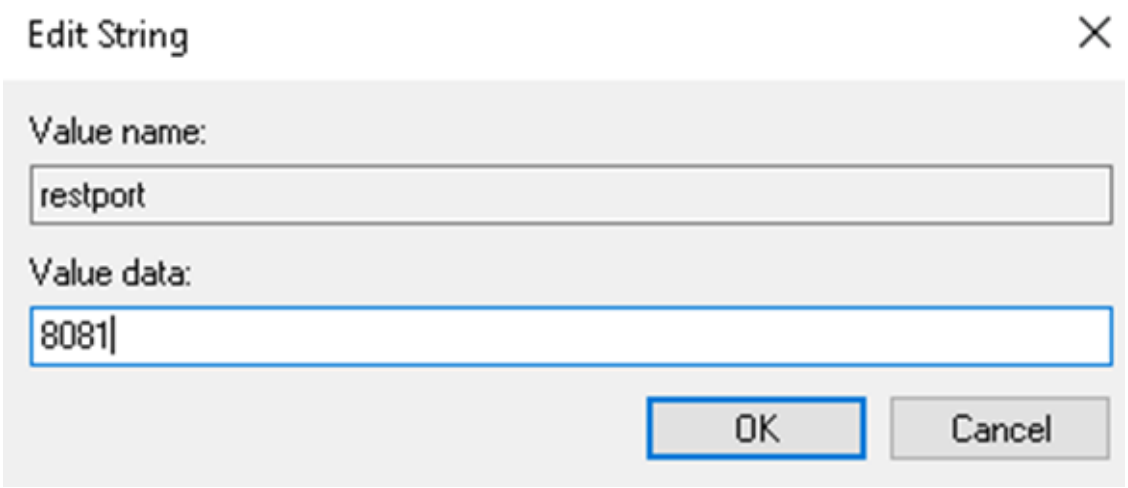
1. Press **Win+R**, type **regedit** and press Enter to open **Registry Editor**:



2. Navigate to **HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global**, select **restport** Key and select **Modify...**



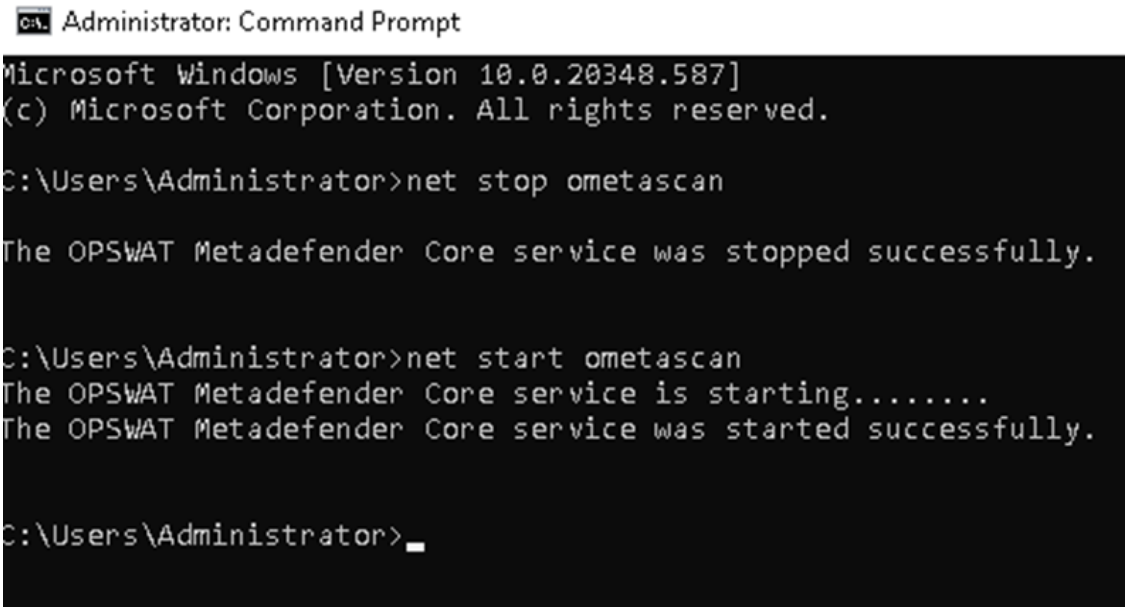
3. Enter the desired port and click the **OK** Button



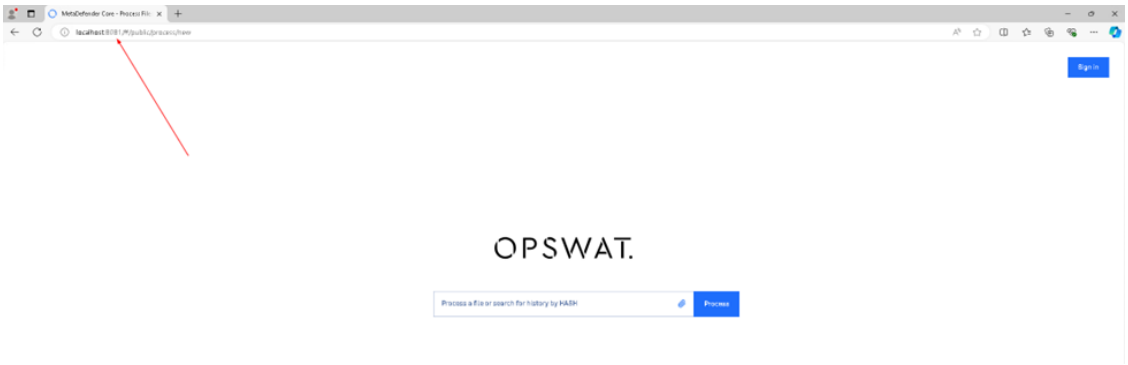
4. Open a **Command Prompt** as an **Administrator** and run the following commands to restart **MetaDefender Core Service**:

Command Prompt · bash

```
net stop ometascan
net start ometascan
```



5. You can now access the **MetaDefender Core Web Management Interface** via configured **Port**:



Linux:

1. Open the /etc/ometascan/ometascan.conf file in edit mode:

Example · bash

```
sudo vim /etc/ometascan/ometascan.conf
```

2. Edit the **restport** value with the desired port:

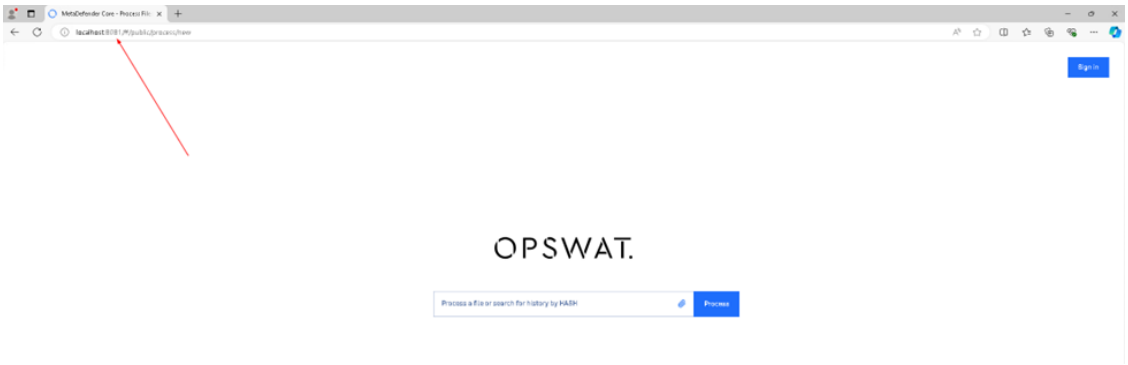
```
[global]
dbmode=1
restaddress=*
restport=8081
[logger]
log_rotation=1
logfile=/var/log/ometascan/ometascan.log
loglevel=info
nginx_log_rotation=1
nginx_logfile=/var/log/ometascan/nginx-ometascan.log
```

3. Save the changes and restart the **MetaDefender Core Service** using the following command:

Code · bash

```
sudo systemctl restart ometascan
```

4. You can now access the **MetaDefender Core Web Management Interface** via configured **Port**:



Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Does MetaDefender Core store original files?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

MetaDefender Core does not store original files permanently.

Files are temporarily stored while waiting in the queue for processing. During processing, they are kept in a designated folder and handled by MetaDefender Core and its engines. Once processing is complete, the files are

automatically removed.

Default Temporary Folder Locations:

- **Windows: <installation directory>\data\resources**
- **Linux:** `/var/tmp/ometascan/resources/` (MetaDefender Core creates a subfolder named `ometascan` and `ometascan/resources` in this directory.)

To change the temporary folder location, please refer to this article for more information How can the Temp folder be changed?

Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

How to install pre-built libgdiplus on RedHat 9?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Linux systems.

Installing the pre-built **libgdiplus** package on **Red Hat Enterprise Linux (RHEL) 9** is necessary for running **Deep CDR**. Since **libgdiplus** is not available in the default RHEL repositories and some users may not be able to install it from **EPEL** or **Mono**, using a pre-built version is the best alternative. This guide provides step-by-step instructions for downloading, extracting, and configuring the pre-built **libgdiplus** package to ensure smooth operation of **Deep CDR** on your system.

- Download the prebuilt libgdiplus6.0.4.tar.gz .
  - File size: 1,857,019 bytes
  - MD5: 759a6a3641a1bc83363336f860cd7f25
  - SHA256: 222ec848c62bdc886ebed980f87d4e56ff3341c5e9af1c242a2e9706b674a4b5
- Run these commands to install, be aware of the “Copy and extract libgdiplus6.0.4.tar.gz ...” step.

Terminal · bash

```
### Install runtime dependency
dnf install -y libjpeg-devel giflib libjpeg libpng cairo libtiff libexif glib2
### Copy and extract libgdiplus6.0.4.tar.gz file to your installed folder (this folder can be
read by metascan) (for example :/var/lib/libgdiplus6)
mkdir /var/lib/libgdiplus6
tar -xvf libgdiplus6.0.4.tar.gz -C /var/lib/libgdiplus6
echo "/var/lib/libgdiplus6/6.0.4/lib" | sudo tee /etc/ld.so.conf.d/libgdiplus.conf
ldconfig
```

```
### Double check libgdiplus was installed successfully
ldconfig -p | grep libgdiplus
### Restart Core to make change applied
systemctl stop ometascan
systemctl start ometascan
```

Support:

If you have followed the steps above but are still having trouble installing pre-built **libgdiplus**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Does Deep CDR support sanitizing split archives?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

Currently Deep CDR only supports sanitizing split CAB archives fully. While other split archives are supported for extraction and AV engine scanning (ZIP, RAR and TAR), Deep CDR does not have the capability to sanitize any files contained in these types of split archives at this time.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

How to exclude OPSWAT folder from Anti-Virus Windows Defender?

Please Note:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

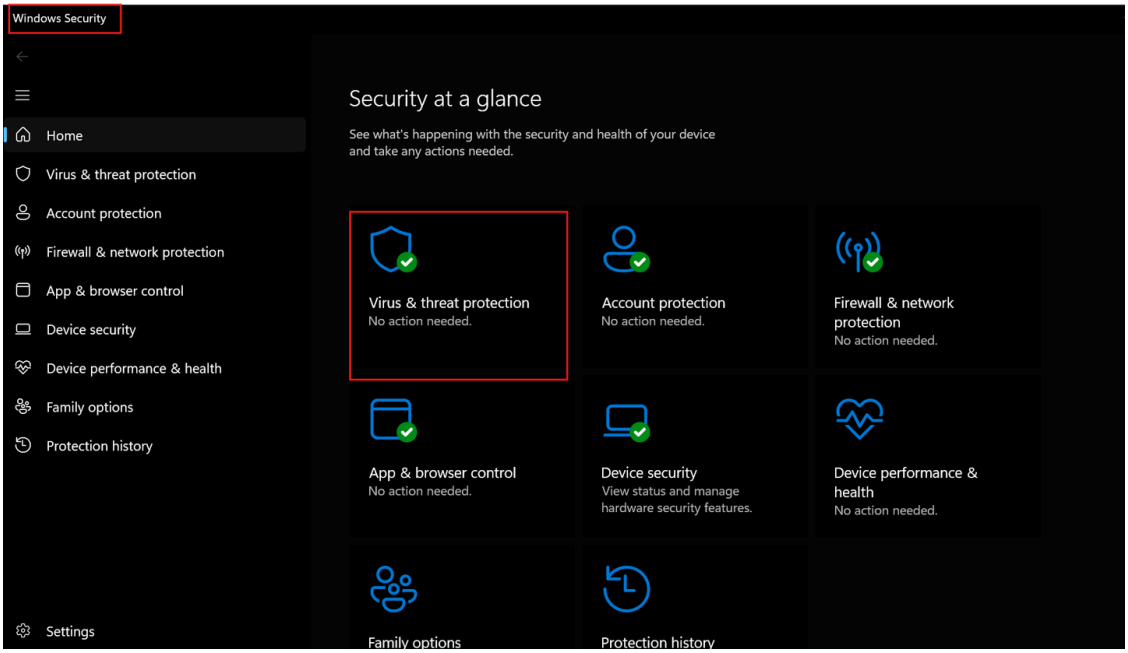
When scanning sample malware files, such as the commonly used EICAR from Download Anti Malware Testfile - EICAR, it is important to exclude the OPSWAT folder from **all** existing Anti-Virus software programs on the machine. Failure to exclude the OPSWAT folder can lead to “Not Scanned” results, as the Anti-Virus software will pick up the malware signatures for processing before the MetaDefender Core software can process. Therefore, MetaDefender Core will be unable to determine the hash, leading to the scan result “Not Scanned”.

Common Anti-Virus softwares to perform exclusions, including but not limited to: McAfee, Norton, Bitdefender, Malwarebytes, SentinelOne

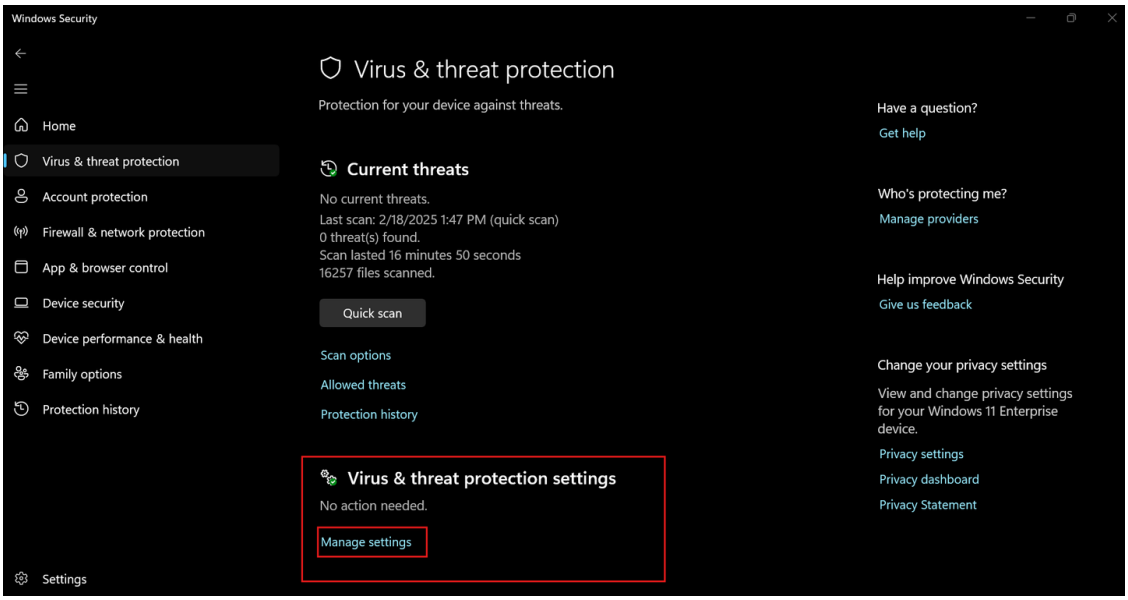
How to exclude on Windows Defender

Please ensure you have the proper permissions to add exclusions, such as administrator rights.

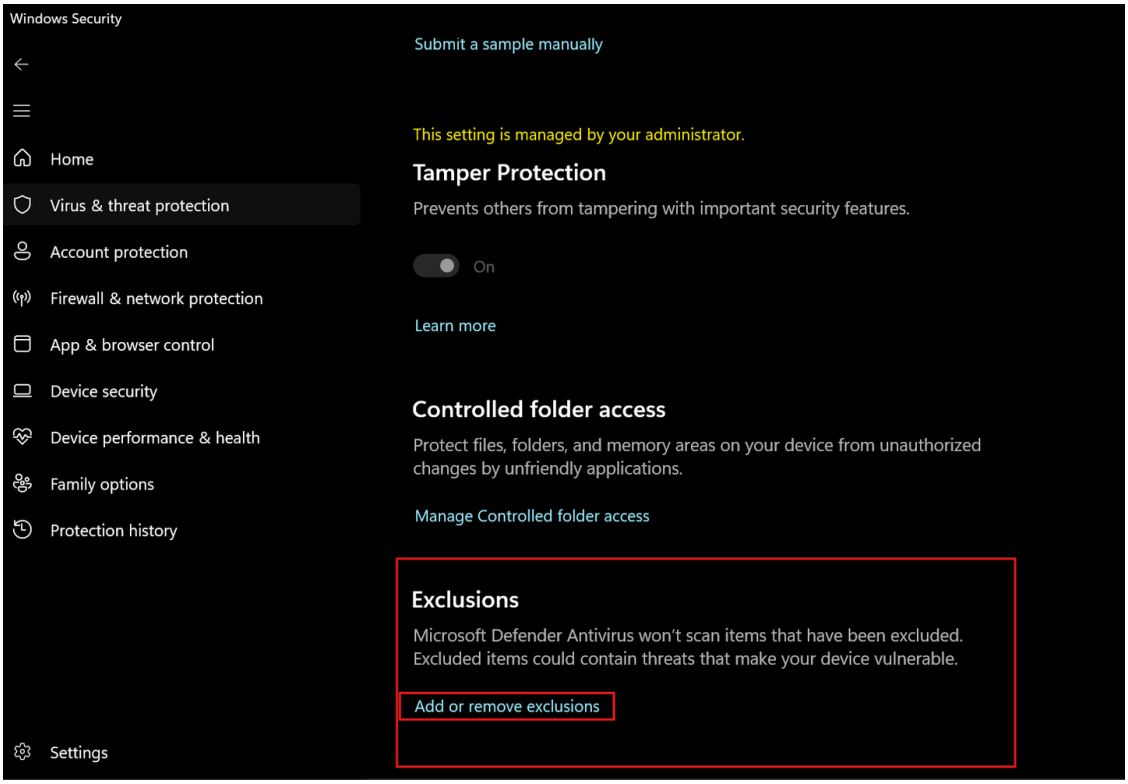
1. Navigate to “Windows Security” > “Virus & threat protection”



2. Under “Virus & threat protection settings” click “Manage settings”



3. Scroll down to “Exclusions” and click “Add or remove exclusions”



4. Add the folder “OPSWAT”, by default it is located in the install directory: "C:\Program Files\OPSWAT" [Your directory may be different depending on if you have performed a customer install, placing the OPSWAT folder inside a separate Drive]
5. Add the folder “Temp” stored at location `C:\Users\<user>\AppData\Local\Temp`.

Important Note regarding engine process exclusion

Important:

- If Windows Defender is used **only** as a local AV (and **not** as an AV engine inside MetaDefender Core), then you **may** additionally exclude these MetaDefender Core processes: `engineprocess.exe`, `engineprocess32.exe`, `ometascan.exe`, `postgres.exe`, and `nginx.exe`.
- However, if Windows Defender is configured **both** as the local AV **and** as an AV engine inside MetaDefender Core, then you **should not exclude** `engineprocess.exe` (and its variant) from the real-time protection exclusion list. Excluding the engine process in that scenario can impair the integrated AV engine’s functionality.

Registry Exclusion (Optional)

You can optionally exclude the following registry path:

`HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global`

For the values: `dlppath`, `quarantinepath`, `sanitizepath`.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Where are MetaDefender Core Install Directories and Registry?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

File Explorer:

By default, the following are the directories and use when MetaDefender Core is installed

- **"C:\Program Files\OPSWAT\MetaDefender Core"** - Main installation folder where tools and support package script are found.
- "**C:\Program Files\OPSWAT\MetaDefender Core\nginx**" - Web management files and logs files for the ability to access MetaDefender core on the browser
- "**C:\Program Files\OPSWAT\MetaDefender Core\data\resources**" **-** engine folders that store temporary files during scanning, files are deleted once the scan is complete
- "**C:\Program Files\OPSWAT\MetaDefender Core\data\updates**" - Folders [av] [db] containing the engines database update packages [zip] and instruction .YML file
- "**C:\Program Files\OPSWAT\MetaDefender Core\data\logs**" - Folder containing rotating core logs and upgrade log.
- "**C:\Program Files\OPSWAT\MetaDefender Core\data\engines**" - Folder containing loaded engines and their necessary components to complete scanning.
- **"C:\Program Files\OPSWAT\MetaDefender Core\data\enginelogs"** - Logs for each engine produced during scans
- "**C:\Program Files\OPSWAT\MetaDefender Core\data\pg_data**" - Postgres Database files and logs [only for local DB]
- "**C:\Program Files\OPSWAT\MetaDefender Core\postgres**" **-** Libraires and executables for the local database

Registry

```
Computer\HKEY_CURRENT_USER\Software\OPSWAT\MetaDefender Core

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan
```

Support:

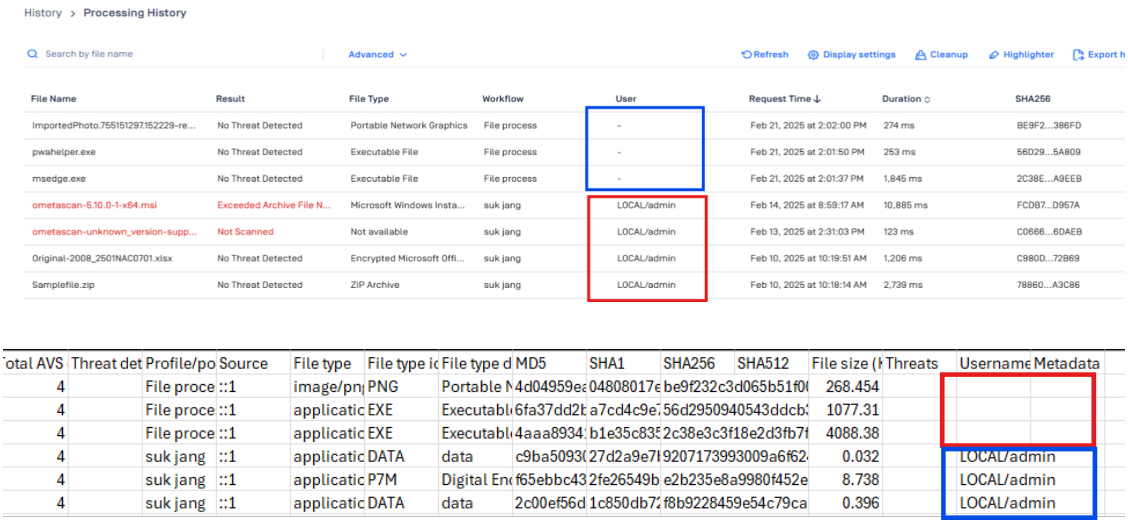
If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Under what circumstances does the scan result not show the username?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

When MetaDefender Core scans the files from each session, each session represents the information on the user that scans the files.



History > Processing History

Search by file name | Advanced | Refresh | Display settings | Cleanup | Highlighter | Export h

File Name	Result	File Type	Workflow	User	Request Time	Duration	SHA256
ImportedPhoto.7551512971522229-re...	No Threat Detected	Portable Network Graphics	File process	-	Feb 21, 2025 at 2:02:00 PM	274 ms	BE9F2...386FD
pwahelper.exe	No Threat Detected	Executable File	File process	-	Feb 21, 2025 at 2:01:50 PM	253 ms	56D29...5A809
msedge.exe	No Threat Detected	Executable File	File process	-	Feb 21, 2025 at 2:01:37 PM	1,845 ms	2C38E...A9EEB
ometascan-5.10.0-1-x64.msi	Exceeded Archive File N...	Microsoft Windows Insta...	suk jang	LOCAL/admin	Feb 14, 2025 at 8:59:17 AM	10,885 ms	FCDB7...D957A
ometascan-unknown_version-supp...	Not Scanned	Not available	suk jang	LOCAL/admin	Feb 13, 2025 at 2:31:03 PM	123 ms	C0666...6DAEB
Original-2008_2501NAC0701.xlsx	No Threat Detected	Encrypted Microsoft Offi...	suk jang	LOCAL/admin	Feb 10, 2025 at 10:19:51 AM	1,206 ms	C980D...72B69
Samplefile.zip	No Threat Detected	ZIP Archive	suk jang	LOCAL/admin	Feb 10, 2025 at 10:18:14 AM	2,739 ms	78860...A3C86

otal AVS	Threat det	Profile/po	Source	File type	File type ic	File type d	MD5	SHA1	SHA256	SHA512	File size (k	Threats	Username	Metadata
4		File proce	::1	image/pn	PNG	Portable N	4d04959e	04808017	be9f232c3d065b51f0		268.454			
4		File proce	::1	applicatic	EXE	Executabl	6fa37dd2	a7cd4c9e	56d2950940543ddcb		1077.31			
4		File proce	::1	applicatic	EXE	Executabl	4aaa8934	b1e35c83	2c38e3c3f18e2d3fb7		4088.38			
4		suk jang	::1	applicatic	DATA	data	c9ba5093	27d2a9e7	9207173993009a6f62		0.032		LOCAL/admin	
4		suk jang	::1	applicatic	P7M	Digital En	f65ebbc43	2fe26549b	e2b235e8a9980f452e		8.738		LOCAL/admin	
4		suk jang	::1	applicatic	DATA	data	2c00ef56d	1c850db7	f8b9228459e54c79ca		0.396		LOCAL/admin	

There are 2 ways to scan without username

1. Scanning Without Logging In

As shown in the image above, **MetaDefender Core allows file scanning without requiring user authentication**. When a file is scanned without logging in:

- No username or user-specific details are stored in the scan record.
- The file can still be processed and analyzed, but there is no user identification in the logs.
- This applies to scans initiated via the **public web interface** where authentication is not required.



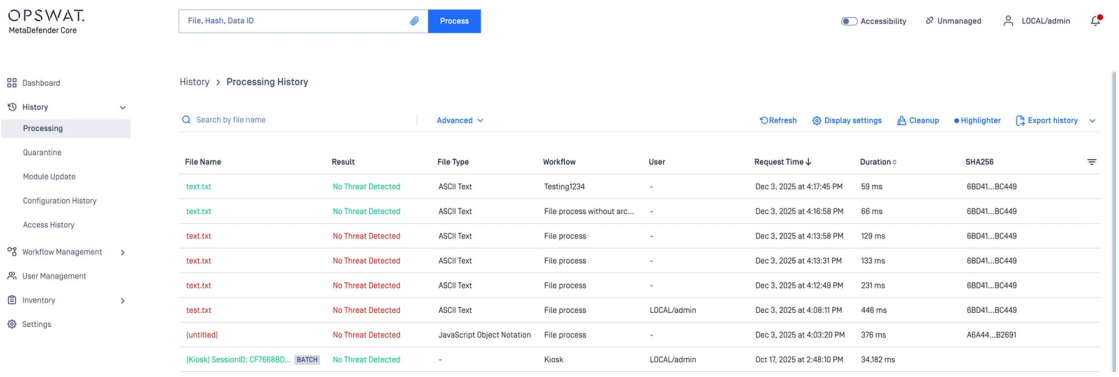
2. Scanning Files Through the REST API

When scanning files through the **MetaDefender Core REST API**, the behavior differs:

- The scan session is tracked by the API authentication token or session key.
- If an authenticated API request is used, the scan can be tied to the system or user executing the request.
- If performed anonymously (without authentication), the scan will not contain identifiable user details, similar to the public web interface.

3. After a User has been Deleted

If a certain user has been deleted after performing multiple scans, that user won't appear in the Processing History page anymore.



- This also applies if the scan request has been made through the REST API using the deleted user's credentials in order to obtain a session ID.

Recommendations

Export reports before deleting users If user accounts must be removed, export Scan History beforehand to preserve

user attribution.

Restrict WebScan workflows to authenticated users Set workflow access to **Restrict access to the following roles: Every authenticated** to prevent anonymous scans.

Use separate API credentials For REST API integrations, use different API keys for each application or service to improve scan traceability.

Track user attribution externally If scans are initiated through applications, pass end-user identifiers through metadata or custom headers and correlate them using external logging systems such as SIEM platforms.

Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

How to send MetaDefender Core logs to NAS (Network Attached Storage) Path?

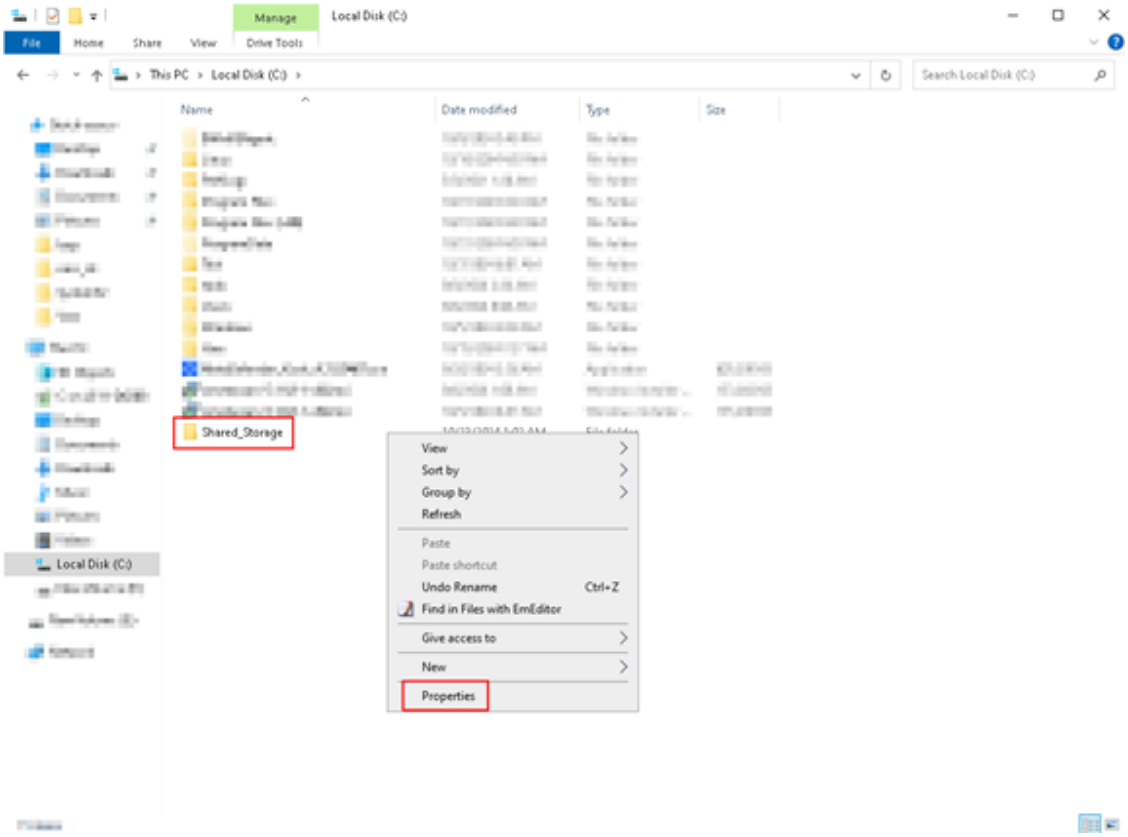
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

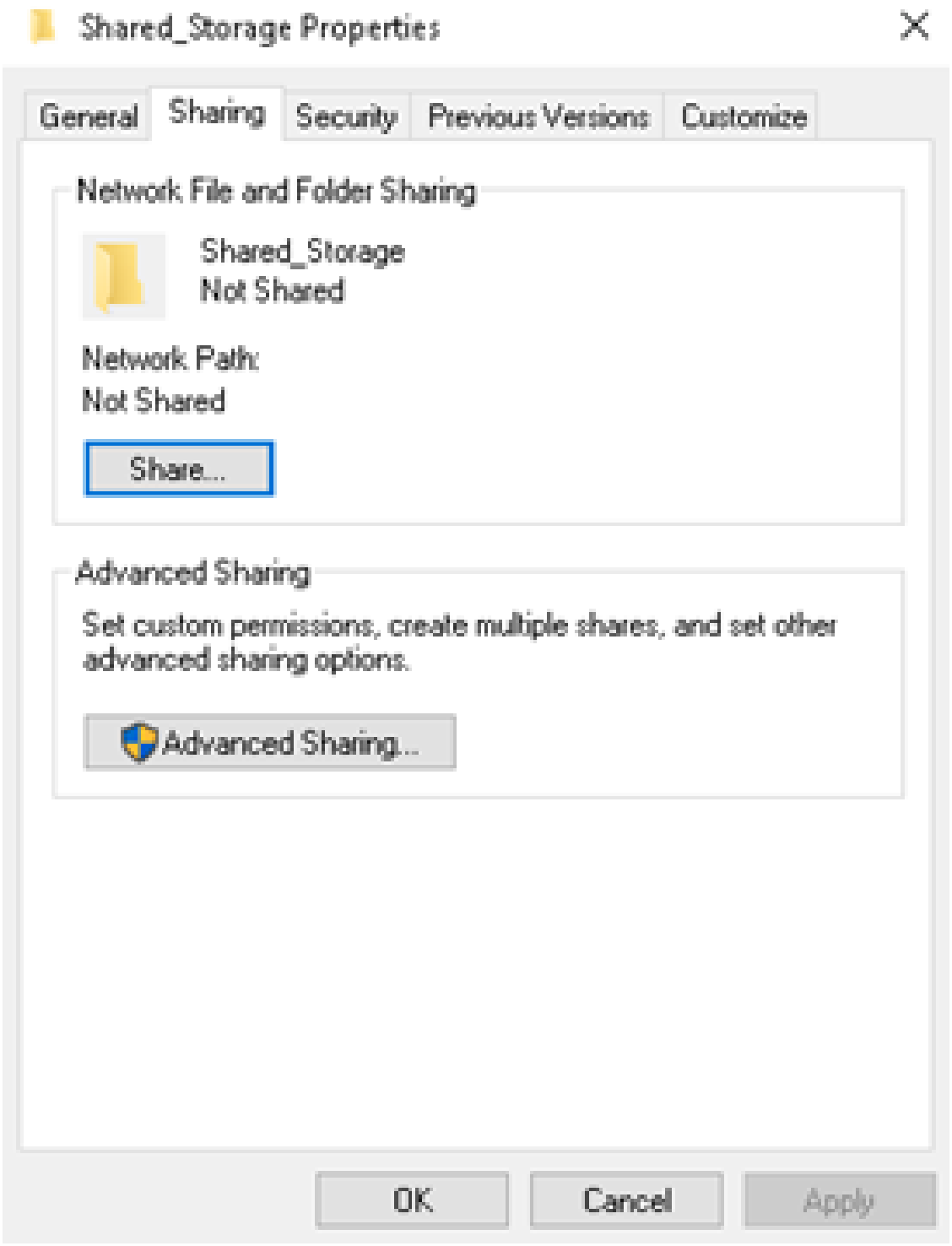
Creating a Network Attached Storage

In our example, we will use a **Windows Machine** to create a remote shared directory where we will store the logs of **MetaDefender Core** from a different machine.

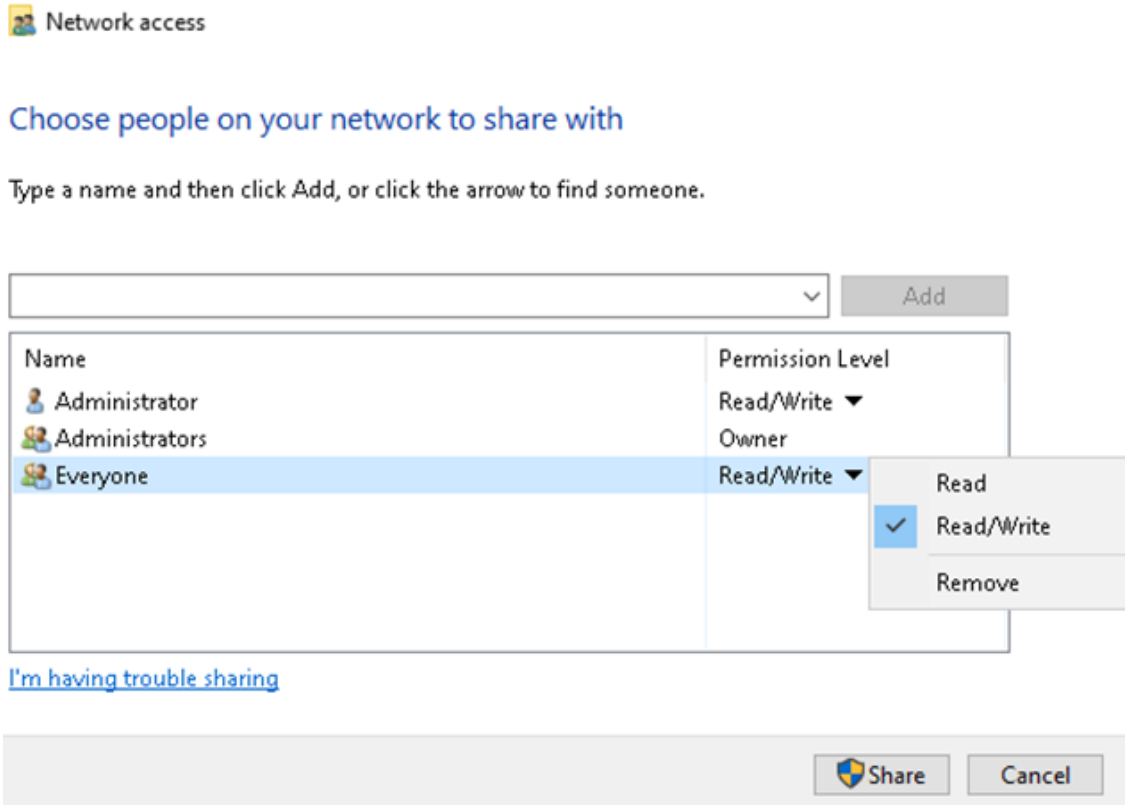
Step 1: Open the **File Explorer** and create a folder that is going to be shared.



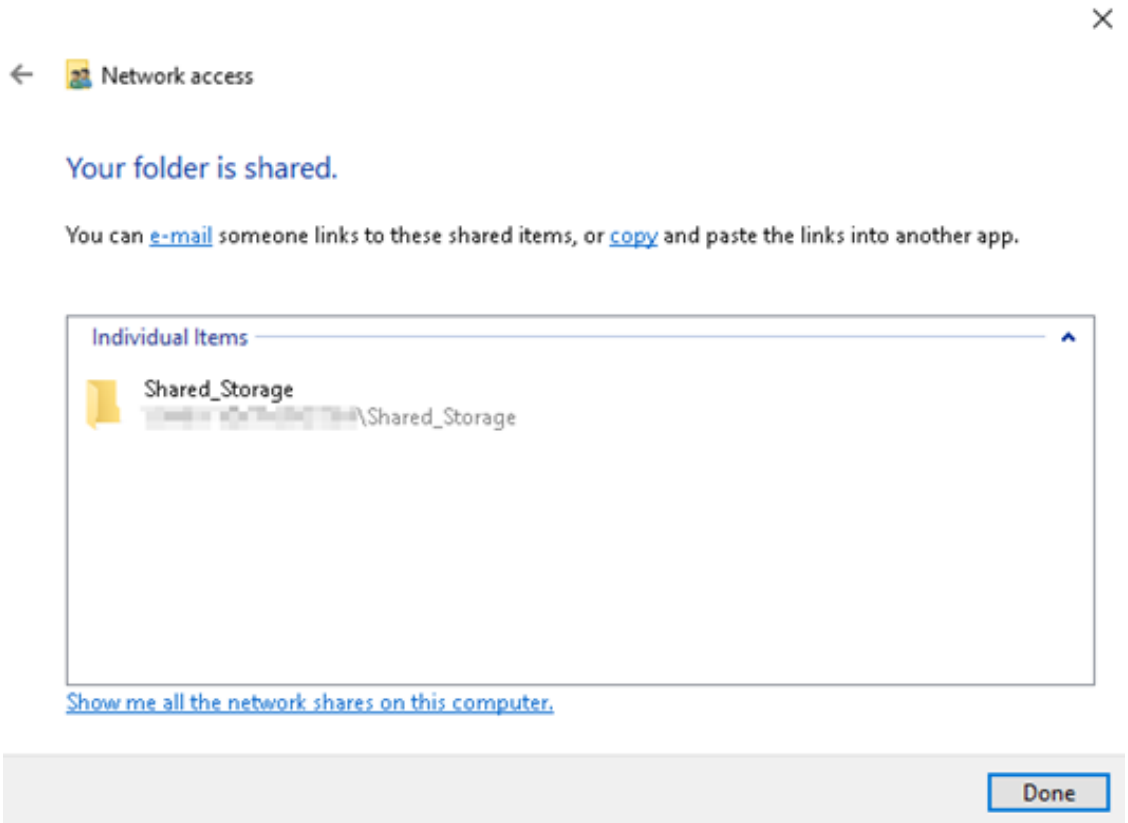
Step 2: *Right click* the created folder, open **Properties** and under **Sharing** tab click button **Share...**



Step 3: A Network Access tab will pop-up. Here you can add any users and set the permissions as needed



The folder is now accessible over the network:



Now let's configure MetaDefender Core to send the logs to this storage.

Linux:

Step 1: We will need a tool to allow us to mount the **Network Attached Storage**. Run the following commands to install **cifs-utils**:

```
sudo apt-get install cifs-utils
```

Step 2: Create a directory where you want to mount the storage, along with the log file:

```
sudo mkdir /core_logs
```

```
sudo touch /core_logs/core.log
```

Step 3: Mount the **Windows Share**, considering the **minimum permissions** ometascan service user will need:

> **Note**
> Permissions to set:
>
> - The directory that contains the logs: read, write and execute permissions for ometascan user or at least for ometascan group
> - The other directories on the path to the logs: at least read and execute permissions for ometascan user and/or ometascan group

Code · bash

```
sudo mount -t cifs //windows_server_ip/Shared_Storage /core_logs -o
username=windows_user,password=windows_password,domain=windows_domain,dir_mode=0777,file_mode=077
7
```

Step 4: Modify /etc/ometascan/ometascan.conf file to send the logs to the desired directory.

a. Open the file:

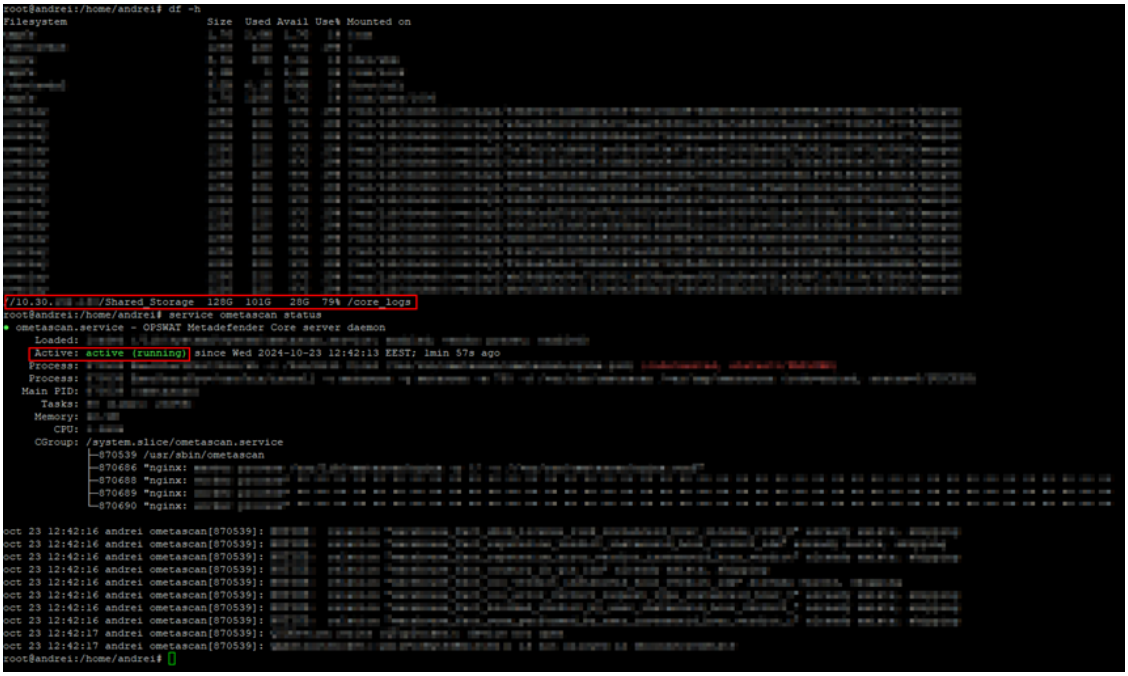
```
sudo vim /etc/ometascan/ometascan.conf
```

b. Change the logfile path to the configured one:

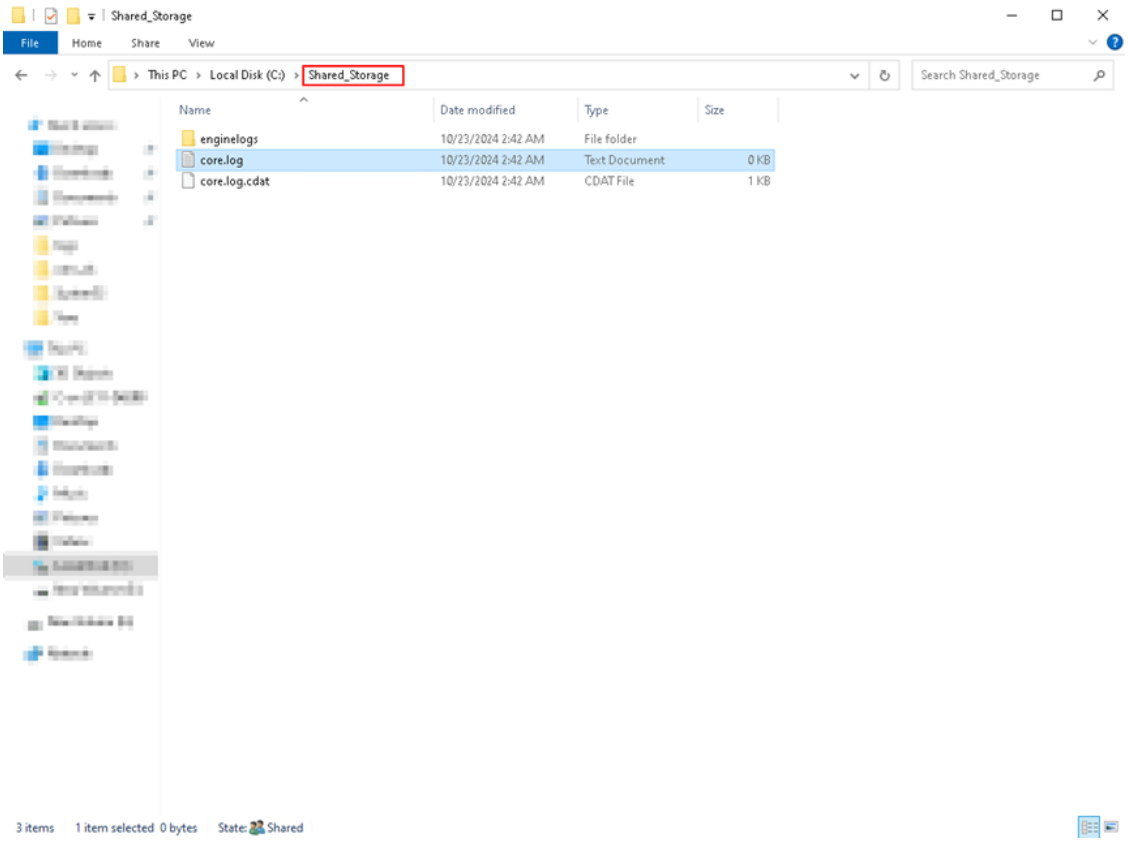
```
logfile=/core_logs/core.log
```

Step 5: Restart the MD Core Service:

```
service ometascan restart
```



Core will now send all the logs to the Network Attached Storage:



Windows:

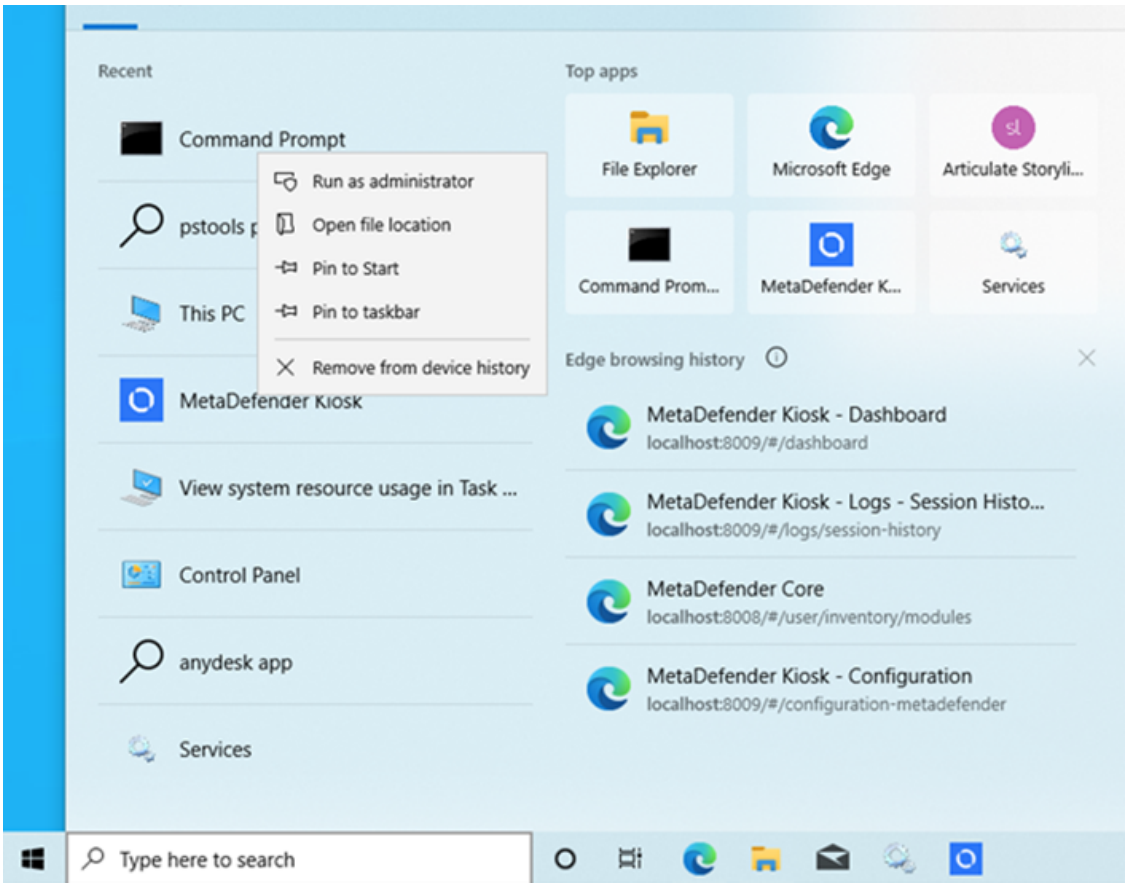
Step 1: In order to save logs to a mapped drive, the mapped drive must first be created under the **SYSTEM** account.

To do this, please follow the instructions below:

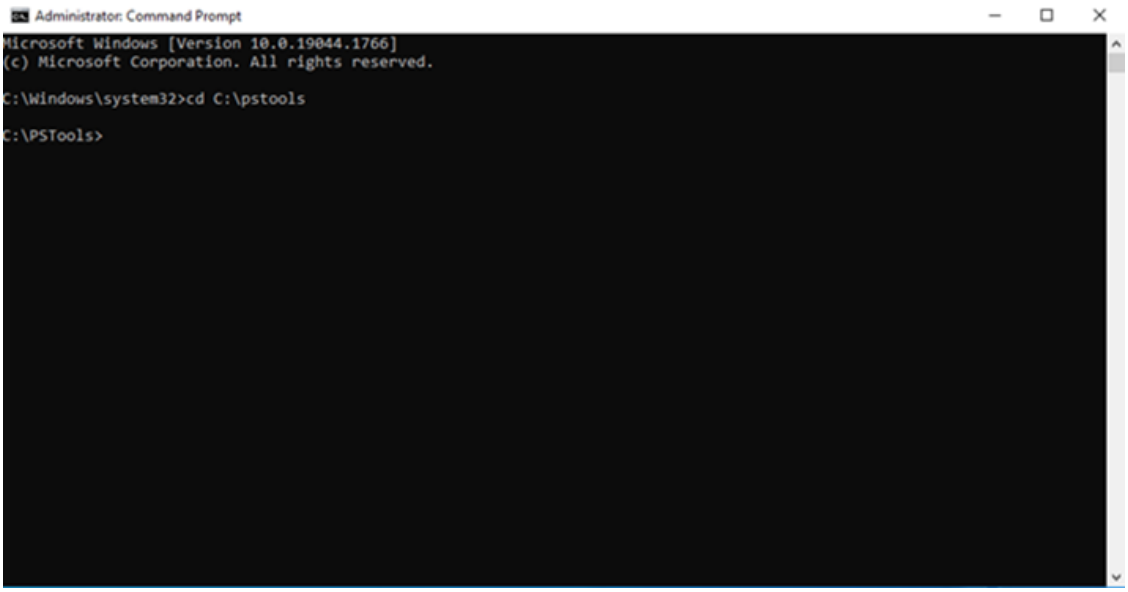
- Download the **psexec.exe.zip** file from Microsoft's Sysinternals Security Utilities suite by visiting this Webpage,

then extract it and delete the original zip file.

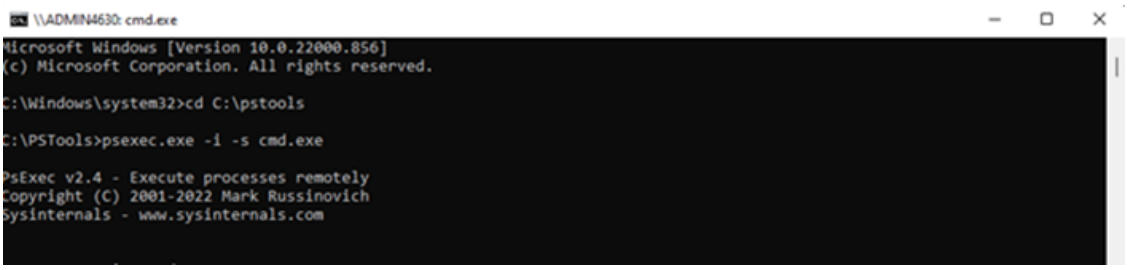
- Now, place the psexec.exe tool on the machine where you want the mapped drive to be created (e.g. **C:\PSTools\psexec.exe**).
- Open the command prompt, right click and select: **Run as Administrator.**



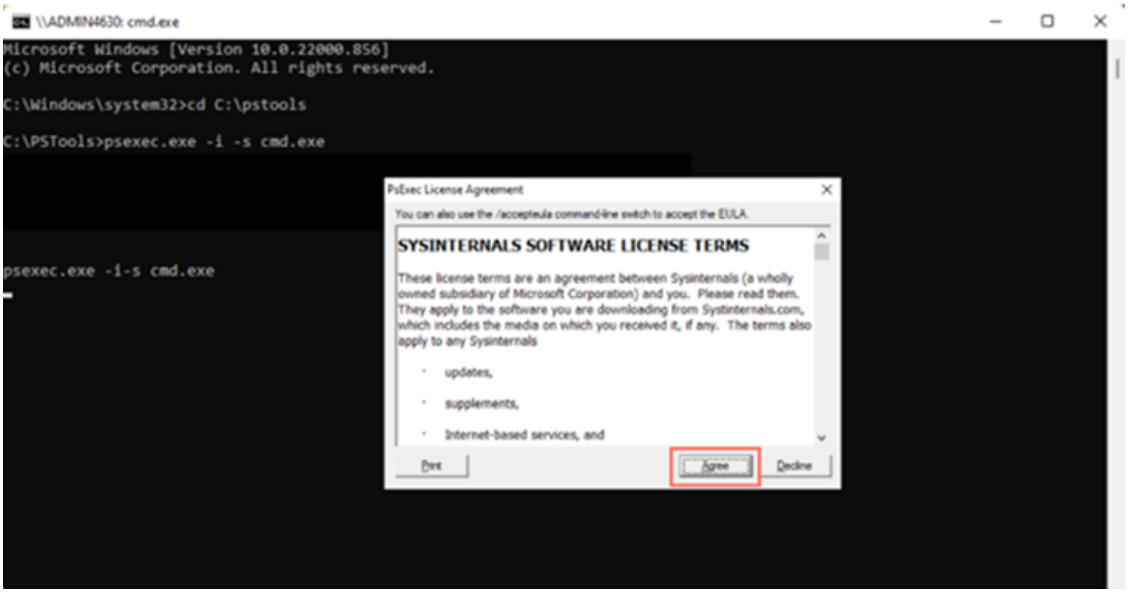
- In the command prompt, navigate to the directory where psexec.exe was copied (e.g. `cd C:\pstools`, as illustrated in the screenshot below).



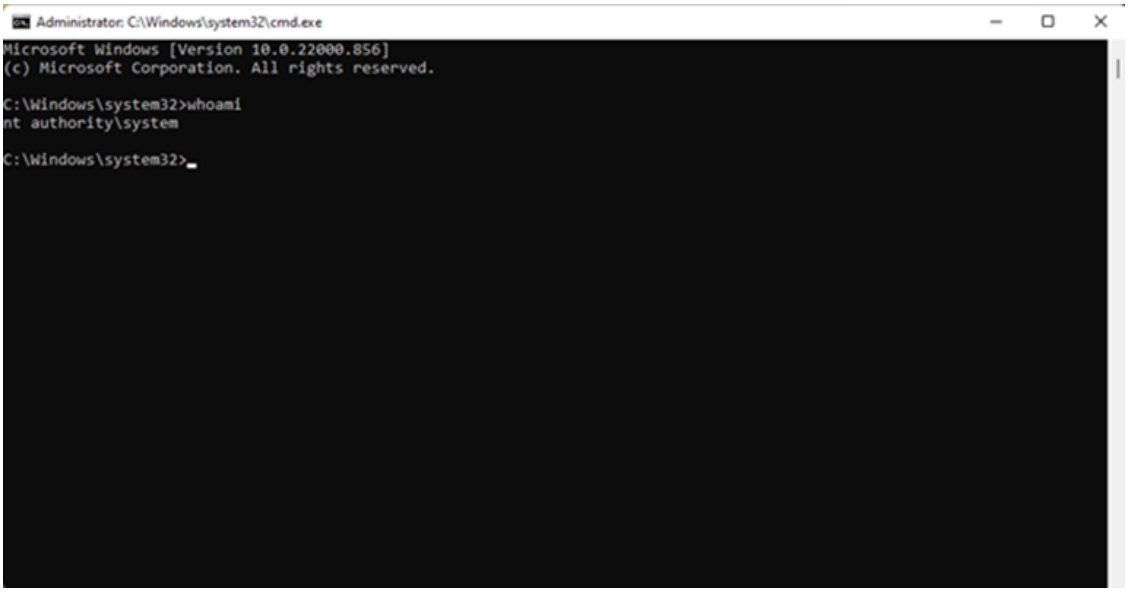
- Enter the following command `psexec.exe -i -s cmd.exe`, as illustrated in the screenshot below:



- If the Sysinternals pop-up appears, click **Agree**, as illustrated in the screenshot below.



- An elevated command prompt window will open.
- Enter the command: **whoami** to confirm that the new command prompt is running as: **nt**

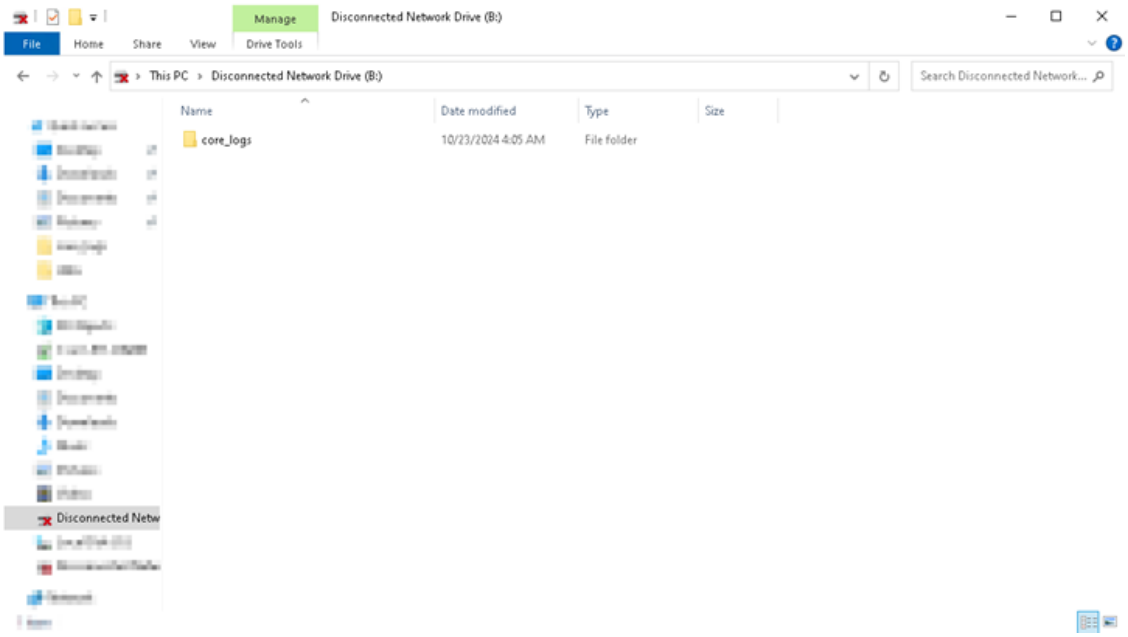


- Enter the following command to mount the mapped drive: **use <desired drive letter> <UNC path to map the drive**

to> (e.g. net use Z: \file-server-1\share).

```
C:\Windows\system32>net use B: \\10.30.[redacted]Shared_Storage
The command completed successfully.
```

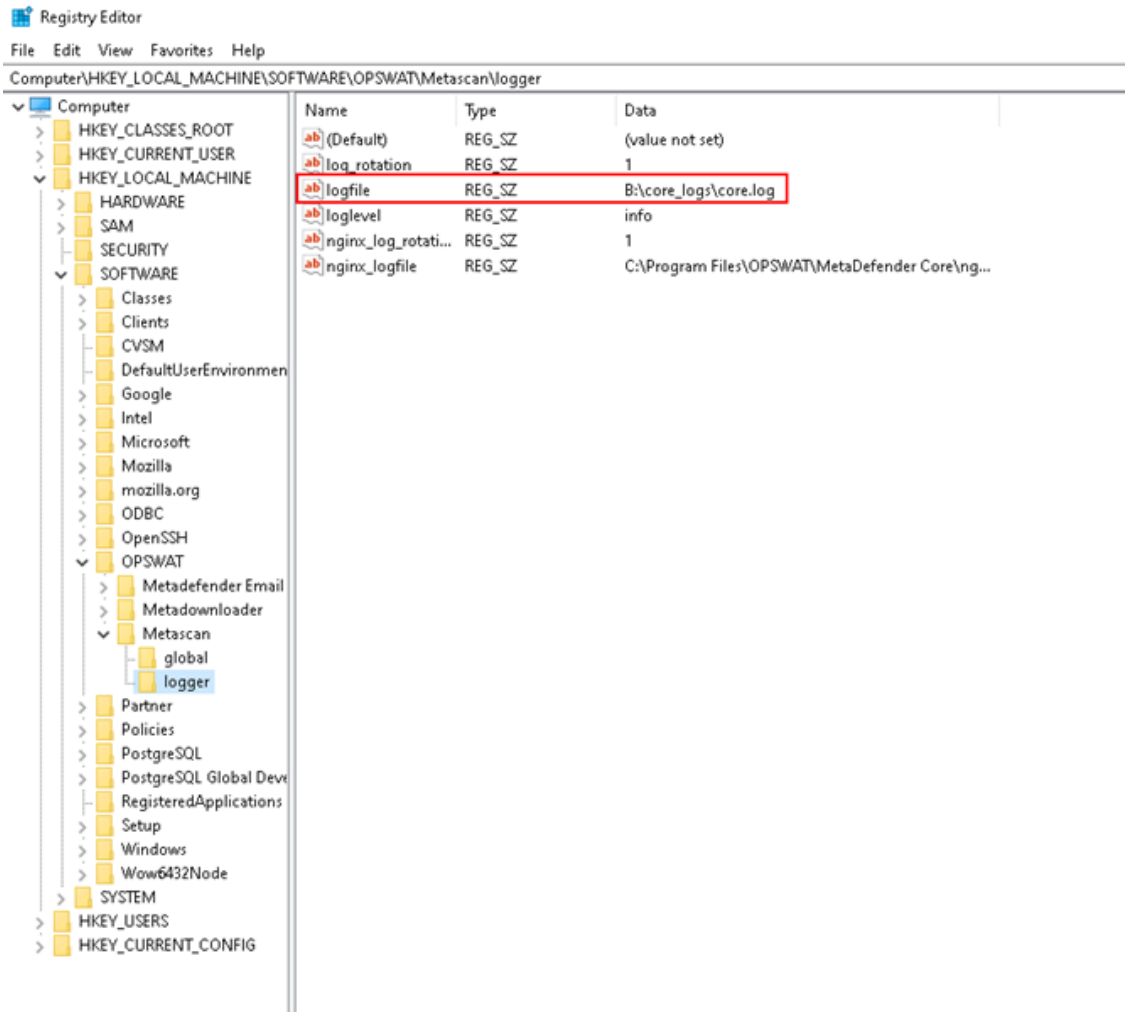
Step 2: Now that we have mapped the **Shared_Storage**, we have created a new **core_logs** folder under which we have created **core.log** file, where we are going to send the logs.



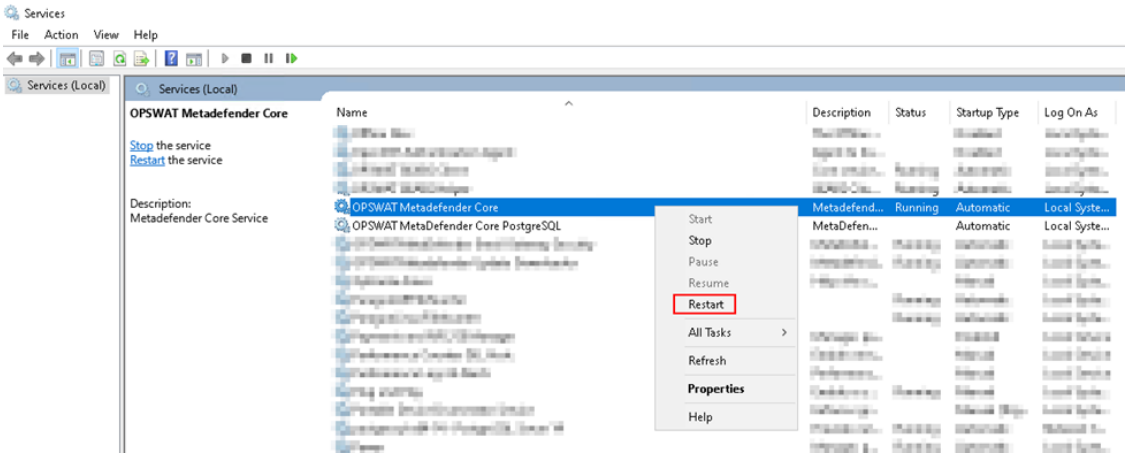
Step 3: Open **Registry Editor** and navigate to the following path:

`Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`

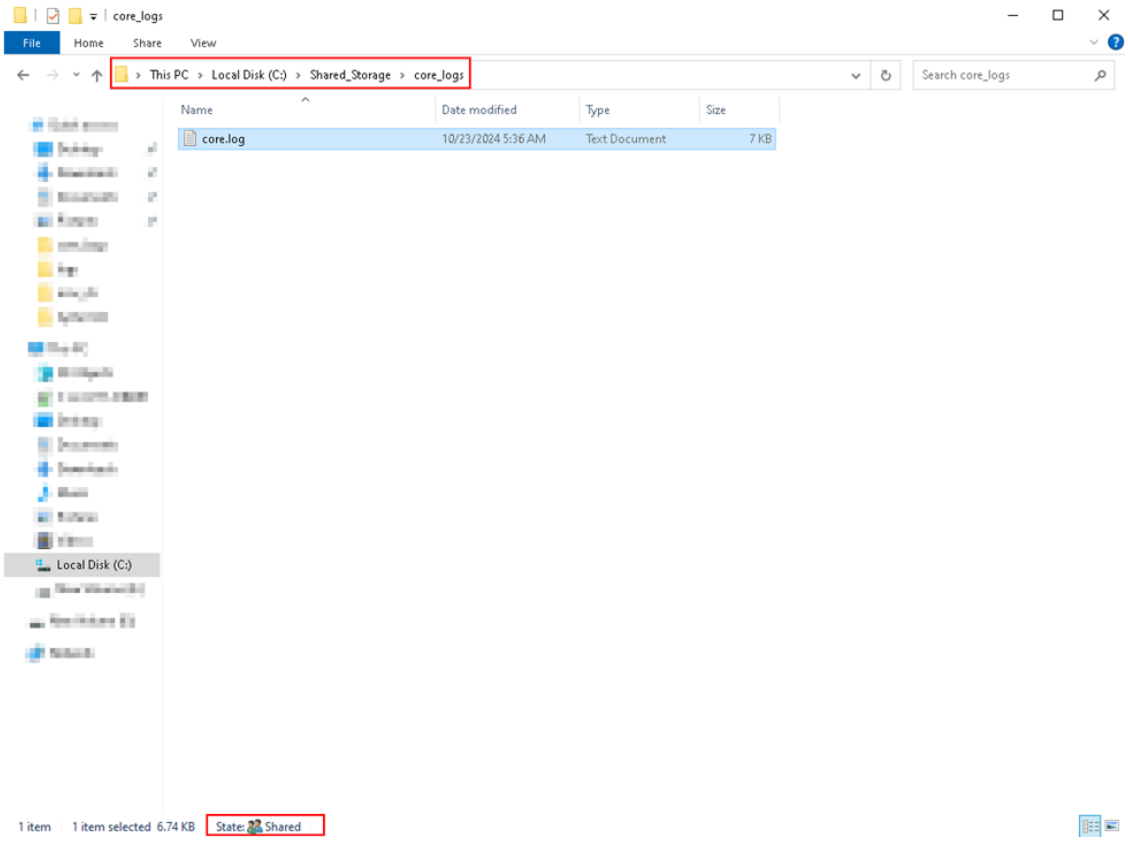
Here you will need to change the **logfile** value to the path we have just configured:



Step 4: Open **Services** and restart **OPSWAT MetaDefender Core** service for the changes to take place:



MetaDefender Core will now send the logs to the configured NAS (Network Attached Storage Path):



Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Why is a file blocked by Deep CDR or Proactive DLP but I can still download the sanitized version?

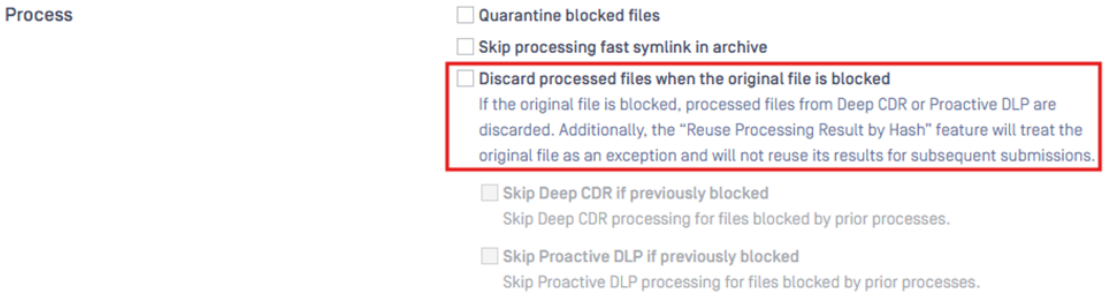
Check Your Version:

This applies to MetaDefender Core version 5.13.1 and newer.

In many use cases, Deep CDR and Proactive DLP will mark files as blocked due to the detection of sensitive information as configured in the workflow. However, sanitized versions of these files will still be generated and will be available for retrieval as per the sanitized file availability configuration under the data retention policy.

This is expected behavior, as we intend to offer a rich customization experience so that the workflows can be tailored to each specific use case.

To ensure that if Deep CDR or Proactive DLP blocks a file it will not be available to be retrieved even if sanitized, we have introduced a new setting under the workflow configuration, in the General tab:



This setting will cause the download functionality for any files that are blocked by Deep CDR or Proactive DLP to be completely inaccessible, even if accessing these files in the MetaDefender Core UI.

In addition, we have introduced two more settings for improving scanning time, so that you can configure Deep CDR or Proactive DLP to not even attempt to process files if they are blocked by any other engines that run first, such as the AV engines.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Does Deep CDR support Adobe Illustrator (AI) files in PostScript format?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Answer:

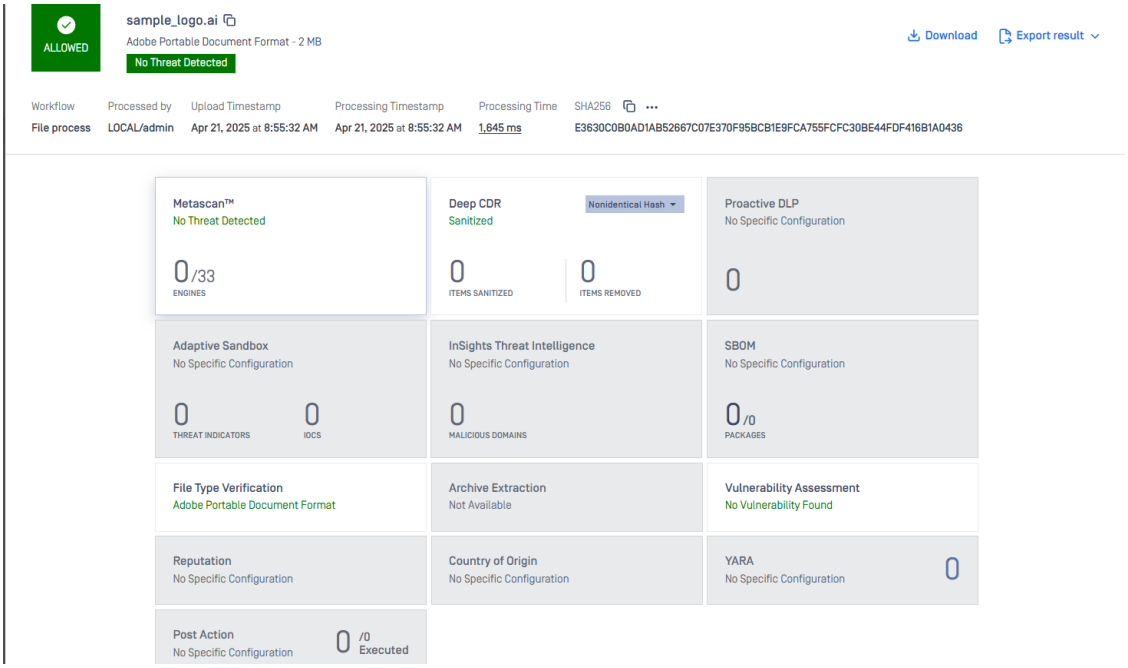
No, Deep CDR does not support AI files in PostScript format.

Adobe Illustrator (AI) files can be saved or exported in two formats:

✅ **AI file in PDF format**: This file format is essentially a PDF that contains vector graphics and design elements created using Adobe Illustrator. It is widely used for sharing and viewing vector designs while retaining high quality.

❌ **AI file in PostScript format**: This file format contains vector data and design elements described in the PostScript language, which is primarily intended for high-quality printing.

In Deep CDR, only **AI files in PDF format** are supported. Due to the nature of the PostScript format, AI files in PostScript format are not compatible with the Deep CDR engine.



For the complete list of supported file types for DeepCDR, please refer to the following article: Supported File Types

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to add the source IP in scan details?

Overview:

By default, when triggering scan from client through a load balancer or proxy server, MetaDefender Core's Nginx web server will not identify the original client source (but the load balancer or proxy server's address instead).

This article provides step-by-step instructions to configure MD Core's Nginx web server to display the true source IP address in the scan details.

Instructions:

Step 1: Create or Modify the extra.conf File

1. Access the configuration folder:

- **Windows**: Navigate to the directory where MD Core is installed, typically found at: *C:\Program Files\OPSWAT\MetaDefender Core\nginx*
- **Linux**: Navigate to the configuration directory: */etc/ometascan/nginx.d/*

2. Create or Edit the extra.conf file:

3. Add Configuration for Real IP Handling:

- Insert the following lines into the extra.conf file:

set_real_ ip_from 192.168.1.1;

real_ip_header X-Forwarded-For;

real_ip_recursive on;

- **Important**: Replace 192.168.1.1 with the actual IP address of your Load Balancer or proxy server.
- These settings instruct the Nginx web server used by MD Core to trace back and record the original client IP from the X-Forwarded-For HTTP header, which is commonly used by Load Balancers and proxies to pass along the client's real IP.

Step 2: Restart the OPSWAT MetaDefender Core Service

1. Restart the Service:

Windows:

- Open the Services management console by typing services.msc in the Run dialog (Win + R).
- Locate and select the **OPSWAT MetaDefender Core** service.
- Click **Restart**.

Linux:

Use the following command to restart the service:

sudo systemctl restart ometascan

2. Verify the Configuration:

- After restarting the service, initiate a scan through the Load Balancer or proxy server.
- Check the scan details in the MD Core interface to confirm that the "Source" IP address now reflects the true IP of the client that initiated the scan.

Troubleshooting:

- **Configuration Not Taking Effect**: Double-check the extra.conf file for typos, particularly in the IP address and Nginx directives.
- **Service Restart Issues**: Ensure you have the necessary permissions to restart services on the server. If problems persist, consult the logs for errors related to Nginx or MD Core.

Conclusion:

By following these steps, you can successfully configure MD Core to display the original client's IP address in scan details, even when behind a Load Balancer or proxy server. This configuration enhances the accuracy of your logging and provides better visibility into the true source of scan requests.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I integrate Jenkins with MetaDefender Core?

Check Your Version:

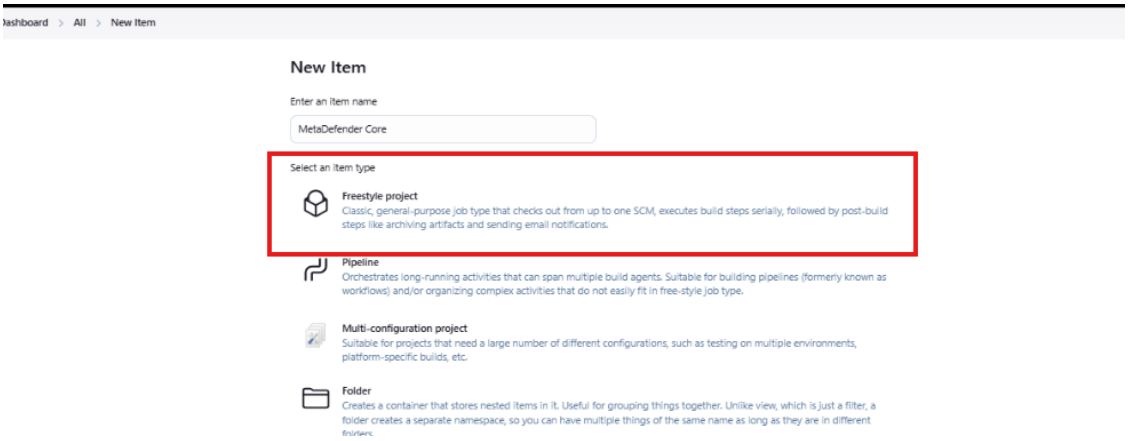
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Summary:

MetaDefender Core offers a free Jenkins plugin via the MetaDefender Plugin for Jenkins, a free tool from OPSWAT. However, this tool is currently not documented in the MetaDefender Core Knowledge Base (KB).

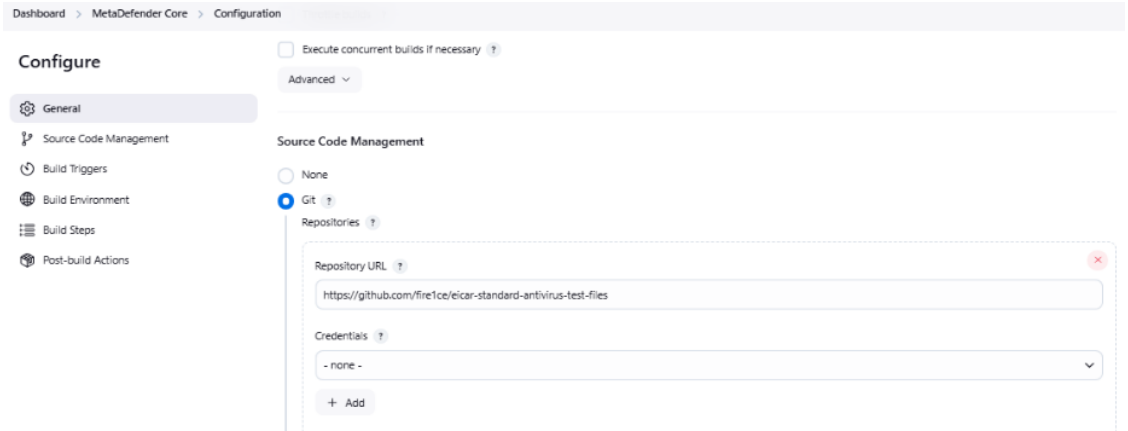
New Pipeline

Create a new pipeline using the 'Freestyle project' option. Use the name 'MD Core' for testing purposes.

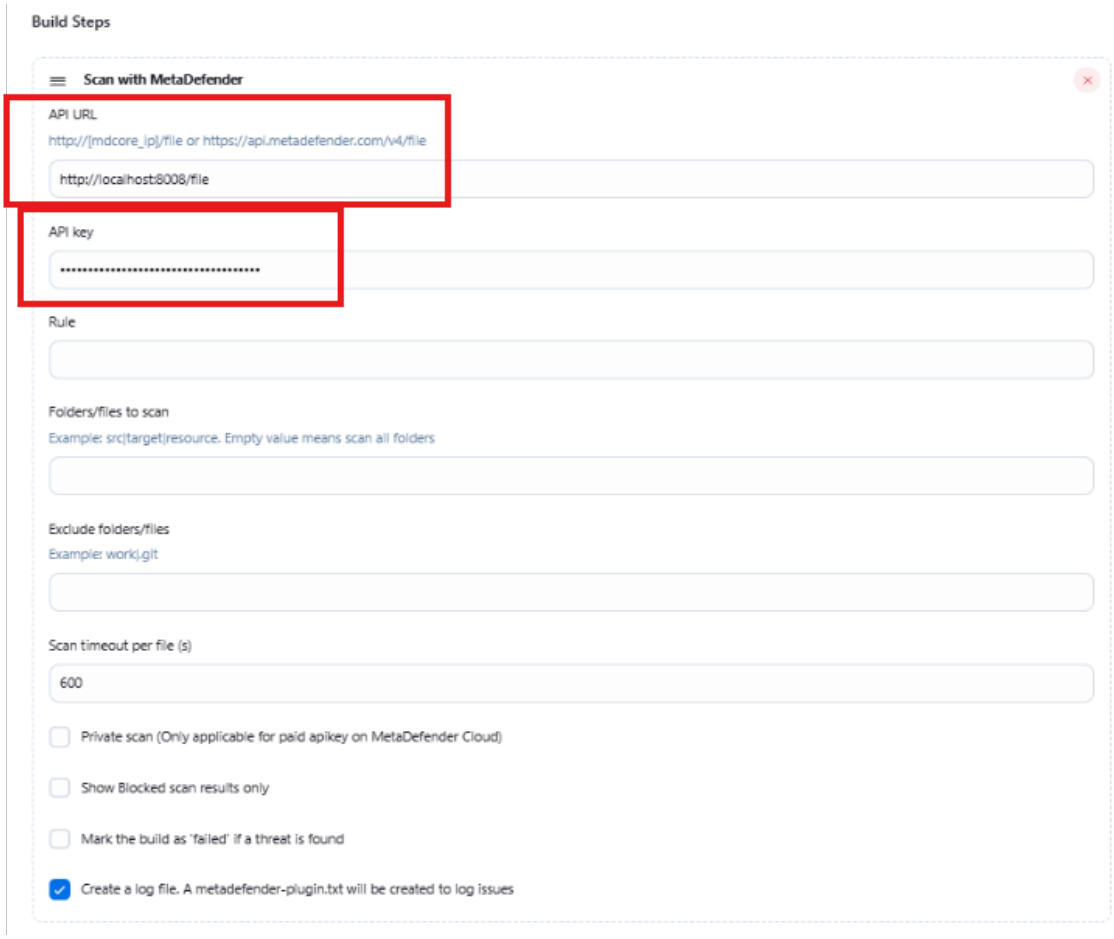


Test files

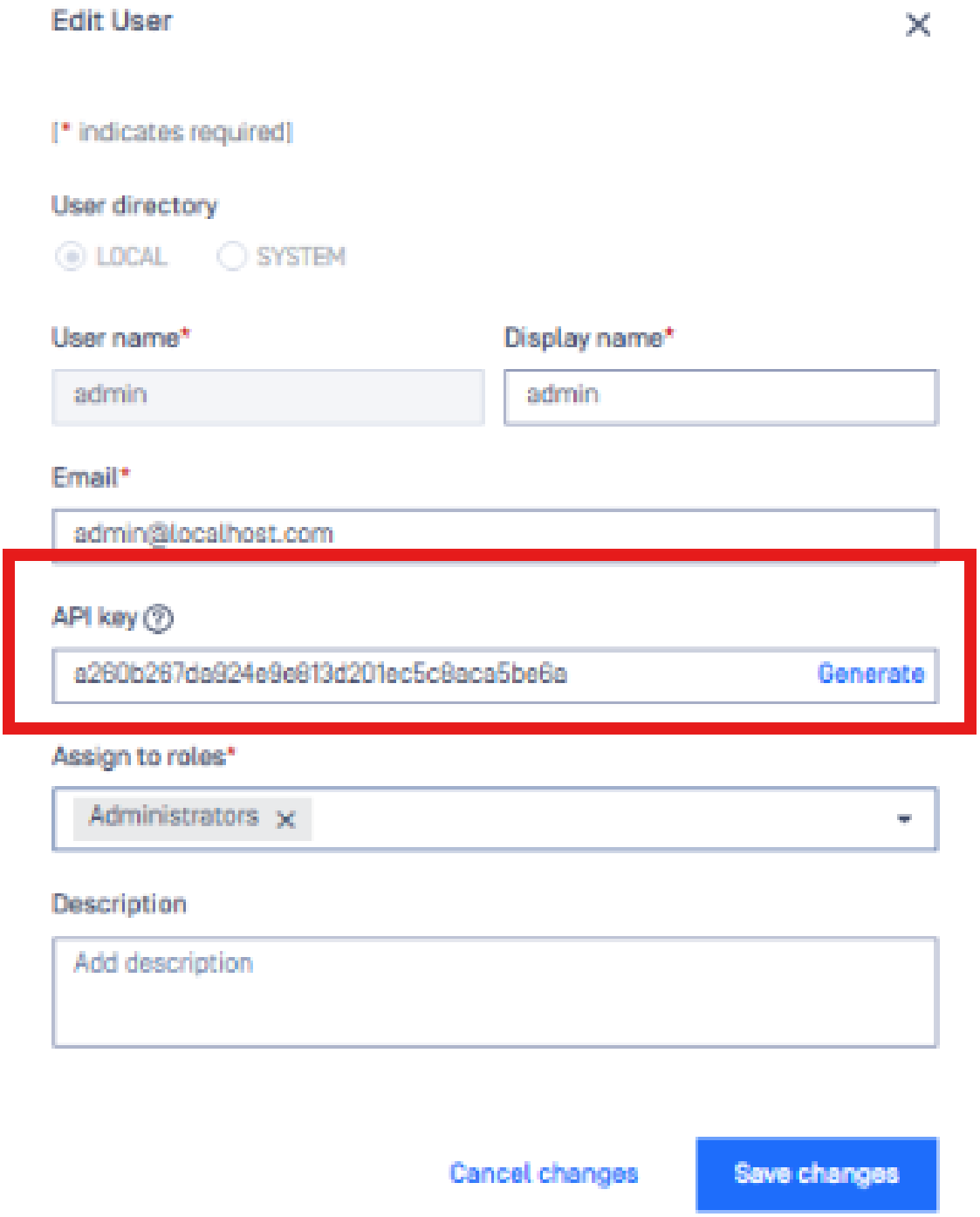
You can scan the test files from the EICAR GitHub repository. Here's the link for reference: GitHub - fire1ce/eicar-standard-antivirus-test-files: eicar standard antivirus test files



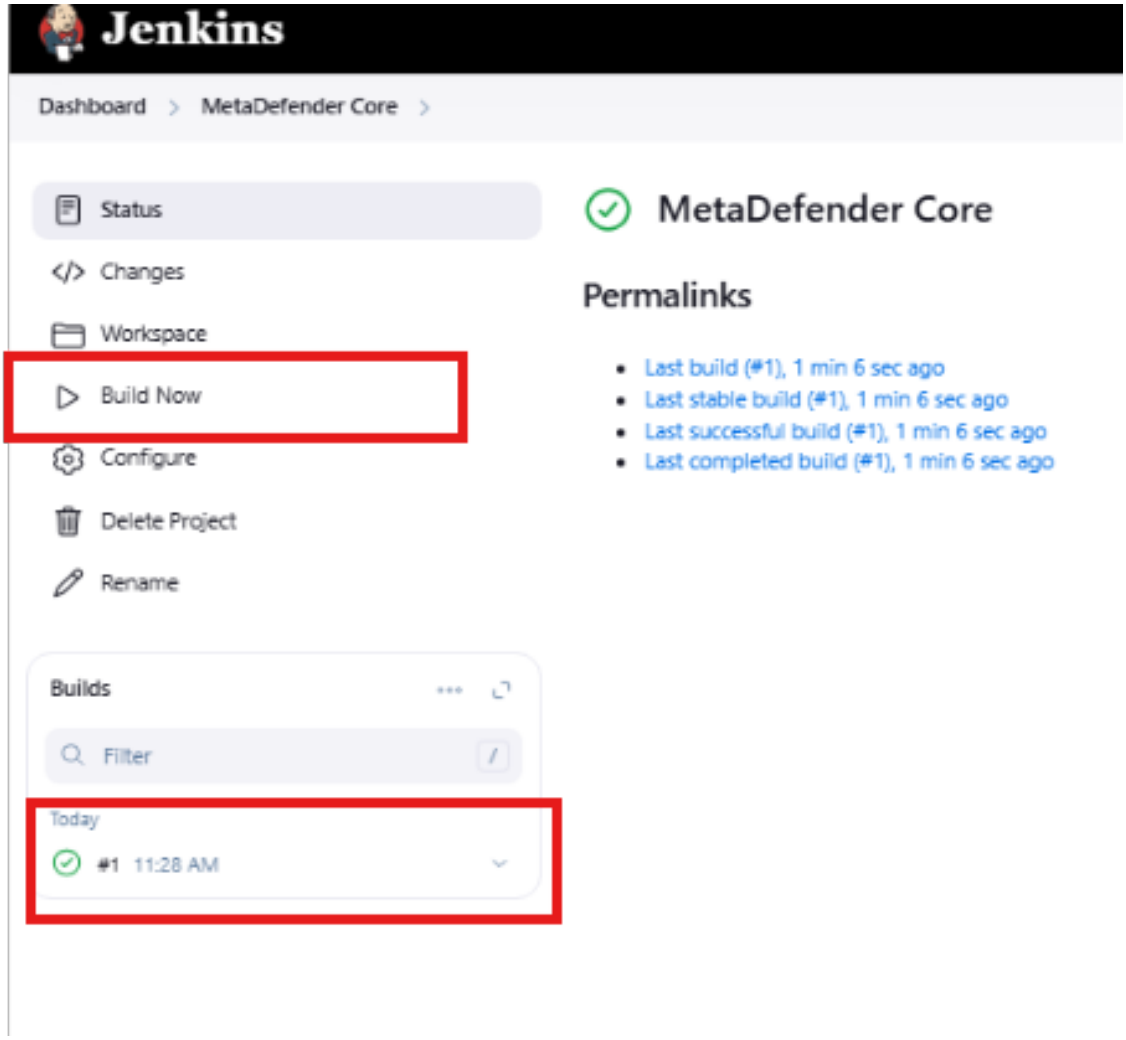
We will then add the steps to build the pipeline with MetaDefender Core



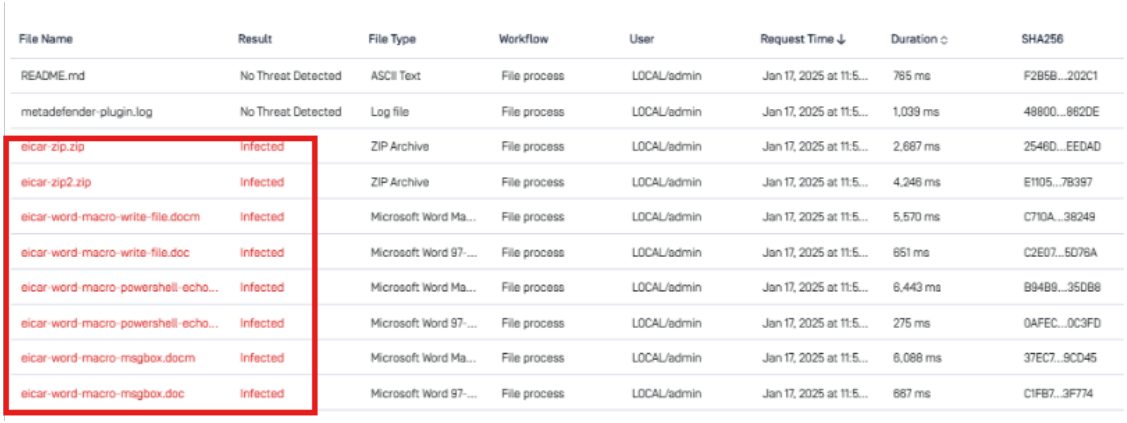
Since this test is performed on a local machine, the API URL is set to http://localhost:8008/file and the API key will be used from the user management API key.



Now, save the configuration and start testing the pipeline by clicking 'Build Now'.




```
[MetaDefender Plugin] [INFO] .git\objects\07\6caa47e105c3a1732298444bc3c28a9313f0dc |
http://localhost:8008/#/public/process/dataId/724e348bd9174c98a7d14400ae38b7c8 | Infected
[MetaDefender Plugin] [INFO] .git\objects\0c\de84f06a5dc24809569e0dce08f998c5820bc1 |
http://localhost:8008/#/public/process/dataId/21a66213dfb244d7b5c96e891954c292 | No Threat Detected
[MetaDefender Plugin] [INFO] .git\objects\11\3635e190f1d0b7e24baaefcc6d128c2d8b3607 |
http://localhost:8008/#/public/process/dataId/a4c4202149e94778b43f23660136f060 | Infected
[MetaDefender Plugin] [INFO] .git\objects\1c\203f130f9c624f85f2c64fbb0335f4f940a330 |
http://localhost:8008/#/public/process/dataId/cb9a88c6a54f4c729ba377f3ae671ad7 | Infected
[MetaDefender Plugin] [INFO] .git\objects\24\bbdfaa7b90dfea34c95abf886b5d8a6132b73f |
http://localhost:8008/#/public/process/dataId/46eea8ce2c4e4b71a676e1204df97a5b | Infected
[MetaDefender Plugin] [INFO] .git\objects\28\6abd47f0d7e5fbe20bc9f7b3331bbc0a3c1286 |
http://localhost:8008/#/public/process/dataId/074d9ec339c24441a4ff87dc6a47150c | No Threat Detected
[MetaDefender Plugin] [INFO] .git\objects\2c\0513291bdddeded6be4845e40c543d37cc4a2f |
http://localhost:8008/#/public/process/dataId/372695c7632a4334843b47f066847ffc | Infected
[MetaDefender Plugin] [INFO] .git\objects\2c\d88ba3a0988ac44ddec256468207f349c63977 |
http://localhost:8008/#/public/process/dataId/5cd63e965d834afea7b038c6eff5f49d | Infected
[MetaDefender Plugin] [INFO] .git\objects\2e\0f94f6ca198617a368f51a00e6bb51ac939d5f |
http://localhost:8008/#/public/process/dataId/b67a932bcd9642d79617c05272e7e980 | No Threat Detected
[MetaDefender Plugin] [INFO] .git\objects\37\dfdbc562b3c4beb9b8958df52c381ff7a88e7e |
http://localhost:8008/#/public/process/dataId/e920a59785b247788a6f376b07db9c2f | Infected
```



File Name	Result	File Type	Workflow	User	Request Time ↓	Duration ◇	SHA256
README.md	No Threat Detected	ASCII Text	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	765 ms	F2B5B...202C1
metadefender-plugin.log	No Threat Detected	Log file	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	1,039 ms	48800...862DE
eicar-zip.zip	Infected	ZIP Archive	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	2,687 ms	2546D...EEDAD
eicar-zip2.zip	Infected	ZIP Archive	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	4,246 ms	E1105...7B397
eicar-word-macro-write-file.docm	Infected	Microsoft Word Ma...	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	5,570 ms	C710A...38249
eicar-word-macro-write-file.doc	Infected	Microsoft Word 97-...	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	651 ms	C2E07...5D76A
eicar-word-macro-powershell-echo...	Infected	Microsoft Word Ma...	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	6,443 ms	B94B9...35DB8
eicar-word-macro-powershell-echo...	Infected	Microsoft Word 97-...	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	275 ms	0AFEC...0C3FD
eicar-word-macro-msgbox.docm	Infected	Microsoft Word Ma...	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	6,088 ms	37EC7...9CD45
eicar-word-macro-msgbox.doc	Infected	Microsoft Word 97-...	File process	LOCAL/admin	Jan 17, 2025 at 11:5...	687 ms	C1FB7...3F774

Support:

If you have followed all of the steps above but are still unable to resolve the issue. Please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Can I apply a new license key before the current one expires?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Yes, you can apply a new license key in advance as part of your license renewal process. OPSWAT provides a grace period prior to the official start date of your renewed license to ensure a smooth transition and avoid last-minute license renewal updates.

To confirm the start date of your new key and ensure a seamless renewal, feel free to contact OPSWAT Technical Support within the week of your current license's end date. You may also reach out to your Account Owner for further assistance.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to automate backup configuration settings for MD Core? (API)

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows.

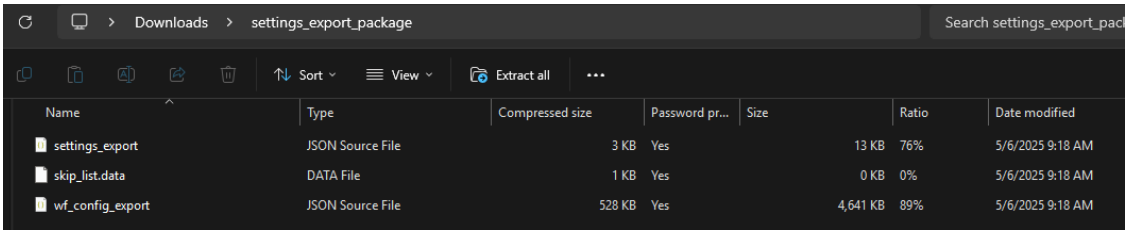
This article provides a simple Bash script template that allows users to automate the export of MetaDefender Core configuration using the `/admin/export/v2` API endpoint. For reference: GET - Export configuration

Prerequisites

- A valid **MetaDefender Core API key** (`session_id`), which the user has received from the post request for Login POST - Login

Export settings:





Bash script template:

```
#!/bin/bash

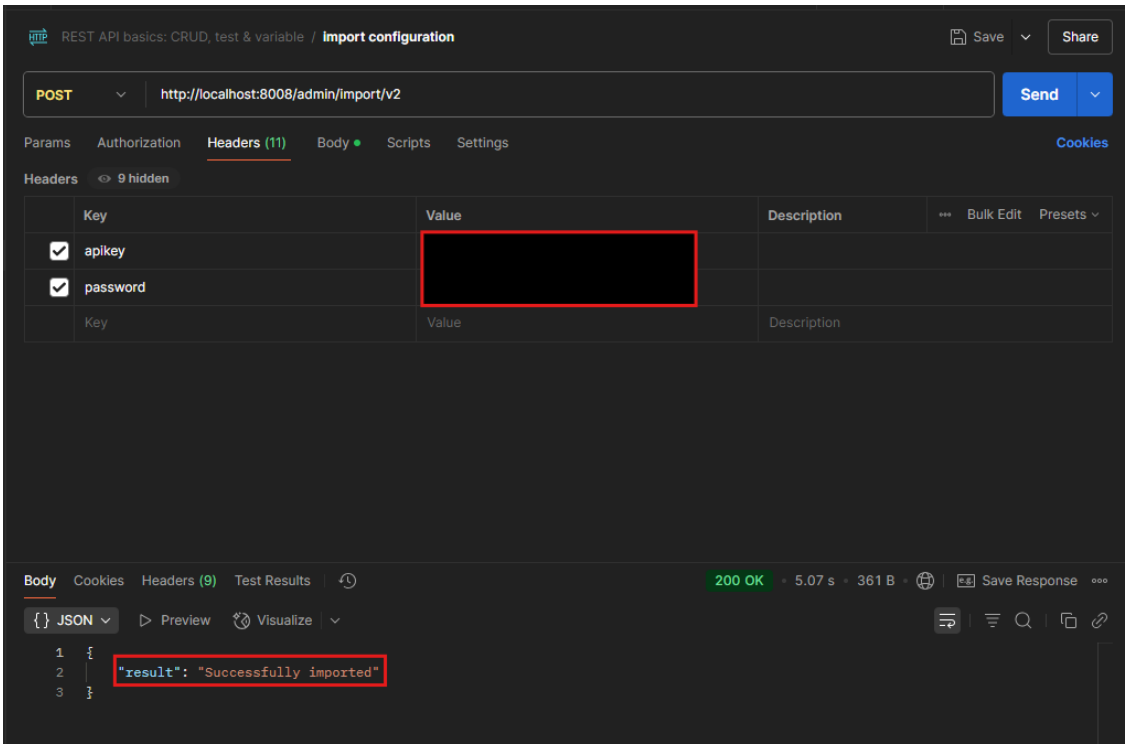
# === CONFIGURATION ===
API_URL="http://localhost:8008/admin/export/v2"
API_KEY="your_api_key_here"            # Use session_id from /login
PASSWORD="your_password_here"          # Password to encrypt exported config
OUTPUT_FILE="mdcore_config.zip"        # The name you wanted for the zip file
SETTING_JSON='{"yara":{"value":false}}'  # Optional: true or false

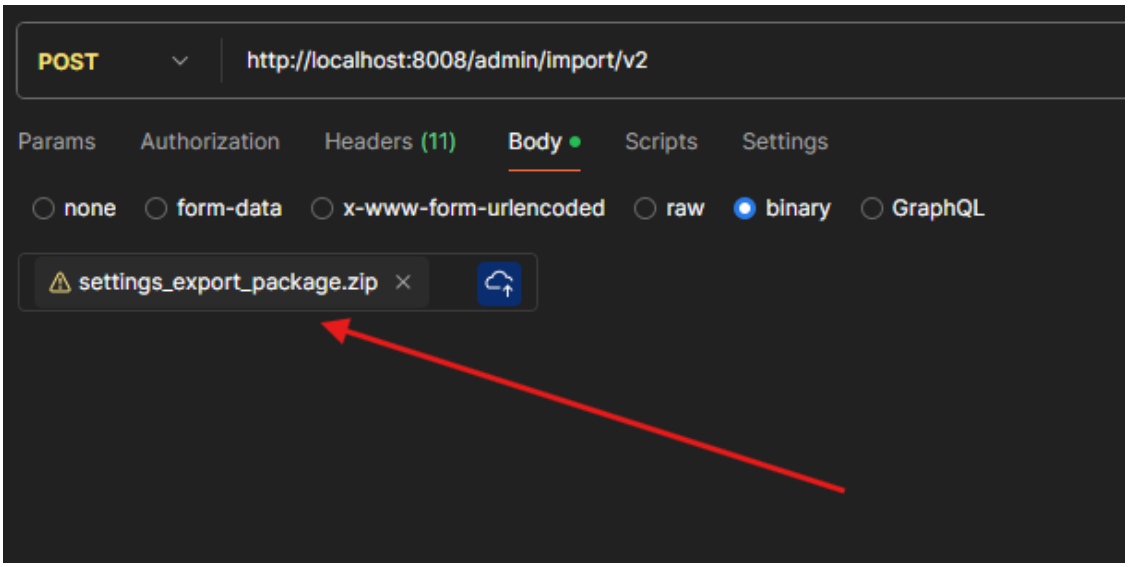
# === EXPORT CONFIGURATION ===
curl -X GET "$API_URL" \
     -H "apikey: $API_KEY" \
     -H "password: $PASSWORD" \
     -H "setting: $SETTING_JSON" \
     --output "$OUTPUT_FILE"

echo "Configuration export complete. Saved to $OUTPUT_FILE"
```

Import settings

- The prerequisites for this also required the login `session_id` to make the API call. For reference: POST - Import configuration





Bash scripts template:

```
#!/bin/bash

# === CONFIGURATION ===
API_URL="http://localhost:8008/admin/import/v2"
API_KEY="your_api_key_here"            # Use session_id from /login
PASSWORD="your_decryption_password"    # Password used during export
CONFIG_FILE="mdcore_config.zip"        # The ZIP file you exported
FORCE_UPDATE_YARA="false"               # Optional: true or false

# === IMPORT CONFIGURATION ===
curl --request POST "$API_URL" \
     --header "Content-Type: application/octet-stream" \
     --header "apikey: $API_KEY" \
     --header "password: $PASSWORD" \
     --header "import_force_update_yara: $FORCE_UPDATE_YARA" \
     --data-binary "@$CONFIG_FILE"

echo "Configuration import request sent."
```

Support:

If you have followed all of the steps above but cannot set the desired RAM Drive security permissions, please follow these instructions on How to Create Support Package With Bundle Tools? before creating a support case or chatting with our support engineer.

Why I can't push configuration due to password storage?

Check Your Version:

This article applies to versions OCM 7.x, Core 5.11 and above on both Windows and Linux platforms.

Introduction:

This knowledge base article provides details regarding Core Versions in OCM

Details:

If you have older versions of MD Core connected to OCM that do not have the Password Storage feature and you upgrade to a version that has the feature, you must remove the Core from OCM and re-add the core for the settings to take effect. Otherwise, you will receive the following error when you try to push the set settings:

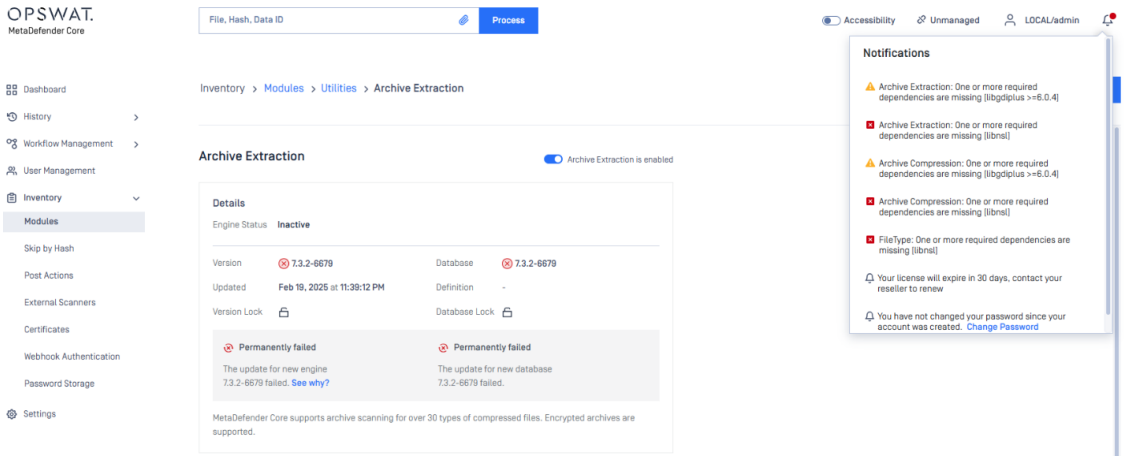
""err": "Invalid item for type: rule", "err_details": {"errors": [{"error":"Contains none-existing item","key":"archive.archive_handling.password_storage.storage","value":[1737987671157]}], "name": "Email Policy - Default"}"

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I install MetaDefender Core in Rocky Linux 9

When you first install MetaDefender Core on Rocky Linux 9 and successfully complete the installation, some engines may fail to install after activation due to missing dependency packages, as shown below:



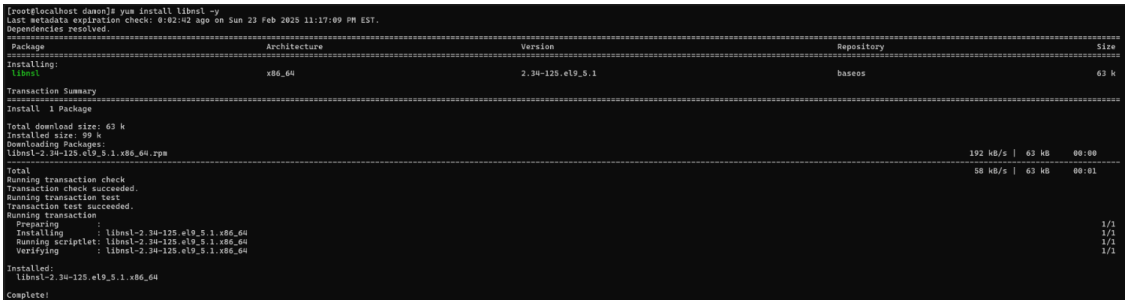
The screenshot shows that the Extract Archive Engine failed to install due to missing libraries: **libgdiplus** and **libnsl**. (these dependency packages are requested)

You can install them using the following command:

Code · bash

```
curl https://download.mono-project.com/repo/centos7-stable.repo | tee /etc/yum.repos.d/mono-centos7-stable.repo
yum install -y epel-release
yum install -y libgdiplus --nobest
yum install libnsl -y
```

Libnsl is also a dependency for Deep CDR and other engines. After installing the library, the output should match the screenshot below:

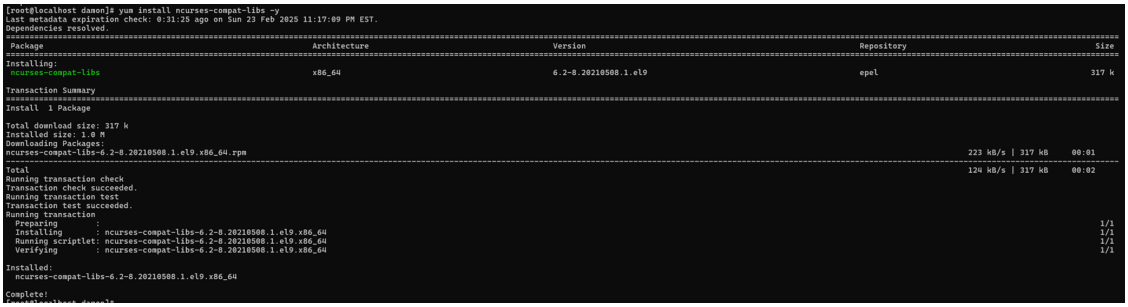


Besides that, the Deep CDR engine requires additional dependency packages (**libgomp**, **ncurses-compat-libs**), which can be installed using the command below:

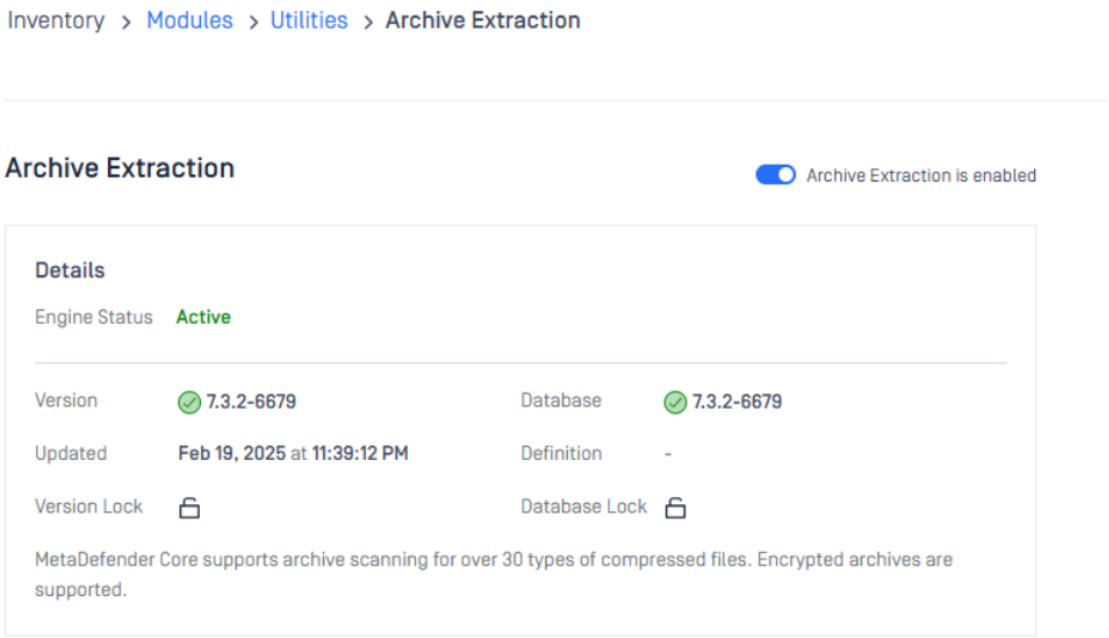
Code · bash

```
yum install libgomp -y
yum install ncurses-compat-libs -y
```

The output should be as the image below:



Next, you need to disable and then re-enable the Archive Engine to prompt MetaDefender Core to reinstall it. After doing this, the Archive Engine should install successfully.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to properly configure Syslog Server for MetaDefender Core?

Check Your Version:

This articles apply to all MetaDefender Core releases deployed on Windows and Linux.

Starting from **MetaDefender Core v5.13.0**, Syslog server configuration can be performed using REST API endpoints in addition to registry settings. While both methods are supported, **API-based configuration takes precedence**.

Problem

When configuring MetaDefender Core to send log history to a logging system or a SIEM via syslog, you may encounter issues where the logs are not being sent, and unknown errors are logged without capturing scan history.

Troubleshooting Steps

1. Using REST API (Core v5.13.0+)

If you are using MetaDefender Core v5.13.0 or later, it is recommended to configure syslog settings using the REST API. For more details on API usage, please refer to: POST - Forcefully apply the new syslog server configurations

2. Using Registry (All Versions)

If you're using an older version or prefer registry-based configuration, ensure the following keys are correctly set under: `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`.

- `syslog`: Your syslog server, in the format of `<protocol>://<syslog_server_address>:<port>`.
- `syslog_level`: Level of logging. Supported values are: `debug, info, warning, error`.

> **Info:**
>
> Syslog typically uses the UDP protocol on port 514. Please ensure that the protocol (tcp/udp) matches what your Syslog server is configured to receive.

> **Success:**
>
> **After making the changes to the registry values, make sure to restart the service of MetaDefender Core, to apply the new configuration.**

Additionally, you can confirm that the MetaDefender Core server can reach the Syslog server over the specified port and protocol. You should ensure that no firewall or network ACL is blocking outbound traffic from MetaDefender Core.

> **Support:**
>
> If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why MetaDefender Core's nginx ssl.conf changed to ssl.mdcore.conf?

> **Info:**
>
> This article applies to all MetaDefender Core versions starting with 5.13.

Starting with version 1.24.0 of Nginx, the ssl directive has been deprecated and can no longer be used for configuring SSL. To address this, MetaDefender Core updated the naming convention of **ssl.conf to ssl.mdcore.conf**.

Implications and limitations

The implications of this are that the **ssl, ssl_certificate** and **ssl_certificate_key** directives will be placed in the **ssl.mdcore.conf**. The current limitation is that the file does not support the following directives: **ssl_client_certificate, ssl_verify_client, ssl_verify_depth**.

In this scenario, the configuration can be achieved as follows:

ssl.conf.mdcore file:

`ssl`

`ssl_certificate`

`ssl_certificate_key`

ssl.conf file:

`ssl_client_certificate`

`ssl_verify_client`

`ssl_verify_depth`

Info:

Starting with MetaDefender Core version 5.14.0, advanced configuration can be achieved directly via the User Interface, without the need of having an individual config file: Enabling HTTPS

This way the configurations are stored internally and not being exposed to an external file.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to configure SMTP server with authentication under MetaDefender Core to send notification to M365?

Solution

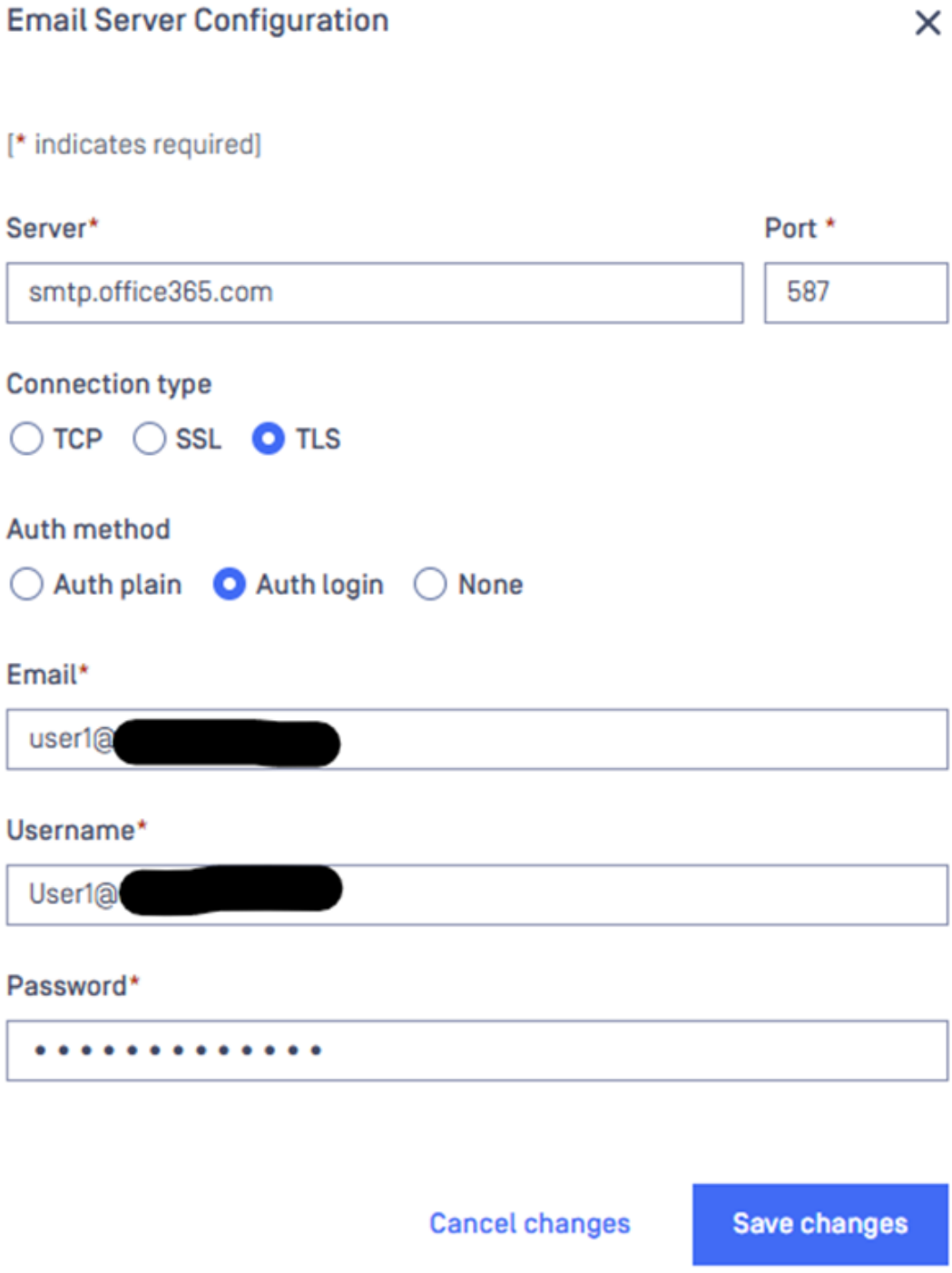
To be able to send notification to your M365 tenant from MetaDefender Core, the following fields should be configured properly under MetaDefender Core:

- Server: smtp.office365.com
- Port: 587

- Connection type: TLS
- Auth Method: Auth Login
- Email: Email address for sender of notification, it should be a real mailbox which is licensed on M365 side.
- Username: username of the mailbox provided under “email” field.
- Password: password of the mailbox provided under “Email” field.

More details about M365 URLs can be found checking following link: Microsoft 365 URLs and IP address ranges - Microsoft 365 Enterprise | Microsoft Learn

Example:



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to manage Disk Space Usage in pg_data\base directory in MetaDefender Core for Windows?

Issue

The directory located at OPSWAT\MetaDefender Core\data\pg_data\base continues to grow in size until the disk becomes full, eventually causing scans to fail or stop running altogether.

Cause

This folder stores critical components such as:

- Scanning history
- Executive report
- Other related data used by MetaDefender Core

Without proper data retention policies in place, this data accumulates indefinitely, consuming excessive disk space.

Solution

To manage and reduce the size of this directory, configure **automatic deletion** through the **Data Retention** settings in MetaDefender Core. **Steps to Configure Automatic Deletion:**

1. Navigate to the **Settings** page in the MetaDefender Core UI.
2. Go to the **Data Retention** section.
3. Set a retention period—**we recommend 1 to 4 weeks**, depending on your organizational needs and storage capacity.
4. For Deep CDR and DLP, these are recommended to remain set to the 4 h default.

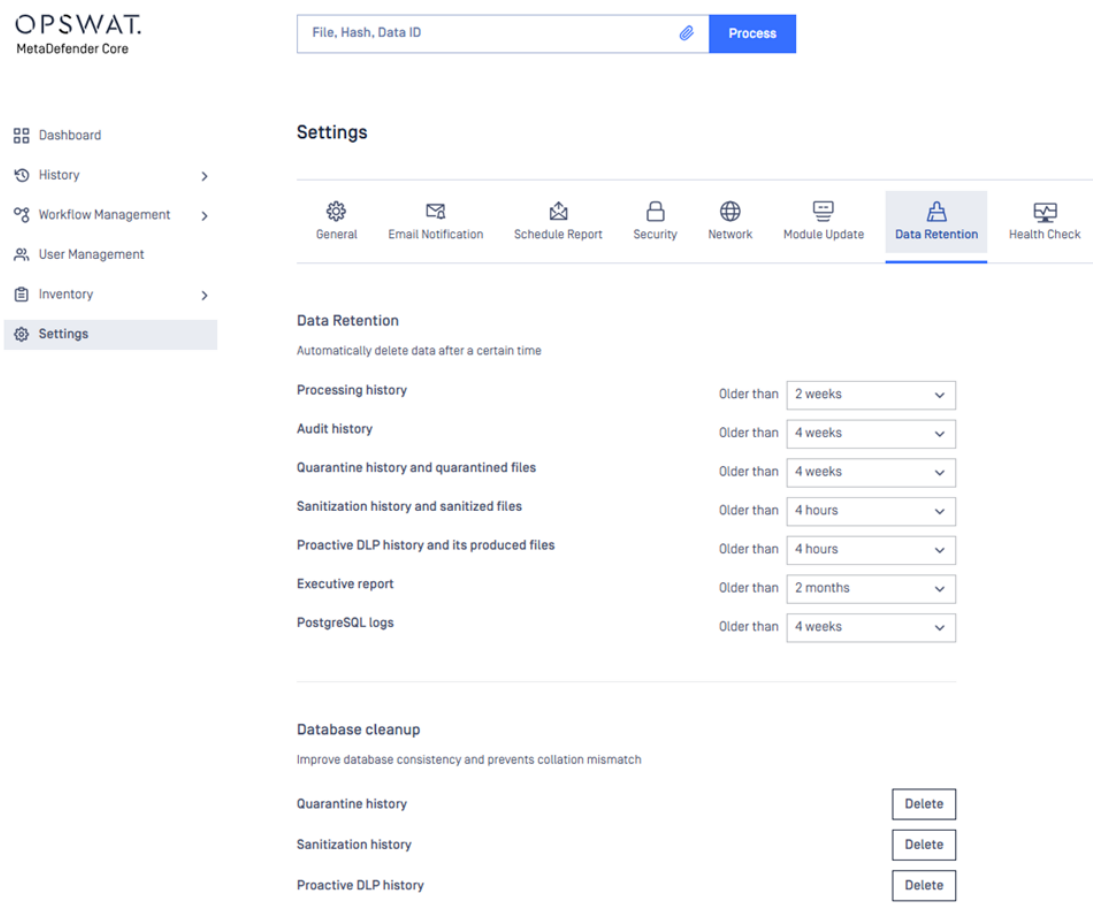
Optional: Immediately Free Up Space

To quickly recover disk space after updating the settings:

- Use the **Delete** button at the bottom of the **Data Retention** settings page to remove existing data that falls outside the new retention period.
- Use the database vacuum tool (instructions can be found here Database Maintenance | MetaDefender Core 4.19.0 or above)

Note:

Please consider that even after data retention changes, it will take some time for the storage to be cleaned (usually up to 24 h, due to the internal database vacuum process).



Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

What storage type should I use on MetaDefender Core in K8S?

Introduction:

This knowledge base article provides details on what types of storages to use in K8S

Details:

If the Database is remote and there is no incentive to keep permanent data like sanitized files with CDR or DLP, an ephemeral storage can be used. This can be done from the values.yaml

md-core: env: resources: ephemeral-storage: "60Gi"

If you need persistence, in that case the persistence storage would be the best option. So, make sure that you comment the storage_configs from null and use the following code from values.yaml, comment out line 61, and uncomment lines from 62 to 74

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What permissions are needed for the user of the storage of Google Cloud?

Introduction:

This knowledge base article provides information on what roles the user that maps the Google Cloud Storage should be.

Permissions Overview

For MDSS to operate correctly when integrating with Google Cloud Storage, the user account must have sufficient privileges. The documentation initially recommends that the user be granted the Owner role. This role provides comprehensive permissions across the Google Cloud project, including the ability to:

- Create, modify, and delete buckets and objects
- Configure access controls and security settings
- Manage project resources and billing

Important Note: While granting the Owner role ensures all necessary permissions are available, it is generally best practice to adhere to the principle of least privilege. In cases where only storage operations are required, consider using more specific roles such as:

- Storage Admin (roles/storage.admin): Grants full control over Cloud Storage resources without broader project permissions.
- Storage Object Admin (roles/storage.objectAdmin): Provides full control over objects in the storage buckets while limiting administrative capabilities on bucket settings.

Select the role based on the required functionality and security guidelines of your organization.

Detailed Configuration Steps

1. Evaluating Requirements

- Assess MDSS Needs: Determine if MDSS requires full project management capabilities or just access to storage resources. For most scenarios where the sole purpose is mapping storage, the Storage Admin role may suffice.
- Review Security Policies: Consult your organization's IT and security policies to ensure compliance with role assignments and the principle of least privilege.

2. Assigning Permissions via the Google Cloud Console

- Log into the Google Cloud Console: Access your Google Cloud project.
- Navigate to IAM & Admin: Select “IAM & Admin” from the navigation menu.
- Locate the User Account: Find the user account that MDSS will use for mapping the storage.
- Assign the Appropriate Role:
  - For comprehensive permissions, assign the Owner role.
  - Alternatively, assign Storage Admin or Storage Object Admin if full project permissions are not required.

- Save Changes: Confirm and save the changes to update the permissions.

3. Verifying Permissions

- Test Storage Access: Use MDSS to perform operations such as creating, reading, updating, and deleting objects to verify that the assigned role meets the required functionality.
- Audit Logs: Regularly review the Cloud Audit Logs to ensure no unintended or unauthorized actions are performed.

Additional Considerations

- Security Best Practices: Always follow the principle of least privilege. Grant only the permissions that are necessary for the MDSS operations to minimize security risks.
- Periodic Review: Regularly review IAM roles and permissions in your Google Cloud project. Reassess whether the granted permissions still match the operational requirements of MDSS.
- Role Customization: If none of the predefined roles align with your needs, consider creating a custom role that aggregates only the required permissions for storage mapping. This approach can offer a more tailored security posture while meeting functional requirements.

Conclusion

Mapping Google Cloud Storage for MDSS requires careful consideration of the permissions assigned to the user account. While the initial recommendation is to use the Owner role, assessing the operational needs and following best security practices might suggest using more limited roles such as Storage Admin. By following the configuration steps and periodic reviews outlined in this article, you can ensure both functionality and security in your MDSS deployment.

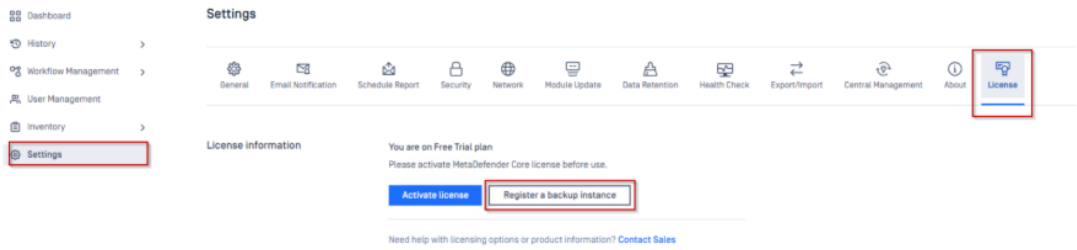
Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

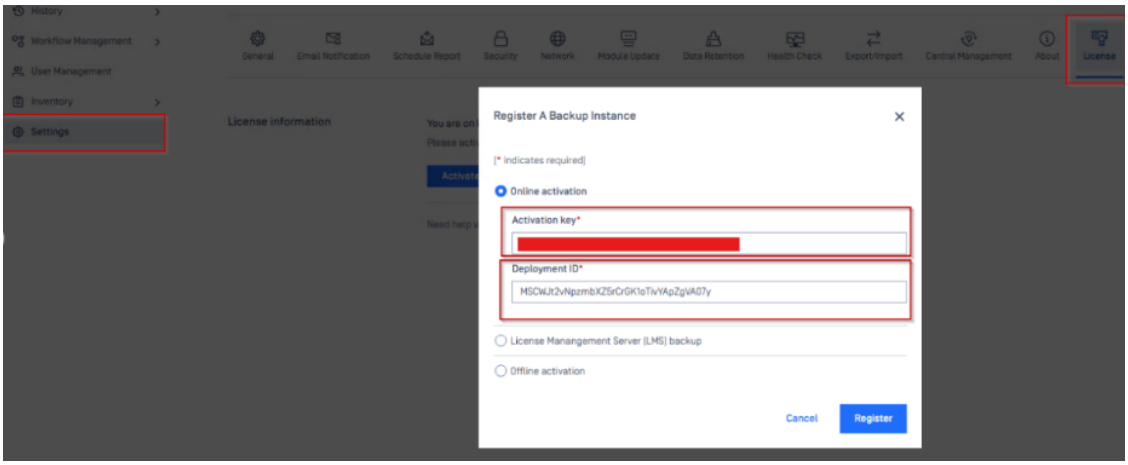
How do I setup Hot Swap Backup Configuration between MetaDefender Core instances?

Online procedure

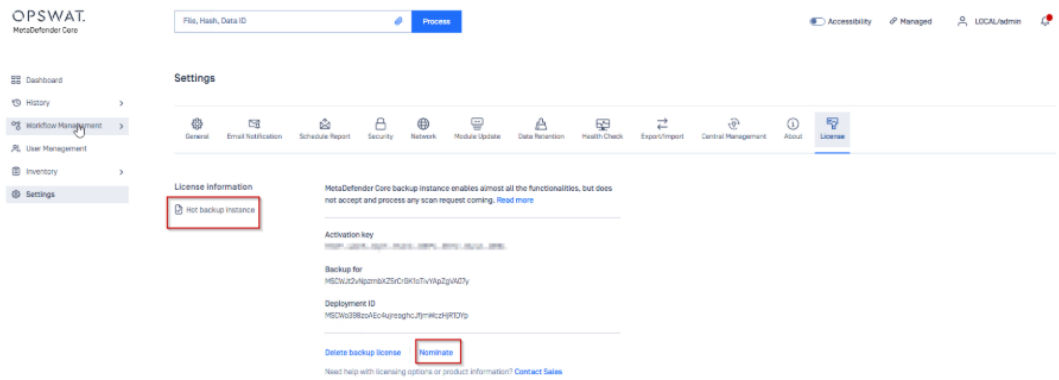
- Make sure the primary MetaDefender Core deployment is activated
- Gather the deployment ID and Activation key from the primary MetaDefender Core deployment
- Navigate to the secondary MetaDefender Core instance and take the following steps
  - Select the Register as back-up instance option on the back-up MetaDefender Core machine



- - Add the necessary variables from the primary MetaDefender Core deployment and “register” (Primary Core deployment ID and Activation key)



Once the steps are completed, we can observe the successful result on the back-up MetaDefender Core console.



Api call (online environment)

In order to successfully set up hot swap backup configuration between 2 or more MetaDefender Core deployments via API calls the following steps need to be followed:

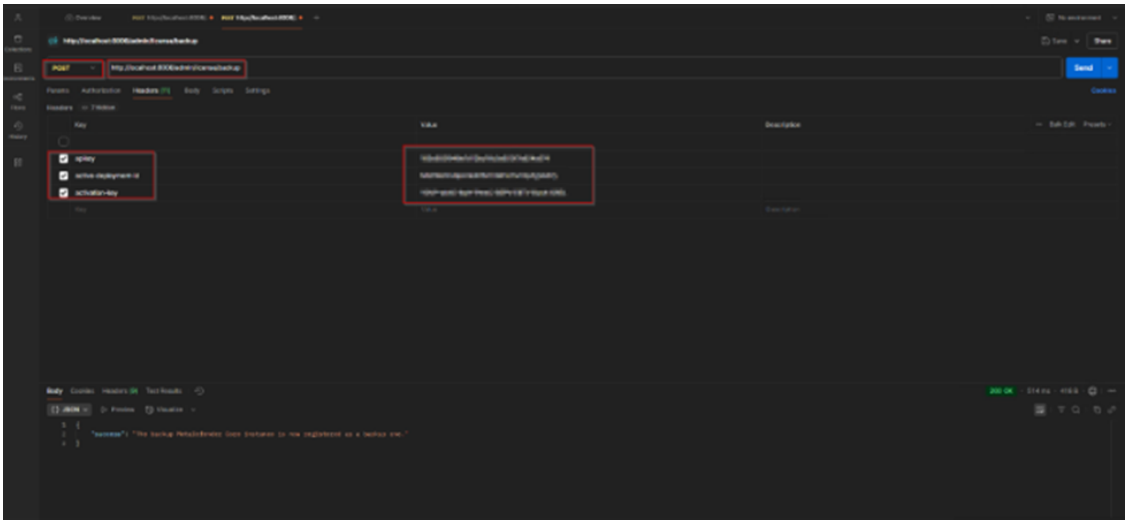
- Make sure the primary MetaDefender Core deployment is activated
- Gather the deployment ID and Activation key from the primary MetaDefender Core deployment
- Use the API call listed below with any preferred solution.

Code · bash

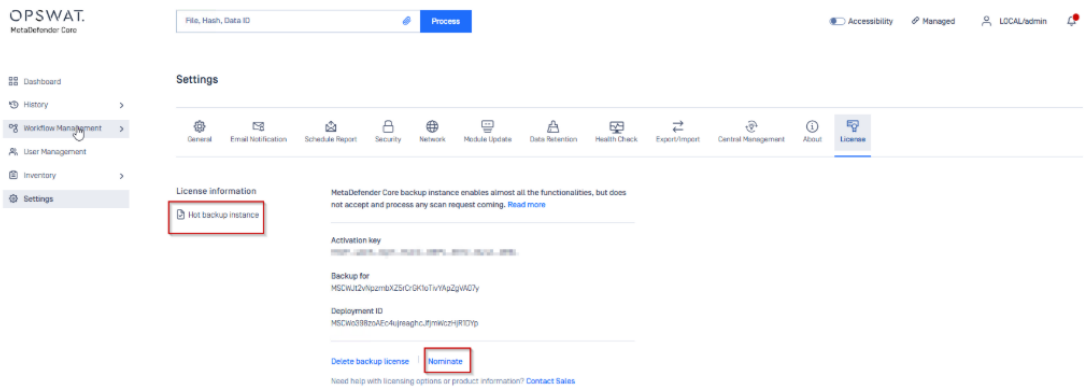
```
curl --request POST \ --url '__http://localhost:8008/admin/license/backup' \ --header 'Content-Type: application/octet-stream' \ --header 'apikey: {apikey}' \ --header 'active-deployment-id: {active-deployment-id}' \ --header 'activation-key: {activation-key}' \ --data-binary '{ "schema": "{string}" }'
```

A Postman example is highlighted below. Make sure the following information is available before the API call is send:

- API Key from the backup MetaDefender Core server
- Activation key
- Primary MD Core instance deployment ID



Once the steps are completed, we can observe the successful result on the back-up MetaDefender Core console.



Api call (offline environment)

In order to successfully set up hot swap backup configuration between 2 or more MetaDefender Core deployments via API calls the following steps need to be followed:

- Make sure the primary MetaDefender Core deployment is activated
- Gather the deployment ID, Activation key, and the offline yml activation file from the primary MetaDefender Core deployment

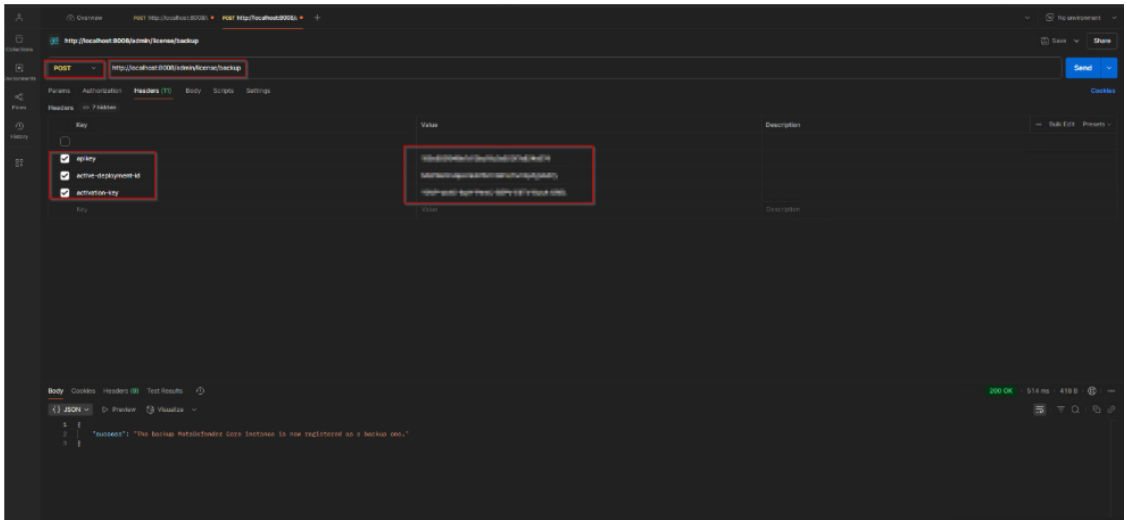
- Use the API call listed below with any preferred solution.

Code · bash

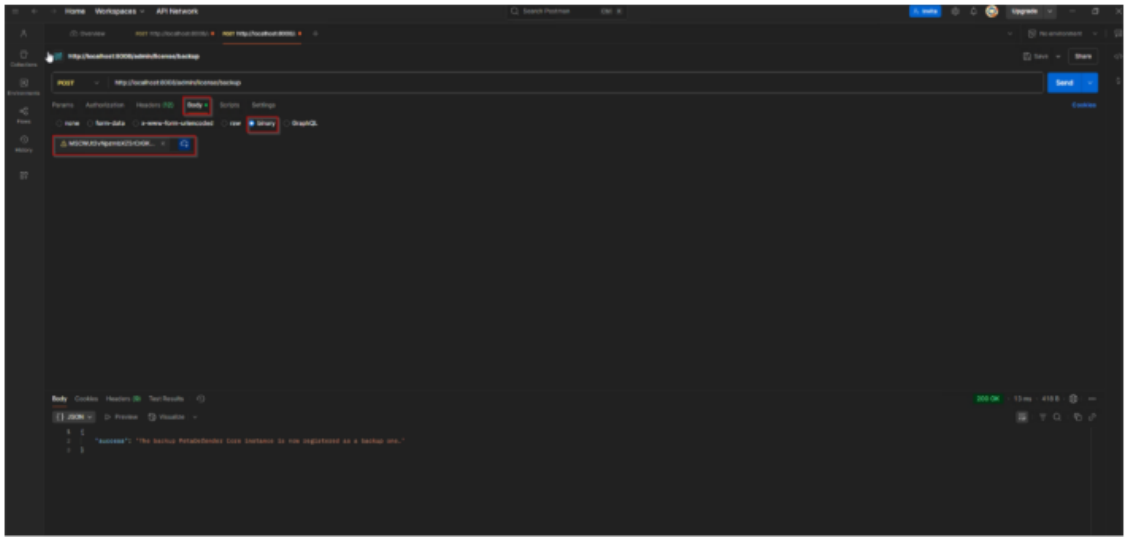
```
curl --request POST \ --url '__http://localhost:8008/admin/license/backup' \ --header 'Content-Type: application/octet-stream' \ --header 'apikey: {apikey}' \ --header 'active-deployment-id: {active-deployment-id}' \ --header 'activation-key: {activation-key}' \ --data-binary '{ "schema": "{string}" }'
```

A Postman example is highlighted below. Make sure the following information is available before the API call is send:

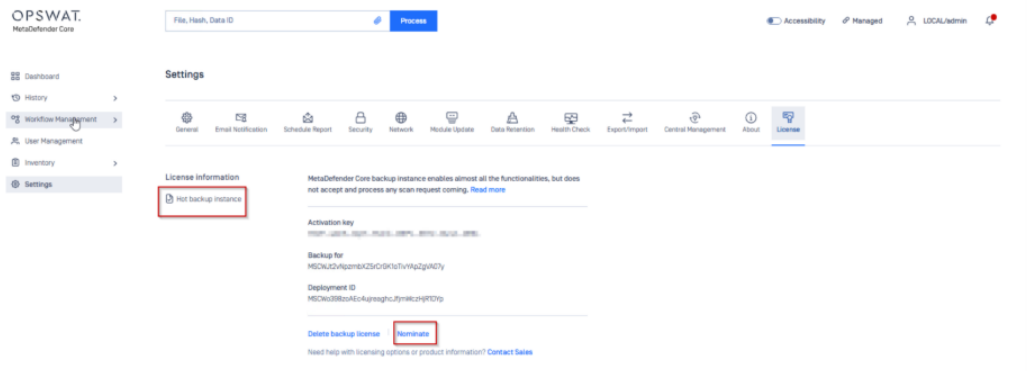
- API Key from the backup MetaDefender Core server
- Activation key
- Primary MetaDefender Core instance deployment ID
- .yml activation file from the Primary MetaDefender Core instance



- Use that yml file content in body content of the REST API



Once the steps are completed, we can observe the successful result on the back-up MetaDefender Core console.



Support:

If you encounter any issues during the activation process or require further assistance, please contact OPSWAT support via My OPSWAT at: https://my.opswat.com/home

Our support team will be able to provide you with the necessary guidance and resolution for any license-related concerns.

Where are MetaDefender Core Install Directories and Advanced Configurations?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems.

Windows

By default, the following are the directories used when MetaDefender Core is installed:

Directory	Content
C:\Program Files\OPSWAT\MetaDefender Core	Main installation folder where tools and support package script are found.
C:\Program Files\OPSWAT\MetaDefender Core ginx	Web management files and logs files for the ability to access MetaDefender core on the browser.
C:\Program Files\OPSWAT\MetaDefender Core\data\resources	Engine folders that store temporary files during scanning, files are deleted once the scan is complete.
C:\Program Files\OPSWAT\MetaDefender Core\data\updates\av C:\Program Files\OPSWAT\MetaDefender Core\data\updates\db	Folders [av] [db] containing the engines database update packages [zip] and instruction YML file.
C:\Program Files\OPSWAT\MetaDefender Core\data\logs	Folder containing rotating core logs and upgrade log.
C:\Program Files\OPSWAT\MetaDefender Core\data\engines	Folder containing loaded engines and their necessary components to complete scanning.
C:\Program Files\OPSWAT\MetaDefender Core\data\enginelogs	Logs for each engine produced during scans.
C:\Program Files\OPSWAT\MetaDefender Core\data\pg_data	Postgres Database files and logs (only for local DB).
C:\Program Files\OPSWAT\MetaDefender Core\postgres	Libraires and executables for the local database.

Additional configuration located in this Registry path:

Code · html

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan
```

Linux

By default, the following are the directories used when MetaDefender Core is installed:

Directory	Directory

Directory	Directory
/var/lib/ometascan	Main installation folder where tools and support package script are found.
/etc/ometascan /etc/ometascan/nginx.d	Folder containing configuration files for Core and web server.
/var/run/ometascan	Folder containing runtime variable files for Core services.
/var/tmp/ometascan/resources	Engine folders that store temporary files during scanning, files are deleted once the scan is complete.
/var/lib/ometascan/updates/av /var/lib/ometascan/updates/db	Folders [av] [db] containing the engines database update packages [zip] and instruction .YML file.
/var/log/ometascan	Folder containing rotating core logs, upgrade log, and web server logs.
/var/lib/ometascan/engines	Folder containing loaded engines and their necessary components to complete scanning.
/var/log/ometascan/enginelogs	Logs for each engine produced during scans.
/var/lib/ometascan/pg_data	Postgres Database files and logs (only for local DB).
/usr/lib/ometascan/postgres/bin /usr/lib/ometascan/postgres/lib	Libraires and executables for the local database.

Additional configuration is located in this file:

Code · html

```
/etc/ometascan/ometascan.conf
```

Support:

If you have followed through this article but are still unable to locate **MetaDefender Core Install Directories and Advanced Configurations**, please follow these instructions on How to Create Support Package With Bundle Tools?, before **creating a support case or chatting with our support engineer**.

How to retrieve the CRL and OCSP URLs for the Server and ICA Certificates?

Purpose

This document explains how to retrieve the **CRL (Certificate Revocation List)** and **OCSP (Online Certificate Status Protocol)** URLs from a server's SSL/TLS certificate and its Intermediate Certificate Authority (ICA) certificate, using a web browser. These URLs are often required for configuring firewall rules to allow outbound certificate validation.

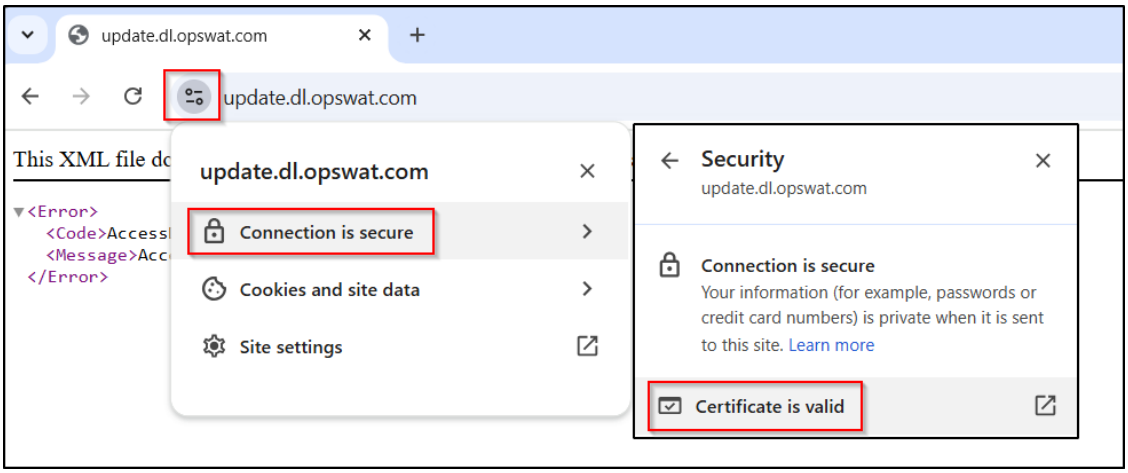
Steps for Chrome or Edge (similar steps for Firefox)

1. Open the Website

Go to the website in question using Chrome or Edge (e.g., https://activation.dl.opswat.com, https://update.dl.opswat.com).

2. View the Certificate

- Click the **site information icon** in the address bar.
- Click **“Connection is secure”** or **“Certificate is valid”**.
- A **certificate window** will appear.



3. Examine the Certificate Chain

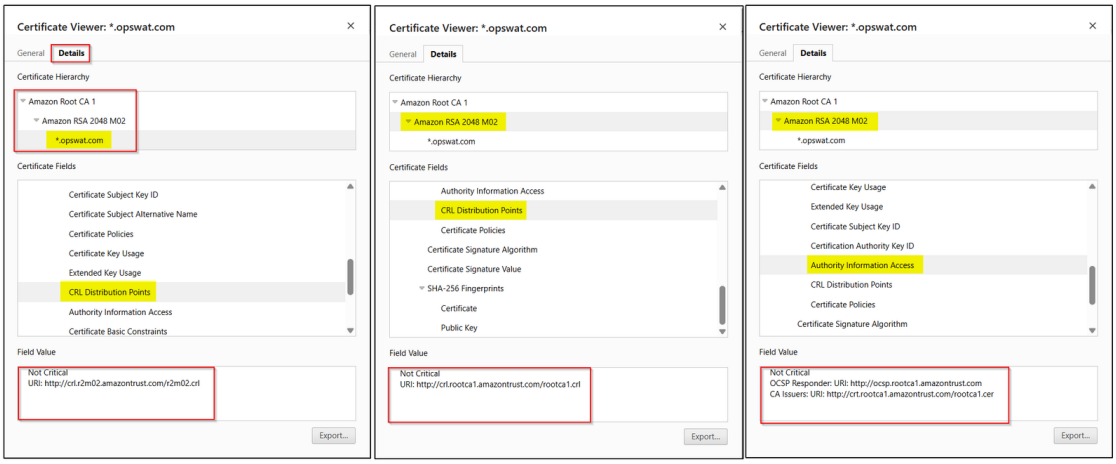
- In the certificate window, go to the "**Details**" tab (or similar, depending on the browser).
- Select each certificate in the chain one by one (server, intermediate CA) to see the details.

4. Find CRL and OCSP URLs

For each certificate, search for the following fields::

- **CRL Distribution Points**

- **Authority Information Access**



These URLs are used by clients to check if a certificate has been revoked.

---

5. Compile URLs for Firewall Whitelisting

For each certificate in the chain, list all:

- CRL URLs
- OCSP URLs

Example Whitelist Entries:

- `http://crl.example-ca.com/example-ca.crl`
- `http://ocsp.example-ca.com`

Make sure these are accessible from inside your firewall, typically over **port 80 (HTTP)**.

Notes

- These URLs are not always present; absence means revocation may not be supported.
- Most revocation services (CRL/OCSP) are still served over **HTTP**, not HTTPS.
- If OCSP stapling is enabled on the server, external OCSP queries may be reduced—but firewall whitelisting is still recommended.

Troubleshooting

- If CRL/OCSP URLs are unreachable, try accessing them directly in the browser to test connectivity.
- Use browser developer tools (F12) and Network tab to watch OCSP requests in real time, if needed.

Support:

If **Further Assistance** is required regarding this topic, please proceed to log a **support case or chat with a support engineer**.

Is Import Password Mandatory When Using Ignition File for MetaDefender Core installation?

Check Your Version:

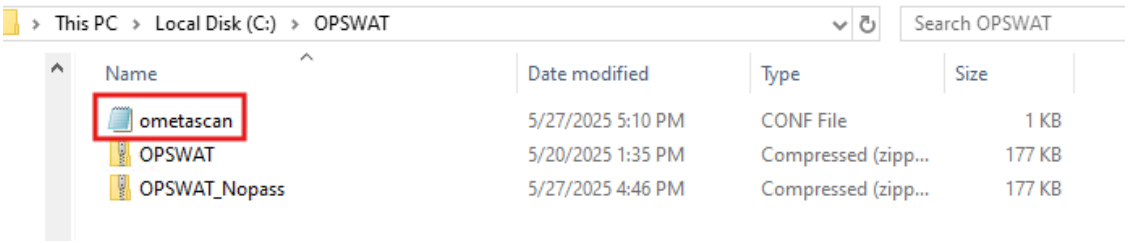
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Summary

When installing MetaDefender Core, the system uses an ignition file to apply the initial configuration. This article explains whether including an import password is mandatory when the configuration package is referenced within the ignition file.

Here we have an example of a configuration file with OPSWAT.zip as in password file, and OPSWAT_Nopass will be the one with the password.

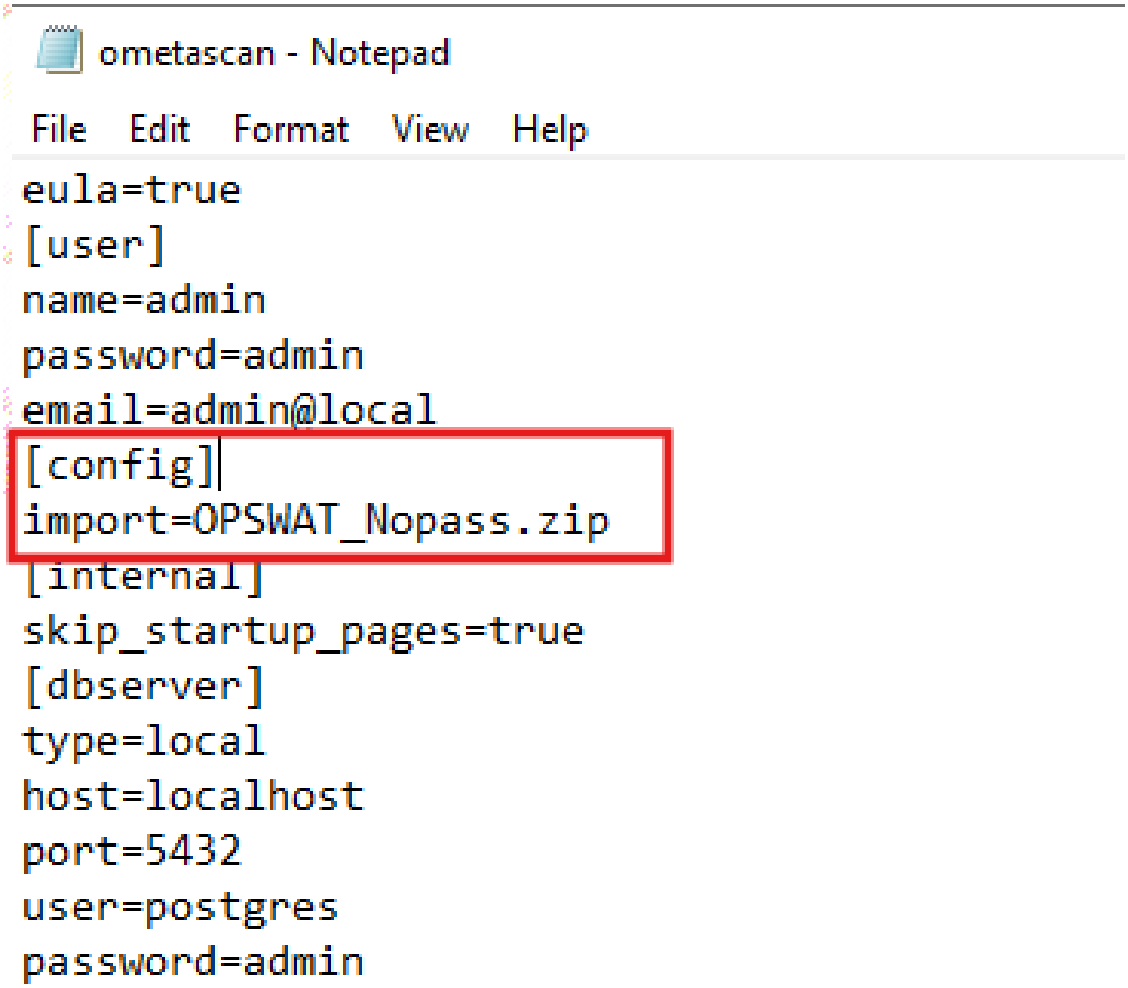
Example:

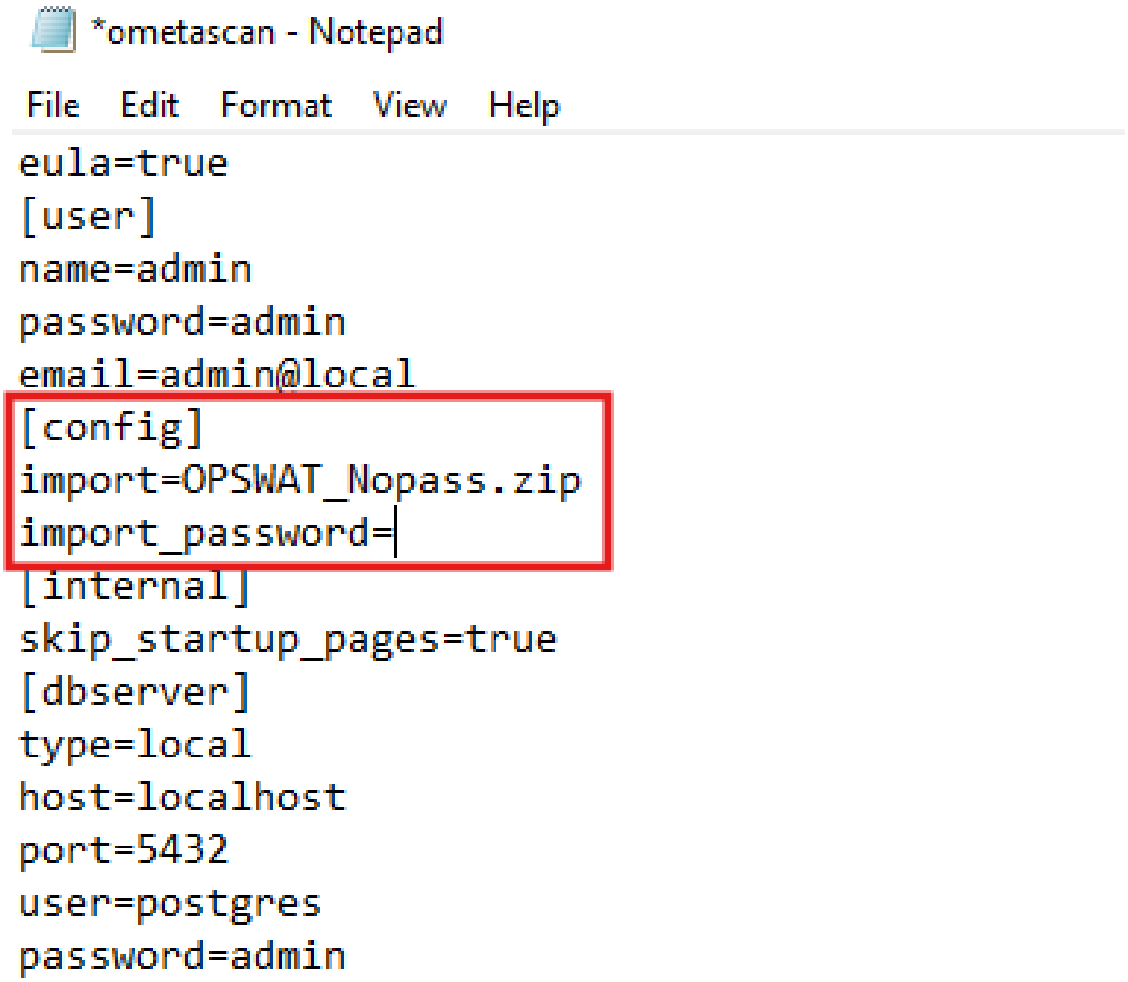


Here we have an example of two configuration files:

- `**OPSWAT.zip**`: This configuration file **requires an import password**.
- `**OPSWAT_Nopass.zip**`: This configuration file **does not have a password**.

Scenario 1: No password configuration file. (OPSWAT_Nopass.zip)


ometascan - Notepad
File Edit Format View Help
eula=true
[user]
name=admin
password=admin
email=admin@local
[config]
import=OPSWAT_Nopass.zip
[internal]
skip_startup_pages=true
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=admin



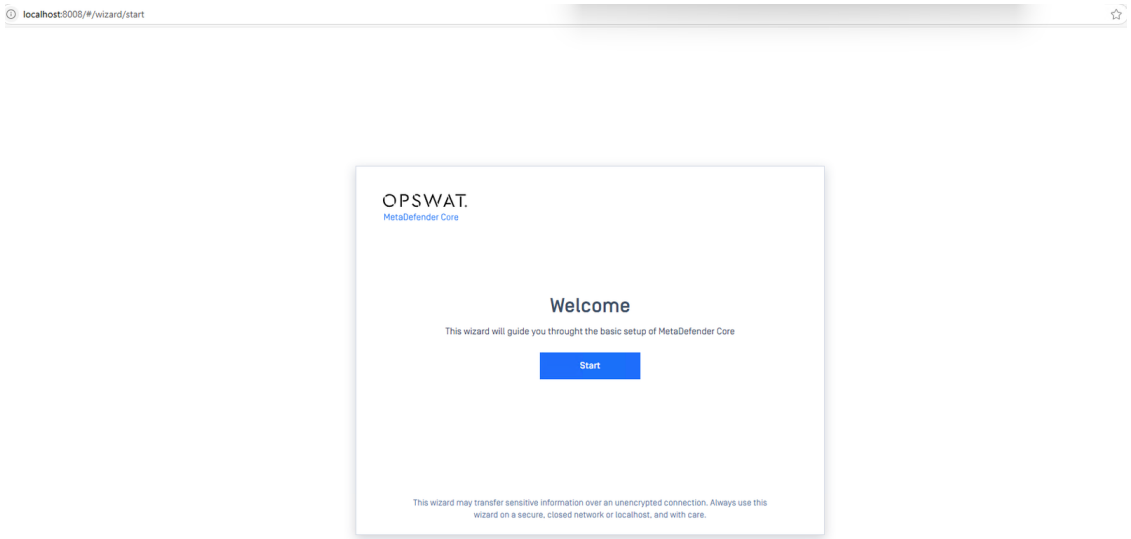
When importing the configuration file above, MetaDefender Core reports it as an invalid format.

Code · html

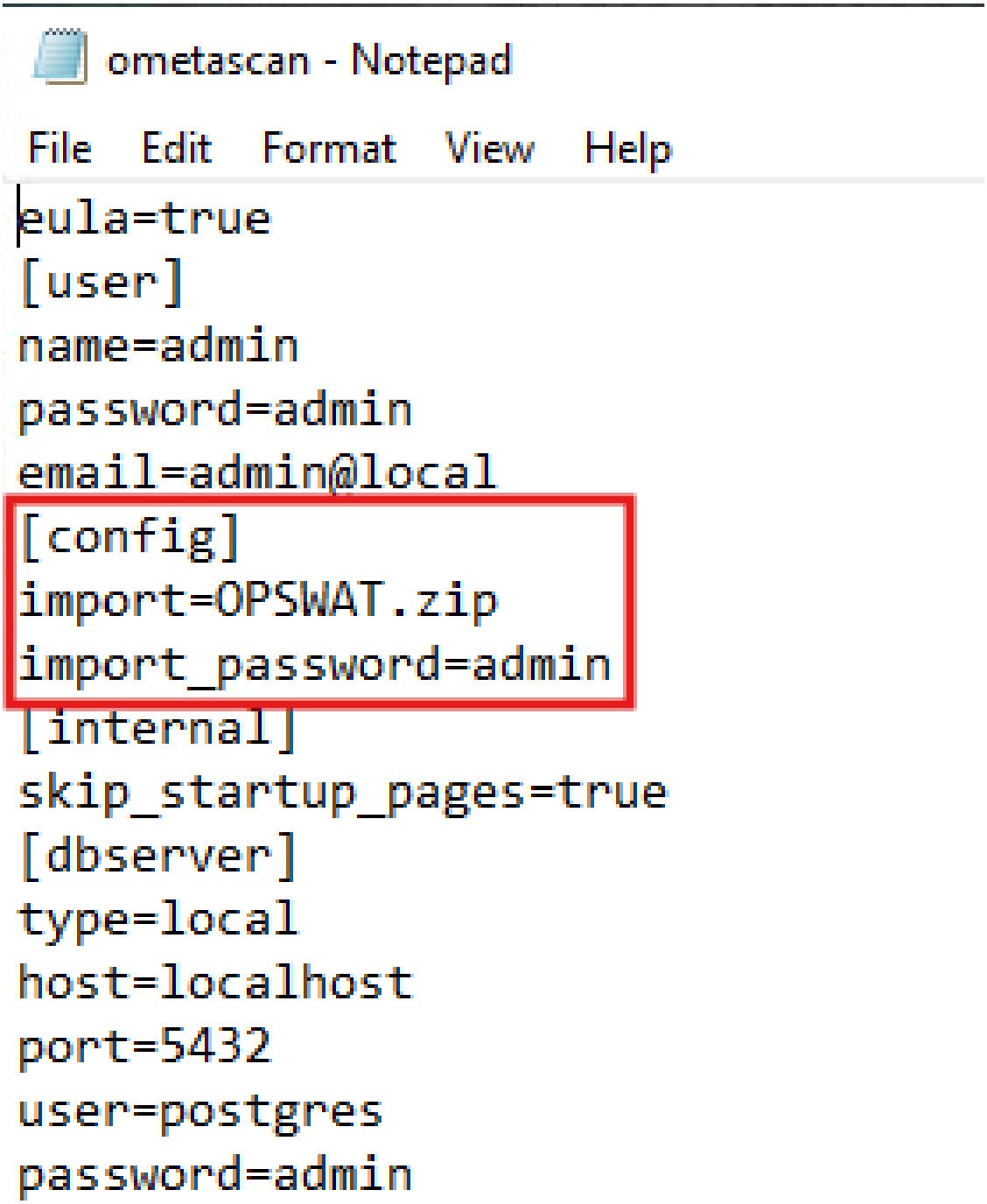
```
Failed to initialize the product with config file, reason='import file format invalid, C:/
opswat/OPSWAT_Nopass.zip'
```

Error:

One of the other ways to confirm if the ignition file fails is to access MetaDefender Core, is to see if the login screen appears or starts with the wizards to set up the account.



Scenario 2: Password-protected configuration file. (OPSWAT.zip)

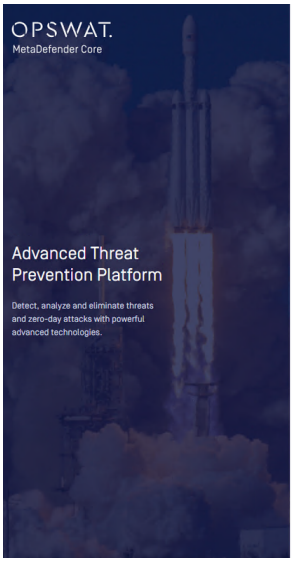


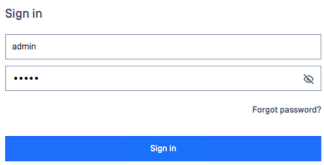
Once the ignition file is confirmed when installing MetaDefender Core, the configuration file is in the right format.

Code · html

```
Ignition: Config package has been imported!, schemaImportPath='C:/opswat/OPSWAT.zip'
```

Since the import was a success, we can access the login page to proceed.





Support:

If you have followed all of the steps above but are still unable to resolve the issue. Please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How does the recursion level work in MetaDefender Core?

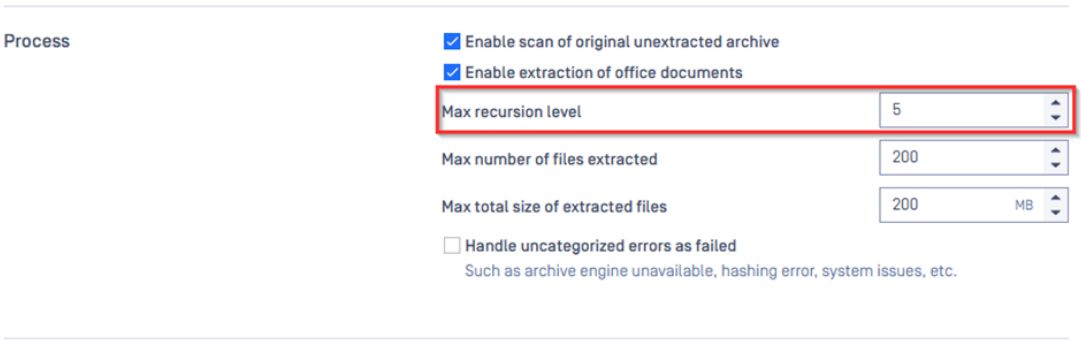
Check Your Version:

This article applies to all MetaDefender Core version 5.0 and newer releases deployed on Windows and Linux systems.

Understanding Archive Recursion Level in MetaDefender Core

 Overview

The recursion level can be configured in the MetaDefender Core workflow, in the Archive tab.



When MetaDefender Core scans files, it doesn't just check the top-level file — it also scans inside archives (like ZIP, RAR, DOCX files). Some archives can contain other archives or files inside them. The **archive recursion level** setting

controls **how many layers deep** MetaDefender Core will go to extract and scan files.

This article explains how it works.

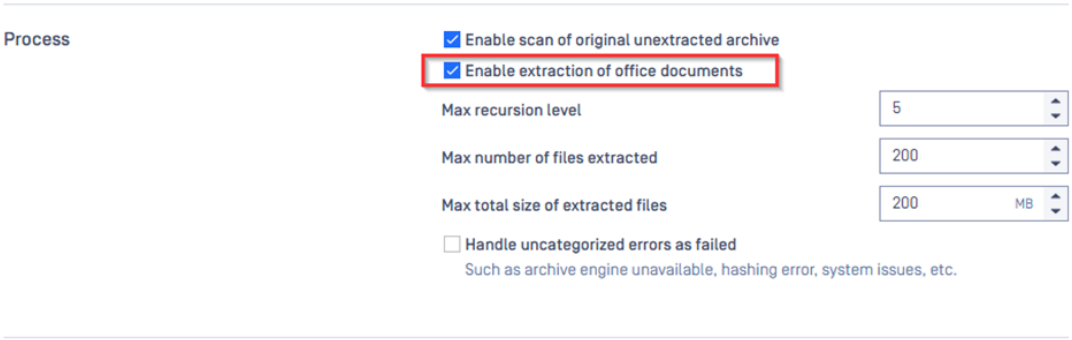
What Is Archive Recursion Level?

Archive Recursion Level = How deep MetaDefender Core will **unpack nested archives** during a scan.

- Archives: .zip, .rar, .7z, .tar, .gz, .docx, .xlsx, .pptx, .eml, etc.
- Regular files: .txt, .jpg, .pdf, .exe, etc. — **do not affect recursion level**
- Folders: Directory structures inside archives **do not affect recursion level**

! Important note:

MetaDefender Core extracts the office documents by default due to the option “Enable extraction of Office Documents”. If this option is disabled, the office documents will not be extracted.



Recursion Level Breakdown

Recursion Level Description	
Level 0	The top-level archive being scanned
Level 1	Archives inside the first archive
Level 2	Archives inside level 1 archives
Level 3	Archives inside level 2 archives
...	and so on...

Detailed Example for Clarity

Let's say we scan the following file:

```
pgsql
CopyEdit
MainArchive.zip                → Level 0
├── ReadMe.txt                 → Regular file (no impact)
├── FolderA/                   → Folder (no impact)
│   ├── Invoice.pdf            → Regular file (no impact)
│   └── Image.jpg              → Regular file (no impact)
├── FolderB/
│   └── Level1.zip             → Level 1
│       ├── Document.docx      → Level 2 (treated as archive)
│       │   └── word/document.xml  → Contents inside docx (scanned)
│       └── Level2.zip         → Level 2
│           ├── Video.mp4      → Regular file (no impact)
│           └── Level3.zip     → Level 3
│               ├── FolderX/   → Folder (no impact)
│               │   └── Note.txt  → Regular file (scanned)
│               └── Level4.zip  → Level 4
│                   └── secret.xlsm  → Level 5 (treated as archive)
│                       └── workbook.xml
```

 Recursion Level Impact Summary

File/Folder	Affects Recursion?	Recursion Level
MainArchive.zip	✅ Yes	0
ReadMe.txt	❌ No	0
FolderA	❌ No	0
Level1.zip	✅ Yes	1
Document.docx	✅ Yes	2
Level2.zip	✅ Yes	2
Level3.zip	✅ Yes	3
Level4.zip	✅ Yes	4
secret.xlsm	✅ Yes	5
Note.txt, Video.mp4	❌ No	scanned but do not increase depth

⚠️ **Important Notes**

- Recursion is about **nested archives only**.
- Regular files and folders inside archives are **scanned**, but do **not increase** the recursion level.
- If the recursion level limit is reached (e.g., set to 3), any deeper archives (like Level4.zip in this example) will **not be extracted or scanned**.

Configuration Tip

You can set the maximum archive recursion level in the MetaDefender Core UI or configuration file:

- **Default**: 5
- **Recommended**:
- **10-30** for high-security needs
- **5-10** for environments needing better performance

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

How to Enable Debug Mode in a MetaDefender Core Kubernetes Deployment?

Check Your Version:

This article is applied to MetaDefender Core releases deployed on Linux systems in a Kubernetes container environment.

Overview

This article explains how to enable debug mode in a MetaDefender Core deployment running in a Kubernetes

environment. Enabling debug mode increases the verbosity of log output, which can be helpful for troubleshooting and diagnostics. This setting should only be used when actively investigating issues, as it can produce large volumes of logs.

Steps to Configure Debug Mode

1. **Add the Debug Logging Environment Variable**: Modify your Kubernetes deployment to include the following environment variable in the container specification:

CORE_CONF_JSON: {"logger/loglevel":"debug"}

This variable instructs MetaDefender Core to set the logger level to 'debug' by injecting a JSON configuration at container startup.

Example Snippet in Deployment YAML:

Insert the following under the `env`: section of your container in the deployment manifest:

Code · yaml

```yaml
env:
  CORE_CONF_JSON: '{"logger/loglevel":"debug"}'
```

In an AWS Kubernetes deployment the variable can be set as below:



2. **Apply the Updated Deployment:**

After saving your modified deployment YAML, apply it using the following command:

`kubectl apply -f <your-deployment-file>.yaml`

This will restart the pod with the updated environment variable. If the variable was added in the POD creation, there is no need for an apply as shown above.

3. **Verify Debug Logging:**

Once the pod is running, you can check if debug mode is enabled by inspecting the logs:

`kubectl logs <pod-name> -n <namespace>`

Look for log entries that contain [`DEBUG`]. These confirm that the log level has been successfully elevated to debug.

4. **Revert to Standard Logging:**

Once debugging is complete, it is recommended to revert the log level to 'info' by changing the environment variable value to:

`CORE_CONF_JSON: {"logger/loglevel":"info"}`

Troubleshooting Notes

- Use debug mode temporarily, as it can increase log volume significantly. - Always monitor disk usage if debug logging is left enabled for extended periods.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

How to Select a Specific MetaDefender Core Instance in a PostgreSQL Database with Multiple Instances?

Check Your Version:

This article is applied to MetaDefender Core releases deployed on Windows and Linux systems.

Issue:

When connecting MetaDefender Core to a PostgreSQL database that contains multiple MetaDefender Core instances, there is currently no option within the Core installation interface to select a specific instance. In a Shared Database configuration, MetaDefender Core will connect to and use the first available MetaDefender Core shared instance found within the database. The selection is made automatically by the application and cannot be overridden.

Resolution:

If it is mandatory to use a database with multiple instances, use the Docker or Kubernetes environments, where this option is available. For example, in a docker deployment, the environmental variable `MDCore_DB_NAME` can be used to point to a specific database instance.

Support:

Support: If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Can you import configuration settings from the latest version to the previous version of Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows/Linux systems.

Summary

Importing configuration settings from a newer version (e.g., **5.14.2**) of MetaDefender Core into a previous version

(Example: **5.9 or below**) may lead to the following errors:

Error 1: `Invalid configuration file`.



Error 2: `Configuration import failed.`



OR



Solution:

It is recommended to **upgrade MetaDefender Core to the latest version** before importing configuration settings. Each new version of Core introduces updated schemas and feature support, which may not be backward-compatible with older versions.

Support:

If you have followed all of the steps above but are still unable to resolve the issue. Please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How to Local whitelist a file using AllowList?

Check Your Version:

This article applies to all MD Core versions

Purpose:

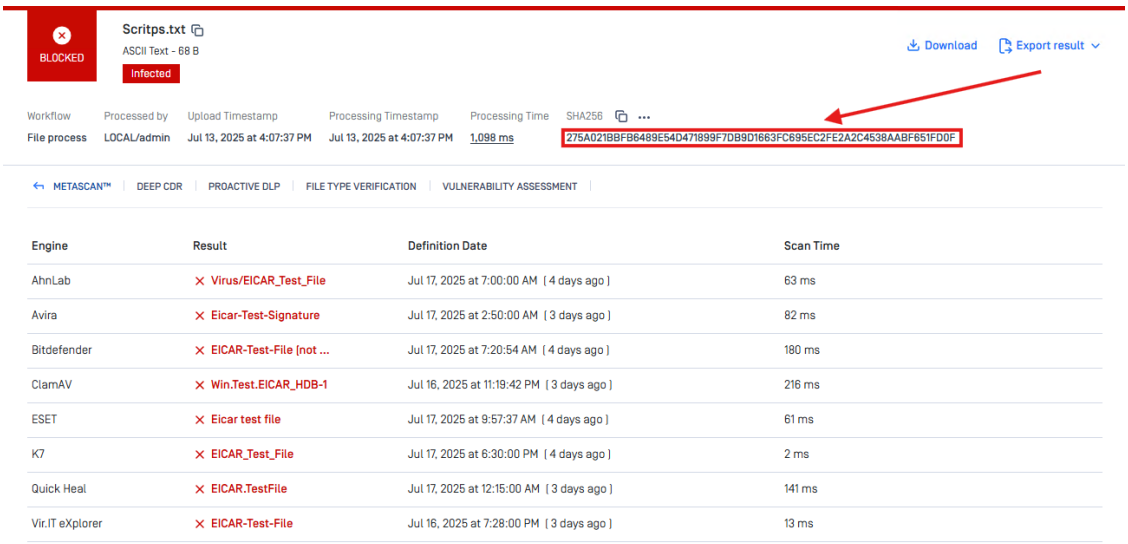
This article provides a step-by-step guide on how to **whitelist a file using the allowlist feature** in MetaDefender Core.

When to Use:

Local whitelisting is recommended when the user has verified and fully trusts the file, even if it is flagged by antivirus engines. This method is especially useful in environments where trusted internal tools or utilities are mistakenly identified as potentially harmful.

Scenario:

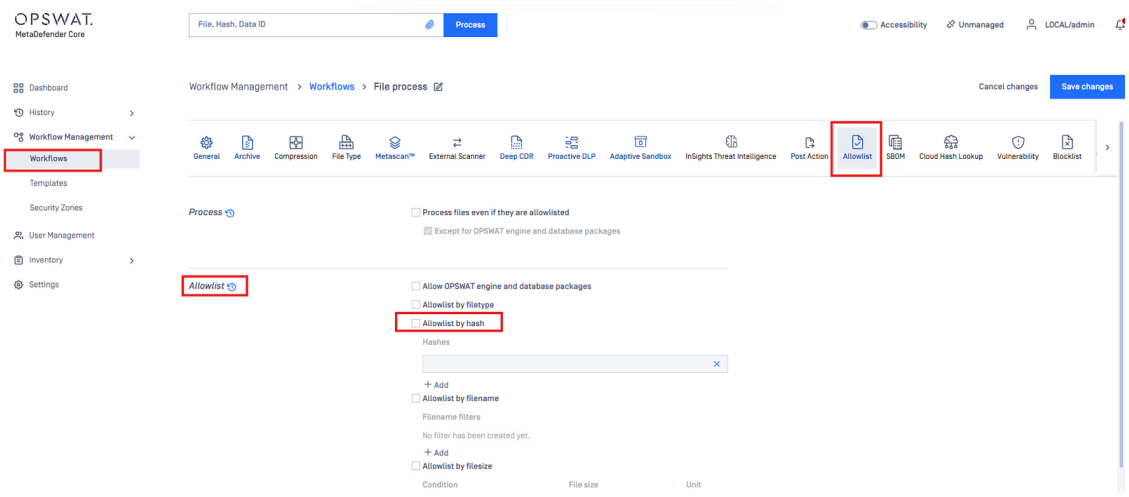
In this use case, we will use a text file containing the **EICAR test string** as the sample file.



Engine	Result	Definition Date	Scan Time
AhnLab	Virus/EICAR_Test_File	Jul 17, 2025 at 7:00:00 AM [4 days ago]	63 ms
Avira	Eicar-Test-Signature	Jul 17, 2025 at 2:50:00 AM [3 days ago]	82 ms
Bitdefender	EICAR-Test-File (not ...	Jul 17, 2025 at 7:20:54 AM [4 days ago]	180 ms
ClamAV	Win.Test.EICAR_HDB-1	Jul 16, 2025 at 11:19:42 PM [3 days ago]	216 ms
ESET	Eicar test file	Jul 17, 2025 at 9:57:37 AM [4 days ago]	61 ms
K7	EICAR_Test_File	Jul 17, 2025 at 6:30:00 PM [4 days ago]	2 ms
Quick Heal	EICAR.TestFile	Jul 17, 2025 at 12:15:00 AM [3 days ago]	141 ms
Vir.IT eXplorer	EICAR-Test-File	Jul 16, 2025 at 7:28:00 PM [3 days ago]	13 ms

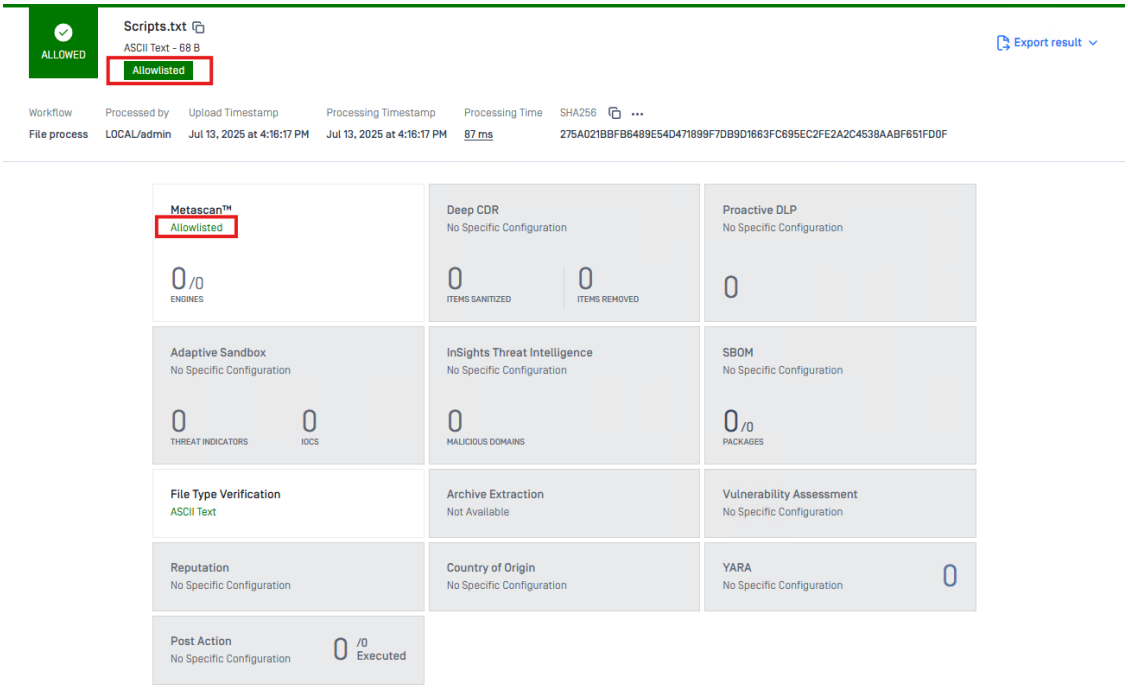
Steps to Allowlist a File

1. **Access Your Workflow:** Navigate to the workflow used for your scanning session in MetaDefender Core.
2. **Open the Allowlist Tab:** Select the **Allowlist** section from the workflow settings.
3. **Allow by Hash:** Add the hash value of your trusted file to the **Allowlist by Hash** section.



Result:

Once added, the file will be recognized as allowlisted during future scans and will not be flagged by AV engines configured in your workflow.



Warning:

If the file is confirmed to be malicious or poses a potential risk according to AV vendors, **use the allowlist at your own discretion**. Local whitelisting bypasses standard detection and should only be used when absolutely certain the file is safe.

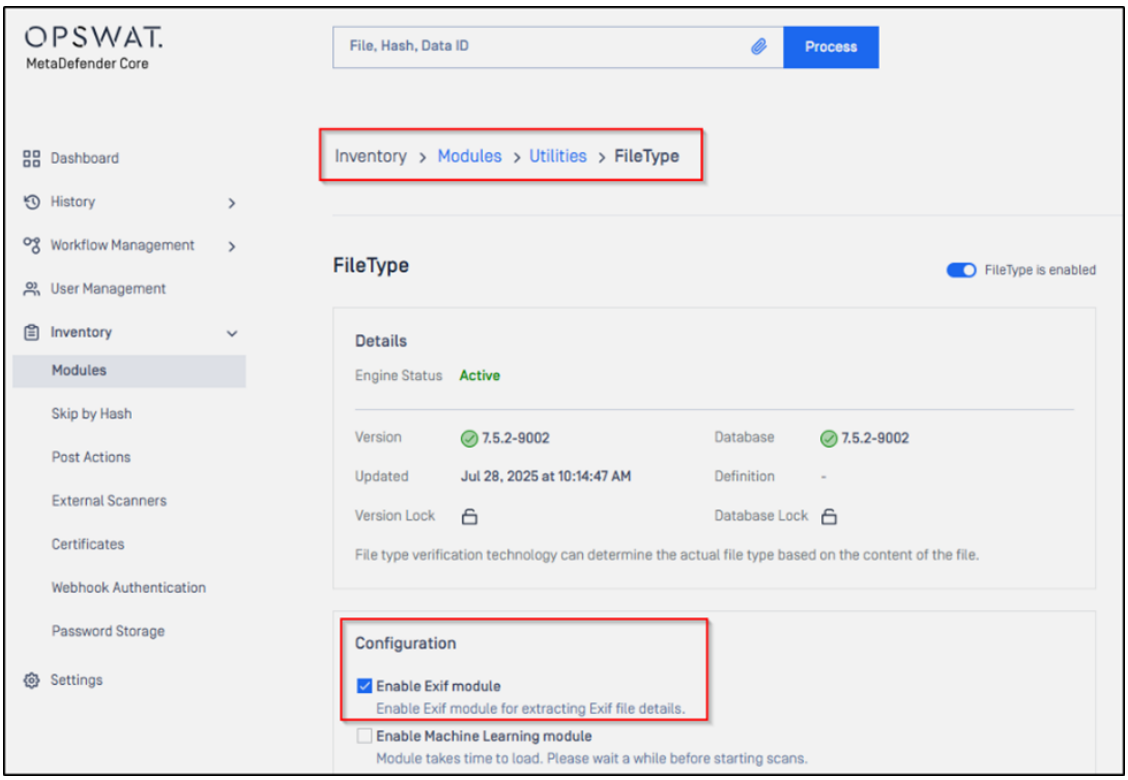
Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with our support engineer**.

What Is exiftool_win.exe and Why Is It Falsely Detected?

Overview

When the **Exif feature** is enabled in **MetaDefender Core's Filetype engine settings**, customers may notice the presence of the executable **exiftool*win.exe*** *under the Core resource directory (e.g., data\resources\filetype* 15_windows_<ID>\exif_cache_<ID>).



This file may be flagged by certain antivirus engines as a potential threat due to its unsigned status and self-extracting behavior.

This article explains why exiftool_win.exe exists, how it behaves, and what can be done to manage false positive detections.

Why Is exiftool_win.exe Present?

exiftool_win.exe is a **third-party utility** integrated into MetaDefender Core to extract metadata from files. It is a core component of the Exif processing functionality.

- **Tool Origin:** ExifTool is an open-source utility developed by Phil Harvey, widely used for reading and writing metadata.
- **Windows Packaging:** On Windows, ExifTool is distributed as a **self-extracting Perl executable**. When run:
- It extracts embedded Perl scripts to a **temporary folder**.
- This temporary location is defined during engine initialization by MetaDefender Core.

Example of the folder path:

C:\Program Files\OPSWAT\MetaDefender Core\data\resources\filetype_15_windows_\exif_cache_

Why Is It Flagged by Antivirus Software?

Some antivirus (AV) engines may detect exiftool_win.exe as malicious for the following reasons:

- **It is not digitally signed** – some AVs treat unsigned executables as suspicious.
- **Heuristic detection** – the self-extraction behavior is commonly seen in malicious droppers or downloaders.
- **False positive patterns** – occasionally, the hash or behavior matches known malware patterns erroneously.

Example:

- A customer reported detection as TrojanDownloader.Banload.btij by two AV engines on VirusTotal.
- Hash of the file in question:
- **SHA256**: 190fa21bca88e2c4ac0ebea467be79f2fda63ccd45915d72a2582ca30a3c7c67

What Can Be Done About It?

✅ Recommended Actions:

1. **Report False Positives:**
    1. Report the detection to the affected AV vendors.
    2. One vendor (Zillya) confirmed it as a false positive and resolved it in a database update.
2. **Allowlist MetaDefender Core Folder:**
    1. If real-time protection (RTP) is enabled on the server, we recommend allowlisting the **entire MetaDefender Core installation directory**, e.g.:

C:\Program Files\OPSWAT\MetaDefender Core\

1. **Monitor & Communicate**:
    1. This behavior is expected and known. Inform customers and security teams as necessary.
    2. Refer them to this article or similar documentation to clarify the usage and behavior of the file.

How to relocate installation folder in MetaDefender Core on Windows?

Info:

- This article applies to all MetaDefender Core v5

Overview

Some users prefer to install or move the installation folder to a specific storage location but are unsure which files or folders will be created after running the installer. Creating a procedure and listing the necessary paths can help users make informed decisions when selecting or relocating installation directories.

Use Case

Relocating the installation folder from C:\Program Files\OPSWAT\MetaDefender Core\ to a new location, e.g., E:\OPSWAT\MetaDefender Core.

Prerequisites

Users need to prepare a directory in advance for storing temporary files.

Steps to Configure

1. Stop the MetaDefender Core Service

Stop the MetaDefender Core service (ometascan or OPSWAT MetaDefender Core) before proceeding.

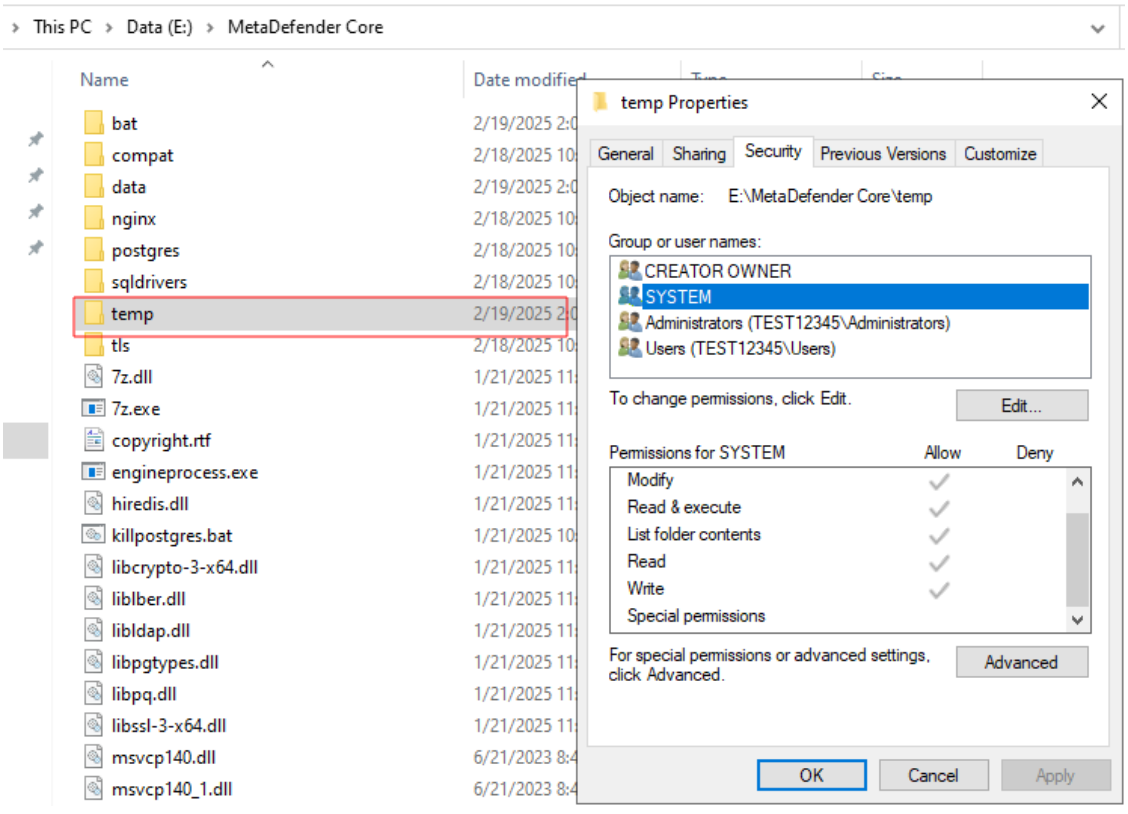
2. Move the Installation Folder

Manually move the entire **MetaDefender Core** folder from: C:\Program Files\OPSWAT\MetaDefender Core to the new location: E:\OPSWAT\MetaDefender Core\

3. Create new temp folder with full permission in the new installation folder:

Full path of a directory to use for storing temporary files. Users need to prepare this directory in advance.

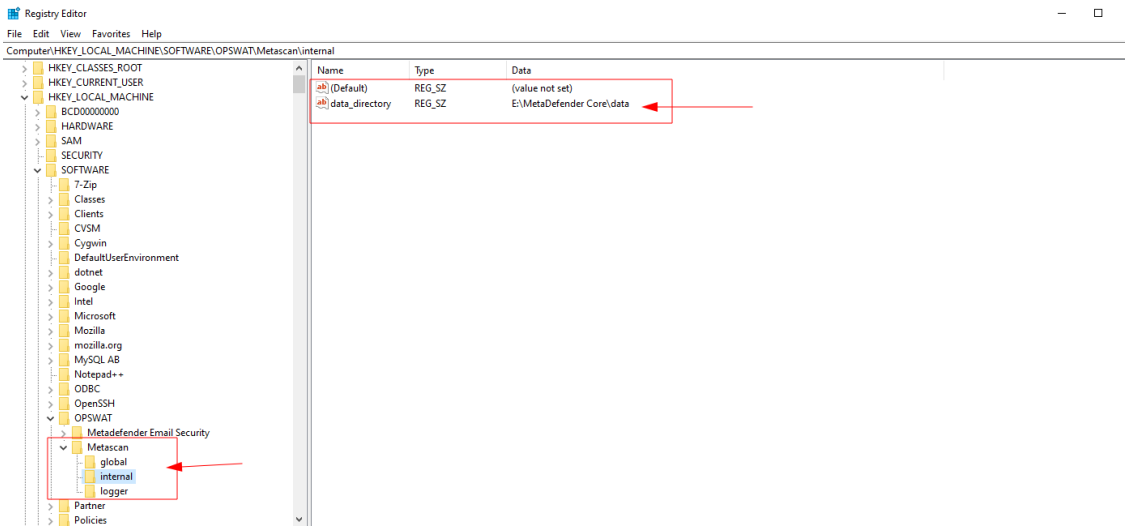
MetaDefender Core creates a subfolder called resources in this folder.



4. Update the Windows Registry

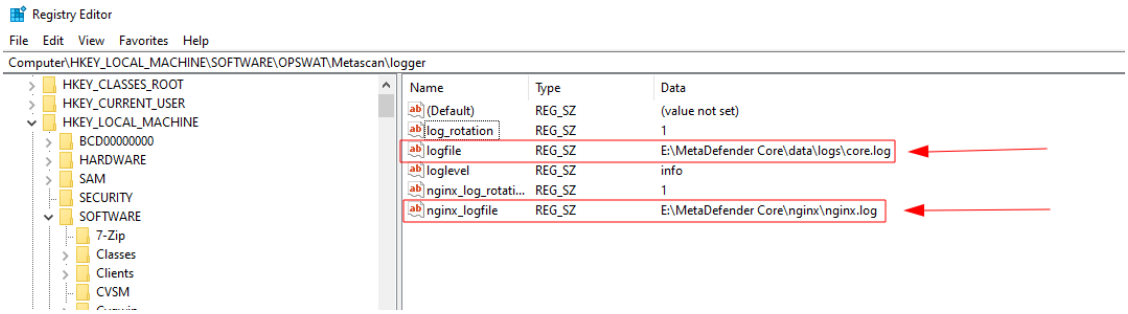
- Open the **Registry Editor** (regedit).
- Navigate to *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan*
- Update the **InstallationFolder** value: Edit the **InstallationFolder** key and change its value to the new path: E:\OPSWAT\MetaDefender Core

- In *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global,* add new value for tempdirectory with new data: *E:\MetaDefender Core\temp*
- Under Metascan, create a **new key** and name it **internal**.
- Inside **internal**, create a **new String value** and name it **data_directory**.
- Set the **data_directory** value to: *E:\OPSWAT\MetaDefender Core\data*
- This change ensures that MetaDefender Core correctly references the new data path for its database, updates, and other critical files.

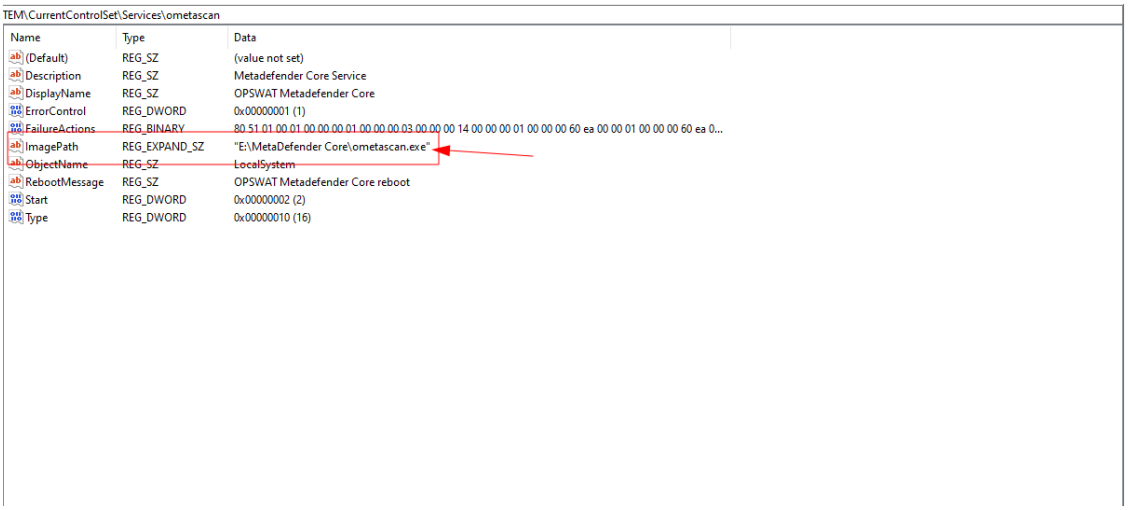


- In *Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger,* edit the new path for logfile and nginx_logfile

  1. **logfile**: E:\MetaDefender Core\data\logs\core.log
  2. **nginx_logfile**: E:\MetaDefender Core\nginx\nginx.log



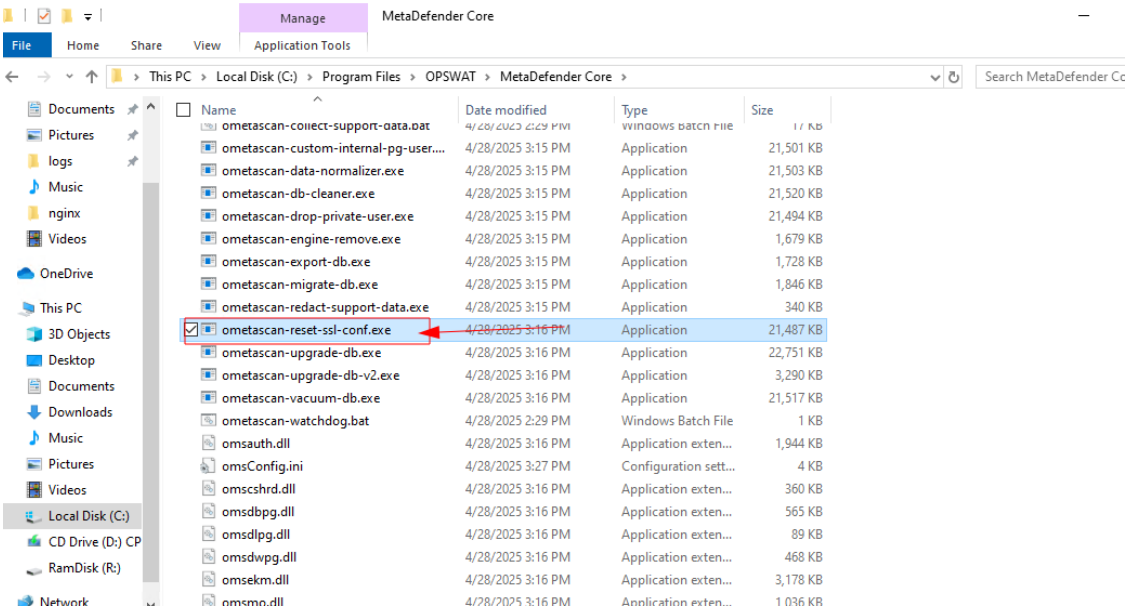
- Change a Windows service executable path:
  - Navigate to *Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ometascan*
  - Change data value of Image Path to “E:\OPSWAT\MetaDefender Core\ometascan.exe”



5.SSL configuration reset:

MetaDefender Core relies on nginx for its web interface. On startup, nginx will check the configured location for the SSL certificate, will attempt to load it into C:\Windows\Temp\ometascan and then will start and be ready to accept connections.

If at any time the certificate is deleted or removed, nginx will no longer be able to start and this will cause Core to completely fail. the best option will be to reset the SSL configuration and, once the Core service is successfully restarted using HTTP, the certificate can be imported back into the MetaDefender Core certificate inventory and the HTTPS configuration can be reenabled.



6. Restart MD Core service:

Start the MetaDefender Core service (ometascan or OPSWAT MetaDefender Core) and verify that the application functions correctly in the new location.

Support:

If you require further assistance, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How to Set Up MetaDefender Core to Connect to Syslog via TLS?

Check Your Version:

This article applies to MetaDefender Core version **5.15.1** releases deployed on Windows and Linux systems.

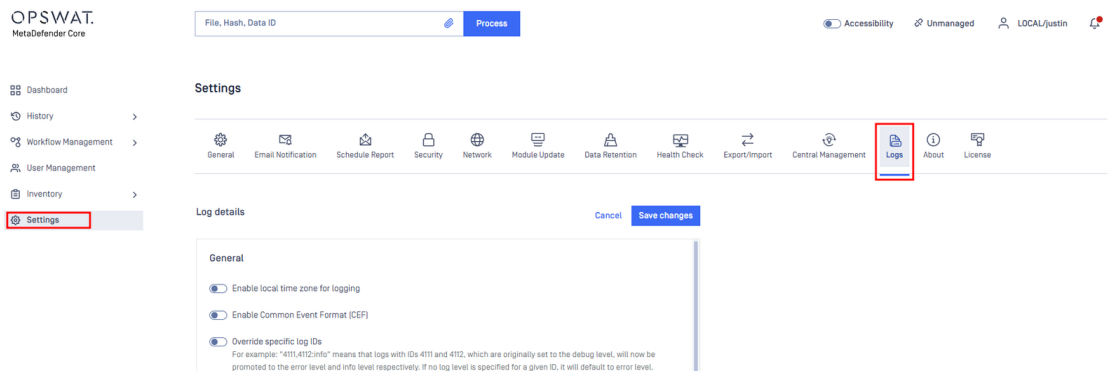
In this version, MetaDefender Core introduces a user interface for configuring syslog settings (refer to the Release notes).

Your Splunk server is configured to listen on a TLS port and is using a self-signed certificate.

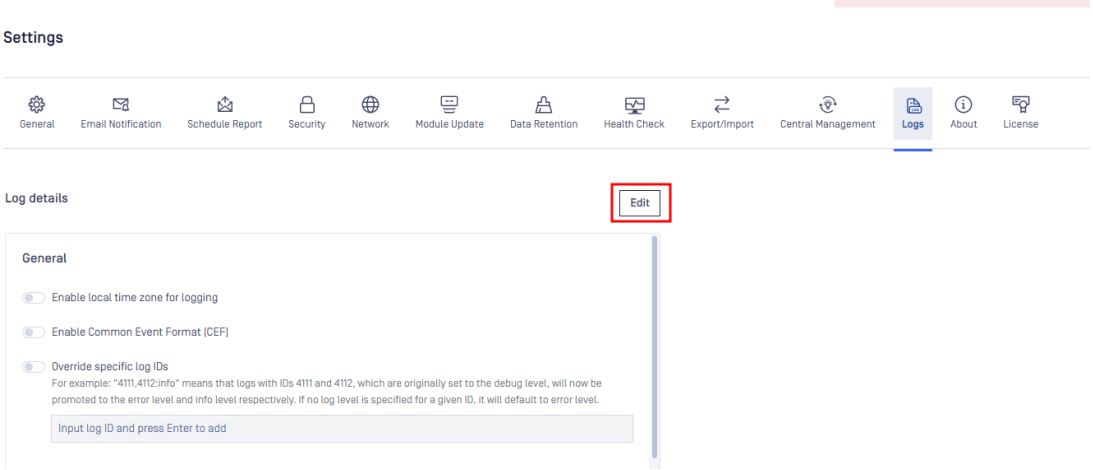
You can easily set up the syslog integration through the MetaDefender Core web UI:

1. Access the **Web Dashboard**.

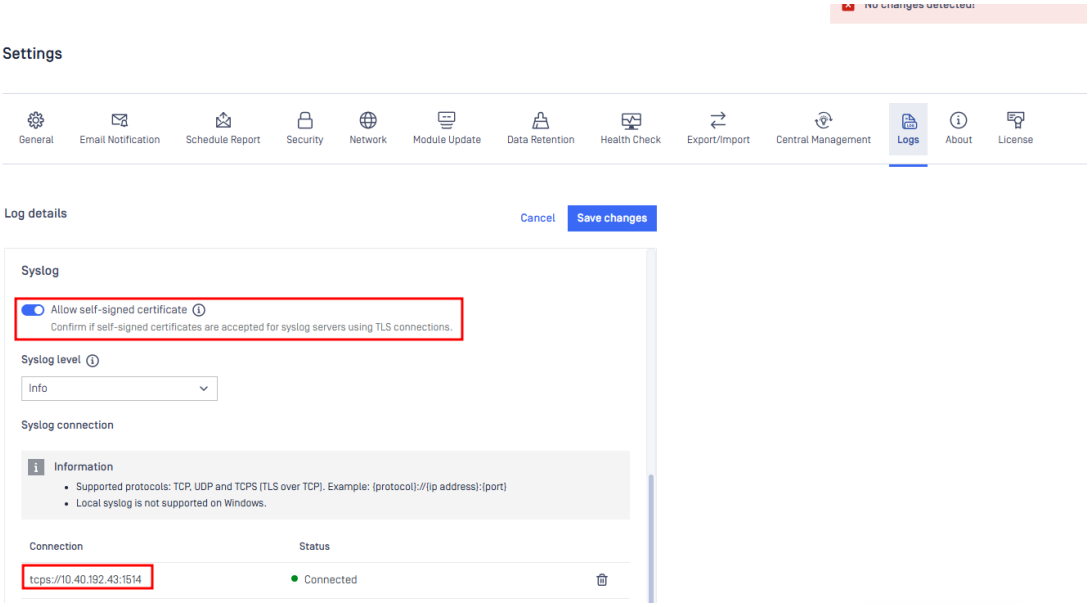
2. Navigate to **Settings > Logs** to begin the configuration.



3. Next, click on “Edit” to modify the syslog settings.



4. You will now see an option in the settings to allow the use of self-signed certificates.



5. Once enabled, the syslog connection status will show as “Connected,” indicating that MetaDefender Core has successfully established a connection with the syslog server.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to Work with Role Restrictions in MetaDefender Core Workflows?

Check Your Version:

This article is applied to MetaDefender Core releases deployed on Windows and Linux systems.

Understanding Role Restrictions in MetaDefender Core Workflows

Role Restrictions in workflows function differently than you might expect. They do **not** control who can view or edit workflows in the GUI. Instead, Role Restrictions determine which workflows are available to specific roles when submitting files for scanning.

How Role Restrictions Work

Role Restrictions limit workflow availability during the file submission process. Users can only select workflows that include either:

- Their assigned role
- The "Everyone" role

Testing Role Restrictions

To verify role restrictions are working correctly:

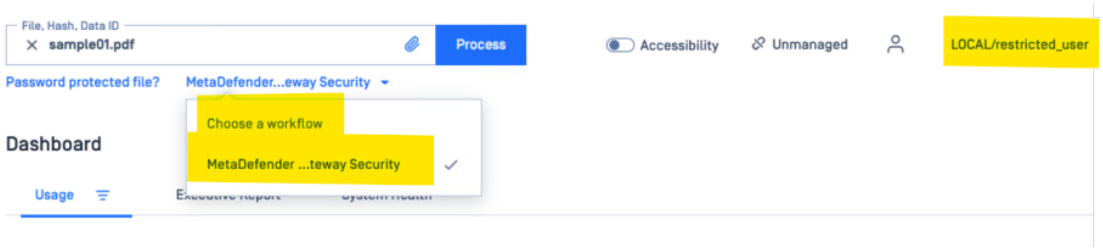
1. Log in as a user with a specific role (e.g., restricted_user)
2. Navigate to the file scanning interface
3. Drag a file to the scan box at the top of the page
4. Click the workflow selector (located to the lower left of the "Process" button)
5. Review the available workflows—only those containing the user's role will appear

Example Configuration

In this example, we configured role restrictions as follows:

- Added the "restricted_role" role to the Email Gateway Security workflow
- Removed both "restricted_role" and "Everyone" roles from all other workflows

Result: When logged in as a user with the "restricted_role" role, only the Email Gateway Security workflow appears in the workflow selector (see screenshot).



Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How Do I Setup Syslog connect to Splunk Cloud by using Universal Forwarder?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Linux systems.

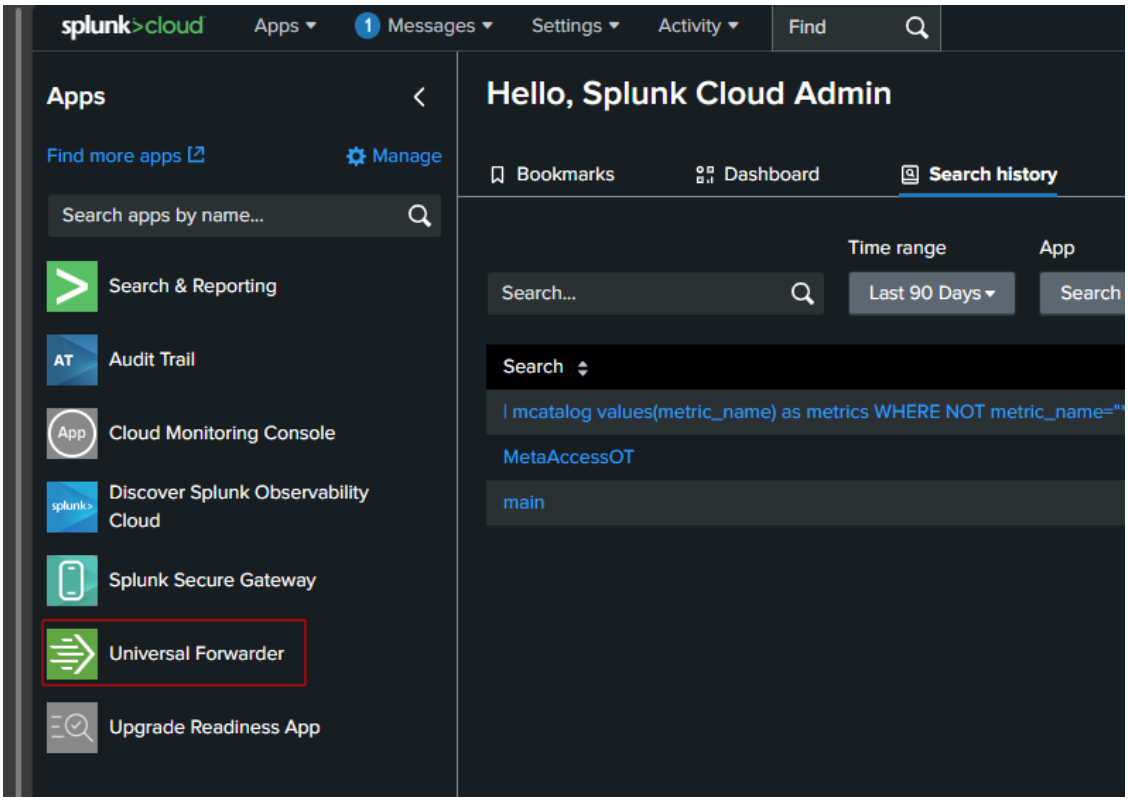
This KB, I will provide the guideline to setup integrate with Splunk Cloud. Splunk Cloud requires Universal Forwarder, therefore, we need to setup Universal Forwarder first. In this KB I am using RedHat 9.2 (Plow) to setup so the command in this KB will be runned on RedHat and package system should be RPM.

Code · html

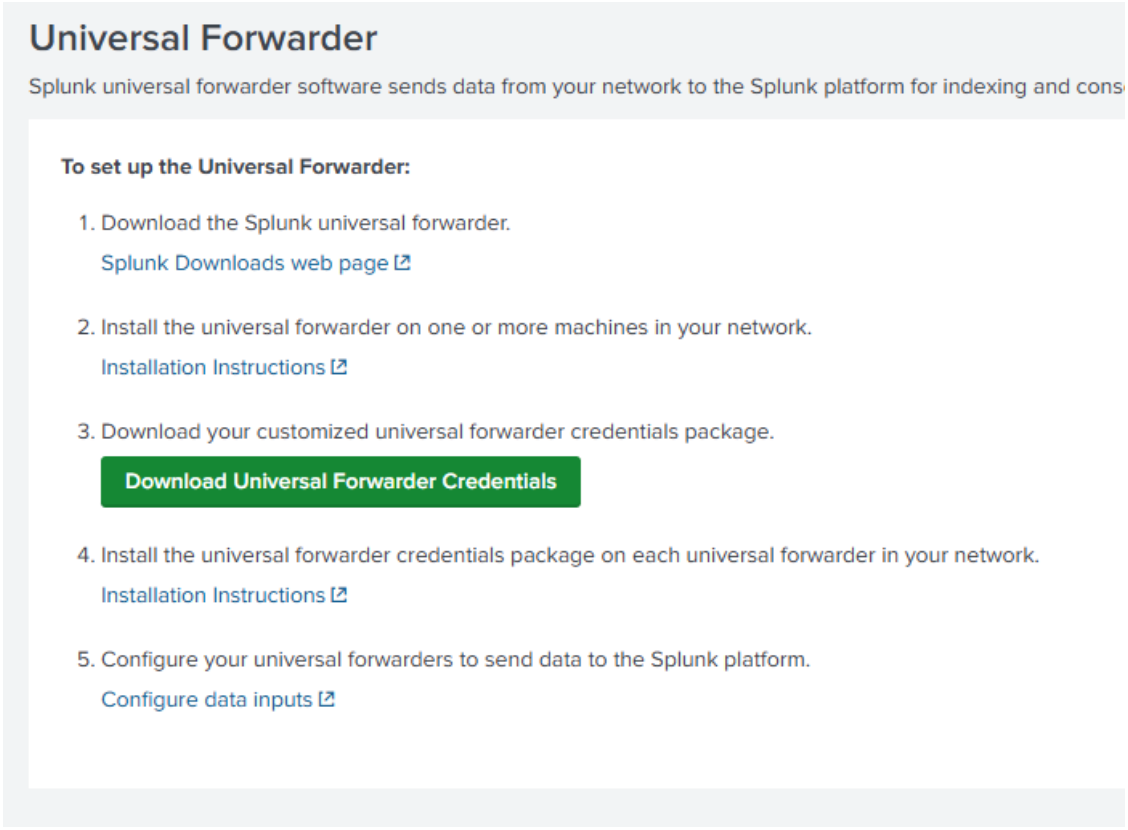
```
root>15:50:19 (Core) # cat /etc/redhat-release
Red Hat Enterprise Linux release 9.2 (Plow)
```

1 - Install Universal Forwarder

In your Splunk Instance, you can select this to setup Universal Forwarder

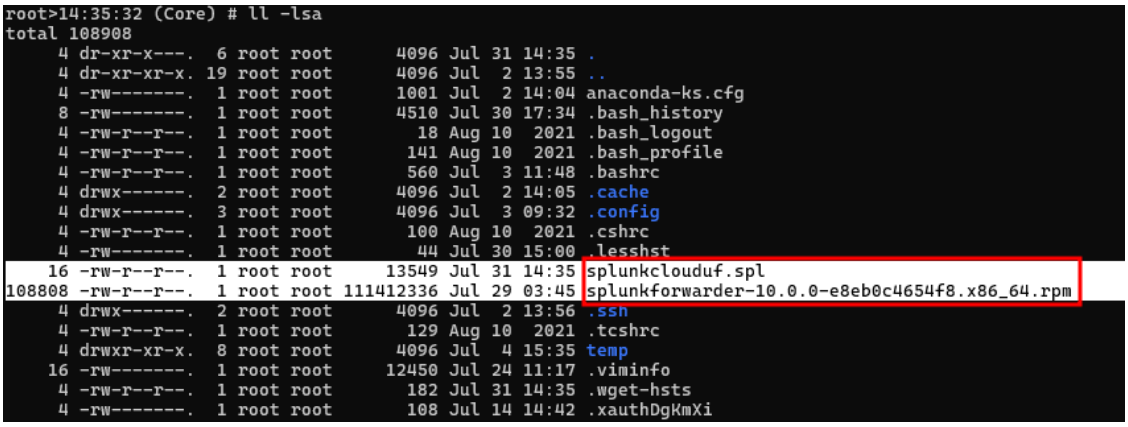


Then, you can follow this Splunk instruction to setup



After downloading the installer and credential files as shown in the screenshot below, you can proceed with setting up the Splunk Universal Forwarder.

- **Installer file:** `splunkforwarder-10.0.0-e8eb0c4654f8.x86_64.rpm`
- **Credential file:** `splunkclouduf.spl`



Install the Splunk Universal Forwarder using the following command:

Step 1: Create the Splunk user and group.

Code · html

```
useradd -m splunkfwd
groupadd splunkfwd
```

Step 2: Install the Splunk software, as described in the installation instructions for your platform in Installation instructions. Create the `$SPLUNK_HOME` directory wherever desired.

Code · html

```
export SPLUNK_HOME="/opt/splunkforwarder"
mkdir $SPLUNK_HOME
```

Step 3: Running command to install package.

Code · html

```
rpm -i splunkforwarder-10.0.0-e8eb0c4654f8.x86_64.rpm
```

The output should be

Code · html

```
rpm -i splunkforwarder-10.0.0-e8eb0c4654f8.x86_64.rpm
warning: splunkforwarder-10.0.0-e8eb0c4654f8.x86_64.rpm: Header V4 RSA/SHA256 Signature, key ID
b3cd4420: NOKEY
verify that this sytem has all the commands we will require to perform the preflight step
```

```
no need to run the splunk-preinstall upgrade check
complete
```

Step 4: Run the chown command to change the ownership of the Splunk directory and all its contents to the user that will run the Splunk software.

Step 5: run the command below to start

Code · html

```
$SPLUNK_HOME/bin/splunk start
```

or this command

Code · html

```
$SPLUNK_HOME/bin/splunk start --accept-license
```

You will be prompted to create a new administrator username and password. After entering the required information, you should see the following output:

Code · html

```
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"

This appears to be your first time running this version of Splunk.

Splunk software must create an administrator account during startup. Otherwise, you cannot log
in.
Create credentials for the administrator account.
Characters do not appear on the screen when you type in credentials.

Please enter an administrator username: admin
Password must contain at least:
   * 8 total printable ASCII character(s).
Please enter a new password:
Please confirm new password:
Creating unit file...
Important: splunk will start under systemd as user: splunkfwd
The unit file has been created.
...
...
All preliminary checks passed.

Starting splunk server daemon (splunkd)...
Done
```

To verify, you can run this command below

Code · html

```
$SPLUNK_HOME/bin/splunk version
```

The output should be:

Code · html

```
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Splunk Universal Forwarder 10.0.0 (build e8eb0c4654f8)
```

2 - Install the forwarder credentials

To install the forwarder credentials, run the following command:

Code · html

```
$SPLUNK_HOME/bin/splunk install app splunkclouduf.spl
```

It will prompt you for the username and password you set during the Splunk Forwarder installation.

Once completed, you should see an output confirming the action, followed by a prompt to restart Splunk.

Code · html

```
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Couldn't change ownership for /opt/splunkforwarder/etc/deployment-apps: Operation not permitted
App '/opt/splunkforwarder/etc/deployment-apps/splunkclouduf.spl' installed
You need to restart the Splunk Server (splunkd) for your changes to take effect.
```

To restart Splunk, you can run this command

Code · html

```
$SPLUNK_HOME/bin/splunk restart
```

The output should be

Code · html

```
All preliminary checks passed.

Starting splunk server daemon (splunkd)...
Done
```

To configure Splunk to listen for syslog, create the file `/opt/splunkforwarder/etc/system/local/inputs.conf`

and add the following settings:

Code · html

```
cat /opt/splunkforwarder/etc/system/local/inputs.conf
# For TCP syslog (optional)
[tcp://1514]
index = main
sourcetype = syslog
disabled = 0
```

To verify whether the Splunk Universal Forwarder (UF) port is open, you can use the following command:

Code · html

```
netstat -ntlp | grep 1514
tcp        0      0 0.0.0.0:1514            0.0.0.0:*               LISTEN      1071882/splunkd
```

3 - Integrate MD Core syslog with Splunk Universal Forwarder.

If you are using MD Core on Linux (Red Hat, Ubuntu, or Debian), you can configure syslog by adding the settings below to the /etc/ometascan/ometascan.conf file.

These settings should be placed directly beneath the [logger] section.

Code · html

```
syslog=tcp://10.40.192.43:1514
syslog_level=info
```

For example

Code · html

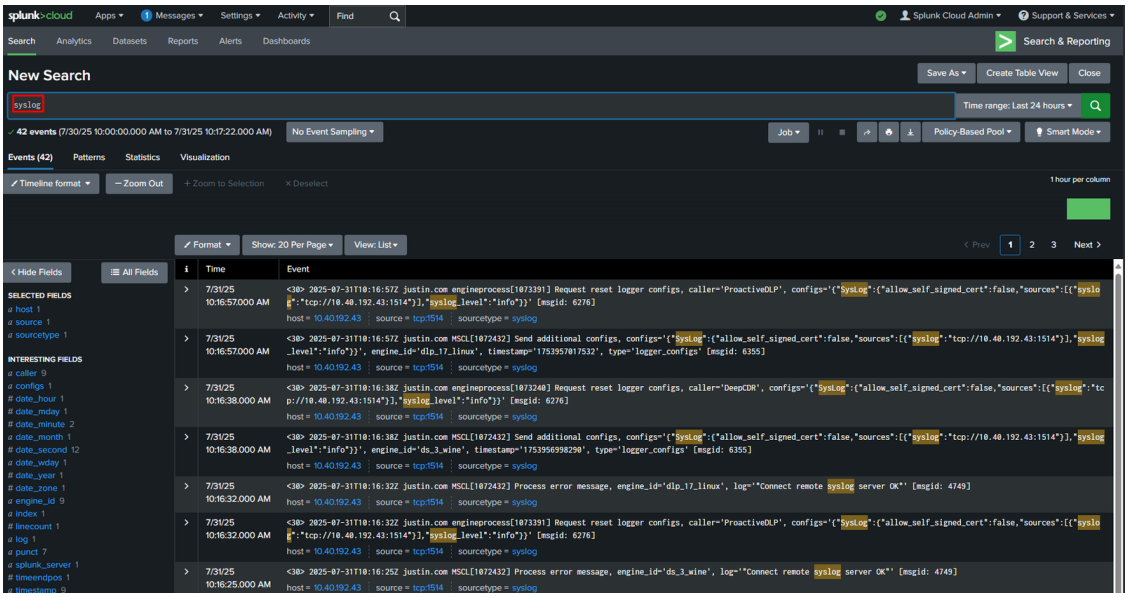
```
[global]
dbmode=1
restaddress=*
restport=8008
[logger]
log_rotation=1
logfile=/var/log/ometascan/ometascan.log
loglevel=info
nginx_log_rotation=1
nginx_logfile=/var/log/ometascan/nginx-ometascan.log
syslog=tcp://<IP_SPLUNK_UNIVERSAL_FORWARDER>:1514
syslog_level=info
```

Then, restart MD Core service.

Code · html

```
systemctl restart ometascan
```

Now, go back to your Splunk instance and search for "syslog" — you should see entries related to MD Core appearing there.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I customize the bundled PostgreSQL server configuration in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

This article describes how to customize the PostgreSQL configuration for the bundled PostgreSQL server that comes with MetaDefender Core.

For example, one goal of making such configurations is to meet requirements in the CIS benchmark for PostgreSQL server.

Steps to Configure

To resolve the issues and achieve CIS benchmark compliance without affecting MetaDefender Core operations,

follow these steps:

Important:

Do not modify the bundled `postgresql.conf` located at `OPSWAT\MetaDefender Core\data\pg_data\postgresql.conf` because it is controlled by MetaDefender Core and resets on service restart.

1. Instead, create and edit a new configuration file, which overrides the default configuration, and changes made here will persist after restarts:
   1. Windows: `<installation directory>\OPSWAT\MetaDefender Core\postgres\postgresql.conf`
   2. Linux: `/usr/lib/ometascan/postgres/postgresql.conf`
2. Add any configuration parameters needed, such as `ssl_ciphers`, and save the file.
3. Restart the MetaDefender Core service (which also restarts the bundled PostgreSQL server).

Note: if SSL configuration for PostgreSQL server needs to be made, such as enabling SSL connection, please refer to SSL Connection for bundled PostgreSQL.

Verify the Change

After the service restarts, check and verify that the custom parameters have overridden the default ones.

If you're making these configurations according to CIS benchmarks, refer to the relevant CIS documentation for the audit steps to verify that PostgreSQL is using the recommended settings.

Remote PostgreSQL server:

For more control over PostgreSQL configuration, consider deploying MetaDefender Core with a **remote PostgreSQL server** that you manage. This avoids limitations of the bundled PostgreSQL server.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How do I detect PUP/PUA as infected?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems with the latest engine updates.

Description:

PUP stands for Potentially Unwanted Program

PUA stands for Potentially Unwanted Application

Software that isn't malicious by definition but could be, it is often:

- Installed without clear user consent
- Bundled with other software
- Performs unwanted behavior (e.g., showing ads, tracking activity, slowing performance)

Common Examples:

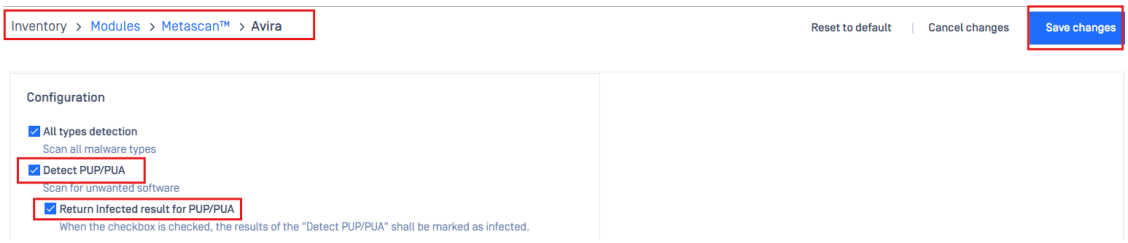
- **Toolbars or browser hijackers** (e.g., search engine changers)
- **Adware** (injects ads or pop-ups)
- **System optimizers** or **registry cleaners** (that fake "problems" to upsell)
- **Fake antivirus or driver updaters**
- Software that installs **background processes without notice**

How do I mark these detections as infected?

By default, PUP/PUA are labeled in their own category. MetaDefender Core separates unwanted detections into three categories: Suspicious, Infected, and PUP/PUA.

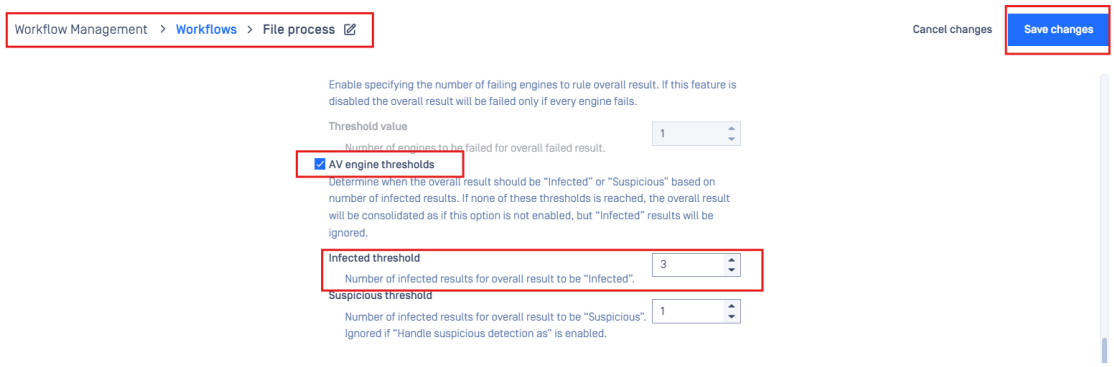
To place PUP/PUA into the infected category, please check the box below on the engine [not available for all engines].

Example:



Use case:

If you have a threshold built into the workflow to only mark a file as “Infected” if x or more engines find a malware signature, PUP/PUA will now be included in this detection threshold.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What is the difference between "Engine Current Status" and "Engine Advanced Configuration" export options?

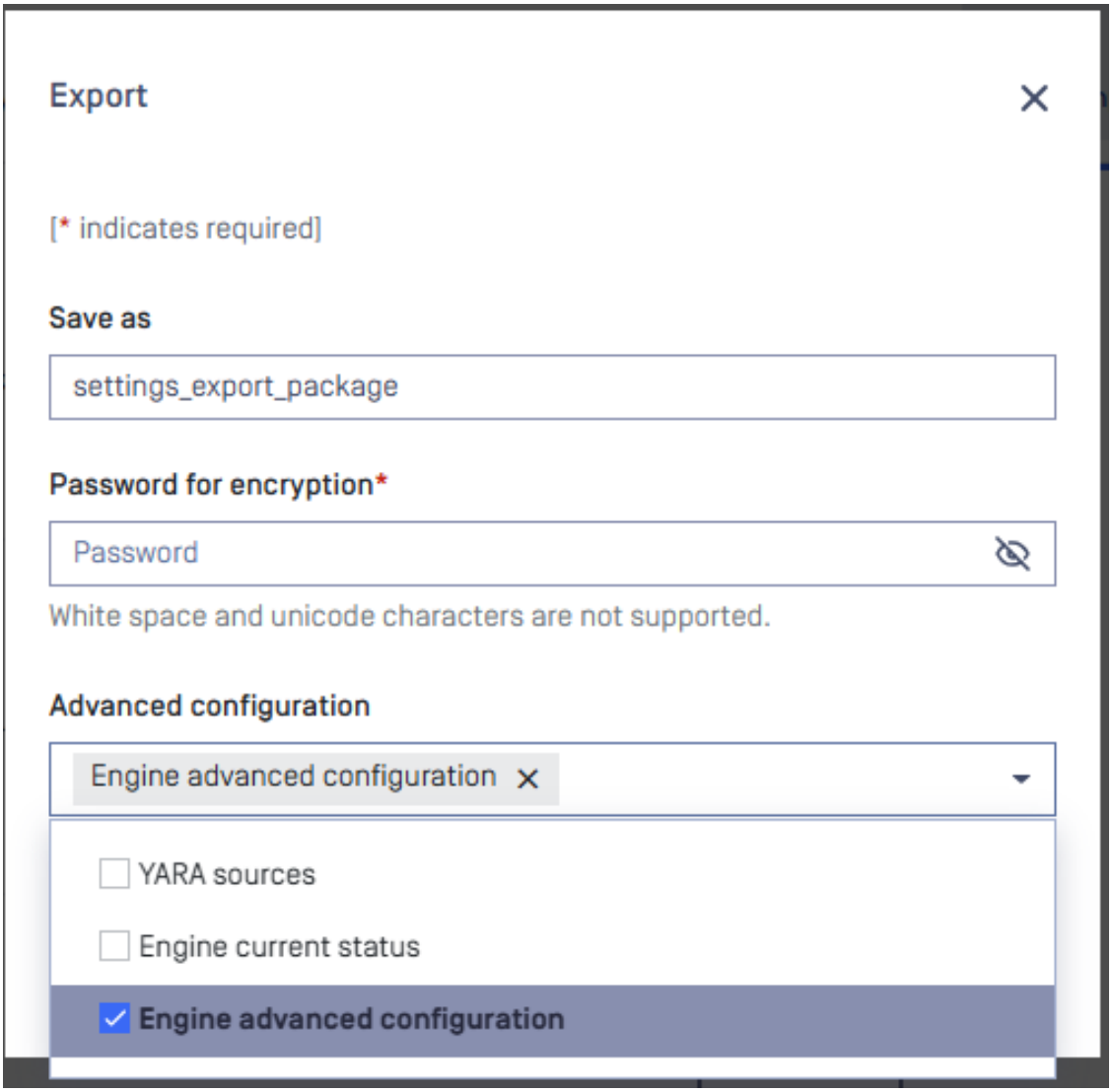
Check Your Version:

This article applies to all MetaDefender Core v5.15.1 releases deployed on Windows and Linux systems.

Overview

In the 5.15.1 version of MetaDefender Core, a new option for exporting called "Engine advanced configuration" was introduced

Unlike the older Engine Current Status export, which only provided the basic operational state of engines (such as enabled or disabled), the new Engine Advanced Configuration export gives you full visibility into all engine-level parameters, including custom detection rules, logging levels, and module configurations.



Export Options Explained

1. Engine Current Status

- Purpose: Exports the basic operational state of each engine (enabled/disabled).

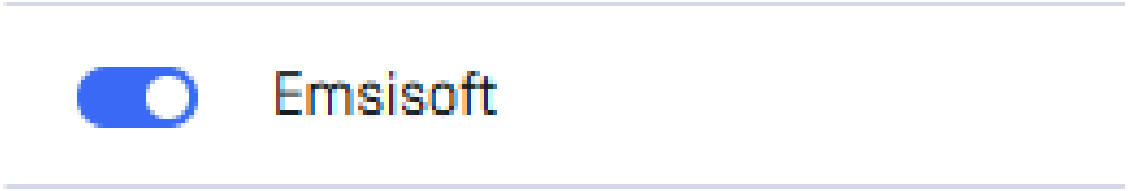
Example Export Config:

Code · html

```
{
    "engine": {
        "engine_status": {
            "name": "Engine status",
            "status_list": [
                {
                    "engine_id": "emsisoft_1_windows",
                    "status": 1
```

```
                },
                {
                    "engine_id": "ikarus_1_windows",
                    "status": 0
                }
            ]
        }
    }
}
```

Emsisoft Engine Enabled Example:



Ikarus Engine Disabled Example:



2. Engine Advanced Configuration

- Purpose: Exports all custom and detailed settings per engine.

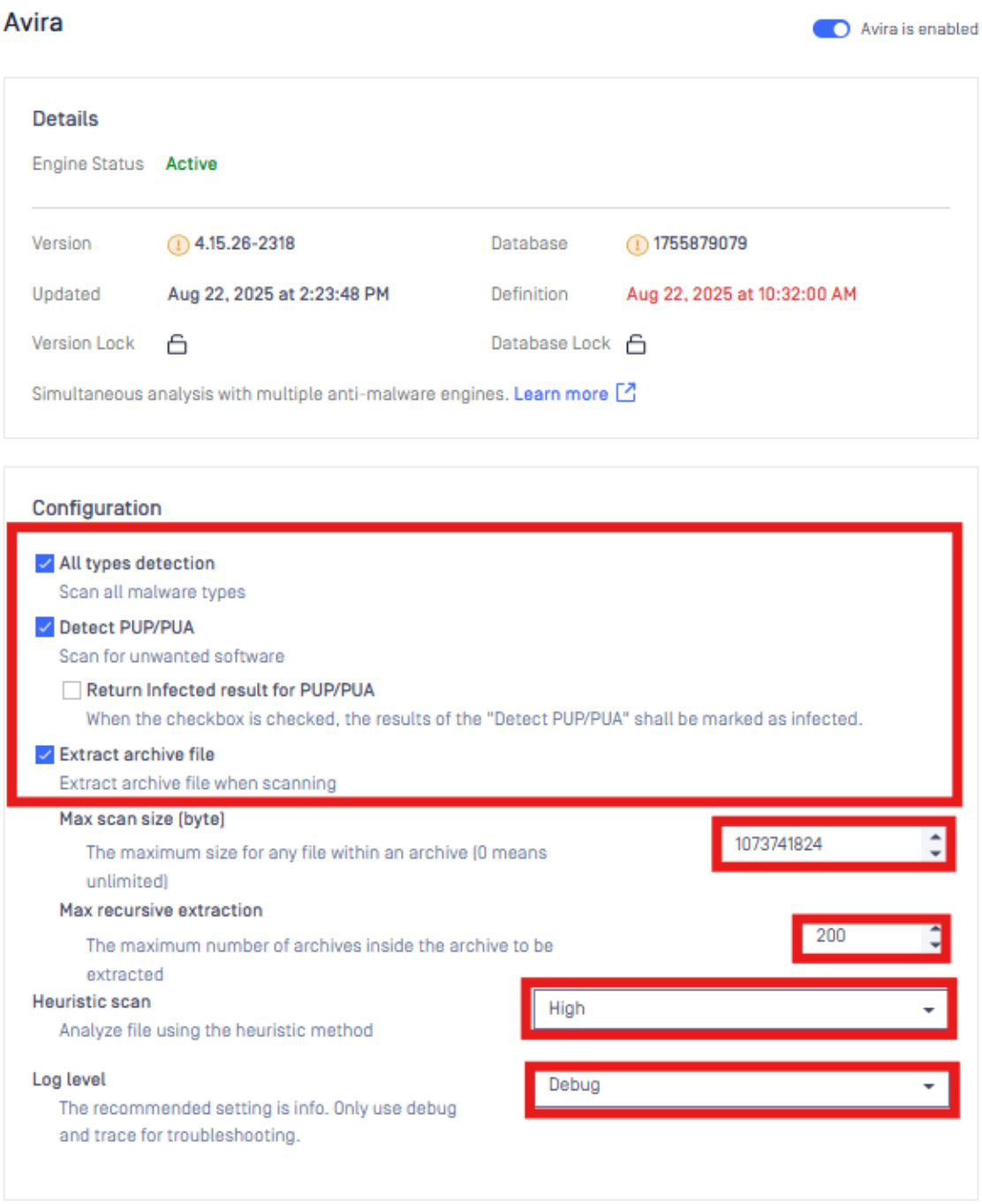
Avira Engine Export Config:

Code · html

```
{
"engine":{
"engine_config":{
"full_name":"engine.engine",
"name":"EngineConfiguration",
"schema":"engine",
  "table": "engine",
            "values": {
                "3": {
                    "configuration": "{\"config\":{\"engine.avira_1_windows.config\":
{\"values\":{\"detect_all_types\":true,\"enable_pup_scan\":true,
\"enable_pup_scan.enable_pup_infected\":false,\"extract_archive\":true,
\"extract_archive.max_extract_archive_file_size\":1073741824,
```

```
\"extract_archive.max_extract_archive_recursion_depth\":200,\"heuristic_scan\":3,
\"log_level\":2}}},\"config-serial\":1,\"schema-version\":6}",
                    "engine_prototype": "avira_windows",
                    "engine_type": "av",
                    "instance_id": "1",
                    "pin_database": "0",
                    "pin_engine": "0",
                    "schema": "{\\\"product\":{\n\"display_name\":\"Avira\",\n\"id\":
\"avira_1_windows\"\n},\n\"schema\":[\n{\n\"default_value\":true,\n\"display_name\":
\"Alltypesdetection\",\n\"id\":\"detect_all_types\",\n\"info\":\"Scanallmalwaretypes\",
\n\"type\":\"bool\"\n},\n{\n\"default_value\":true,\n\"display_name\":\"DetectPUP/PUA\",
\n\"fields\":[\n{\n\"default_value\":false,\n\"display_name\":\"ReturnInfectedresultforPUP/
PUA\",\n\"id\":\"enable_pup_infected\",\n\"info\":\"Whenthecheckboxischecked,theresultsofthe\\
\"DetectPUP/PUA\\\"shallbemarkedasinfected.\",\n\"type\":\"bool\"\n}\n],\n\"id\":
\"enable_pup_scan\",\n\"info\":\"Scanforunwantedsoftware\",\n\"type\":\"feature\"\n},
\n{\n\"default_value\":false,\n\"display_name\":\"Extractarchivefile\",\n\"fields\":
[\n{\n\"default_value\":1073741824,\n\"display_name\":\"Maxscansize(byte)\",\n\"id\":
\"max_extract_archive_file_size\",\n\"info\":
\"Themaximumsizeforanyfilewithinanarchive(0meansunlimited)\",\n\"max\":9223372036854775807,
\n\"min\":0,\n\"type\":\"number\"\n},\n{\n\"default_value\":200,\n\"display_name\":
\"Maxrecursiveextraction\",\n\"id\":\"max_extract_archive_recursion_depth\",\n\"info\":
\"Themaximumnumberofarchivesinsidethearchivetobeextracted\",\n\"max\":1000,\n\"min\":0,
\n\"type\":\"number\"\n}\n],\n\"id\":\"extract_archive\",\n\"info\":
\"Extractarchivefilewhenscanning\",\n\"type\":\"feature\"\n},\n{\n\"default_value\":1,
\n\"display_name\":\"Heuristicscan\",\n\"id\":\"heuristic_scan\",\n\"info\":
\"Analyzefileusingtheheuristicmethod\",\n\"infos\":[\n\"Disable\",\n\"Lazy\",\n\"Normal\",
\n\"High\"\n],\n\"labels\":[\n\"Disable\",\n\"Lazy\",\n\"Normal\",\n\"High\"\n],\n\"type\":
\"select\",\n\"values\":[\n0,\n1,\n2,\n3\n]\n},\n{\n\"default_value\":3,\n\"display_name\":
\"Loglevel\",\n\"id\":\"log_level\",\n\"info\":
\"Therecommendedsettingisinfo.Onlyusedebugandtracefortroubleshooting.\",\n\"infos\":
[\n\"Trace\",\n\"Debug\",\n\"Info\",\n\"Warning\",\n\"Error\",\n\"Critical\",\n\"Disable\"\n],
\n\"labels\":[\n\"Trace\",\n\"Debug\",\n\"Info\",\n\"Warning\",\n\"Error\",\n\"Critical\",
\n\"Disable\"\n],\n\"type\":\"select\",\n\"values\":[\n1,\n2,\n3,\n4,\n5,\n6,\n7\n]\n}\n],
\n\"version\":6\n}\n"
                }
            }
        }
    }
}
```


Avira Engine Advanced Configuration UI:



Can I Export Both?

Yes, both options can be exported individually or together, depending on your needs. This provides flexibility to retain the engine's operational status and advanced configuration for backup/restoring purposes.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Should I Use Subject Alternative Name (SAN) Certificates for Core and Kiosk GUIs?

Check Your Version:

This article applies to all MetaDefender product releases deployed on Windows and Linux systems.

Overview

Customers may prefer to use a Subject Alternative Name (SAN) in their SSL certificate to simplify the process of securing multiple instances of Core or Kiosk. Using a SAN certificate, customers can secure as many hostnames or IPs as they like using the same SSL certificate.

As the Common Name (CN) implementation ages out from current generation browsers, customers may also be concerned that OPSWAT will deprecate support of CN-based certificates. OPSWAT currently has no plans to deprecate Common Name certificates.

However, both the Chrome browser and Microsoft Edge deprecated support for Common Name certs in 2017, and Firefox followed suit in 2022.

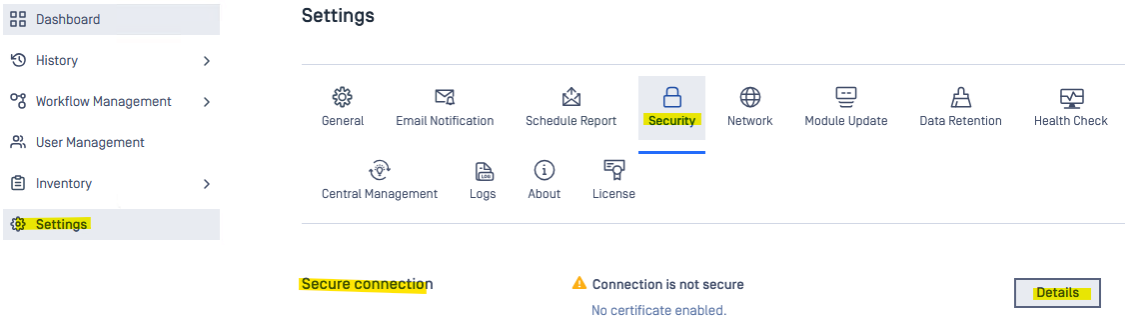
Recommendation

For convenience and greater security, OPSWAT recommends transitioning away from Common Name-based certificates and adopting the use of Subject Alternative Names instead.

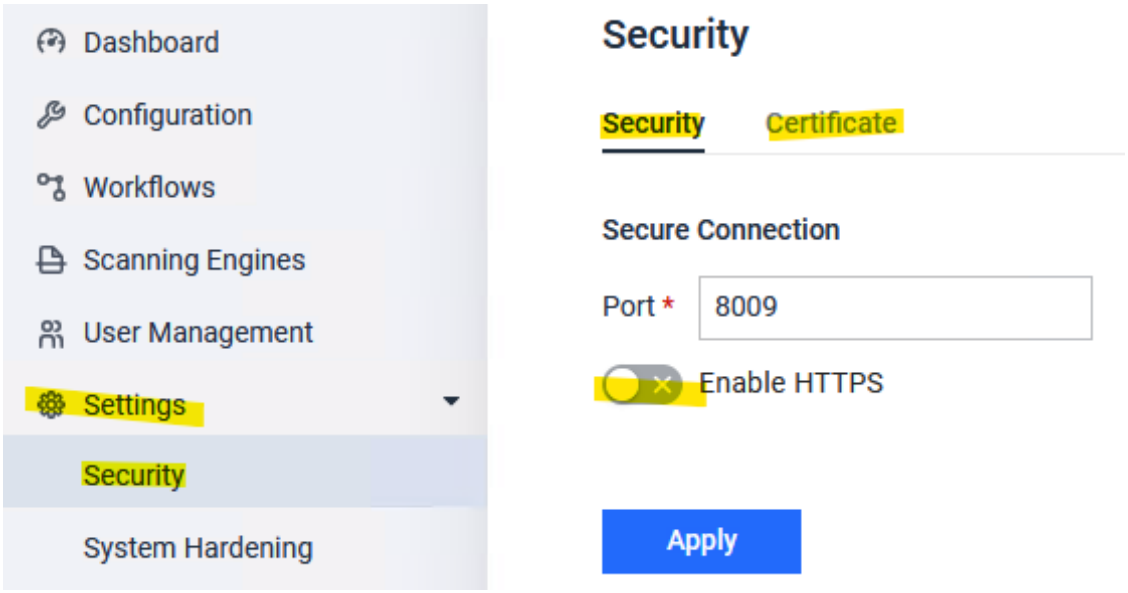
Steps to Configure

After generating your SAN certificate and private key in X509, DER-encoded format:

- Go to the Security section of each OPSWAT product's GUI
- For MD Core that's here:



- For Kiosk, the settings are here:



- Choose the new certificate/private key pair.

Verify the Change

If Chrome or Firefox previously reported a Name Mismatch error when using a Common Name certificate, you should now notice that they allow connection without throwing that error.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to customize Nginx logs to identify mTLS client connections?

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows and Linux systems.

Currently, **MetaDefender Core does not support editing the existing log format.** However, it supports creating a new custom log format where Nginx can write the needed details.

Follow the steps below to generate a new log file that lets you differentiate between mTLS and regular requests:

1. **Create the configuration files:**

- Windows:
  - C:\Program Files\OPSWAT\MetaDefender Core\nginx**newlog.conf** (to configure the new log format)
  - C:\Program Files\OPSWAT\MetaDefender Core\nginx**built-in/mtls.conf** (to configure where nginx will write the log to)

- Linux:
  - /etc/ometascan/nginx.d/**newlog.conf** (to configure the new log format)
  - /etc/ometascan/nginx.d/**built-in/mtls.conf** (to configure where nginx will write the log to)

2. **Add the following data to the config files:**

newlog.conf:

Code · html

```
log_format mtls '$remote_addr - $ssl_client_s_dn [$time_local] "$request" '
                '$status $body_bytes_sent "$http_referer" '
                '"$http_user_agent" ssl_client_verify "$ssl_client_verify" ssl_client_s_dn
"$ssl_client_s_dn" ssl_client_i_dn "$ssl_client_i_dn" ssl_client_serial "$ssl_client_serial"';

log_format message_log_file_batch_mtls '$remote_addr - $remote_user [$time_local] ' '"$request"
$status $body_bytes_sent ' '"$batch_id" ' '"$http_referer" "$http_user_agent" ' '$request_time'
'ssl_client_verify "$ssl_client_verify" ssl_client_s_dn "$ssl_client_s_dn" ssl_client_i_dn
"$ssl_client_i_dn" ssl_client_serial "$ssl_client_serial"';

log_format message_log_file_mtls '$remote_addr - $remote_user [$time_local] ' '"$request"
$status $body_bytes_sent ' '"batch_id:$http_batch" ' '"$data_id" ' '"$http_referer"
"$http_user_agent" ' '$request_time' 'ssl_client_verify "$ssl_client_verify" ssl_client_s_dn
"$ssl_client_s_dn" ssl_client_i_dn "$ssl_client_i_dn" ssl_client_serial "$ssl_client_serial"';
```

mtls.conf

Code · html

```
ssl_client_certificate "path_to_ca_cert/ca.crt";
ssl_verify_client optional;

access_log "path_to_nginx_folder\\mtls.log" mtls;

location ~ "^/file/sync$" {
            access_log "path_to_nginx_folder\\mtls.log" message_log_file_mtls;

            set $data_id "";
            header_filter_by_lua_block {
                if ngx.header["data-id"] then
                        ngx.var.data_id = "data_id" .. ":" .. ngx.header["data-id"];
                end
            }

            if ( $request_method !~ ^(POST)$ ) {
                return 404;
            }
            more_set_headers 'Cache-Control: no-cache';
```

```
            content_by_lua_file resource/resourcehandler.raw;
        }

location ~ "^/file/batch$" {
            access_log "path_to_nginx_folder\\mtls.log" message_log_file_batch_mtls;

            set $batch_id "";
            body_filter_by_lua_block {
                ngx.ctx.buffered = (ngx.ctx.buffered or "") .. ngx.arg[1]
                if ngx.arg[2] then
                    ngx.ctx.buffered = string.sub(ngx.ctx.buffered, 1, 5000)
                end
                local brief_batch_id = string.gsub(ngx.ctx.buffered, "%\x22", "")
                local resp_batch_id = brief_batch_id:match('^{batch_id:([a-f0-9]+)')
                if resp_batch_id then
                    ngx.var.batch_id = "batch_id" .. ":" .. resp_batch_id
                end
            }

            if ( $request_method !~ ^(POST)$ ) {
                return 404;
            }

            more_set_headers 'Cache-Control: no-cache';
            content_by_lua_file resource/resourcehandler.raw;
        }
location ~ "^/(file|process)$" {
            access_log "path_to_nginx_folder\\mtls.log" message_log_file_mtls;

            set $data_id "";
            body_filter_by_lua_block {
                ngx.ctx.buffered = (ngx.ctx.buffered or "") .. ngx.arg[1]
                if ngx.arg[2] then
                    ngx.ctx.buffered = string.sub(ngx.ctx.buffered, 1, 5000)
                end
                local brief_data_id = string.gsub(ngx.ctx.buffered, "%\x22", "")
                local resp_data_id = brief_data_id:match('^{data_id:([a-f0-9]+)')
                if resp_data_id then
                    ngx.var.data_id = "data_id" .. ":" .. resp_data_id
                end
            }

            if ( $request_method !~ ^(POST)$ ) {
                return 404;
            }
            more_set_headers 'Cache-Control: no-cache';
            content_by_lua_file resource/resourcehandler.raw;
        }
```

3. **Restart the MetaDefender Core service**

4. **Verify the new configuration by sending requests**

- Perform a scan request **with mTLS** and **without mTLS**.
- The result will be as follows:

With mTLS:

Code · html

```
10.40.50.228 - CN=client\x5C0A [29/Jul/2025:13:20:21 +0700] "POST /file/async HTTP/1.1" 404 948
"-" "curl/7.81.0" ssl_client_verify "SUCCESS" ssl_client_s_dn "CN=client\x5C0A" ssl_client_i_dn
"CN=my_ca" ssl_client_serial "2553AE8B5C0360FA6A015EF3F53B4A2EF59A664E"
```

Without mTLS:

Code · html

```
::1 - - [29/Jul/2025:13:19:30 +0700] "POST /file HTTP/1.1" 200 57 "batch_id:-"
"data_id:a00d24282a7d4e8098bab2fd042ccfcd" "https://localhost:8008/" "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
0.093ssl_client_verify "NONE" ssl_client_s_dn "-" ssl_client_i_dn "-" ssl_client_serial "-"
```

Notes

- The logging for the endpoints `/file/sync`, `/file/batch`, `^/(file|process)` will **no longer output to the official** `nginx.log` in MetaDefender Core. Instead, logs will be written to the new log file.
- It is advised to place the new log file (**nginx_mtls.log**) in the same folder as `nginx.log`.
  - This ensures it will be included in the **support package**.
  - The log filename **must start with** `nginx` (e.g., `nginx_mtls.log`).

Limitations

- This is a **workaround, not an official solution**. OPSWAT does not provide product support for this customization.
- You will need to maintain the settings manually after upgrades.
- Log rotation is **not supported** for the new log file.
  - The file will continuously grow in size.
  - You must implement your own rotation/cleanup process if needed.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

How to configure Role Mapping per groups in SSO Integration with Entra ID

Overview

When integrating **MetaDefender Core** with **Microsoft Entra ID** for Single Sign-On (SSO) authentication over SAML, user roles can be mapped in two primary ways:

1. Assigning a default role to all users
2. Configuring custom role mappings to support multiple roles across various user groups

This guide demonstrates how to assign specific MetaDefender Core roles to Entra ID groups.

Prerequisites

Before proceeding, ensure the following configurations are in place:

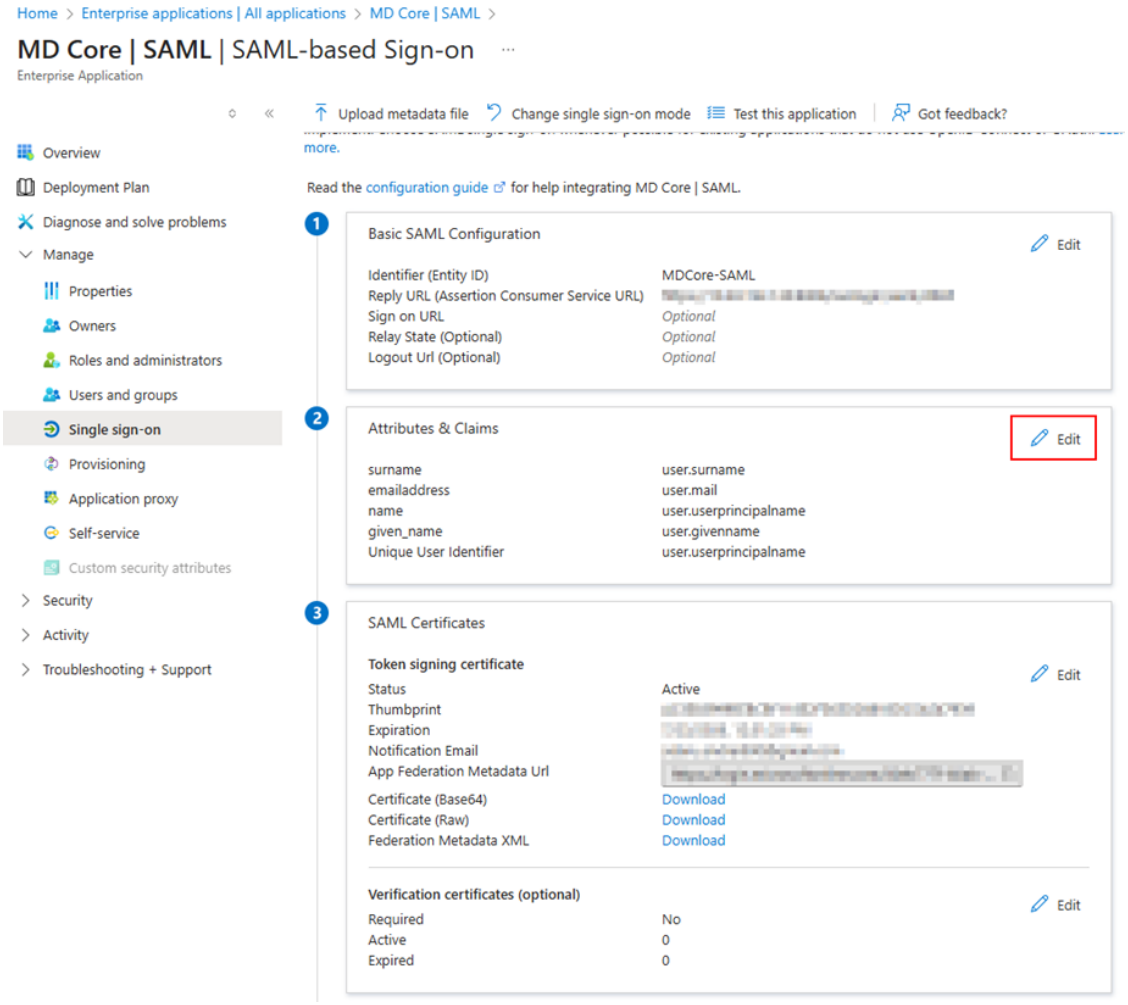
- Entra ID user-group mappings are already configured
- Groups are assigned to the SSO application
- MetaDefender Core SAML integration is completed

Step-by-Step Instructions

Step 1: Add a Group Claim in the SSO Application

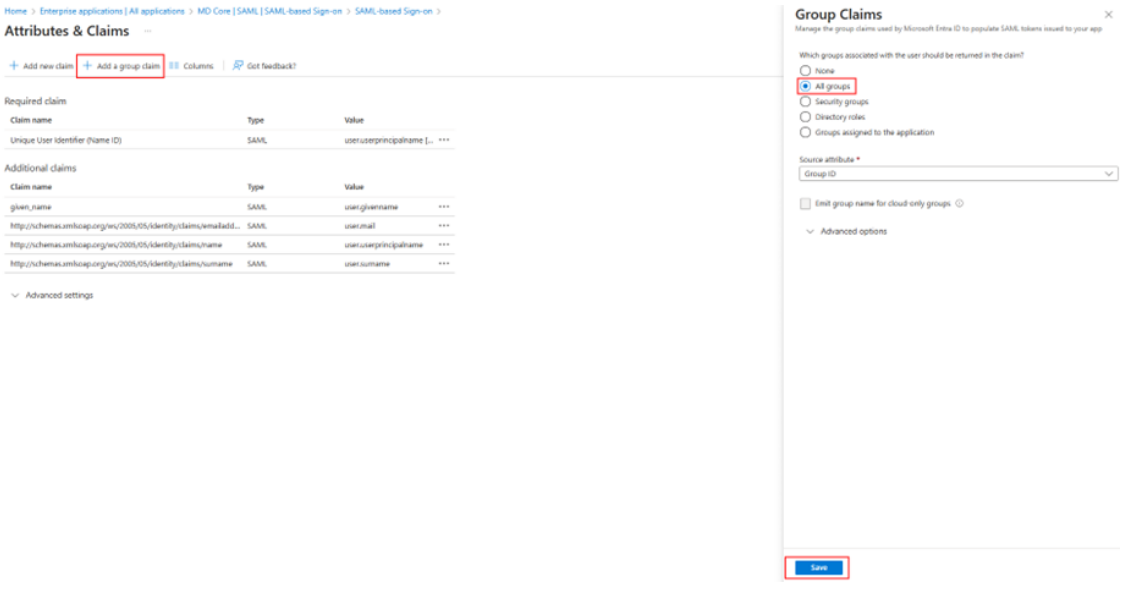
Go to:

SAML Application Single sign-on Attributes & Claims Edit

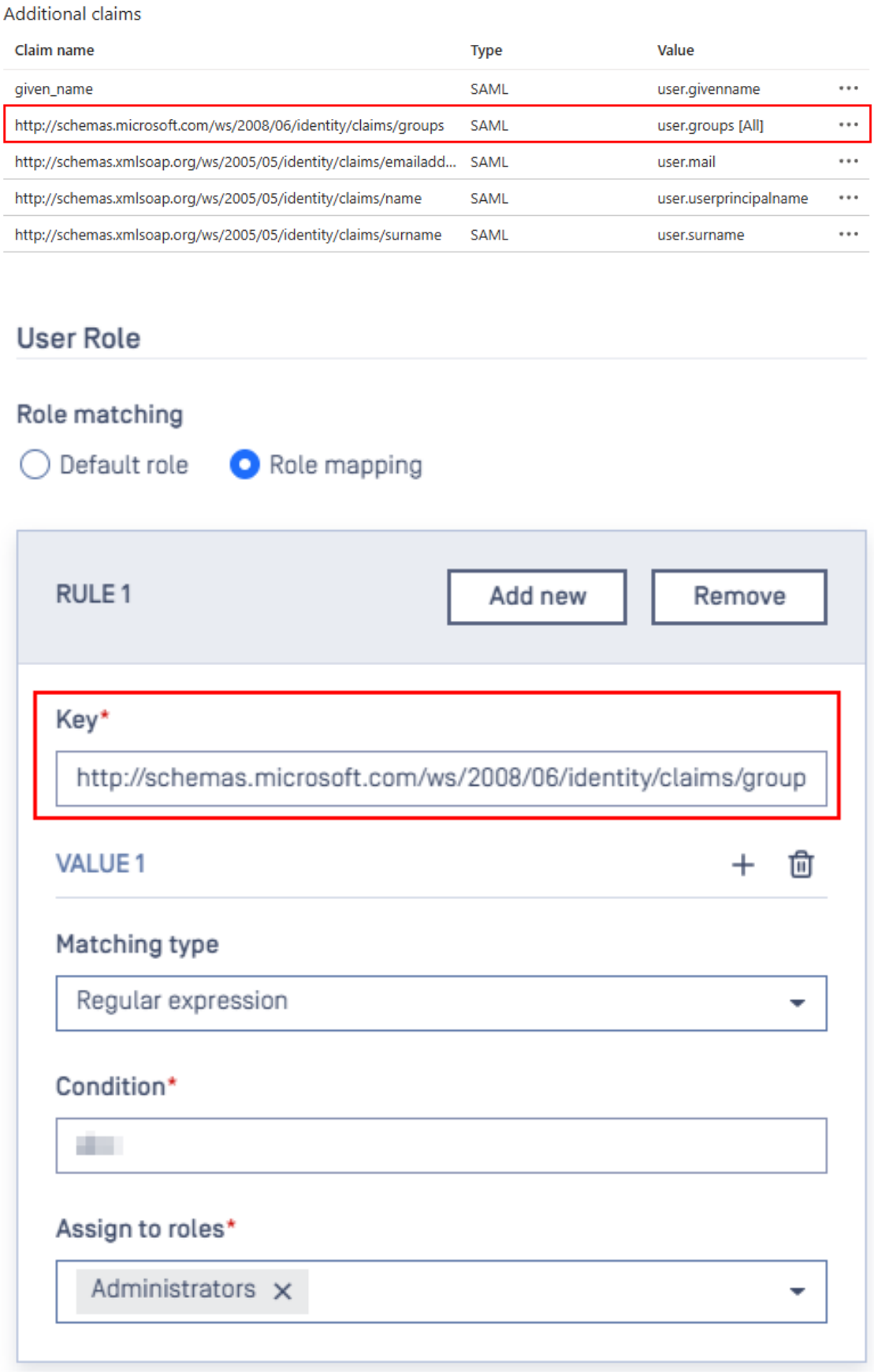


Then:

1. Click **Add a group claim**
2. Select **All groups**
3. Click **Save**

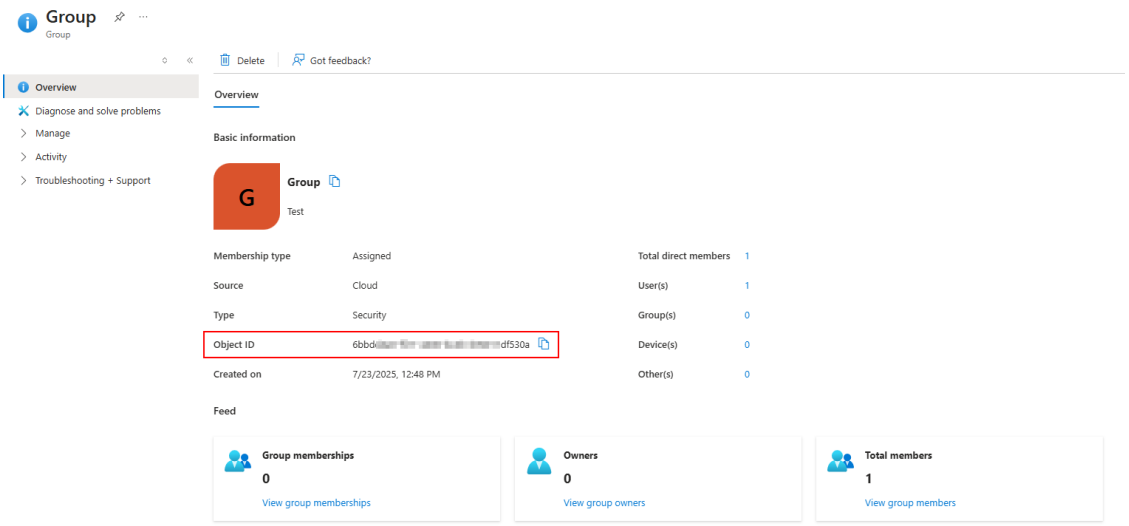


This creates the `user.groups [All]` claim, which provides a **Claim URL**. This URL will be used as the **Key** in the MetaDefender Core configuration.



Step 2: Retrieve the Group's Object ID

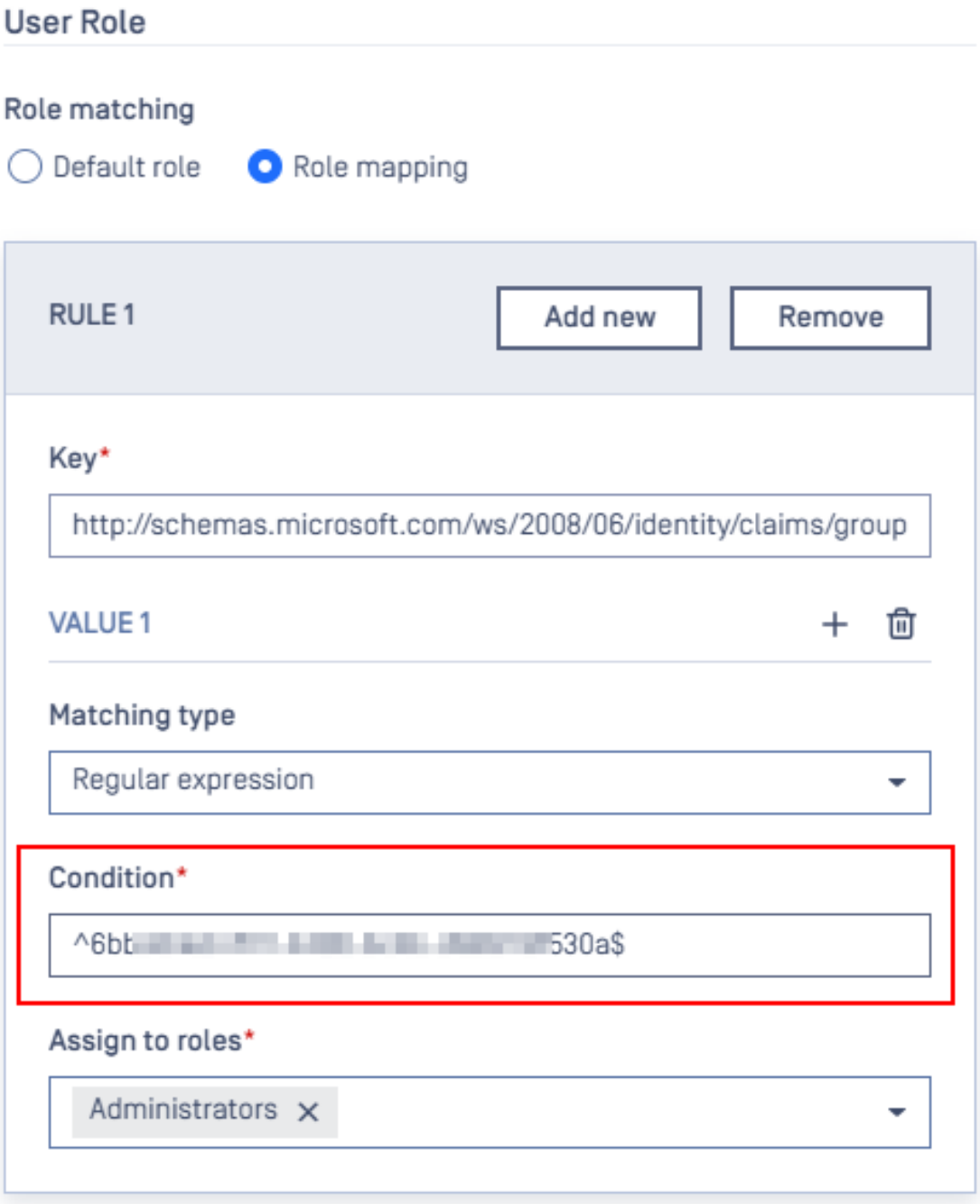
Return to the configured SAML application and select one of the assigned groups to view its details. This displays the group's **Object ID**, which is needed to map roles in MetaDefender Core.



Use the following format in the **Condition** field to match the group by its Object ID:

Code · html

```
^object_id$
```



This pattern ensures only users from that specific group are matched.

Role Assignment Example

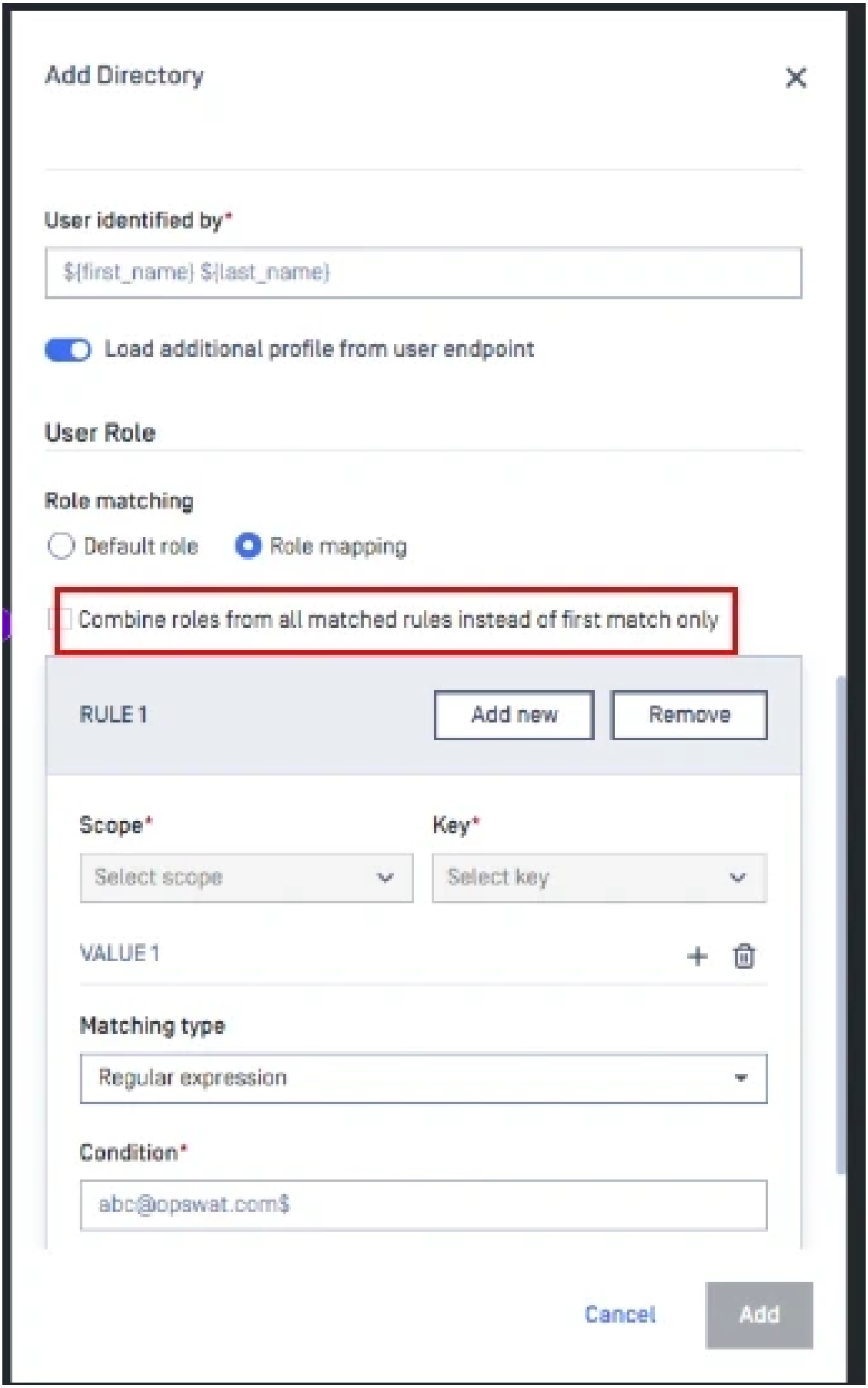
If a user belongs to multiple groups and multiple role-mapping rules are configured:

- By default, MetaDefender Core applies **only the first matching rule**
- If **"Combine roles from all matched rules instead of first match only"** is enabled, **all matching roles will be assigned to the user**

Combine Roles from Multiple Matching Rules

How to enable

1. Go to: **Configuration Role Matching Role Mapping**
2. Enable: **Combine roles from all matched rules instead of first match only**
3. Click **Save**


Add Directory
User identified by*
${first_name} ${last_name}
Load additional profile from user endpoint
User Role
Role matching
Default role
Role mapping
Combine roles from all matched rules instead of first match only
RULE 1
Add new
Remove
Scope*
Key*
Select scope
Select key
VALUE 1
Matching type
Regular expression
Condition*
abc@opswat.com$
Cancel
Add

Example

- Group A Administrator role
- Group B Read-only role

If a user belongs to both groups:

- **Disabled:** User gets *Administrator only* (first match)
- **Enabled:** User gets *Administrator + Read-only*

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

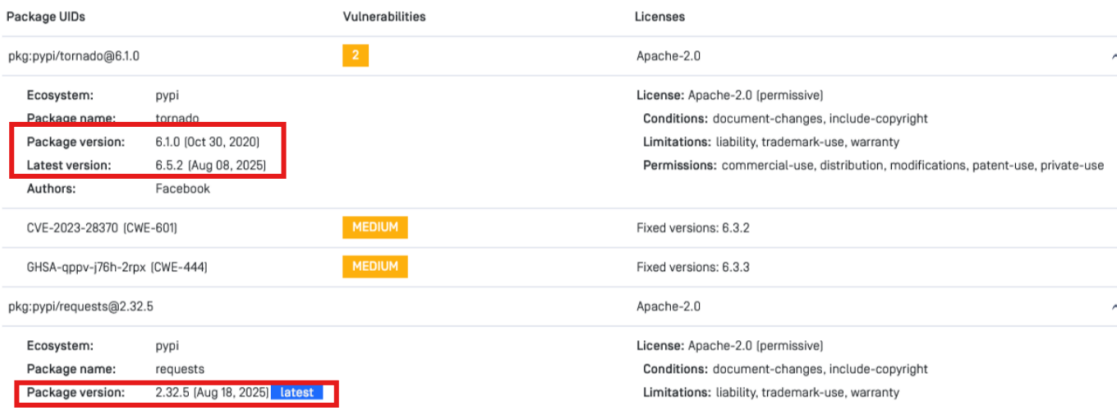
Does MetaDefender Core Detect EOS/EOL Status for Open-Source Software?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

MetaDefender Core does not directly detect whether an open-source library or software package has reached EOS/EOL. This is mainly because most open-source projects do not explicitly provide EOS/EOL information. In many cases, the absence of new releases for a long period may suggest that the project is no longer actively maintained.

However, MetaDefender Core can still help address this concern. The SBOM (Software Bill of Materials) feature provides visibility into both the current package version and the latest available version.



You can use this information to:

- Compare the current version in use with the latest version available to determine how outdated it may be.
- Assess whether they are running the latest version (which reduces operational risk).
- Identify potential risks if their version is significantly behind the latest release.

This allows you to make informed decisions about whether the software they are using may pose an operational risk

due to being outdated or unsupported.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

How Does MetaDefender Core Rotate PostgreSQL Logs?

Check Your Version:

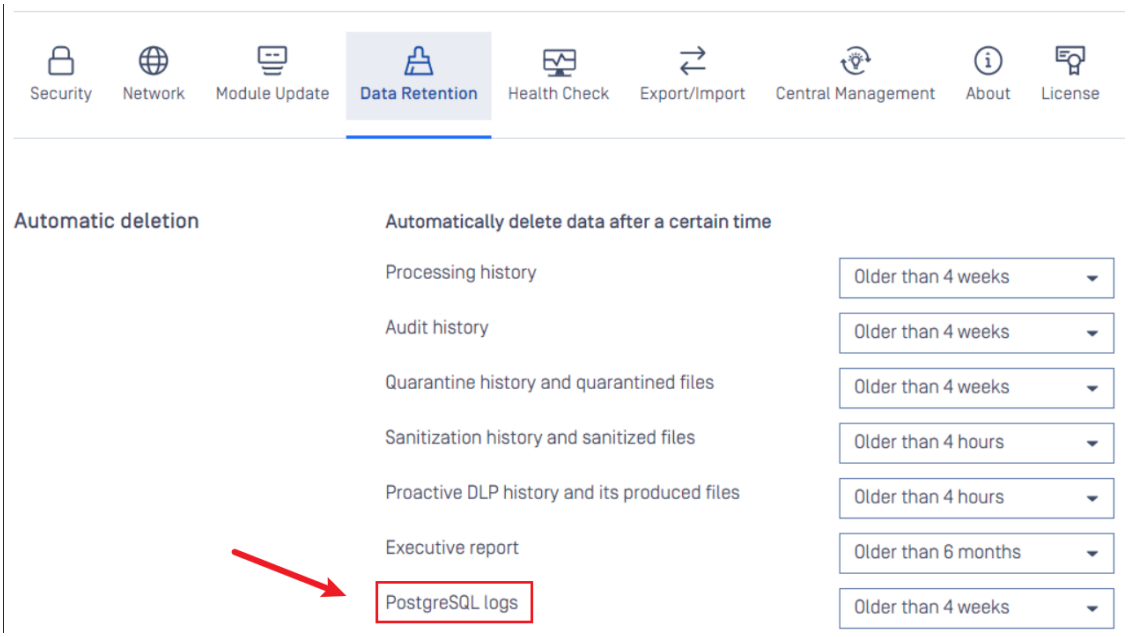
This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

MetaDefender Core uses PostgreSQL default settings for log rotation. By default:

- Logs are rotated **after 1 day** or when the PostgreSQL log reaches **10MB**.
- Restarting the `ometascan-postgresql` service will also rotate logs.

MetaDefender Core includes a Data Retention mechanism to retain PostgreSQL logs.



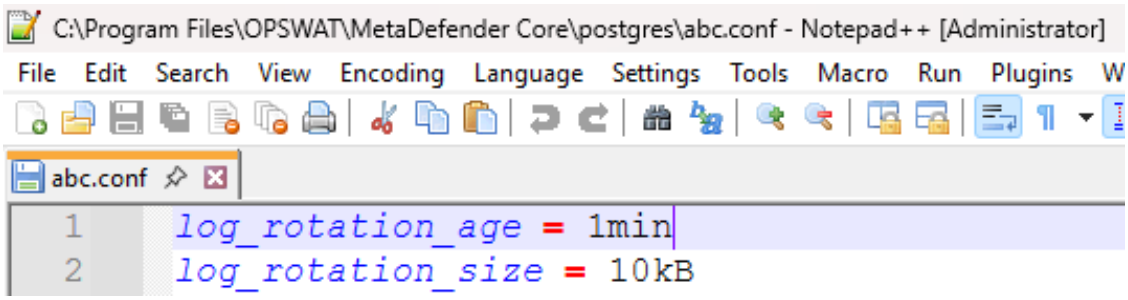
For more information, please refer to this article: Data Retention

Configuration

You can configure log rotation settings by creating a `*.conf` file in the `<MetaDefender Core>\postgres` folder if you are using the bundled database.

Configuration Example

- `log_rotation_age = 1min`
- `log_rotation_size = 10kB`



Please note that the configuration and example provided apply **only to the bundled local PostgreSQL database**. If you are using a **remote database**, you will need to configure log rotation directly on that remote PostgreSQL instance.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why Must Each MetaDefender Core Installation Have a Unique Instance Name?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

The instance name is a key configuration parameter in MetaDefender Core that identifies each installation uniquely — especially in environments where multiple Core instances share the same database.

This article explains what the instance name is, how it is used, and why it must be unique for each Core installation.

What Is the Instance Name and When Is It Used?

The instance name is a unique identifier assigned to each MetaDefender Core installation. It is primarily used in shared database environments where multiple MetaDefender Core servers operate together, typically for scalability or redundancy.

When multiple Core installations share a single database:

- The instance name distinguishes between individual Core servers.
- It allows each instance to maintain its own local configuration settings, such as:
  - Application configuration (e.g., queue size, parallel count, etc)

  - Module configuration (e.g., enabled engines, module parameters)
- Meanwhile, global components such as workflows and user accounts are shared across all instances in the shared database.

Why the Instance Name Must Be Unique

Each MetaDefender Core instance must have a unique instance name when sharing the same database. If two or more Core installations use the same instance name:

- Configuration data between those instances will overlap.
- Changes made in one installation may overwrite or conflict with those in another.
- This can result in abnormal or unstable functionality, including:
  - Module misconfiguration
  - Database connection issues

In short, duplicate instance names in a shared database environment are not supported and will cause operational conflicts.

Support:

If further assistance is required, please proceed to log a support case or chatting with our support engineer.

Why is my Excel document not showing hidden sheets after Deep CDR sanitization?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

When processing Excel documents with OPSWAT MetaDefender Core Deep CDR (Content Disarm and Reconstruction), users may notice that **hidden sheets are missing** from the sanitized output. This behavior is expected and depends on both the file format and the **Deep CDR configuration** settings.

Cause

The Deep CDR engine currently supports the removal of hidden sheets only for Excel files in the .XLS format (legacy Excel format). During sanitization, hidden sheets in .XLS files are detected as embedded objects. When the configuration setting remove_embedded_object is enabled in the Excel section of the Deep CDR policy, these hidden sheets are automatically removed as part of the sanitization process.

This is done to ensure that no potentially malicious or hidden content remains in the file.

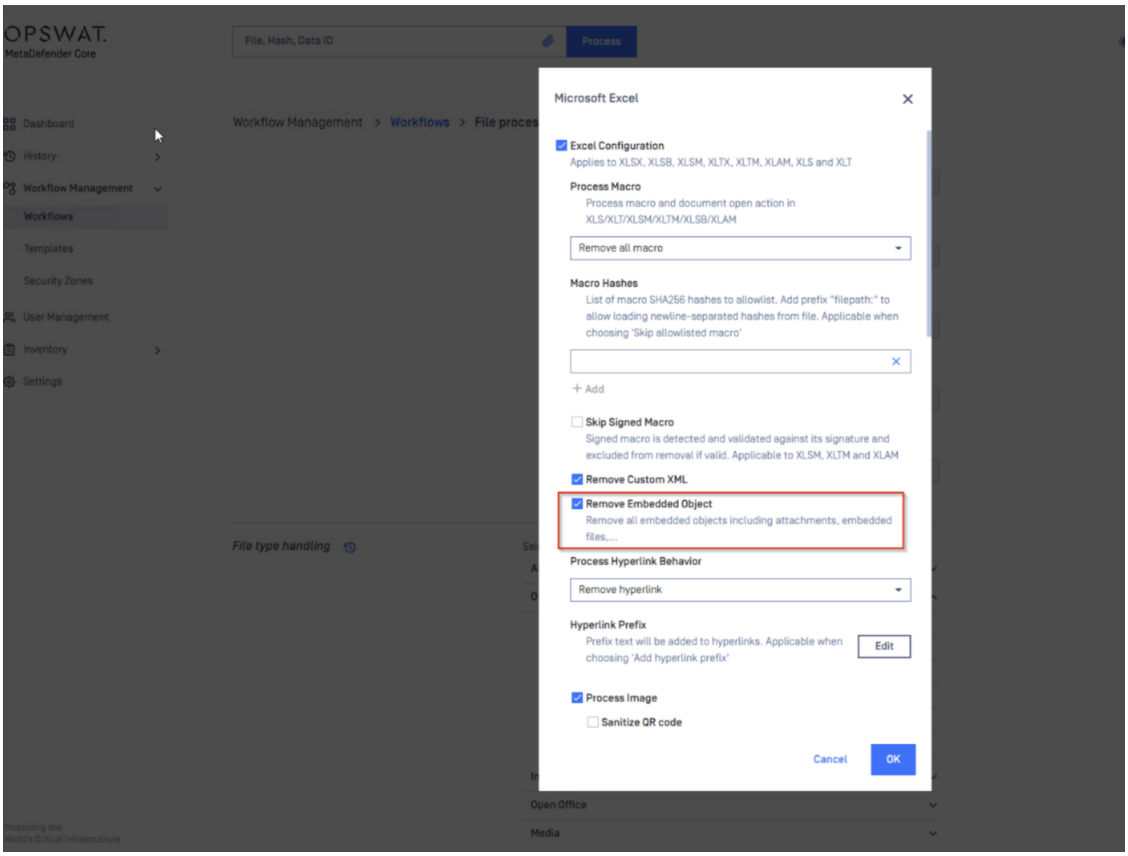
Example

For example, if an .XLS file contains a hidden sheet, and the Deep CDR configuration has the default remove_embedded_object setting enabled, the hidden sheet will be removed from the sanitized file.

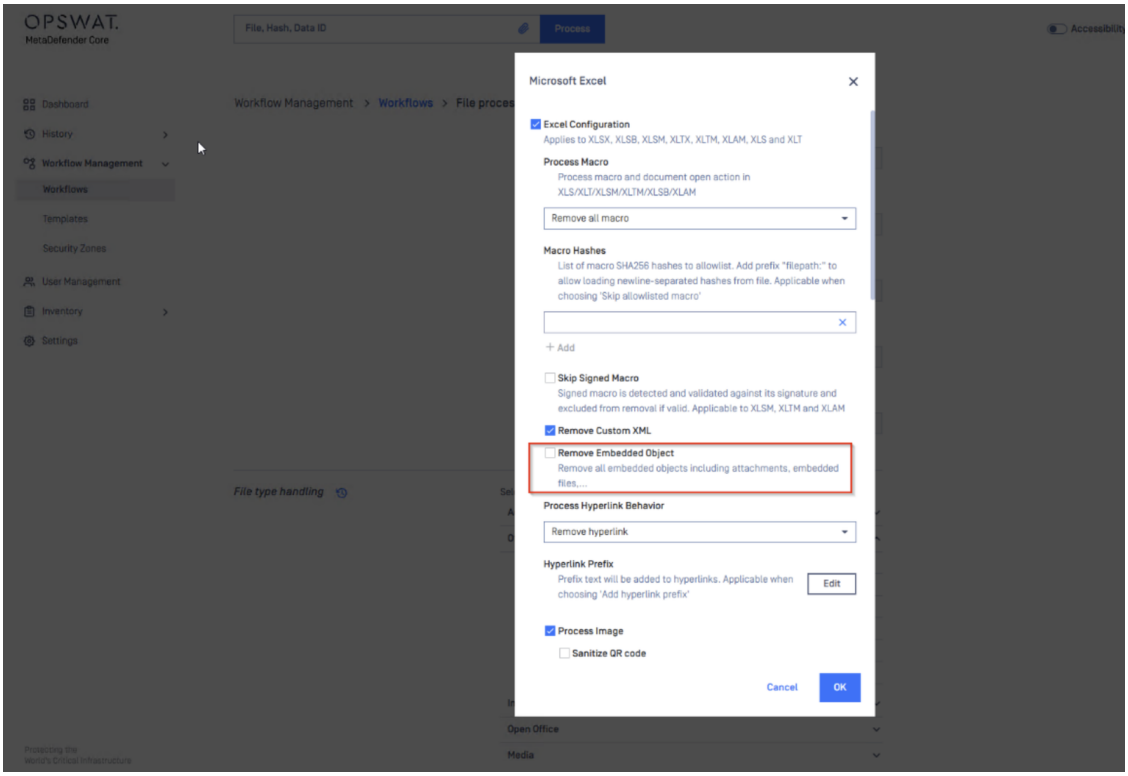
How to Keep Hidden Sheets

If you need to preserve hidden sheets during sanitization, you can modify the Deep CDR configuration:

- Open your Deep CDR configuration file or navigate to the Excel section in your MetaDefender Core policy settings.
- Locate the following setting: “"remove_embedded_object": true”



- Change it to: “"remove_embedded_object": false”



- Save the configuration and reprocess the file.

With this change, the Deep CDR engine will no longer remove hidden sheets, and they will be preserved in the sanitized output.

Important Notes

- This behavior applies only to **.XLS** files.
- For modern Excel formats such as .XLSX, hidden sheets are not currently modified or removed by the Deep CDR engine.
- Always ensure that disabling embedded object removal aligns with your organization's security policy, as it may increase the risk of allowing hidden or embedded malicious content.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

How to Limit the Number of CPU Cores Used for Scanning in MetaDefender Core

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

In some environments, administrators may observe that **MetaDefender Core's scanning processes consume 100% of CPU resources**, even when the number of concurrent scans is below 200. When CPU usage remains at maximum capacity, the **Core API may become unresponsive**, affecting health checks and subsequent file scanning requests.

This article explains how MetaDefender Core manages CPU utilization and provides **tuning recommendations** to maintain system responsiveness.

Can CPU Usage Be Limited Directly?

Currently, **MetaDefender Core does not provide an option to directly limit CPU usage** for scanning processes. The scanning engines are designed to utilize available CPU resources as efficiently as possible, while CPU allocation is handled by the **operating system scheduler**.

That said, there are several tuning methods that can help reduce CPU contention and keep the API and queue management processes responsive.

Recommended Tuning Methods

1. Adjust Scanning Concurrency

Modify the `parallelcount` parameter in the engine configuration to control how many threads each engine uses concurrently.

- Lower values reduce CPU load but may also lower scanning throughput.
- The optimal value depends on your system's hardware and workload characteristics.

> ***Example:*** *Decreasing `parallelcount` from 20 to 15 can significantly lower CPU utilization during high-load periods.*

2. Reduce Workload Complexity

Simplify scan workloads to reduce CPU consumption:

- Limit **archive extraction depth**.
- Define **maximum file size limits** for scanning.
- Avoid enabling **deep archive scanning** unless absolutely necessary.

3. Optimize Infrastructure and Configuration

Ensure MetaDefender Core runs on an optimized environment:

- Deploy on hardware with **sufficient CPU cores and memory** beyond the baseline recommendation.
- Enable **high-performance power profiles** and verify that all CPU cores are active.
- Keep MetaDefender Core and all scanning engines **up to date** to take advantage of performance improvements.

Refer to this support article for more detailed information: MetaDefender Core Performance Tuning Guide

4. Scale Horizontally

In high-load environments, distribute scanning tasks across multiple MetaDefender Core instances to ensure

optimal performance and scalability.

This approach reduces per-instance CPU usage and improves overall throughput. You can achieve horizontal scaling by:

- Deploying a **load balancer** or other task distribution mechanism to efficiently route scanning requests across instances.
- Utilizing **MetaDefender Distributed Cluster** for centralized management, job scheduling, and load distribution across multiple Core nodes.

For more detailed configuration guidance, refer to Using external Load Balance and MetaDefender Distributed Cluster documentation.

Verify the Change

After applying the tuning recommendations, you can verify the improvements by:

1. **Monitoring CPU utilization** on the MetaDefender Core host via your OS tools (e.g., Task Manager, `top`, `htop`, or system metrics).
2. **Observing API responsiveness** by performing health check requests (`/metadefender/status`) and test file scan submissions.
3. **Observing API responsiveness** by
    1. **Performing health check requests** using `GET /readyz` to verify that the service is up and responding.
    2. **Reviewing the** `nginx.log` file located in your Core log directory. This log records every API request along with its total processing time. Each log entry includes details such as the timestamp, request type, response status, and processing duration.

Example log entry:

```
[dd/MMM/yyyy:HH:mm:ss +0800] "GET /api/scan HTTP/1.1" 200 512 "-" "Java/17.0.10" 0.245
```

In this example, the value `0.245` at the end indicates that the request took **0.245 seconds** to complete.

4. **Comparing throughput and response time** before and after tuning using the same workload.
5. If CPU usage stabilizes and APIs respond consistently, the tuning changes are effective.

Summary

While MetaDefender Core does not support direct CPU throttling for scanning processes, administrators can achieve better performance stability by **tuning concurrency, simplifying workloads, optimizing system configuration, and scaling horizontally**.

Support:

If you continue to experience issue with the upgrade and have followed the instructions above, **please create a support case or chat with our support engineer for further assistance.**

What Protocols Does MetaDefender Core Use for Active Directory Integration?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows/Linux systems.

Overview

NTLM:

- Not supported.
- The application does not rely on NTLM hashes for authentication.

Steps to Configure

Refer to the official documentation for detailed configuration steps: User directories

Verify the Change

Refer to the official documentation for verification steps: User directories

Troubleshooting

If you have queries, concerns, or issues regarding Metadefender Core AD integration methods, please open a Support Case with the OPSWAT team via phone, online chat, or form, or feel free to ask the community on our OPSWAT Expert Forum.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why Is the Time Range on MetaDefender Core's Executive Report Different from the Time Range Selected Before Exporting?

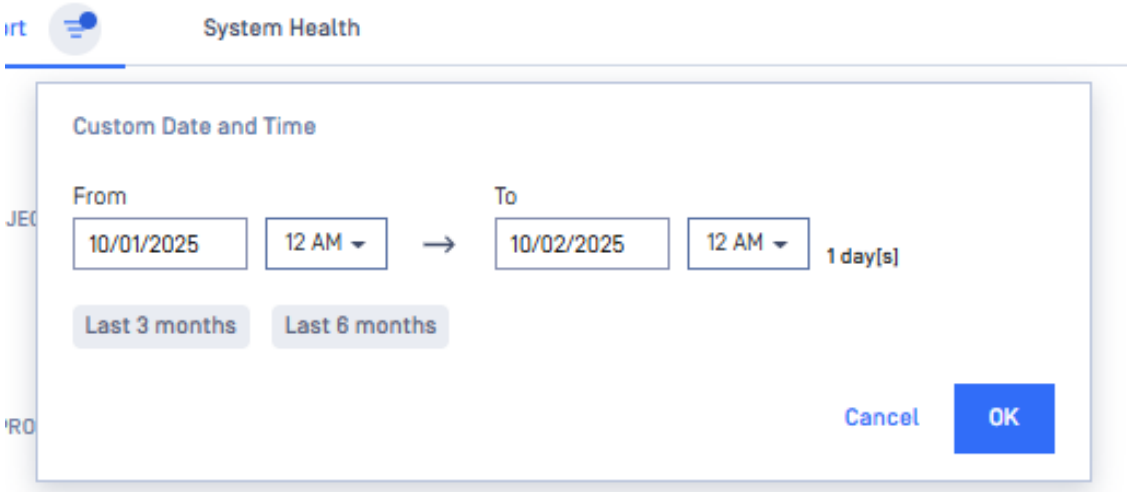
Check Your Version:

This article applies to all MetaDefender Core versions.

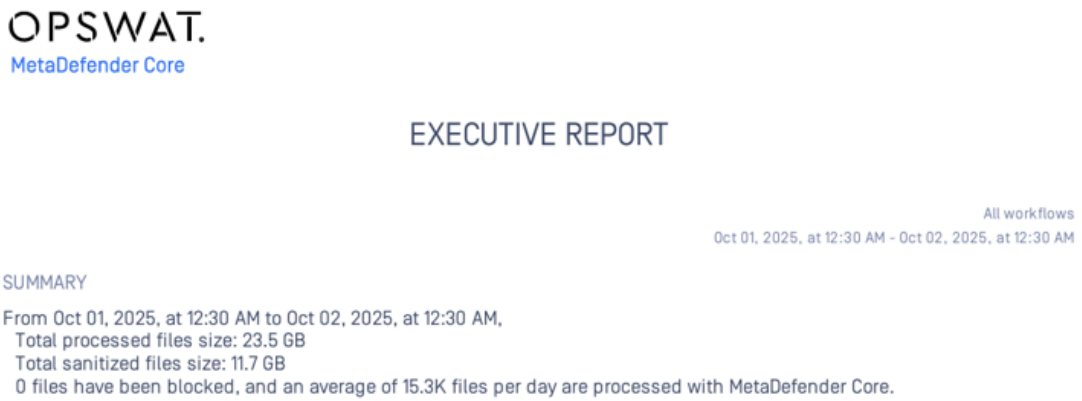
Users may notice a time difference between the selected time range in the MetaDefender Core UI and the time

displayed in the exported Executive Report (PDF). This behavior is caused by a difference between the local time zone of the user's browser and the system time zone of the MetaDefender Core server.

- The MetaDefender Core UI displays time based on the local system time zone of the web browser used to access MetaDefender Core.
  - For example, if your browser's system time is UTC +07:00, and you select a time range from 12:00 AM to 12:00 AM (one day), the UI will display data accordingly.



- When exporting the report to PDF, MetaDefender Core applies the server's system time zone.
  - For example, if the Core server time zone is UTC +07:30, the exported report will show 12:30 AM instead of 12:00 AM, resulting in a 30-minute difference.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What are the AI-Powered Components in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

MetaDefender Core includes several components that leverage artificial intelligence (AI) and machine learning technologies to enhance detection accuracy, contextual understanding, and adaptive threat analysis. This article describes which components have AI-powered features and how they contribute to overall threat prevention.

1. Metascan (Anti-Malware Engines)

Depending on the antivirus (AV) engines licensed and enabled in your environment, MetaDefender Core can utilize AI-driven detection models provided by certain vendors. These models are used to detect previously unseen or polymorphic malware based on behavior or heuristic patterns rather than static signatures.

For details on which AV engines include AI capabilities, refer to the official documentation below: MetaDefender Core – Anti-Malware Vendors

2. File Type Engine

The File Type Engine can apply AI-enhanced text-based detection models to strengthen file classification accuracy — for example, when analyzing documents or scripts.

This AI-based enhancement is **disabled by default** and can be enabled for environments requiring higher precision in text content recognition.

3. Proactive DLP

The Proactive Data Loss Prevention (DLP) component applies AI models to identify and classify sensitive or inappropriate content in documents. Key AI-assisted capabilities include:

- **Document identification** (e.g., business, personal, confidential)
- **Detection of "not suitable for work" content**
- **PII (Personally Identifiable Information) detection**

These AI models improve detection accuracy and context awareness beyond traditional keyword or rule-based approaches.

4. Adaptive Sandbox

The Adaptive Sandbox uses AI-driven analysis to interpret behavioural patterns and generate enriched threat intelligence. Examples of AI-powered features include:

- **Executive Summary Generation** – Automatically produces a human-readable summary of observed behavior.
- **Phishing Detection** – Identifies suspicious document or email patterns using machine-learning classifiers.
- **Remote Sandbox Integration** – Some advanced AI features are available only when connected to a standalone **remote sandbox** environment.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to remediate significantly increased scanning times during peak hours due to engine updates failing?

Check Your Version:

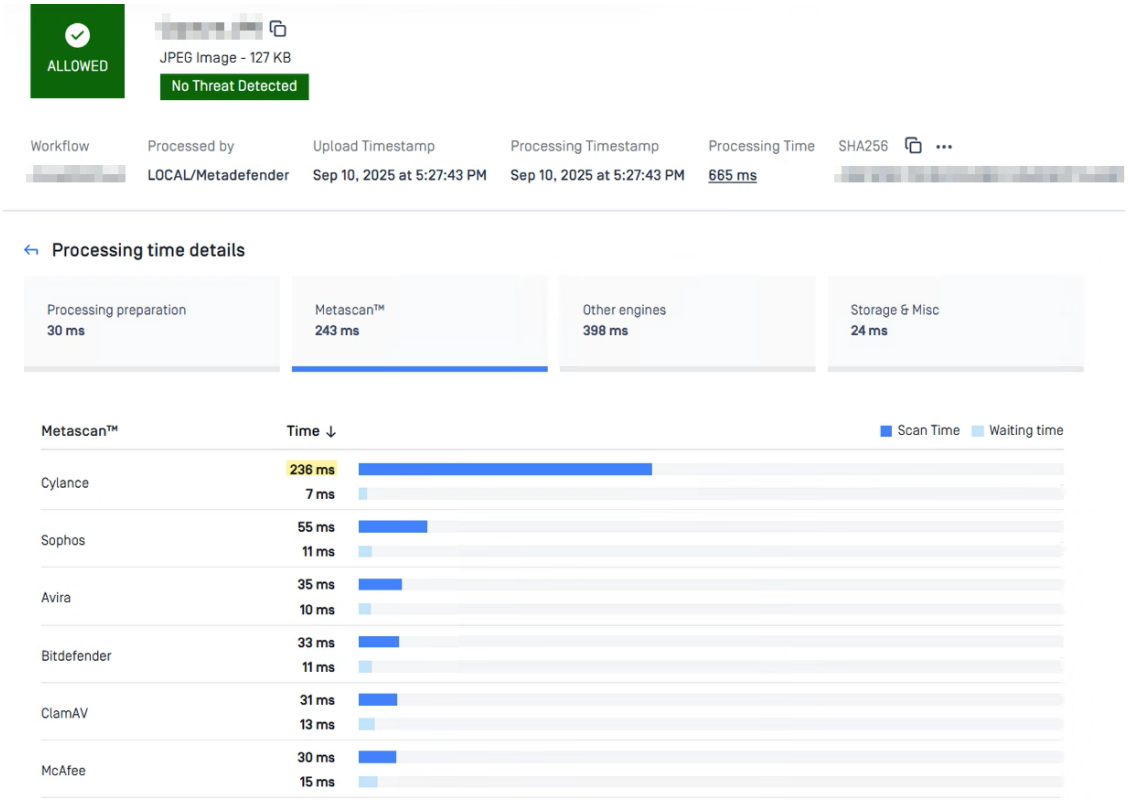
This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue:

In some cases, automatic engine updates may fail to deploy, causing the engine to be temporarily unavailable. During this time, on-going scans will wait for the response from the failed engine, until it reaches the per AV engine timeout for that engine, or until the engine recovers (whichever comes first).

If this happens during peak hours, it can result in a significant increase in file scanning times, even after the engine recovers.

Example: The Cylance engine is the main contributor to the delay (Cylance ~236ms scan time). Logs show Cylance engine occasionally crashes during updates, causing timeouts and delays at peak hours. This slow processing time can impact production.



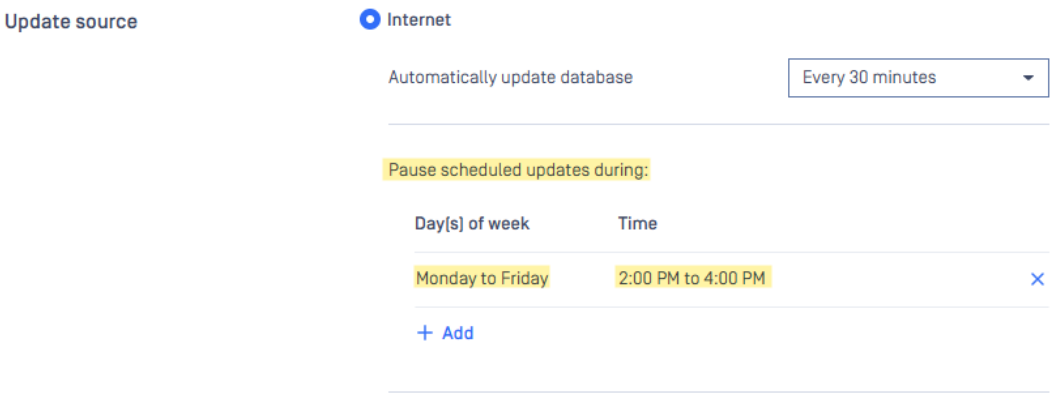
Resolution:

To avoid such failures from impacting production, we can adjust the engine automatic update schedule to **pause updates during peak hours** to prevent crashes.

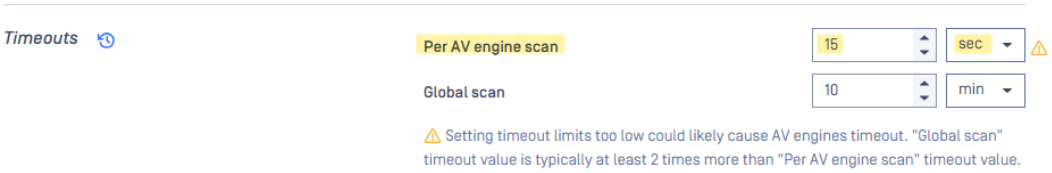
Additionally, lowering the Per AV engine timeout, which is a **fail-fast approach**, can avoid lengthy wait times.

Steps:

- Navigate to: `Settings > Module update`
- `Pause scheduled updates during:` Add your expected peak hours here



- Navigate to: `Workflow > Metascan > Timeouts`
- `Per AV engine scan:` change to about 15-30 seconds (default is 1 minute)



Info

Lowering the Per AV engine timeout should only be used when the machine specs are not sufficient, and you prioritize speed over a comprehensive result from most licensed engines.

Verify the change:

Monitor scan times and engine status during peak hours to ensure the configuration is working as expected and doesn't cause unwanted AV engine timeouts.

For more information regarding operational impact during engine updates, please refer to this knowledge base: Will MetaDefender Core Experience Downtime During Engine Updates?

Info

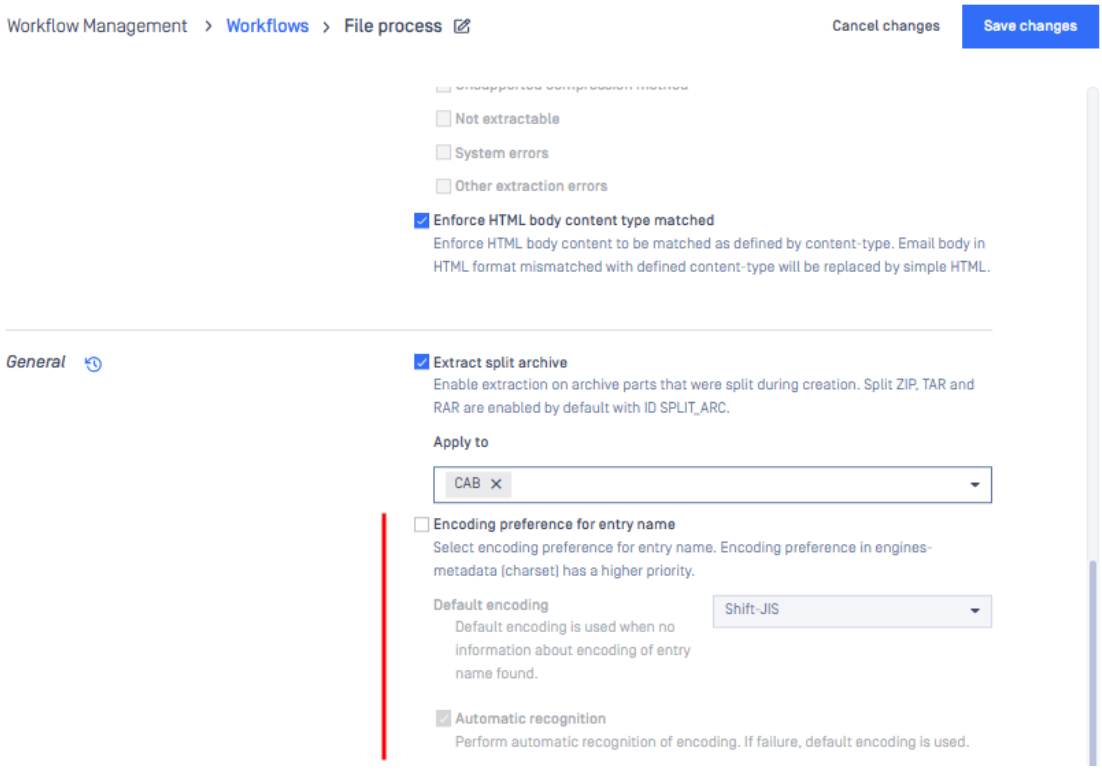
If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

What More to Know About Encoding Preference for Entry Name?

Check Your Version:

This article applies to Archive Engine version 6.2 and later, which requires MetaDefender Core version 5.13.1 or higher to function properly.

- *Default Behavior When Encoding Preference is Disabled** If "Encoding preference for entry name" is disabled, the engine will use UTF-8 as the default character encoding when extracting file names. (Release Notes: MetaDefender Core 5.4.1)



- **Environment-Dependent Behavior** The files being extracted might have been created in an environment that uses a different character encoding from the one used by the MetaDefender Core. As a result, the Archive Extraction engine might not be able to correctly interpret or display filenames that use system-specific encodings (such as Shift-JIS) during extraction. In these cases, users should enable the "Encoding preference for entry name" setting and set the "Default encoding" option to match the character set used in the original environment or enabling "Automatic recognition." This will help ensure that filenames are extracted and shown correctly.
- **Garbled Characters on Different Systems** In certain cases, after sanitization, the Japanese filenames are stored in UTF-8, and they appear correctly on systems that use UTF-8. When these files are opened in a Japanese environment, the names may appear garbled due to the difference between the environment's expected character set (Shift-JIS) and the actual character set used (UTF-8). If this occurs, customers are encouraged to open a support ticket for further investigation.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Is MetaDefender Core Fully SELinux Compliant?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on RHEL 9.

Issue:

MetaDefender Core is not currently SELinux compliant, although it functions on systems where SELinux is enabled. It does not adhere to full SELinux confinement policies or provide officially supported SELinux policy modules.

Resolution:

At this time, OPSWAT does not provide or recommend SELinux confinement policies for MetaDefender Core. For full functionality, Core services should remain unconfined.

Workarounds:

Most Linux customers operate MetaDefender Core on RHEL systems with SELinux enabled without any issues, even when services run unconfined.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why Does MetaDefender Core Require Removing the SSL Private Key Passphrase for PostgreSQL?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

When enabling SSL for PostgreSQL, the documentation instructs removing the passphrase from the private key (`server.key`). This can raise concerns, as passphrases are usually seen as a security measure.

Why It's Required

PostgreSQL runs as a background service and starts automatically on system reboot. It cannot prompt for a passphrase during startup. If the key is still protected, the database fails to start, preventing MetaDefender Core

from running.

Removing the passphrase allows the key to be loaded automatically, ensuring reliable, hands-off operation.

Can the Passphrase Be Kept?

Technically yes, but it's impractical:

- You would need to manually start PostgreSQL and enter the passphrase after each reboot.
- Secure automation would require advanced solutions like hardware key management, which are not part of standard deployments.

How the Key Is Secured

Even without a passphrase:

- Run MetaDefender Core under a dedicated service account.
- Restrict file permissions so only that account and administrators can access `server.key`.

SSL encryption remains fully active; only the key access method changes.

Conclusion

Removing the passphrase is necessary for automatic startup and reliable operation. Security is maintained through OS-level access controls, while SSL encryption continues to protect database communication.

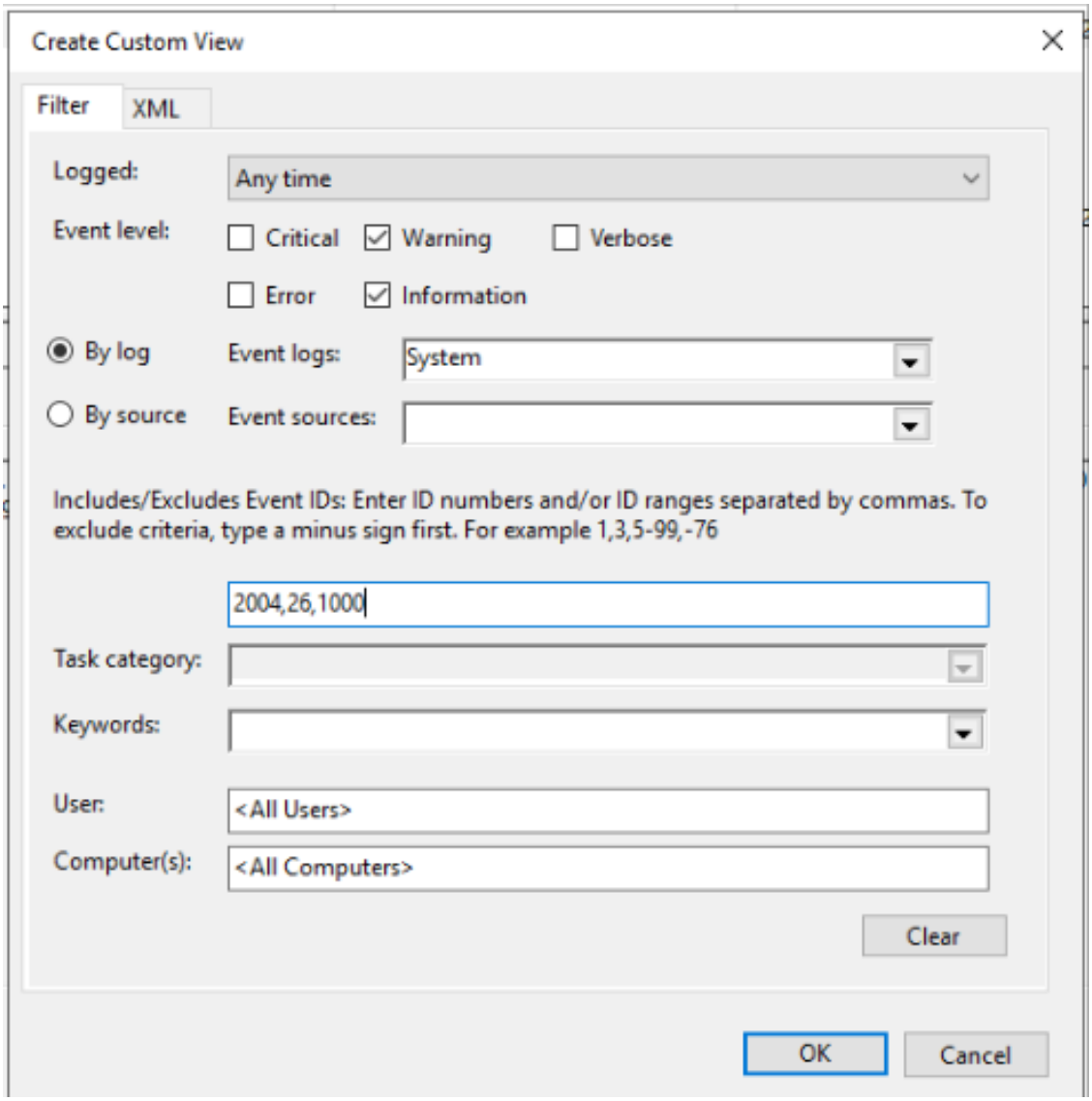
Support:

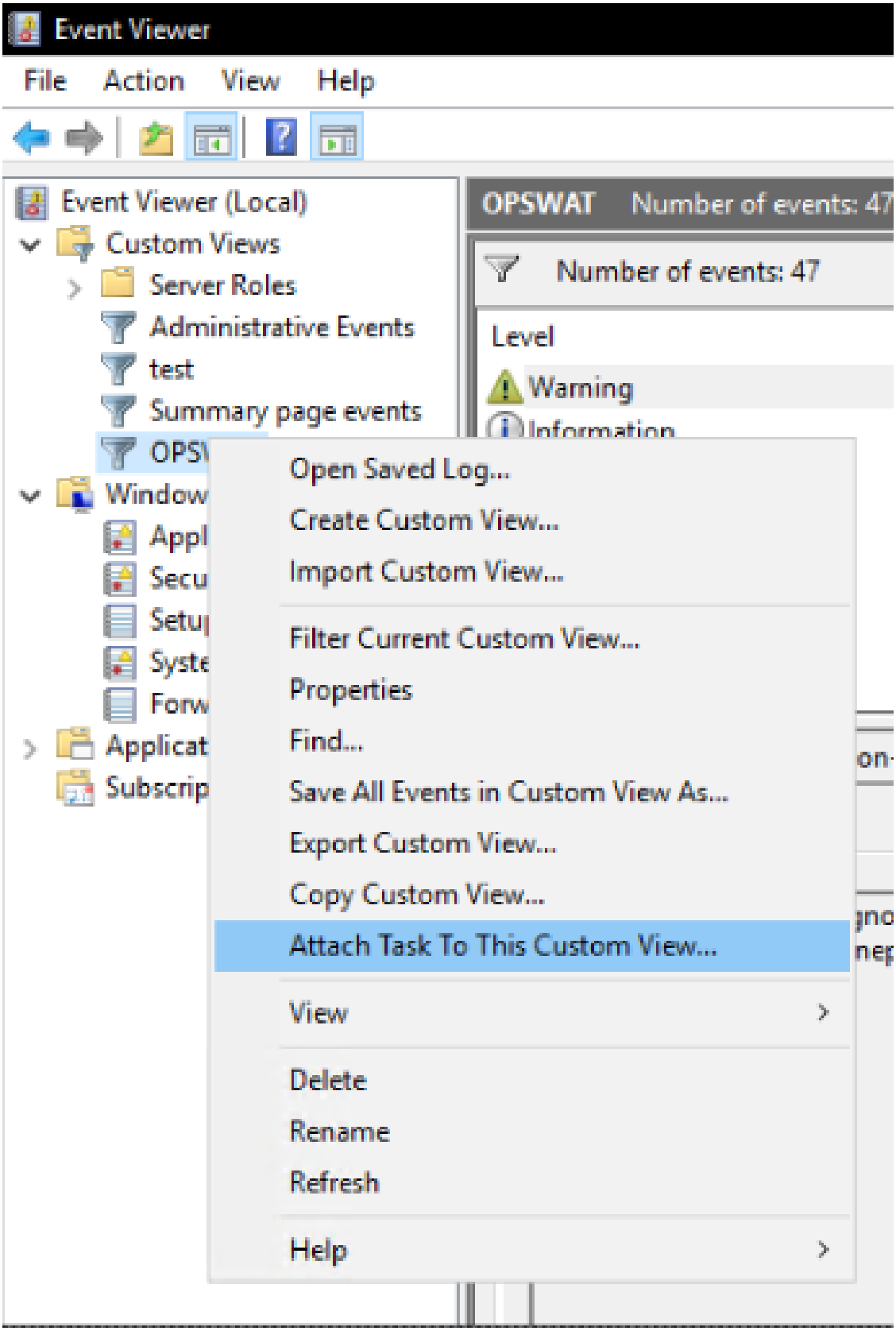
If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

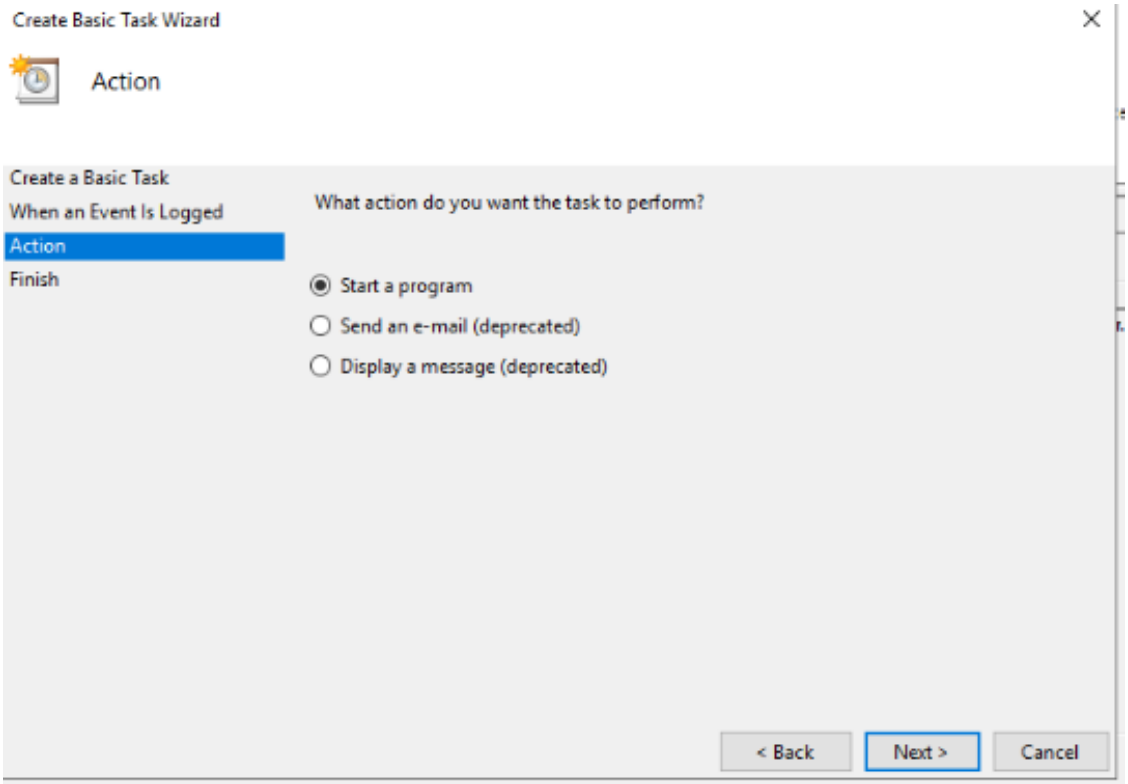
When to set an alarm and how?

1. In what situation should we define an alarm?

The scope of an alarm is to raise awareness when an issue appears, thus the alarm should be always set.


Create Custom View
Filter
XML
Logged:
Any time
Event level:
Critical
Warning
Verbose
Error
Information
By log
Event logs:
System
By source
Event sources:
Includes/Excludes Event IDs: Enter ID numbers and/or ID ranges separated by commas. To exclude criteria, type a minus sign first. For example 1,3,5-99,-76
2004,26,1000
Task category:
Keywords:
User:
<All Users>
Computer(s):
<All Computers>
Clear
OK
Cancel


Event Viewer
File
Action
View
Help
Event Viewer (Local)
Custom Views
Server Roles
Administrative Events
test
Summary page events
OPSWAT
Number of events: 47
Number of events: 47
Level
Warning
Open Saved Log...
Create Custom View...
Import Custom View...
Filter Current Custom View...
Properties
Find...
Save All Events in Custom View As...
Export Custom View...
Copy Custom View...
Attach Task To This Custom View...
View
Delete
Rename
Refresh
Help



A script can be triggered to follow a specific action.

2. When the event level is selected, which keywords should be specified to determine those related to OPSWAT?

The applications that might fail during such events are listed below:

For MD Core:

- pg_ctl.exe
- ometascan.exe
- Engineprocess.exe
- Engineprocess32.exe
- OMMSProcessor.exe
- postgres.exe
- nginx.exe
- clamd.exe

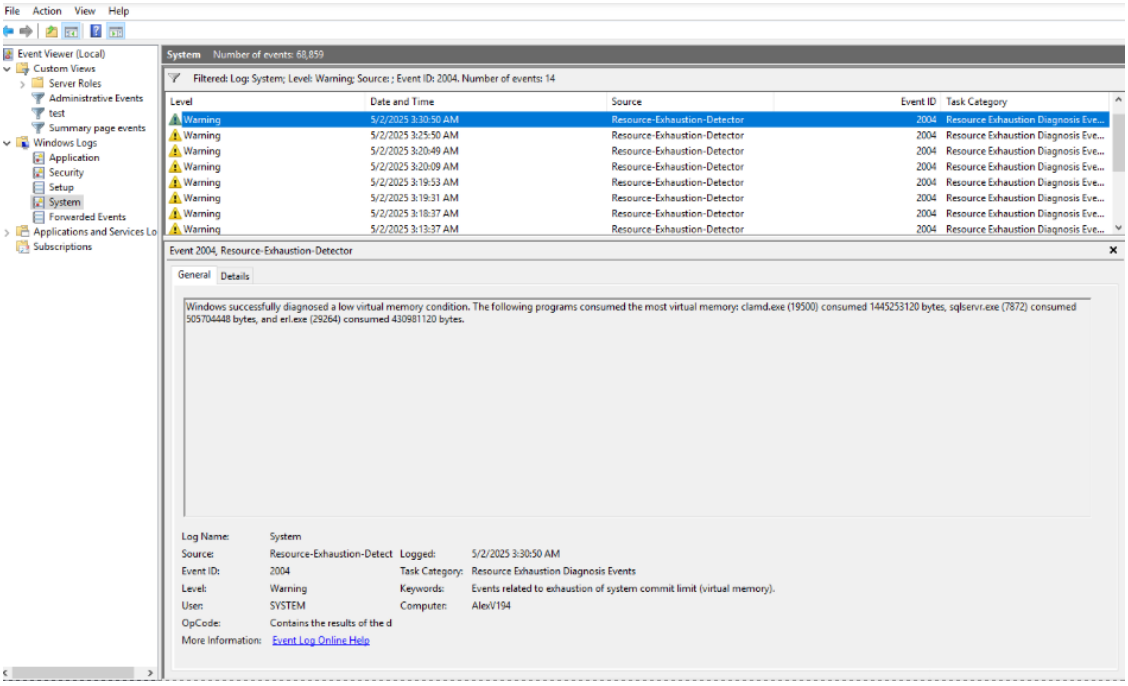
3. In which section does Event ID 2004 (Resource-Exhaustion-Detector) create an alarm? I couldn't find any examples.

The Event ID 2004 will be found in System, with a Warning level. This will provide the list with 3 applications which used most RAM during this event.

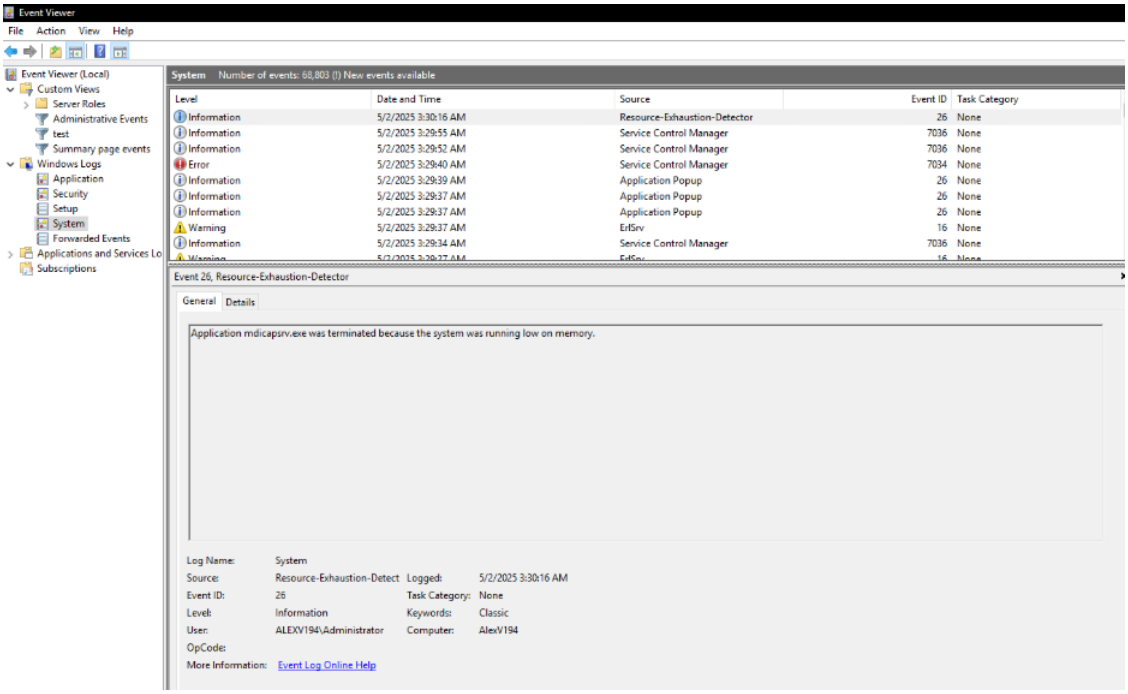
The event itself should trigger a warning or alarm as it states that there is a memory availability issue, and the system might become unstable. If the event also contains any of the mentioned applications at point 2, it means that OPSWAT applications require more resources than available.

4. There is no problem for Application error (Event ID 1000), I saw them in the event viewer and provided detailed information for their definition. I am especially expecting full details and examples from you for event id 2004 and 26.

Please find below some examples for event ID 2004:



Also, for the event ID 26:



Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why are batch operations slow in MetaDefender Core and how are they resolved in version 5.14.0?

Support Version:

This article applies to all MD Core prior to version 5.14.0

Issue:

Significant delays during batch operations, specifically when executing API calls such as:

- `POST /file/batch/<batch_id>/close`
- `GET /file`
- `POST /admin/batch/closeall`

PostgreSQL logs revealed that the function `scan.getFirstRcFile(BIGINT)` was executing repeatedly and taking longer time to query:

```
[11076] LOG: duration: 6552.132 ms execute : SELECT * FROM scan.getFirstRcFile($1::BIGINT); 2025-10-27 00:00:17.705 IST [9804] LOG: duration: 10619.755 ms execute : SELECT * FROM scan.getFirstRcFile($1::BIGINT);
```

These delays led to bottlenecks in the batch-processing workflow.

Root Cause:

In earlier versions, batch-scan requests triggered the `scan.getFirstRcFile(BIGINT)` query multiple times for each file within a batch. When processing large numbers of files, this resulted in cumulative delays and degraded overall performance.

Resolution:

The issue has been addressed in version 5.14.0

https://www.opswat.com/docs/mdcore/v5.14.0/release-notes

The Close Batch workflow has been refactored so that it no longer invokes `scan.getFirstRcFile(BIGINT)`. The `GET /file` API has been optimized to reduce query overhead and avoid repeated calls to this function.

As a result, PostgreSQL logs in version 5.14.0 no longer display long-running `scan.getFirstRcFile($1::BIGINT)` queries during the impacted APIs.

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

Why Are Japanese Filenames Garbled When Submitting Files to MetaDefender Core via API?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

When submitting files to MetaDefender Core via API, filenames containing Japanese characters may appear garbled in **History > Processing History**, even though the file itself is processed successfully.

Cause

For API-based submissions, MetaDefender Core derives the displayed filename from the `filename` and `filepath` values provided in the HTTP request headers. HTTP headers do not reliably support multibyte character sets. As a result, filenames containing Japanese characters may not be interpreted correctly if sent without encoding.

Scope Clarification

Filename encoding behavior differs depending on how the file is submitted:

- **API submissions** Filenames are read directly from HTTP request headers.
- **Archive submissions (ZIP, TAR, CAB)** Filenames of extracted child files are processed during archive extraction and may use automatic UTF-8 or Shift-JIS detection.

Encoding settings configured under **Archive Extraction** apply only to filenames of files extracted from archives and do not affect filenames provided through API requests.

Resolution

To ensure filenames containing Japanese characters are displayed correctly when submitting files via API:

1. Encode the filename using UTF-8.
2. Apply URL encoding (%-encoding) to the UTF-8 encoded filename.
3. Send the encoded value in the `filename` (and `filepath`, if applicable) request header.

This is the recommended and supported approach for handling non-ASCII filenames in API submissions.

Additional Information

- The archive encoding configuration does not influence API-submitted filenames.
- File content processing is not affected; only the displayed filename may appear garbled if encoding is not applied.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

Does MD Core support using a different account type for starting application services

Yes, MD Core does support different account types for starting application services.

The default Local System Account can be changed to a user account for MD Core under:

- Services → OPSWAT MetaDefender Core

Before making this change, please ensure that the selected user has full administrative privileges on the server. Insufficient permissions may cause critical MetaDefender Core functionalities to stop working properly.

We recommend performing this configuration during a maintenance window or low-usage period, as the service restart will temporarily interrupt MetaDefender Core operations. After the change, verify that all dependent services start successfully and that scanning, update, and communication functions are working as expected.

Support:

If you encounter any issues or require further assistance, please contact OPSWAT support via My OPSWAT at: https://my.opswat.com/home → Our support team will be able to provide you with the necessary guidance and resolution for any license-related concerns.

How to export API Keys of MetaDefender Core users

MetaDefender Core provides remote integration and management capabilities through its REST API interface. To perform API operations, each user must have a configured API key with the appropriate privileges for the intended actions.

However, when multiple users access MetaDefender Core simultaneously—each with their own API key—it can become challenging to maintain an accurate record of all existing keys.

To address this, the **GET /admin/user** API endpoint can be used to retrieve a complete list of users within MetaDefender Core, along with their associated details. This allows administrators to programmatically process the response and identify all users who possess an API key.

Furthermore, by leveraging the **GET /admin/userdirectory** API endpoint, it is possible to list all available user directories. This enables correlation between users and their respective directories, providing a clearer and more organized overview of user management within MetaDefender Core.

Below is a PowerShell script that retrieves and displays all existing users configured with an API key in MetaDefender Core. The output is presented in the following structured format:

USER	API KEY
DIRECTORY\user	<api key>

Notes:

- Replace `$apiKey` and `$coreUrl` with your actual MetaDefender Core credentials and instance URL.
- To export the results to a CSV file, **uncomment the last line** and provide a valid file path and name (for example: `C:\Reports\MetaDefender_Users_APIKeys.csv`).

Code

```
# ---- TLS + SKIP CERT CHECK ----
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

Add-Type @"
using System.Net;
using System.Security.Cryptography.X509Certificates;
public class TrustAllCertsPolicy : ICertificatePolicy {
    public bool CheckValidationResult(
        ServicePoint srvPoint, X509Certificate certificate,
        WebRequest request, int certificateProblem) { return true; }
}
"@
[System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy

# ---- CONFIG ----
$apiKey = "Admin API KEY"
$coreUrl = "MD CORE URL"

# ---- FETCH DATA ----
$users        = Invoke-RestMethod -Uri "$coreUrl/admin/user" -Headers @{ apikey = $apiKey } -
Method GET
$directories  = Invoke-RestMethod -Uri "$coreUrl/admin/userdirectory" -Headers @{ apikey =
$apiKey } -Method GET

# Build directory lookup
$dirLookup = @{}
foreach ($dir in $directories) {
    $dirLookup[$dir.id] = $dir.name
}

# ---- BUILD OUTPUT (only users with API keys) ----
$result = foreach ($u in $users | Where-Object { $_.api_key }) {
    $directoryName = $dirLookup[$u.directory_id]

    [PSCustomObject]@{
        User   = "$directoryName\$($u.name)"
        ApiKey = $u.api_key
    }
}

# ---- DISPLAY ----
$result | Format-Table -AutoSize
```

```
# ---- OPTIONAL EXPORT TO CSV ----
# $result | Export-Csv "User_ApiKey_List.csv" -NoTypeInformation
```

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Should Filetype Mismatch Be Blocked or Allowlisted?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Overview

Customers often raise this question when they experience a high volume of *Filetype Mismatch* detections that impact normal business workflows (for example, email attachments, files from legacy systems, or user-renamed files).

This article explains the security implications of allowlisting Filetype Mismatch and OPSWAT's recommended approach.

Why does Filetype Mismatch occur so frequently?

Common legitimate causes include:

- Email gateways or applications renaming attachments
- Legacy systems exporting files with incorrect extensions
- Users manually renaming files
- Archives containing mixed file types

However, the same technique is also commonly used by malware to disguise malicious files.

Does allowlisting Filetype Mismatch affect detection or scanning accuracy?

No.

MetaDefender Core uses **content-based file type detection (magic bytes)**. Even when Filetype Mismatch is allowlisted:

- The real file type is still detected correctly
- The appropriate scanning engines are selected
- Archive extraction and recursive scanning still occur

- Parsers and analysis steps continue to run as expected

From a processing perspective, scanning remains technically correct.

What security risks does allowlisting introduce?

The main risks occur at the **workflow and user-exposure level**:

- **File type spoofing** – executables disguised as documents (e.g., `invoice.pdf.exe`)
- **Policy bypass** – scripts or executables passing through document-only workflows
- **Downstream system trust issues** – systems relying on extensions may mishandle dangerous files
- **Higher chance of user execution** – users are more likely to open files that appear harmless

These techniques are widely used in phishing and malware campaigns.

What is OPSWAT's recommendation?

Default (best practice)

Block Filetype Mismatch.

This provides the strongest protection against disguised malware and workflow bypass.

If allowlisting is required for business reasons

Apply the following compensating controls:

- Allowlist **only specific workflows or trusted sources** (not globally)
- Continue blocking when the detected type is **executable or script**
- Keep **deep and recursive scanning enabled**
- Maintain full engine coverage
- Improve file visibility for end users (for example, show file extensions in Windows via GPO)
- Educate users to be cautious with unexpected attachments

Summary

Blocking Filetype Mismatch provides the strongest protection. Allowlisting does not break scanning, but it increases the risk of disguised malware and policy bypass and should be used only in tightly controlled scenarios.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

Tracking Client Identity Information in MetaDefender Core via

API

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems from version v5.14.2

Overview

The **Client Identity** capability in **MetaDefender Core (MD Core)** lets integrators attach *source/client metadata* to scan requests. This allows administrators to identify **which client system or integration** submitted a file for scanning and improves traceability in multi-client deployments.

Why This Matters

In environments where multiple systems connect to a shared MD Core instance (for example, Kiosks, MFT workflows, or API gateways), it is often necessary to:

- Track the originating client for each request
- Improve auditability of scan traffic
- Speed up troubleshooting and reporting

About the `client-identity` Header

`client-identity` is an optional header designed to carry identification context for scan requests.

In a chained workflow (example: MetaDefender Kiosk MetaDefender MFT MetaDefender Core), this metadata can represent one or more client hops and helps MD Core log who/what initiated the scan.

Typical Metadata Included

The header can contain details such as:

- Client/system name
- Client hostname
- User name associated with the request
- Deployment ID (unique identifier for the client site/instance)
- Timestamp captured at the client side (when the file/request was received)
- Client product/version information

Scan Workflows That Support Client Identity

Client identity metadata can be submitted with common scan flows, including:

- Synchronous analysis
- Asynchronous analysis
- Local scan
- Scan from link
- Batch scanning
- Multipart upload sessions

Common API Operations Where Client Identity Applies

Client identity information can be used with the scan request and later retrieved/validated through API calls such as:

- **Fetch / Status Requests**
  - Fetch analysis result
  - Fetch analysis result by hash
  - Retrieve batch analysis status
- **Submission Requests**
  - Submit file for analysis (async)
  - Submit file for analysis (sync)
  - Create/initiate batch
  - Create multipart upload session

Where You Can View Client Identity Information

Once submitted, client identity details can be reviewed from multiple sources:

- **MetaDefender Core Web Console**
  - View scan request details through the UI (Processing History View Client Identity).
- **Processing Result JSON**
  - The client identity payload is included as part of the processing result output.
- **My OPSWAT Central Management (MOCM)**
  - MD Core forwards client identity as part of scan reporting for centralized monitoring and reporting.

Example JSON Payload

- **Standard Example**

Code · json

```
{
  "clients": [
    {
      "deployment_id": "abcdef1234567890",
      "user_name": "Gemini",
      "product_type": "MetaDefender Kiosk",
      "version": "1.0.0",
      "host": "client.example.com",
      "timestamp": 1744949754890
    }
  ]
}
```

- **Example Used in Testing**

Code · json

```
{
  "clients": [
    {
      "deployment_id": "Test123",
      "user_name": "Test-User",
      "product_type": "PostMan",
      "version": "1.0.0",
      "host": "test.com",
      "timestamp": 1761658500000
    }
  ]
}
```

Parameter Description

Field	Type	Description	Key Types
deployment_id	String	Unique identifier for the originating client deployment.	REQUIRED
user_name	String	User identity associated with the scan request.	REQUIRED
product_type	String	Client integration/ product name (e.g., Kiosk, MFT, Postman).	REQUIRED

Field	Type	Description	Key Types
version	String	Client product version sending the request.	OPTIONAL
host	String	Hostname or domain of the client source system.	OPTIONAL
timestamp	Number	Unix timestamp (milliseconds) captured for the request at the client side.	REQUIRED

Implementation Steps

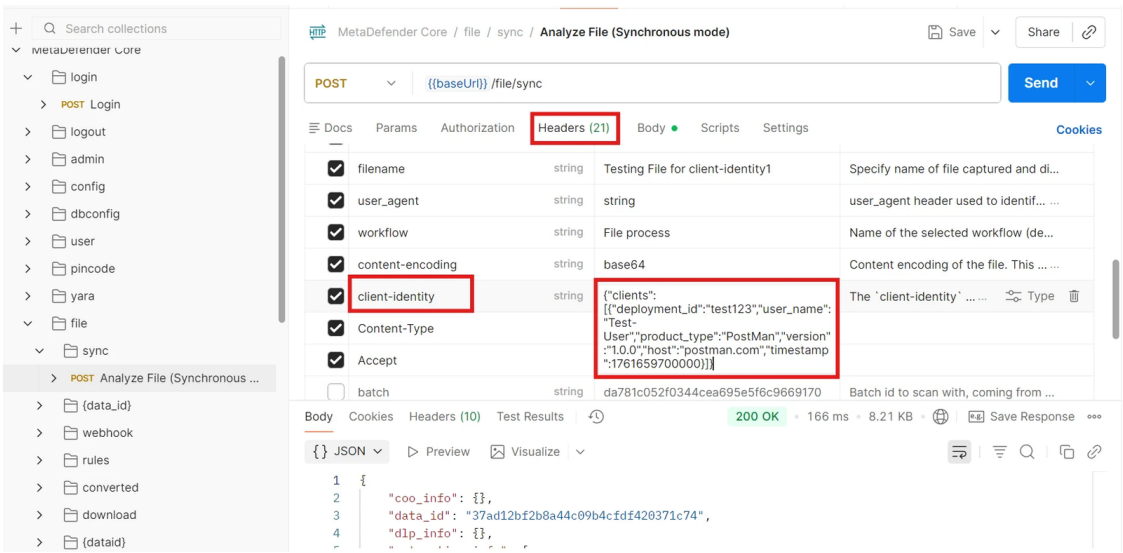
1. Include the `clients` object in the scan request payload sent to MD Core.
2. Add the `client-identity` header (if required by your integration approach).
3. Submit the scan request using Postman or your application.
4. Open Processing History in the MD Core UI and select View Client Identity.

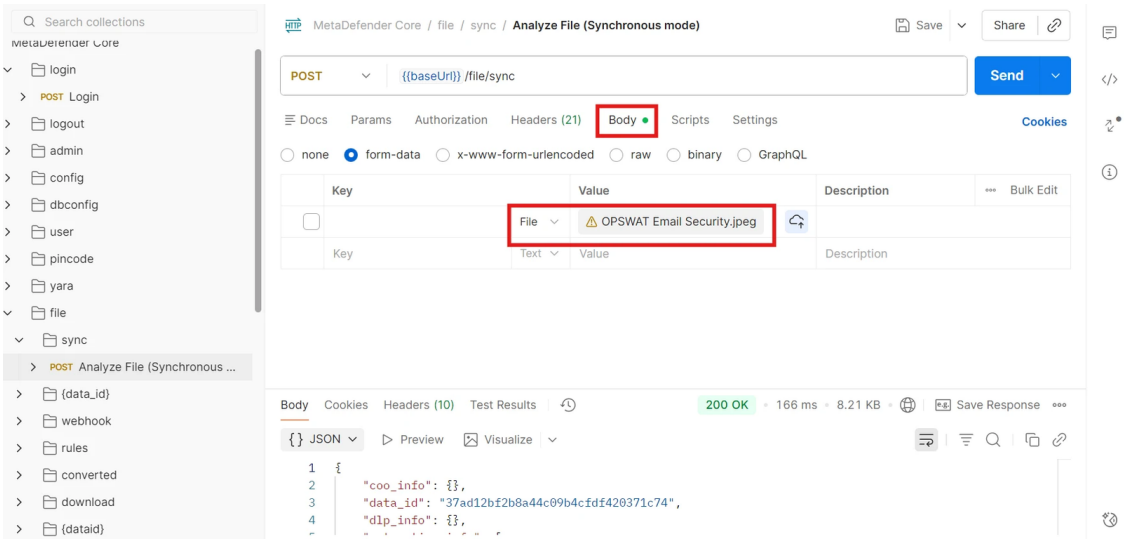
Verification

Validate that the values shown match what was sent from the client.

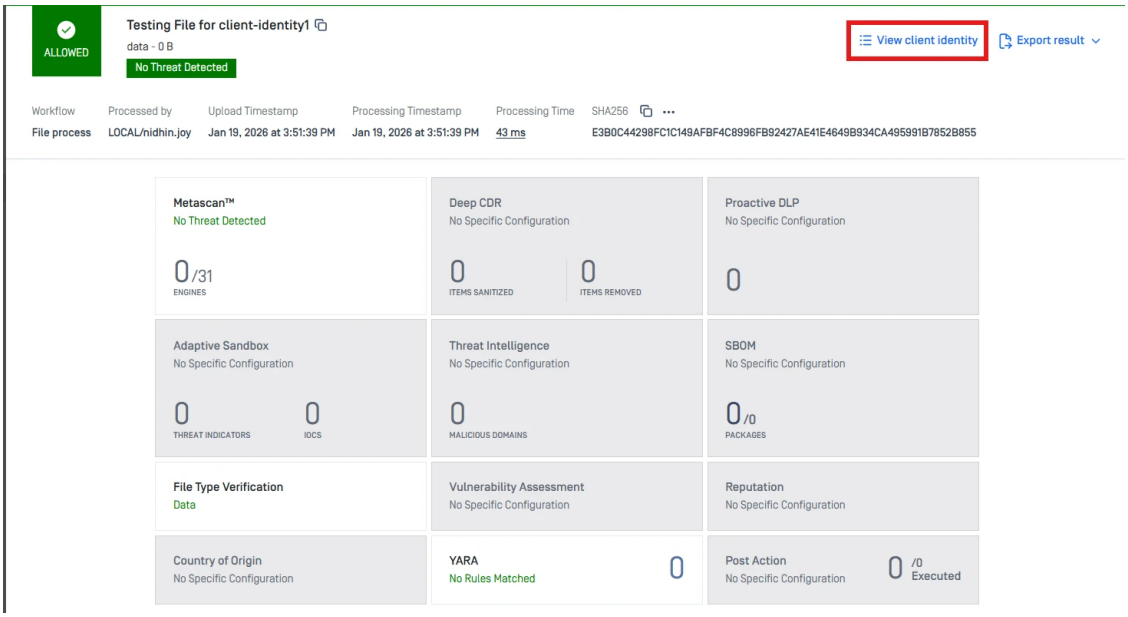
Screenshots:

PostMan/API Client:

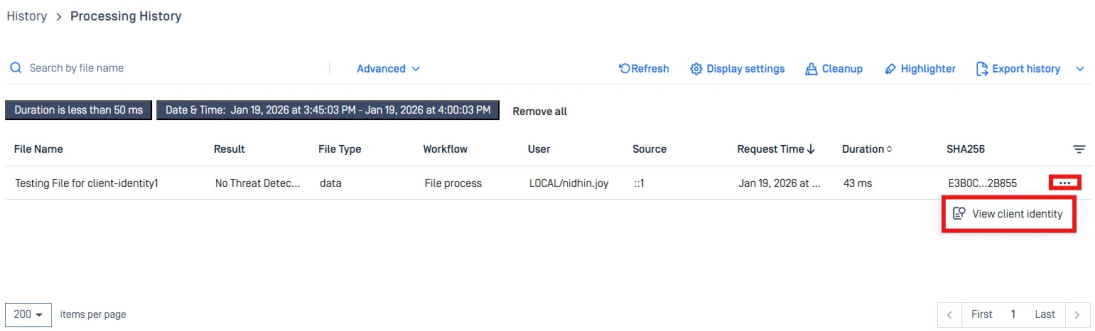


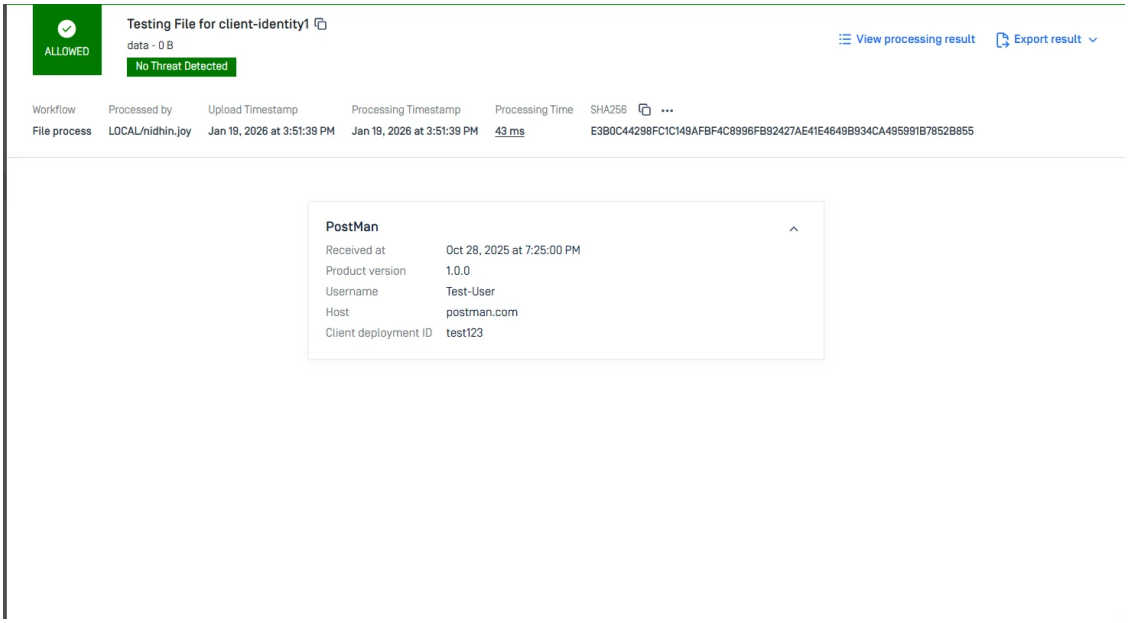

Search collections
MetaDefender Core
login
POST Login
logout
admin
config
dbconfig
user
pincode
yara
file
sync
POST Analyze File (Synchronous ...
{data_id}
webhook
rules
converted
download
{dataid}
MetaDefender Core / file / sync / Analyze File (Synchronous mode)
Save
Share
POST
{{baseUrl}} /file/sync
Send
Docs
Params
Authorization
Headers (21)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Key
Value
Description
Bulk Edit
File
OPSWAT Email Security.jpeg
Key
Text
Value
Description
Body
Cookies
Headers (10)
Test Results
200 OK
166 ms
8.21 KB
Save Response
{} JSON
Preview
Visualize
1 {
2 "coo_info": {},
3 "data_id": "37ad12bf2b8a44c09b4cfdf420371c74",
4 "dlp_info": {},

MD Core:


ALLOWED
Testing File for client-identity1
data - 0 B
No Threat Detected
View client identity
Export result
Workflow
File process
Processed by
LOCAL/nidhin.joy
Upload Timestamp
Jan 19, 2026 at 3:51:39 PM
Processing Timestamp
Jan 19, 2026 at 3:51:39 PM
Processing Time
43 ms
SHA256
E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855
Metascan™
No Threat Detected
0/31
ENGINES
Deep CDR
No Specific Configuration
0
ITEMS SANITIZED
0
ITEMS REMOVED
Proactive DLP
No Specific Configuration
0
Adaptive Sandbox
No Specific Configuration
0
THREAT INDICATORS
0
IOCS
Threat Intelligence
No Specific Configuration
0
MALICIOUS DOMAINS
SBOM
No Specific Configuration
0/0
PACKAGES
File Type Verification
Data
Vulnerability Assessment
No Specific Configuration
Reputation
No Specific Configuration
Country of Origin
No Specific Configuration
YARA
No Rules Matched
0
Post Action
No Specific Configuration
0 /0 Executed

Or:


History > Processing History
Search by file name
Advanced
Refresh
Display settings
Cleanup
Highlighter
Export history
Duration is less than 50 ms
Date & Time: Jan 19, 2026 at 3:45:03 PM - Jan 19, 2026 at 4:00:03 PM
Remove all
File Name
Result
File Type
Workflow
User
Source
Request Time
Duration
SHA256
Testing File for client-identity1
No Threat Detec...
data
File process
LOCAL/nidhin.joy
::1
Jan 19, 2026 at ...
43 ms
E3B0C...2B855
View client identity
200
items per page
First
1
Last



Notes / Best Practices

- Use consistent naming conventions for `deployment_id`, `host`, and `product_type` across all integrations.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why Does My MD Core Deployment on Kubernetes Lose Data After a Restart?

Info

This article applies to all MetaDefender Core releases deployed through OPSWAT MetaDefender Helm Chart on Kubernetes.

PostgreSQL behaviour on Kubernetes

This behavior is typically related to the deployment configuration and how MetaDefender Core manages data persistence in a Kubernetes environment. By design, Kubernetes Pods are stateless. Each time the Core Pod is stopped, restarted, or rescheduled, a new Pod instance is created—with a different name and a fresh container environment. As a result, any data not stored on a persistent volume is lost during this process.



```
default                          md-core-8f7f5c489-lw4h6
default                          md-icapsrv-7789dc59f5-nfvzs
```

Due to this behavior, all files stored within the Core Pod are lost after a Pod restart. If you are using the bundled PostgreSQL database included with the Core Helm chart, it is essential to configure persistent storage for the PostgreSQL Pod.

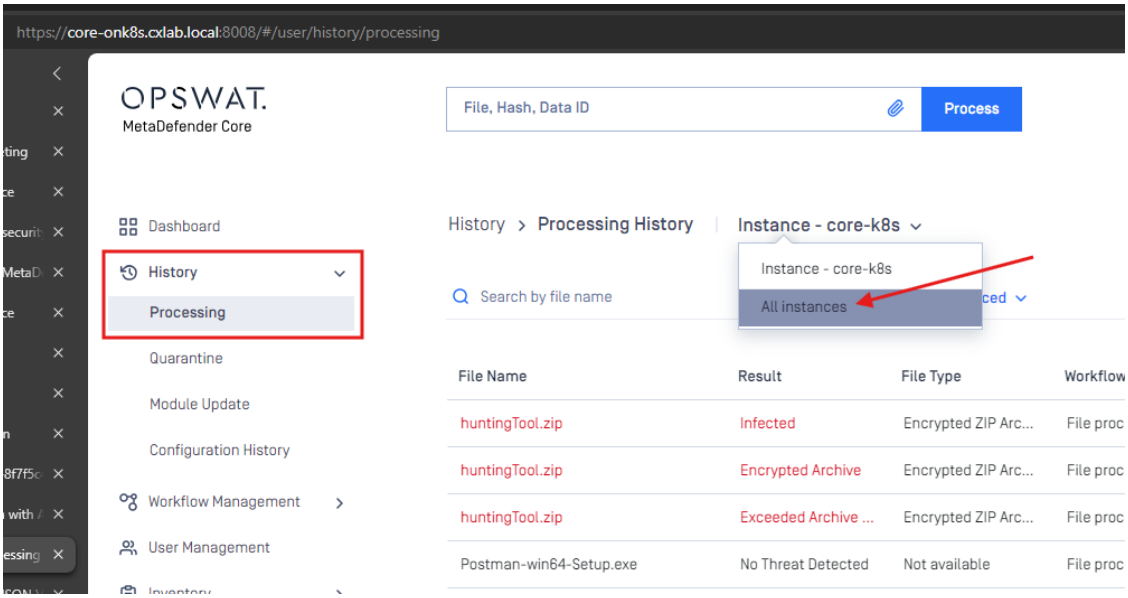
This can be achieved by mapping a Persistent Volume to the path specified by the `persistentDir` value in the `postgres-core` section of the Core Helm chart. This ensures that database data is retained across Pod restarts or re-creations.

```
core_components:
  postgres-core:
    name: postgres-core
    image: postgres:14.8
    env:
      - name: POSTGRES_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mdcore-postgres-cred
            key: password
      - name: POSTGRES_USER
        valueFrom:
          secretKeyRef:
            name: mdcore-postgres-cred
            key: user
    ports:
      - port: 5432
    is_db: true
    persistentDir: /var/lib/postgresql/data
    tls:
      enabled: false
```

A sample config for Persistent Volume Mount can be found on our MD Core Documentation Page:

Storage Configuration on K8S - MetaDefender Core

As long as the PostgreSQL database is persistent, the Processing History will be retained and can be accessed by choosing “All Instance” in the page:



Making DLP, Sanitization files Persistent

According to our Documentation (Docker image published on OPSWAT Docker Hub - MetaDefender Core), MetaDefender Core on containerized platforms supports an environment variable named `STORAGE_PATH`. This variable allows you to specify a full file system path for storing sanitized, DLP-processed, and quarantined files.

To ensure these files persist after a Pod restart, you must set the `STORAGE_PATH` environment variable and map a Persistent Volume to the specified path. This configuration enables the application to retain critical processing data across Pod lifecycles.

1. In the `values.yaml`file, set the STORAGE_PATH environment value to a custom path

```
      secretKeyRef:
        name: mdcore-postgres-cred
        key: user
  - name: DB_PWD
    valueFrom:
      secretKeyRef:
        name: mdcore-postgres-cred
        key: password
  - name: IGNITION_JSON
    valueFrom:
      secretKeyRef:
        name: mdcore-stack-configs
        key: IGNITION_JSON
  - name: STORAGE_PATH
    value: /var/lib/ometascan/mdcore_storage
```

2. Define a `storage_configs` for a persistentVolumeClaim

```
storage_configs:
  mdcore-pvc:
    apiVersion: v1
    kind: PersistentVolumeClaim
    metadata:
      name: mdcore-longhorn-pvc
    spec:
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 30Gi
      storageClassName: longhorn-storage-retain
```

Info

Note that the `storageClassName` should match an available Storage Class in your Kubernetes platform for proper persistent volume provisioning.

3. Set `storage_provisioner`, `storage_name` and `hostPathPrefix (optional)`

```
storage_provisioner: custom       # Type of sto
storage_name: mdcore-longhorn-pvc
hostPathPrefix: mdcore-storage-<APP_NAMESPACE>
```

4. Finally, mount the persistent Storage by using `extraVolumeMounts` and `extraVolumes`

```
    extraVolumeMounts:
      - name: md-core-store-data
        mountPath: /var/lib/ometascan
    extraVolumes:
      - name: md-core-store-data
        persistentVolumeClaim:
          claimName: mdcore-longhorn-pvc
```

Info

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

What is the pg_dump database restore file used by

MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core uses a PostgreSQL database to store critical system information required for normal operation, including processing history and the current state of modules and engines.

When creating a backup, the database restore file is generated through a process called a **database dump**, using the PostgreSQL utility `pg_dump`.

What is the .sql file generated by pg_dump?

If `pg_dump` is executed in its default (plain-text) format, the output is a `.sql` **file**.

This file is a plain-text SQL script that contains all commands required to fully reconstruct the database, including:

- Database schema (tables, indexes, constraints, etc.)
- Data stored in the tables (records/rows)
- Other database objects required by MetaDefender Core

In short, the `.sql` file represents a complete logical backup of the MetaDefender Core database at the time it was created.

How MetaDefender Core uses the restore file

During a restore operation, the `.sql` file is executed against a PostgreSQL database to recreate:

- The database structure
- All stored operational data

This allows MetaDefender Core to resume operation with the same configuration and historical context as the original system.

Standalone database backup and restore overview

For environments using a **standalone database** (local PostgreSQL instance on the same machine as MetaDefender Core):

- Use `pg_dump` to generate the `.sql` backup file.
- Restore the file using PostgreSQL tools (for example, `psql`) on the target system.
- Ensure the new Core instance uses the same database configuration (database name, user, and deployment context).

For detailed step-by-step instructions, refer to the article:

How to Backup and Restore standalone database for MetaDefender Core? - MetaDefender Core

Additional notes

- Restoring to the same machine typically works without issue.
- Restoring to a different machine may fail if environment-specific values (such as deployment ID or database name) differ.
- Always verify whether your deployment uses a standalone or shared database before performing a restore.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

Optimizing PostgreSQL for High-Volume Deployments

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux.

Overview

Symptoms of an Untuned Database

- High CPU usage on the Database process.
- Slow file processing and "stuttering" in the Dashboard.
- Database locks or timeouts.
- Excessive disk I/O is causing system lag.

This guide explains how to tune the bundled PostgreSQL instance for high-volume deployments on Windows and Linux.

Prerequisites

⚠ Prerequisite: Before applying changes, take a Backup of your database or ensure you have a snapshot of the VM.

Tuning Strategy (Read This First)

Do **not** blindly copy the examples below. You must adjust values based on your server's **Total RAM**.

Memory Allocation Guidelines

Use this table to determine the correct values for your server size.

	shared_buffers (25%)	effective_cache_size (75%)	maintenance_work_mem	max_
16 GB	4GB	12GB	1GB	4GB
32 GB	8GB	24GB	2GB	8GB
64 GB	16GB	48GB	2GB	16GB
128 GB	32GB	96GB	4GB	32GB

PostgreSQL Tuning on Windows

A. Locate the Correct Configuration File

MetaDefender Core uses an override mechanism. You must edit the file in the installation directory, not the data directory.

- **File Path:** `<Installation Directory>\\OPSWAT\\MetaDefender Core\\postgres\\postgresql.conf`
- **Default Example:** `C:\\Program Files\\OPSWAT\\MetaDefender Core\\postgres\\postgresql.conf`

Note: If this file does not exist, create it. Settings in this file override the defaults in pg_data.

B. Apply Configuration [This configuration is tuned for a 64GB RAM Server]

Open the `postgresql.conf` file (Run Notepad as Administrator) and add/append the following block. (Values below are optimised for a **64GB RAM server**. Adjust based on the table above.)

Code

```
# -- Memory Settings (Adjust based on RAM Table) --
shared_buffers = 16GB
effective_cache_size = 48GB
work_mem = 64MB
maintenance_work_mem = 2GB

# -- Connection & Write Tuning --
max_connections = 100
wal_buffers = 16MB
max_wal_size = 16GB           # Crucial for high-volume writes
min_wal_size = 4GB
checkpoint_completion_target = 0.9

# -- SSD Optimizations (If using SSD/NVMe) --
random_page_cost = 1.1        # Default is 4.0 (for spinning disks)
effective_io_concurrency = 200
```

```
# -- Autovacuum Tuning (Prevent Bloat) --
autovacuum = on
autovacuum_max_workers = 5
```

IMPORTANT: Please note that the number of workers is totally dependent on the customer's instance **CPU** size.

IMPORTANT: The **random_page_cost** and **effective_io_concurrency** values are only applicable to the **SSD** disk.

C. Restart Service

Restarting the MetaDefender Core service automatically restarts the bundled PostgreSQL.

1. Press `Win + R`, type `services.msc`.
2. Locate `MetaDefender Core`.
3. Right-click > `Restart`.

PostgreSQL Tuning on Linux

A. Locate the Correct Configuration File

Create or edit the override configuration file.

- **Edit:** `/usr/lib/ometascan/postgres/postgresql.conf`

⚠ **Do NOT modify:** `/var/lib/ometascan/pg_data/postgresql.conf` (This file is reset on upgrades/restarts).

B. Apply Configuration [This configuration is tuned for a 64GB RAM Server]

Use a text editor to add the configuration block. (Values below are optimised for a **64GB RAM server**. Adjust based on the table above.)

Code

```
sudo nano /usr/lib/ometascan/postgres/postgresql.conf
```

Paste the following:

Code

```
# -- Memory Settings --
shared_buffers = 16GB
effective_cache_size = 48GB
work_mem = 64MB
maintenance_work_mem = 2GB

# -- Write Ahead Log (WAL) --
wal_buffers = 16MB
max_wal_size = 16GB
min_wal_size = 4GB
```

```
checkpoint_completion_target = 0.9

# -- Connections --
max_connections = 100

# -- Autovacuum --
autovacuum = on
autovacuum_max_workers = 5
```

IMPORTANT: Please note that the number of workers is totally dependent on the customer's instance **CPU** size.

C. Restart Service

Restart the MetaDefender Core service to apply changes.

Code

```
sudo systemctl restart ometascan
```

Verification

After restarting, you must confirm that the database actually loaded your new settings.

Method: Log in to the database (or use the SQL tool in the Core dashboard if available) and run:

Code

```
SELECT name, setting, unit, source
FROM pg_settings
WHERE name IN ('shared_buffers', 'max_connections', 'max_wal_size', 'work_mem');
```

- **Check:** The `setting` column should match your new values.
- **Check:** The `source` column should say `configuration file` (indicating it read your override).

Monitoring & Troubleshooting

Essential Monitoring Queries

1. **Check Active Connections** Ensure you aren't hitting the `max_connections` limit.

Code

```
SELECT count(*) as active_connections FROM pg_stat_activity;
```

2. **Check for Slow Queries** Identify what is slowing down the system.

Code

```
SELECT pid,
       now() - query_start AS duration,
       state,
       query
FROM pg_stat_activity
WHERE state != 'idle'
ORDER BY duration DESC;
```

3. **Check Database Size**

- For a standalone DB

Code

```
SELECT pg_size_pretty(pg_database_size('metadefender_core')) as db_size;
```

- For shared DB

Code

```
SELECT pg_size_pretty(pg_database_size('metadefender_core_xxx')) as db_size;
```

When to Scale Up (Externalize Database)

If tuning PostgreSQL is insufficient, you should move to a **Dedicated PostgreSQL Server**.

Indicators you need a dedicated DB server:

- **CPU:** The `postgres` process consistently consumes > 40% of total system CPU (starving the Scanning engines).
- **Disk I/O:** Disk queue length is high, causing scan timeouts.
- **Database Size:** DB exceeds 500GB.
- **Architecture:** You are running multiple MetaDefender Core nodes and need a central repository.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

How should I correctly restart the Shared Database server?

Check Your Version:

This article applies to the MetaDefender Core using Shared Database mode.

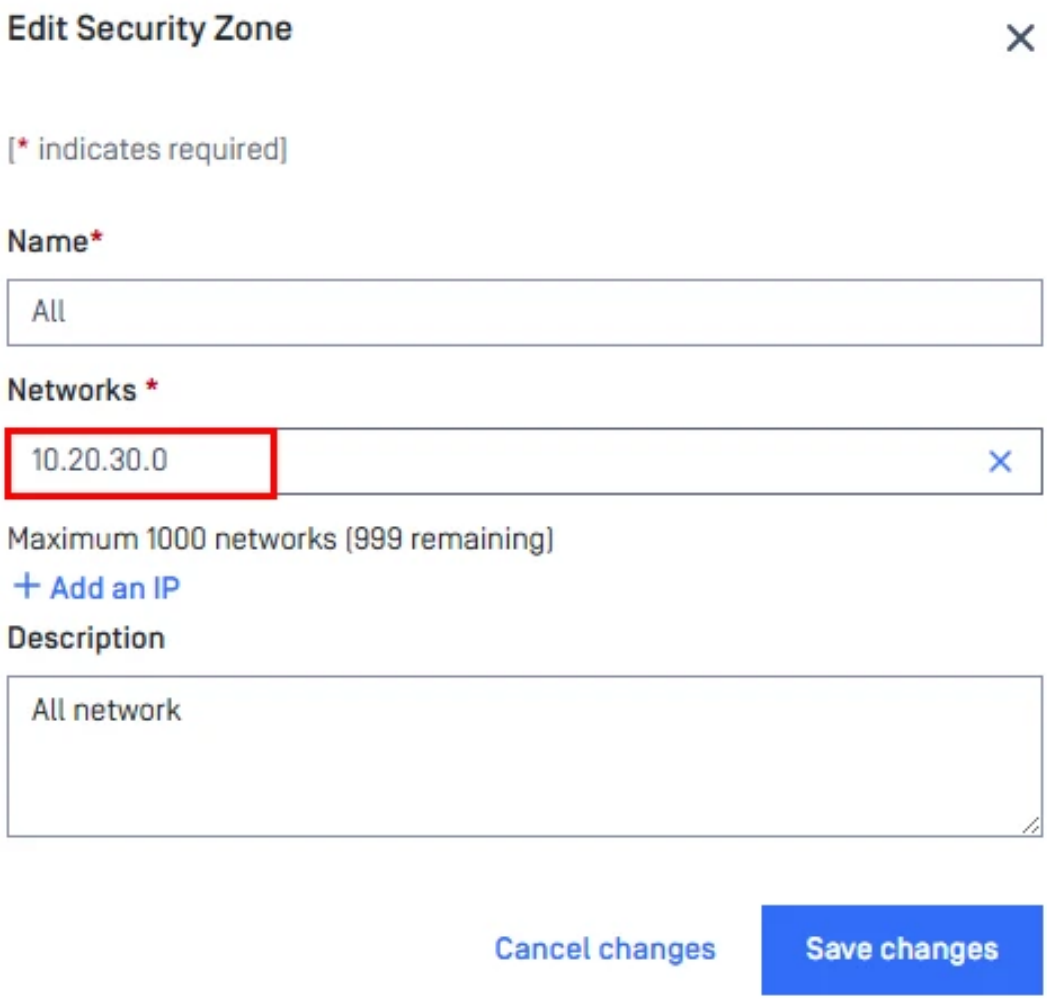
In Shared Database mode, multiple MetaDefender Core instances connect to and operate against the same database. Restarting the PostgreSQL service or the database server will interrupt all connected MD Core instances and may impact the stability of the shared environment.

If you need to restart the PostgreSQL service or the database server, follow the steps below to minimize disruption and reduce the impact on the connected MD Core instances.

Step 1: Block new incoming scan requests to MetaDefender Core.

This prevents new scan requests from being submitted to MetaDefender Core while allowing all in-progress scan tasks to complete normally. You can verify and configure this using the options below:

- Block inbound traffic to TCP port 8008 on the MetaDefender Core instance.
- Modify the MD Core Security Zones configuration to use an unreachable network range (for example, 10.20.30.0) to effectively block incoming connections. *Workflow Management> Security Zones > Edit*



- Assign the newly created user agent to the workflows *Workflow > General > Restrictions > Limit to specified user*

agents >

Step 2: Ensure that there are no active processing entries across all MetaDefender Core instances.

Step 3: Restart the Postgres Service or Database Server.

Step 4: Verify that all MetaDefender Core instances have successfully reconnected to the database and that the web UI is accessible.

Step 5: Revert the changes implemented in Step 1 to restore normal operations.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with our support engineer**.

How can I identify which MetaDefender Core instance sent an alert email?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

Overview

When you have multiple MetaDefender Core instances configured to send alerts, the emails can look identical (same sender, same content). This makes it difficult to identify which Core triggered a specific alert (e.g., an AV engine with outdated patterns).

MetaDefender Core supports **email subject macros** that let you insert a unique identifier (such as **Deployment ID** or **Instance Name**) directly into the subject.

Applies to

- Environments with **two or more MetaDefender Core instances** sending alerts to the same mailbox
- Especially useful for update/engine-related alerts (e.g., “outdated patterns”)

Resolution

Option A (Recommended): Add a unique identifier to the email subject

1. Log in to the **MetaDefender Core** web console on **Core #1**.

2. Go to **Settings** → **Email Notification**.

3. In the relevant alert section, set **Custom Email Subject Format** to include an identifier macro, for example:

- **Most common (works in all modes):** `MDCore ${deployment_id} - ${severity} - AV update alert`
- **Shared database mode (if available):** `MDCore ${instance_name} - ${severity} - AV update alert`

MetaDefender Core supports these macros in the email subject:

- `${severity}` - `${deployment_id}` - `${instance_name}` *(shared database mode only)*

4. Save changes.

5. Repeat steps **1–4** on **Core #2**, using the same subject format.

Result: Each alert email subject will clearly show the originating Core via **Deployment ID** (or **Instance Name**).

Where to find the Deployment ID

On each Core:

1. Go to **Settings** → **License**

2. Locate **Deployment ID**

Option B (Immediate workaround): Identify the sender using email headers

If you already received duplicate emails and need to identify the sender **right now**, open the email's **original headers** and look for the **Received** lines. These typically show the IP/host that connected to your SMTP server.

Validation

- Trigger (or wait for) a known alert (e.g., engine update/pattern alert).
- Confirm the email subject contains `${deployment_id}` (or `${instance_name}`) and that it matches the expected Core in **Settings** → **License**.

Notes / Best Practice

- If both Cores must notify the same recipients, **subject tagging** is the simplest and cleanest way to avoid confusion.
- If you are in shared database mode, `${instance_name}` can be especially readable for operations teams (when properly named).

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

What Is the Recommended Skip Hash List Size in MetaDefender Core Without Impacting Performance?

Check Your Version:

This article applies to MetaDefender Core since version v5.12.0.

The Skip By Hash feature in MetaDefender Core allows administrators to define Allowlist, Blocklist, and Skip Engines rules using file hashes.

This article clarifies system limits, performance behavior, and recommended sizing guidance.

1. **Is There a Maximum Number of Hashes?**

There is no defined hard limit on the total number of hashes that can be stored in the Skip Hash list. However, the CSV import size is limited to a maximum of 500 MB per file.

Performance testing shows:

- Importing **1 million hashes** took approximately **115 seconds**
- Test environment:
  - MetaDefender Core 5.17.1 standalone
  - Windows VM
  - 8 CPUs
  - 16 GB RAM

This confirms that MetaDefender Core can support very large skip lists.

2. **How Skip Hash Checking Works**

When a file is scanned:

- The file hash is calculated.
- MetaDefender Core checks whether the hash exists in the Skip List stored in the local database.

The hash comparison itself is lightweight and does not impact Core performance.

3. **Performance Impact by Hash Volume**

While there is no fixed maximum, practical performance behavior typically follows:

Total Skip Hashes	Expected Performance Impact (Well-Sized DB)
< 500K	Negligible impact

Total Skip Hashes	Expected Performance Impact (Well-Sized DB)
~1M	Very low impact (validated test)
1M–5M	Generally safe with proper indexing
5M–10M	Increased DB dependency; tuning required
>10M	Requires strong database sizing and monitoring

Important: If the PostgreSQL instance is:

- Shared with other applications
- Under-provisioned
- Running on slow disk

Performance degradation may appear earlier (even at 2–3 million entries).

4. **Import Behavior and Runtime Impact**

During CSV import:

- Scan service is not interrupted
- Duplicate cleanup increases processing time (If the CSV contains hashes that already exist in the Skip Hash list, the system needs to check and update those entries. This “duplicate handling” takes extra time, so the import process may be slower depending on how many duplicates are present).

Large imports increase database write activity but do not stop scanning.

5. **Recommended Safe Practice**

For stable and predictable performance:

- Keep Skip Hash list ideally under 5 million entries
- Use a dedicated SQL instance
- Ensure proper indexing
- Use SSD/NVMe storage
- Monitor:
  - DB CPU usage
  - Disk I/O latency
  - Query execution time
- Perform large imports during maintenance windows

- Avoid parallel CSV uploads

6. **Final Recommendation**

There is no official maximum number of hashes, but performance is infrastructure-dependent.

For most production environments:

1–5 million hashes can be handled without noticeable performance impact when properly sized. Above that, performance depends heavily on database capacity and tuning.

If planning to scale beyond 5–10 million entries, performance testing in a staging environment is strongly recommended.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Does MetaDefender Core support sanitizing GIT-BUNDLE files?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

No, Deep CDR does **not** support sanitization of Git object formats.

Explanation:

GIT-BUNDLE files have a rigid, immutable format with SHA-1/SHA-256 integrity checks and a directed acyclic graph (DAG) structure that cannot be altered without invalidating the bundle.

A git bundle is not just an archive. It's a serialized pack file containing:

- Git commit objects
- Trees
- Blobs
- Tags
- Objects within a packfile format with SHA-1/SHA-256 integrity
- Protocol metadata

This format is binary, integrity-checked, and extremely rigid. If any object inside changes even a single byte, the checksums break, and Git will reject the bundle.

Even if OPSWAT added Deep CDR support, the output would fail, and a "sanitized" Git bundle would not be a functioning Git bundle anymore.

Recommendations:

While Deep CDR sanitization is not supported, the format can still be extracted using Archive Extraction engine, and

all the child files should be scanned with Metascan's AV engines.

After extraction and scanning, if no malware is detected, the original GIT-BUNDLE file can be allowed.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Can I configure multiple REST ports in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview:

MetaDefender Core supports only **one** REST port per instance for API calls and web access. Configuring multiple REST ports on the same MetaDefender Core installation is not supported.

Steps to Configure the REST Port:

The REST/web console port can be configured as described in this article: https://www.opswat.com/docs/mdcore/knowledge-base/how-to-change-core-web-management-interface-address-port-#windows

How to Verify the Change:

- Check that the REST API and web console are accessible on the specified port after restarting the MetaDefender Core service.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How do I identify and renew the certificate for Active Directory integration in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview: This article provides guidance on identifying the certificate configured for Active Directory integration and renewing it, whenever StartTLS or SSL encryption is being used.

Prerequisites:

- Access to the MetaDefender Core server.

- Access to the Active Directory domain controller.

Steps to Configure:

1. **Identify the certificate:** Use openssl to query the LDAPS port and see the exact certificate currently serving the connection: `openssl s_client -connect <DC_IP_OR_FQDN>:636 -showcerts`
2. **Renew the Active Directory certificate:**
   - Follow the vendor's documentation to create a certificate request, submit it to a CA, and apply the renewed certificate. For example: Microsoft's Enable LDAP over SSL (external link).
   - Once the Active Directory certificate is renewed, run the above command to confirm that the renewed certificate is serving the connection.
3. **Install the certificate:** Install the renewed certificate and/or the certificate issuer on the MetaDefender Core server, similar to Adding a new Active Directory.

Verify the Change: Run the openssl command after renewing the certificate to confirm the new certificate is being used.

> **Support:**
> If Further Assistance is required, please proceed to log a support case or chat with our support engineer.

Retrieving Username Using an API Key

If the profile username has been lost, it may still be recovered using a valid **admin API key**, which can be used to send a **GET** Request to the **/admin/user** endpoint to list all the available users and identify the users

Retrieve the list of users by sending a **GET request** to the `/admin/user` API endpoint.

GET - List all users

Example request:

Code

```
curl --get \
 --url 'http://localhost:8008/admin/user' \
 --header 'apikey: {apikey}'
```

From the response, identify the user associated with the provided API key and retrieve the corresponding **name**.

> **Support:**
> If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to Configure MetaDefender Core Syslog to an AWS Ubuntu Instance

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems from version v5.15.1 and newer

Overview

This article provides step-by-step instructions to configure a cloud-hosted Ubuntu instance on Amazon Web Services (AWS) as a Syslog receiver and connect it to MetaDefender Core via the Management Console.

Prerequisites

- **AWS Console Access:** Permissions to modify EC2 Security Groups.
- **Ubuntu EC2 Instance:** Root or `sudo` SSH access to the target Linux server.
- **MetaDefender Core:** Administrator access to the Management Console.

Step-by-Step Instructions

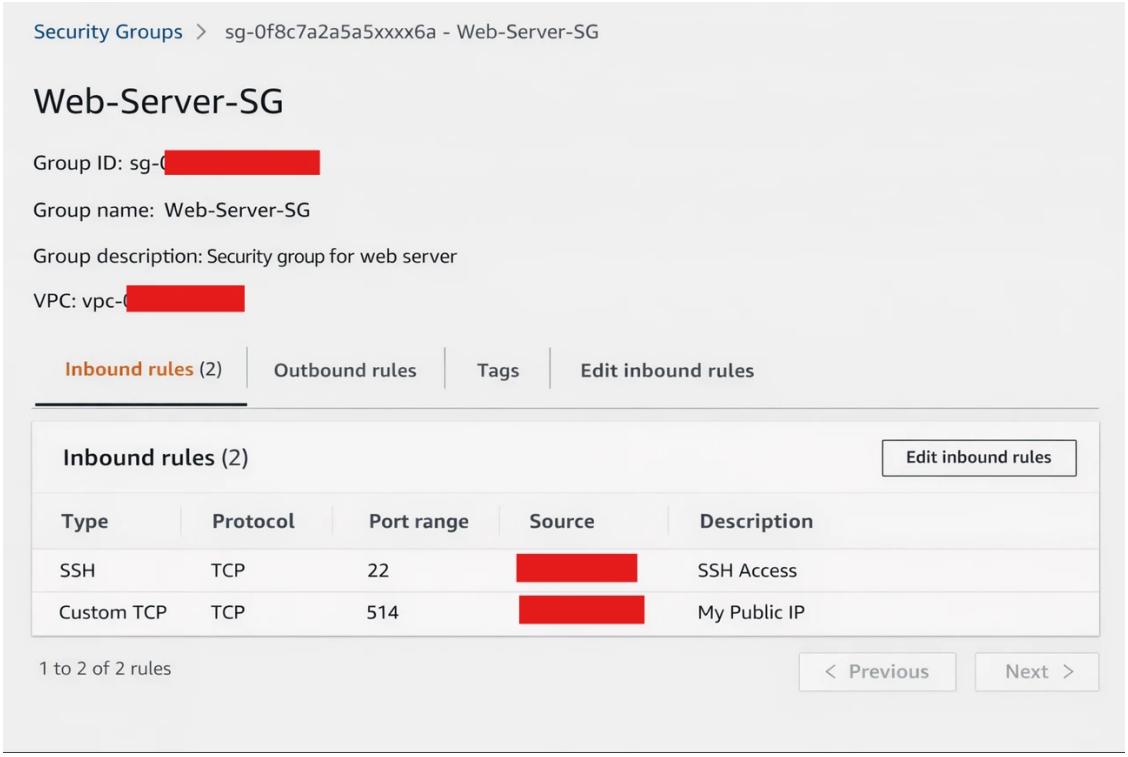
Phase 1: Configure AWS Infrastructure (Security Groups)

By default, AWS blocks all inbound traffic. You must explicitly allow Syslog traffic (Port 514) to reach your Ubuntu instance.

1. Log in to the **AWS Management Console** and navigate to the **EC2 Dashboard**.
2. Select your Ubuntu instance and locate the **Security** tab at the bottom of the screen. Click on the associated **Security Group** link.
3. Click **Edit inbound rules**.
4. Click **Add rule** and configure the following:
   - **Type:** Custom UDP Rule (and/or Custom TCP Rule depending on your preference).
   - **Port Range:** `514`
   - **Source:** Select `Custom` and input the public IP address (or Private VPC IP, if peered) of your MetaDefender Core server.

Pro Tip: Add `/32` to the end of the IP to restrict access strictly to the Core server (e.g., `198.51.100.20/32`).

5. Click **Save rules**.



Phase 2: Configure the Ubuntu Instance (`rsyslog`)

Ubuntu uses `rsyslog` as its default logging service, but it only accepts internal logs out-of-the-box. You must configure it to listen for external connections.

1. Connect to your AWS Ubuntu instance via SSH.
2. Open the `rsyslog` configuration file using a text editor (e.g., `nano`):

```
Code

sudo nano /etc/rsyslog.conf
```

1. Scroll down to the `MODULES` section. Uncomment the following lines (remove the `#` at the beginning) to enable UDP and/or TCP reception:

For UDP (Standard Syslog):

```
Code

module(load="imudp")
input(type="imudp" port="514")
```

For TCP (Reliable Syslog):

```
Code

module(load="imtcp")
```

```
input(type="imtcp" port="514")
```

4. Save the file and exit (`Ctrl+O`, `Enter`, `Ctrl+X` in nano).

5. Restart the `rsyslog` service to apply the changes:

```
Code

sudo systemctl restart rsyslog
```

6. Check for the rsyslog service status and make sure its showing active and running

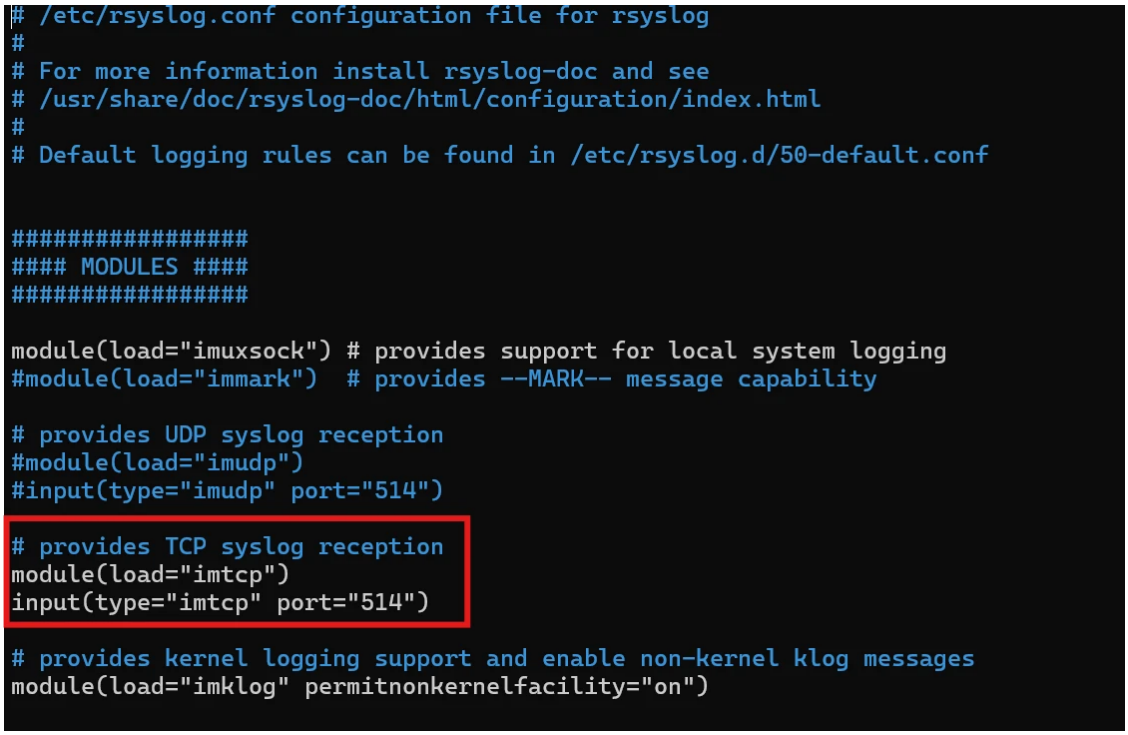
```
Code

sudo systemctl status rsyslog"
```

7. Verify the service is actively listening on port 514:

```
Code

sudo ss -tuln | grep 514
```



Phase 3: Configure MetaDefender Core

Now that the AWS server is ready to receive data, configure MetaDefender Core to forward the logs.

1. Log in to the **MetaDefender Core Management Console**.

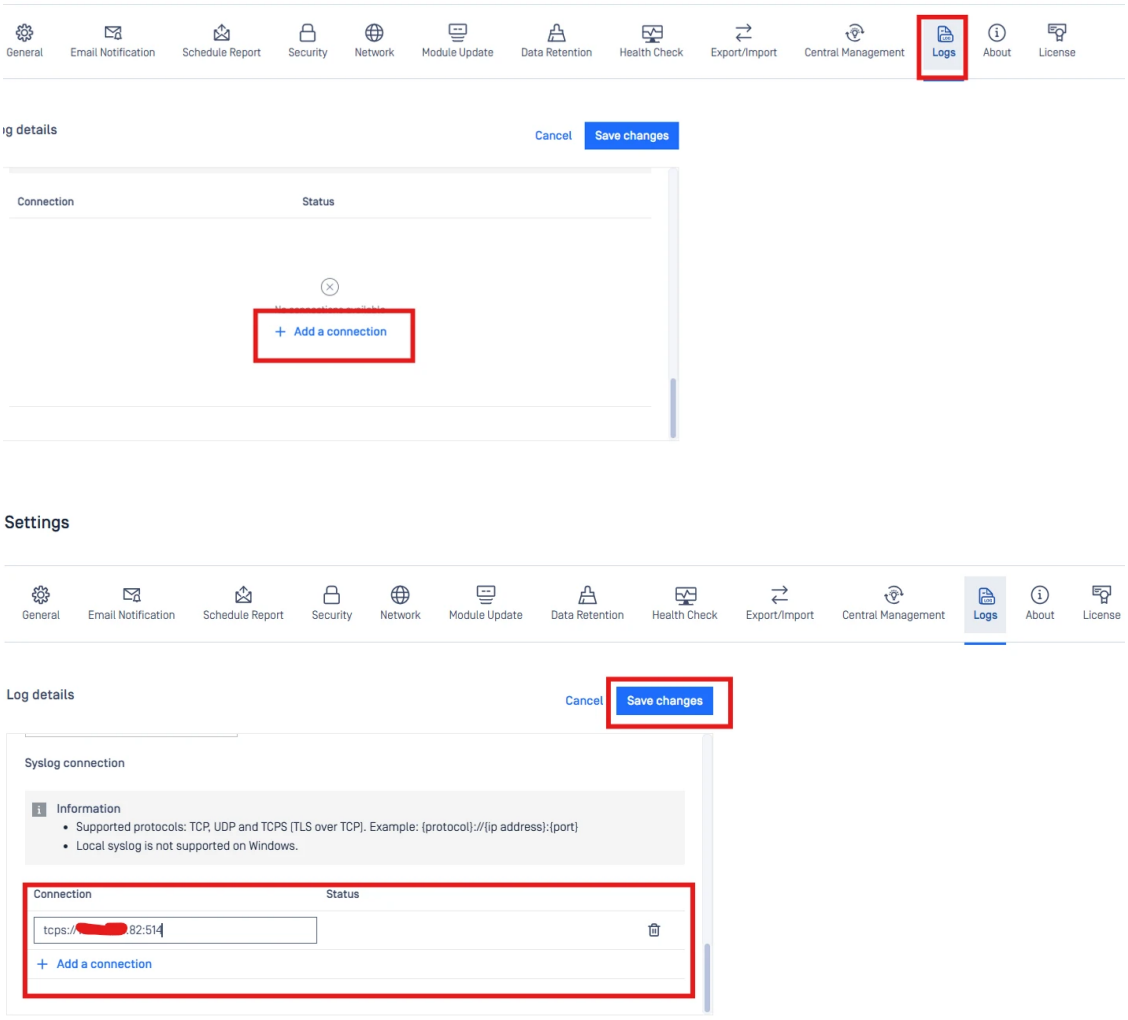
2. Navigate to **Settings** > **Logs >> Edit**.

3. Under **Syslog** tab >> **Add a connection**

4. Input the details for your AWS Ubuntu instance in **{protocol}://{ip address}:{port}** format

   - **Server Address:** The Public IP address (or Private IP if internally routed) of your AWS EC2 instance.
   - **Port:** `514`
   - **Protocol:** `UDP` (or `TCP` if configured in Phase 2).

5. Click on **Save Changes**to save the configuration.



Phase 4: Verification

To confirm that logs are successfully travelling from MetaDefender Core to your AWS instance:

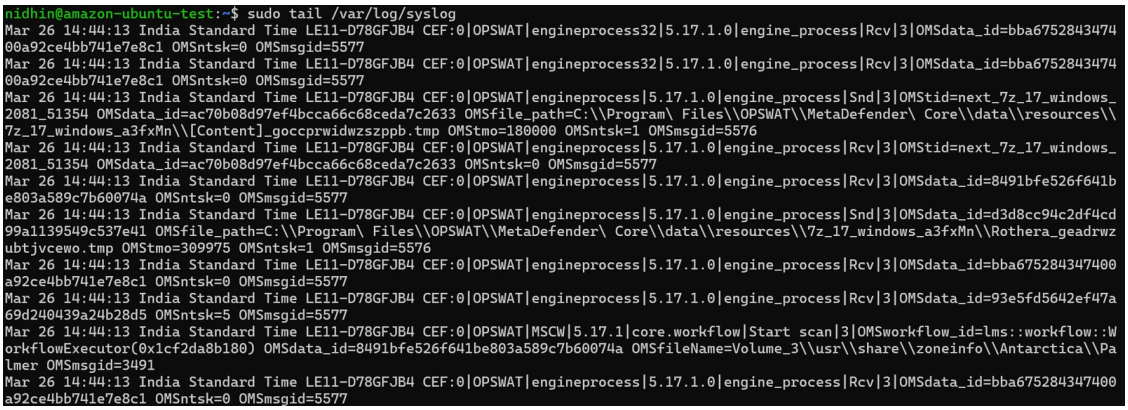
1. Return to your SSH session on the Ubuntu instance.

2. Run the following command to watch the live incoming system logs:

```
Code
```

```
sudo tail -f /var/log/syslog
```

1. Trigger an event in MetaDefender Core (e.g., upload a file for scanning, or log out and log back into the UI).
2. Watch the Ubuntu terminal. You should see new log entries appear containing the `ometascan` or `mdcore` service tags.



Troubleshooting Note:

If logs do not appear, verify that the Ubuntu Uncomplicated Firewall (UFW) is not overriding the AWS Security Group. You can check this by running `sudo ufw status`. If active, run `sudo ufw allow 514/udp`.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I reset all Skip by Hash Entries in MetaDefender Core?

Check you version:

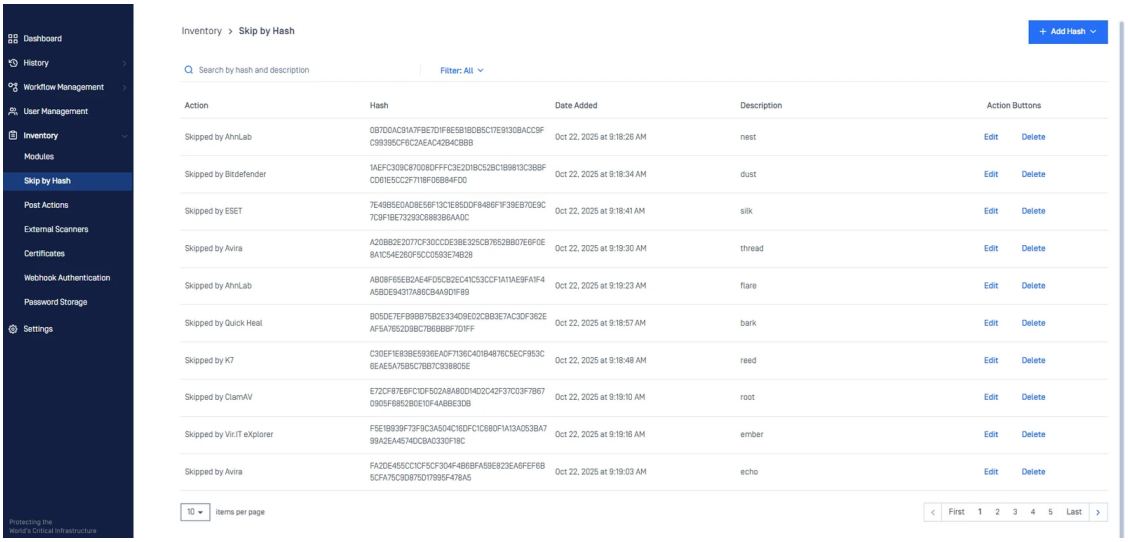
Check Your Version: This article applies to MetaDefender Core and is applicable to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

Warning

Please note that this process will delete all stored entries in the Skip by Hash lists. These lists are global and take precedence during file scans. After the deletion, workflow settings will assume priority.

Overview

When multiple skip-by-hash entries are present, or the volume is too large to manage manually, this method provides the recommended approach of using an API call to efficiently clear the list. Screenshot of a populated Skip by Hash list.



Steps to Configure

The command syntax depends on the operating system where MetaDefender Core is installed.

For Windows (Command Prompt):

Code

```
curl --request PUT ^
  --url "http://YOUR CORE IP HERE:8008/admin/config/skip" ^
  --header "Content-Type: application/json" ^
  --header "apikey: YOUR API KEY HERE" ^
  --data "{ \"blacklist\": {}, \"skip\": {}, \"whitelist\": {} }"
```

For Linux/macOS (Shell / Bash):

Code

```
curl --request PUT \
 --url 'http://YOUR CORE IP HERE:8008/admin/config/skip' \
 --header 'Content-Type: application/json' \
 --header 'apikey: YOUR API KEY HERE' \
 --data '{
  "blacklist": {},
  "skip": {},
  "whitelist": {}}'
```

```
Alacritty
alex@debian:~$ curl --request PUT \
 --url 'https://[redacted]:8008/admin/config/skip' \
 --header 'Content-Type: application/json' \
 --header 'apikey: [redacted]' \
 --data '{
  "blacklist": {},
  "skip": {},
  "whitelist": {}}'
```

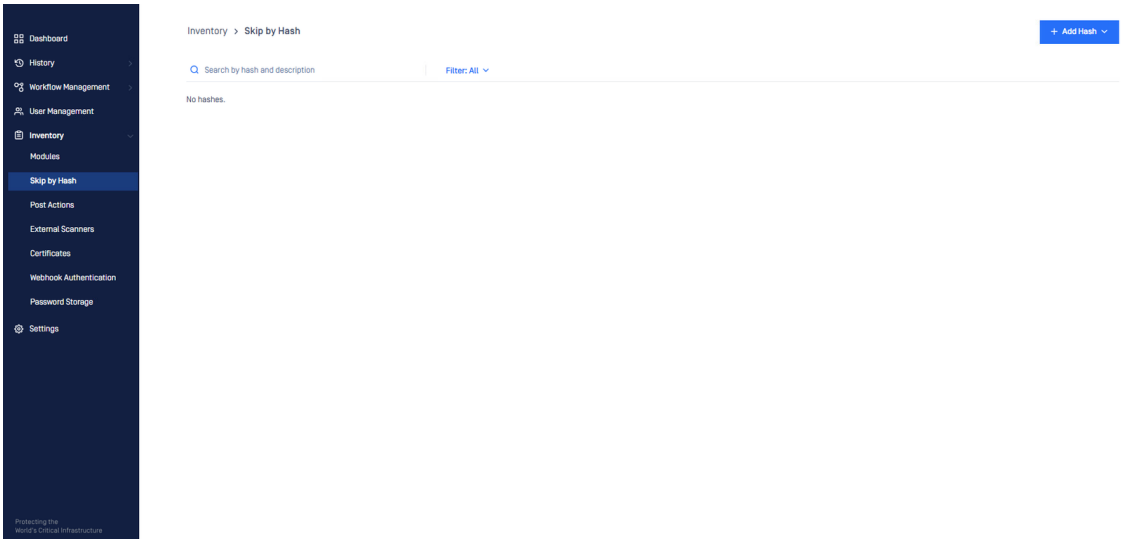
Screenshot of what the API call would look like on the Linux system.

```
{"blacklist":{},"whitelist":{},"skip":{},"total_count":0}
```

After executing the curl command, the shell output returned a JSON-formatted API response.

Verify the Change

After running the API call, verify that all skip by hash lists are cleared.



Screenshot of a cleared Skip by Hash list which shows that the API call has successfully removed all Skip by Hash entries.

Support

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Can I export scheduled scan reports via API calls or email?

Check Your Version:

This article applies to MetaDefender Core versions **5.13.0** and later deployed on Windows and Linux systems.

Overview:

MetaDefender Core does not currently support direct scheduled reporting or exporting of processing history via **API calls**.

However, it is possible to set up **scheduled email reports** of processing history.

Steps to Configure:

1. **SMTP Configuration:** Configure your SMTP server under Settings > Network > Email server.
   - For more details, refer to Email configuration
2. **Schedule Report Configuration:** Configure the schedule report under Settings > Schedule Report > Processing Activity or Executive Report.

Verify the Change:

- After configuration, check that scheduled reports are sent to the configured email address.

Related Articles:

- More details on schedule report can be found here: https://www.opswat.com/docs/mdcore/configuration/schedule-report

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to integrate OpenSSL with MetaDefender Core for decrypting files

This KB explains how to configure MetaDefender Core's Archive Extraction engine with the CLI Extraction feature to decrypt OpenSSL-encrypted files automatically during scanning. It was created from a real customer case and includes working JSON examples, troubleshooting tips, and validation steps.

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows.

Applies to

- Archive Extraction engine enabled.
- CLI Extraction feature is available in your build.

What you will achieve

- Automatically decrypt OpenSSL password-protected files during scans so that child files are inspected by all enabled engines and modules (e.g., Multiscanning, CDR, DLP).
- Map the correct Core file type to OpenSSL via a CLI rule.

Background and key concepts

MetaDefender Core supports an external CLI Extraction capability. You provide a JSON list of tools to run for specific file types. For OpenSSL-encrypted content, Core identifies files by internal file type. For legacy ".des" payloads created via OpenSSL enc, Core's internal file type is OPENSSL-SALT. Therefore, your CLI JSON must target file_type_id: OPENSSL-SALT instead of a generic or extension-based value like "des."

Prerequisites

- MetaDefender Core v5 is installed and licensed with the Archive Extraction engine active.
- Administrative access to Core UI to upload/assign the CLI JSON and to toggle the Archive Extraction engine.
- OpenSSL installed on the Core host (Windows example path below; Linux path may be /usr/bin/openssl).

Step-by-step configuration

1) Create or update the CLI Extraction JSON

Prepare a JSON file (root is an array) defining OpenSSL as the decryptor for OPENSSL-SALT. Ensure that both $input and $output placeholders exist; $password is optional but required for password-based decryption.

Code

```
[
  {
    "file_type_id": "OPENSSL-SALT",
    "executable_path": "C:\\OpenSSL\\bin\\openssl.exe",
    "arguments": [
      "enc",
      "-d",
      "-aes-256-cbc",
      "-pbkdf2",
      "-iter",
      "10000",
      "-in",
      "$input",
      "-out",
      "$output",
      "-k",
      "$password"
    ],
    "patterns_indicating_encrypted": [
```

```
        "bad decrypt",
        "incorrect password",
        "error while reading"
      ]
    }
  ]
```

Notes:

- file_type_id must match Core's internal file type detection. For OpenSSL enc (salted), use OPENSSL-SALT.
- executable_path must be absolute. On Windows JSON, escape backslashes.
- Arguments order mirrors a working OpenSSL decrypt command. Adjust cypher/iterations to your environment if files were encrypted differently.

2) Upload JSON and enable CLI Extraction in Core

1. Open Core UI and navigate to **Inventory** → **Modules** → **Utilities** → **Archive Extraction**.
2. Enable **Enable CLI Processing** >>Edit **Definition File Path** >> and provide the fully qualified path to the **CLI definition JSON file**
3. Save changes. Then **disable and re-enable the Archive Extraction engine** to load the new rules.

3) Test the integration

1. Encrypt a known benign file with OpenSSL on the Core server using the same parameters as your JSON (cipher, pbkdf2, iter).
2. Submit the encrypted file to Core. In Processing History, verify that:
   - The parent file type shows as an archive-like container with internal type OPENSSL-SALT.
   - Archive Extraction runs the CLI and produces decrypted child files under the parent, which then get scanned by engines/CDR/DLP as configured.

Operational guidance and best practices

- Iter count and cipher must match the way files were encrypted. If encryption used a different cipher (e.g., -des3 or -aes-128-cbc) or a different iteration count, mirror those values in the JSON arguments.
- Always re-enable the Archive Extraction engine after editing the CLI JSON so Core loads the new mapping.
- Store known passwords securely in Core's password mechanisms when possible. Avoid embedding secrets directly in JSON.

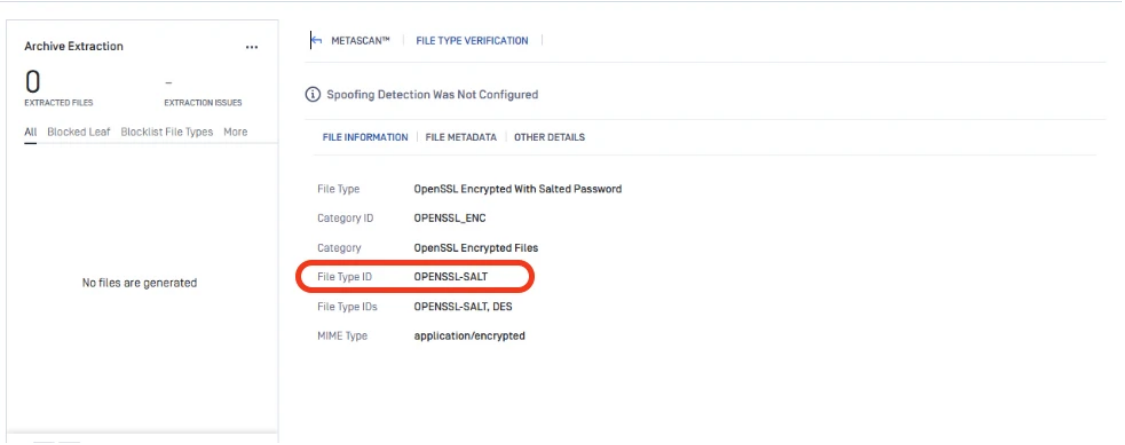
Troubleshooting

Common issues and resolutions drawn from the case and Core documentation:

- Wrong file_type_id: If you configured “des,” Core won’t trigger the CLI. Use OPENSSL-SALT for OpenSSL salted enc files.
- Path escaping on Windows: In JSON, use double backslashes for paths (e.g., O:\OpenSSL\bin\openssl.exe). Single backslashes or unescaped sequences can fail resolution.
- Engine not refreshed: After updating the JSON, disable and re-enable the Archive Extraction engine to apply changes.
- Command alignment: Ensure the decrypt command in JSON exactly matches a known-good manual command tested on the same host with the same OpenSSL build.

Validation checklist

- Scan shows FileTypeID for the scanned file as OPENSSL-SALT



- AV engines, CDR, and DLP process the child files as per the active workflow post its successful decryption

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

FAQ

- Do I need to restart the whole Core service after updating the JSON?
  - No. Disabling and re-enabling the Archive Extraction engine is sufficient to reload the CLI mapping.
- How do I pass the password to the CLI rule?
  - Include the $password placeholder in arguments and configure a password source (e.g., Password Storage or submission metadata). Core will substitute the runtime value.
- Can I use other ciphers or parameters?

- Yes. Adjust the arguments to match your actual encryption method (e.g., -des3, -aes-128-cbc, custom -iter). Ensure your test encrypt/decrypt commands succeed manually before updating the JSON.

How does the MD Core Handle Archive Files?

Check Your Version:

This article applies to all MD Core versions.

Overview:

When an archive file (such as .zip or .7z) contains child files (like Word documents, PDFs, or images), MD Core extracts the archive file, and while the parent file is being extracted, the child files are processed in parallel. After all child files have been extracted and processed, MD Core then processes the parent file with the other engines

Process steps: File intake & initial inspection (MD Core)

- The archive file is ingested by MD Core for initial validation.
- Basic metadata, headers, and file signatures are verified to ensure the archive is valid and not corrupted.

File Type Detection (FileType Engine)

- The FileType Engine determines the archive type (e.g., .zip, .7z, .rar).
- This step helps prevent spoofing attacks where malicious files disguise themselves as harmless archives.

Archive Extraction (Archive Extraction Engine)

- The archive is extracted to access its contents.
- If the archive contains child files (such as .docx or .pdf), these files are treated as individual items.
- For complex document formats (like Office files), note:
  - Office files (.docx, .xlsx, .pptx) are ZIP-based containers with XML and media files inside.
  - The Archive Extraction Engine extracts the container structure and files. Analyzing internal elements like macros, embedded objects, or external links is handled by the Deep CDR engine (and FSV in the future).
- If extraction fails or exceeds configured limits (e.g., file count, recursion depth), the file is blocked without further scanning.

Scanning Child Files (Layered Analysis)

After extraction, each child file is scanned independently. The following engines are processed sequentially in this order:

- Multiscan (Antivirus Engines): Scans each file for known malware signatures and suspicious patterns.

- Data Loss Prevention (DLP): Identifies sensitive or confidential data (such as credit card numbers or personal IDs).
- Deep Content Disarm & Reconstruction (Deep CDR):
  - Removes potential threats from documents and PDFs.
  - Generates a safe, rebuilt version of the file, free from harmful elements.
- Recursive scanning: If a child file is another archive or container file (e.g., a ZIP inside a ZIP), the engine repeats the extraction and scanning process until the configured recursion limit is reached.

Note:

DLP moves before Deep CDR to match the actual processing order.

Recompression & Final Output

After all child files are scanned, cleaned, or blocked, the remaining clean files are:

- Recompressed back into an archive (if necessary).
- Delivered individually, depending on system settings.
- MD Core generates a final result, including scan logs and detailed threat reports.

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

Enabling NGINX Debug Logging in MetaDefender Core

Check Your Version:

This article applies to all MD Core versions.

MetaDefender Core uses an embedded NGINX instance as a reverse proxy in front of the application services. In most cases, troubleshooting should rely on MetaDefender Core application logs. However, NGINX debug logging can be helpful in specific scenarios where issues occur **before the request reaches the Core engine**.

You may need NGINX debug logs when troubleshooting:

- Missing or incorrect **HTTP headers**
- Requests that appear to be **rejected or not processed**
- Differences between **successful and failed API calls**
- **Authentication or proxy-related issues**

- Validation of the **exact request received** by MetaDefender Core (method, headers, timing)

Debug logs provide full visibility into:

- Incoming request headers
- Request processing stages
- Upstream communication
- Connection handling

> ***Note**: Debug logging is very verbose and should be enabled only for temporary troubleshooting.*

1. By default, the runtime NGINX configuration is generated under:

`C:\Windows\Temp\ometascan\nginx.conf`

This should not be modified directly.

However, it includes the following directive:

```
include "C:/Program Files/OPSWAT/MetaDefender Core/nginx/*.conf";
```

This allows additional configuration files to be loaded.

2. Create a debug configuration file:

`C:\Program Files\OPSWAT\MetaDefender Core\nginx\debug.conf`

Add:

```
error_log "C:/Program Files/OPSWAT/MetaDefender Core/nginx/nginx-debug.log" debug;
```

3. Restart MetaDefender Core service:

- Open Services (`services.msc`)
- Restart: `OPSWAT MetaDefender Core`

4. The debug log will be available at:

`C:\Program Files\OPSWAT\MetaDefender Core\nginx\nginx-debug.log`

1. By default, the runtime NGINX configuration is generated under:

`/var/run/ometascan/nginx.conf`

This should not be modified directly.

However, it includes the following directive:

```
include /etc/ometascan/nginx.d/*.conf;
```

This allows additional configuration files to be loaded.

2. Create a debug configuration file:

```
sudo mkdir /etc/ometascan/nginx.d sudo vi /etc/ometascan/nginx.d/debug.conf
```

Add:

```
error_log /var/log/ometascan/nginx-debug.log debug;
```

3. Restart MetaDefender Core service.

```
sudo systemctl restart ometascan
```

4. The debug log will be available at:

```
/var/log/ometascan/nginx-debug.log
```

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

How Is Quarantined File Encryption Handled in MetaDefender Core?

Check Your Version:

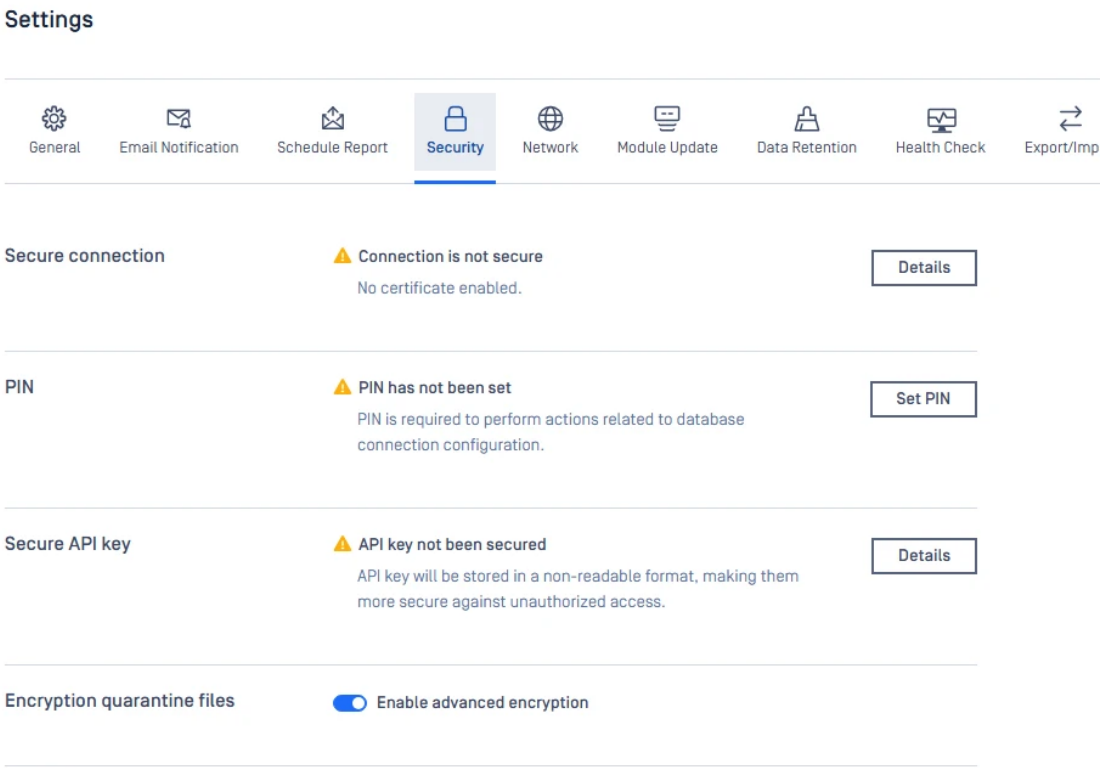
This article applies to all MetaDefender V5 releases deployed on Windows or Linux systems.

Overview

When "**Quarantine blocked files**" is enabled, MetaDefender stores blocked or malicious files in encrypted form. Files are saved without their original filename or extension.

Two encryption modes are available: **Standard Encryption (default, toggle off)** and **Advanced Encryption (optional, toggle on)**.

The configuration can be set in **Settings > Security > Encrypt Quarantined Files**.



Standard Encryption (Default)

Standard encryption is always applied when quarantine is enabled.

- Uses **XOR encryption** with a per-installation salt
- Ensures quarantined files are never stored in plaintext
- Provides basic protection against unauthorized access
- No configuration required

Advanced Encryption (Optional)

Stronger encryption can be enabled via: **Settings > Security > Enable advanced encryption**

- Uses **AES-256-GCM encryption** instead of XOR
- Provides enhanced confidentiality and integrity protection
- Applies only when manually enabled

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

How to exclude .txt files from the File Type Engine?

Overview

To optimize processing time for large flat files (such as .txt or .csv), you may want to prevent the File Type Engine from analyzing these file types. This can significantly reduce total processing overhead, especially in high-volume environments.

This guide explains how to exclude large (10–20 GB size) .txt files by configuring the engine to detect only specific file types.

Use Case

- Large .txt / .csv files are causing increased processing time
- File type detection is not necessary for these formats
- You want to keep File Type Engine enabled for other file types

Solution

Instead of disabling the File Type Engine entirely, configure it to detect only specific file types, effectively excluding .txt.

This is done using the “Detect specific file types only” feature.

Configuration Steps

1. Access your MetaDefender Core configuration.
2. Locate the File Type Engine settings.
3. Enable the option:

- Detect specific file types only

4. Define the list of file types you want the engine to process:

- Include all required formats (e.g., pdf, docx, exe, etc.)
- Do NOT include txt (or csv if also excluding)

5. Save and apply the configuration.

Example

If you configure detection for:

pdf, docx, xlsx, exe

Then:

- ✅ These file types will be analyzed by the File Type Engine
- ❌ .txt files will be skipped by this engine

Expected Result

- Reduced processing time for large .txt / .csv files
- File Type Engine continues to operate for relevant formats

- Improved overall system performance in file-heavy workflows

Additional Notes

- This does not block or reject .txt files — it only skips File Type Engine processing for them
- Ensure other engines or policies still meet your security requirements
- Monitor performance after the change to validate improvement

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

How to Correctly Update a Certificate in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows and Linux.

Introduction

MetaDefender Core 5.x supports HTTPS connections secured with SSL/TLS certificates. When a certificate approaches its expiration date or needs to be replaced for any other reason, it is important to follow the correct update procedure to avoid service disruption or loss of access to the management interface.

Skipping steps or modifying existing certificate files directly can leave the product in a state where the web UI is unreachable. This article describes the recommended steps for updating a certificate, and provides recovery instructions if the UI becomes inaccessible due to a misconfiguration.

Update Procedure

Follow the steps below in the exact order listed. Deviating from this sequence may result in loss of access to the management UI.

Step 1: Add the New Certificate to the Inventory

Navigate to Inventory > Certificates in the MetaDefender Core management interface and add the new certificate entry with the correct path, file name, and any associated configuration. At this stage:

- Do not modify the existing certificate files in any way.
- Do not replace old certificate files with the new ones on disk.
- The old certificate entry in the inventory must remain untouched until the new certificate has been fully activated.

Step 2: Enable HTTPS with the New Certificate

Once the new certificate has been added to the inventory, enable it by navigating to Settings > Security > Secure Connection > Details > Enable Certificate. Select the new certificate from the inventory, fill in all relevant details and options, and save the changes.

Note: Saving the changes will trigger an automatic restart of the MetaDefender Core service to apply the new certificate. Plan for a brief service interruption and notify any affected users accordingly before proceeding.

Step 3: Remove the Old Certificate from the Inventory

If the old certificate is no longer needed, remove it from the inventory first. Navigate to Inventory > Certificates, select the old certificate entry, and delete it from there. This step must be completed before deleting or moving the certificate files on disk.

Step 4: Delete or Move the Old Certificate Files

Only after the old certificate has been removed from the inventory can you safely delete the certificate files from storage or move them to an archive location. Deleting files before removing the inventory entry can cause startup or configuration errors.

Recovery: Resetting SSL Configuration When the UI Is Unreachable

If the steps above were not followed and the management UI is no longer accessible due to a certificate issue, MetaDefender Core provides a command-line utility to reset the SSL configuration back to defaults. Run the appropriate command for your platform.

Windows

Run the following executable from an elevated command prompt:

```
Code

C:\Program Files\OPSWAT\MetaDefender Core\ometascan-reset-ssl-conf.exe
```

Linux

Run the following commands from the terminal:

```
Code

export LD_LIBRARY_PATH=/usr/lib/ometascan/
/usr/sbin/ometascan-reset-ssl-conf
```

After running the reset utility, restart the MetaDefender Core service and reattempt to access the management interface. Once access is restored, follow the update procedure described in this article to correctly apply the new certificate.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Information regarding how files are processed and DB usage by Core

File Processing Workflow and DB Query Behavior

Initial Request Handling

Upon receiving a new file processing request, Metadefender Core performs the following:

- **Query Type:** `INSERT`
- **Purpose:** Log the initial state of the request in the primary request table
- **Timing:** Triggered as soon as a request enters the processing queue

This ensures traceability and readiness for downstream scanning actions.

AV Engine Full Scan Integration

Once we receive the complete scan report from **Metascan AV engines**, a secondary insert operation is performed:

- **Query Type:** `INSERT` (into a different table)
- **Purpose:** Store AV scan results associated with the file
- **Timing:** Executed immediately after all AV engines finish scanning

This result set is stored separately from the initial request metadata.

Final State Update

After all processing stages are completed—including but not limited to:

- Filetype validation engine
- AV engines
- CDR (Content Disarm & Reconstruction), if applicable
- DLP (Data Loss Prevention)

We finalize the request:

- **Query Type:** `UPDATE`
- **Purpose:** Reflect the final status of the request in the DB
- **Timing:** Triggered after successful completion of all components

Summary

Typical file processing involves **three DB queries**:

Stage	DB Operation	Purpose
Request received	`INSERT`	Record initial request state

Stage	DB Operation	Purpose
AV scan results completed	`INSERT`	Store detailed AV findings
All stages finalized	`UPDATE`	Mark the request as complete

MD Core & Reader Instances: Not Compatible

We've evaluated using **reader instances** in the database layer to potentially optimize performance. However:

- **Why we can't use them:** Our system requires frequent `INSERT` and `UPDATE` operations that reader instances typically do not support.
- **Implication:** MD Core must interact with write-enabled database instances to maintain data consistency and support real-time updates.

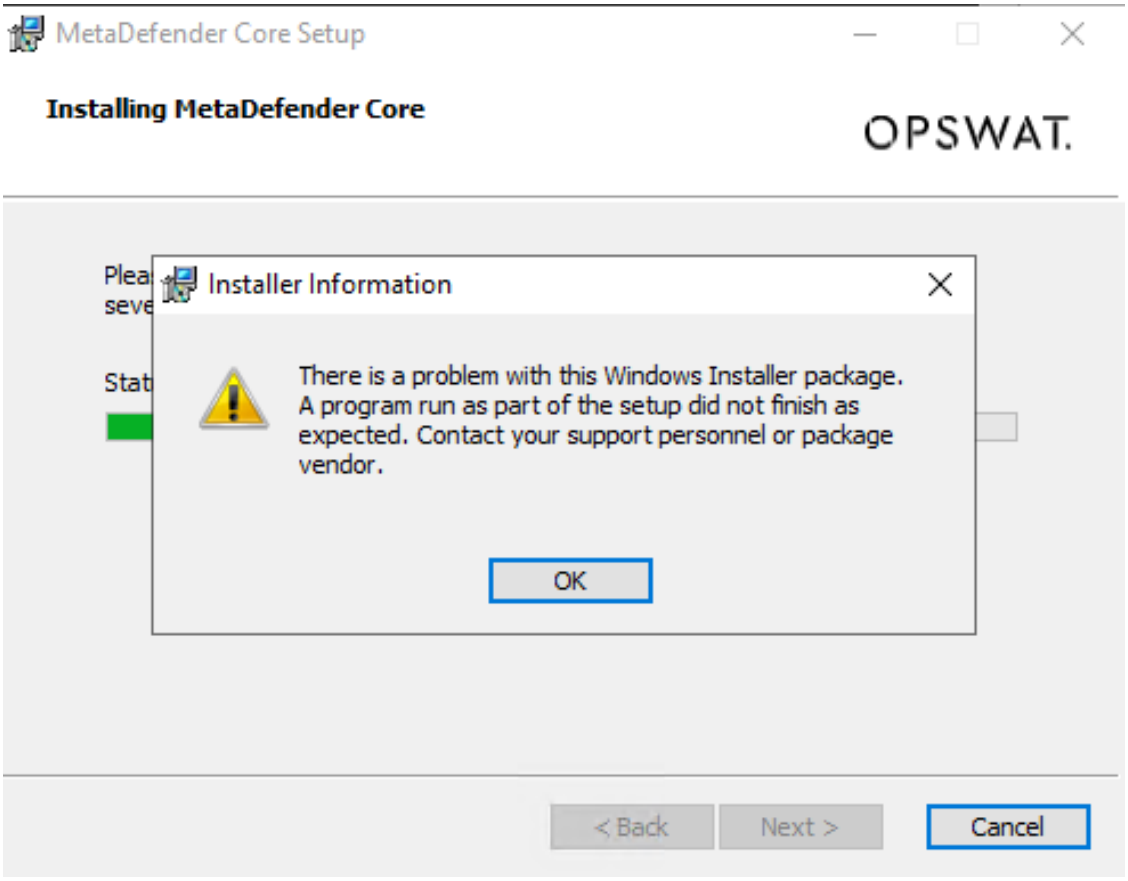
How to resolve "There is a problem with this Windows Installer Package" error during the installation of MetaDefender Core?

Check Your Version:

This article applies to only Windows version of Metadefender Core

Issue:

During the installation of MeteDefender Core on Windows environment, the installer popup window may appear and the MeteDefender application is not installed.



Root Cause:

One or more required prerequisite software packages are not installed on the Windows Server. The installer can't proceed without them.

Solution:

Ensure the Windows Server has all the following required perquisites software installed before running the MeteDefender Core setup.

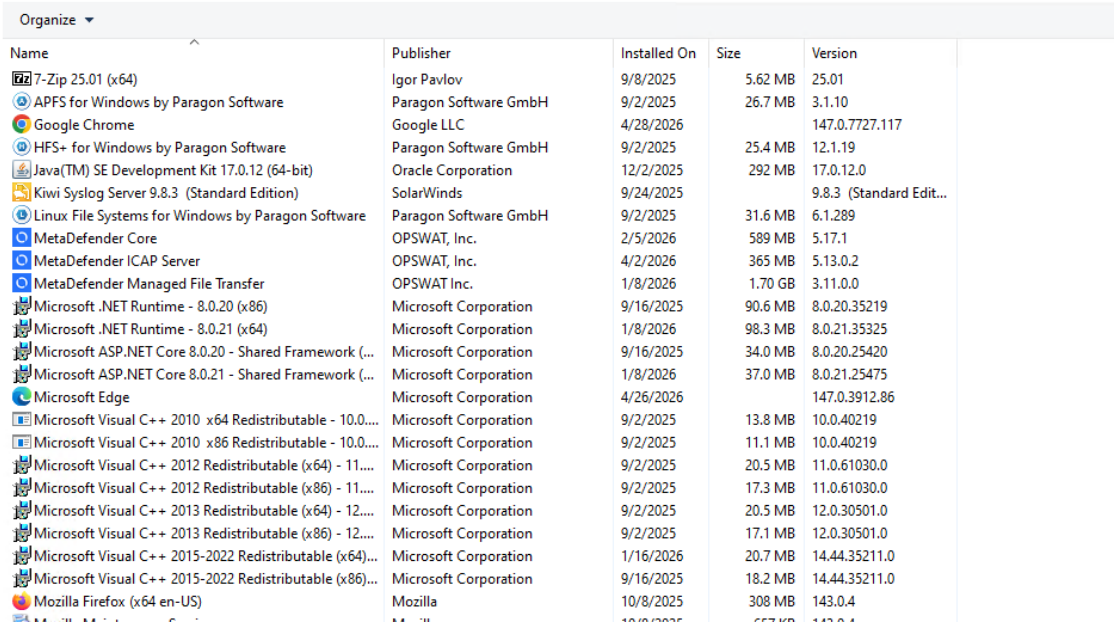
[Windows] Third Party Dependencies

- .NET framework 4.5 or above
- Microsoft Visual C++ Redistributable for Visual Studio 2015-2022 (version 14.38 or higher); x64 version.
- PostgreSQL: Only applicable when using a pre-installed PostgreSQL server running remotely.
  - From 14.5 to 14.22
  - From 15.7
  - From 16.4

Reference: https://www.opswat.com/docs/mdcore/installation/recommended-system-configuration#windows-installation-details

To verified that all the third party dependencies are installed, type **appwiz.cpl** on the command line to open the

installed programs list:



Organize

Name	Publisher	Installed On	Size	Version
7-Zip 25.01 (x64)	Igor Pavlov	9/8/2025	5.62 MB	25.01
APFS for Windows by Paragon Software	Paragon Software GmbH	9/2/2025	26.7 MB	3.1.10
Google Chrome	Google LLC	4/28/2026		147.0.7727.117
HFS+ for Windows by Paragon Software	Paragon Software GmbH	9/2/2025	25.4 MB	12.1.19
Java(TM) SE Development Kit 17.0.12 (64-bit)	Oracle Corporation	12/2/2025	292 MB	17.0.12.0
Kiwi Syslog Server 9.8.3 (Standard Edition)	SolarWinds	9/24/2025		9.8.3 (Standard Edit...
Linux File Systems for Windows by Paragon Software	Paragon Software GmbH	9/2/2025	31.6 MB	6.1.289
MetaDefender Core	OPSWAT, Inc.	2/5/2026	589 MB	5.17.1
MetaDefender ICAP Server	OPSWAT, Inc.	4/2/2026	365 MB	5.13.0.2
MetaDefender Managed File Transfer	OPSWAT Inc.	1/8/2026	1.70 GB	3.11.0.0
Microsoft .NET Runtime - 8.0.20 (x86)	Microsoft Corporation	9/16/2025	90.6 MB	8.0.20.35219
Microsoft .NET Runtime - 8.0.21 (x64)	Microsoft Corporation	1/8/2026	98.3 MB	8.0.21.35325
Microsoft ASP.NET Core 8.0.20 - Shared Framework (...	Microsoft Corporation	9/16/2025	34.0 MB	8.0.20.25420
Microsoft ASP.NET Core 8.0.21 - Shared Framework (...	Microsoft Corporation	1/8/2026	37.0 MB	8.0.21.25475
Microsoft Edge	Microsoft Corporation	4/26/2026		147.0.3912.86
Microsoft Visual C++ 2010 x64 Redistributable - 10.0....	Microsoft Corporation	9/2/2025	13.8 MB	10.0.40219
Microsoft Visual C++ 2010 x86 Redistributable - 10.0....	Microsoft Corporation	9/2/2025	11.1 MB	10.0.40219
Microsoft Visual C++ 2012 Redistributable (x64) - 11....	Microsoft Corporation	9/2/2025	20.5 MB	11.0.61030.0
Microsoft Visual C++ 2012 Redistributable (x86) - 11....	Microsoft Corporation	9/2/2025	17.3 MB	11.0.61030.0
Microsoft Visual C++ 2013 Redistributable (x64) - 12....	Microsoft Corporation	9/2/2025	20.5 MB	12.0.30501.0
Microsoft Visual C++ 2013 Redistributable (x86) - 12....	Microsoft Corporation	9/2/2025	17.1 MB	12.0.30501.0
Microsoft Visual C++ 2015-2022 Redistributable (x64)...	Microsoft Corporation	1/16/2026	20.7 MB	14.44.35211.0
Microsoft Visual C++ 2015-2022 Redistributable (x86)...	Microsoft Corporation	9/16/2025	18.2 MB	14.44.35211.0
Mozilla Firefox (x64 en-US)	Mozilla	10/8/2025	308 MB	143.0.4

After the verification of the required software are installed, proceed with the installation of MeteDefender Core again.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Can an old SSL certificate be safely removed from the Windows Local Machine certificate store after updating the HTTPS certificate for MetaDefender Core?

Check your version:

This article applies to MetaDefender Core and is applicable to all MetaDefender Core V5 releases deployed on Windows systems.

Overview

After updating the SSL certificate used for HTTPS communication by OPSWAT MetaDefender Core on a Windows platform, administrators may notice that the previous certificate is still present in the Windows certificate store. This article explains whether the old certificate can be safely removed and whether any prerequisites are required.

Why the old certificate may still appear on Windows

On Windows systems, updating an SSL certificate does not automatically remove the old certificate from the certificate store. MetaDefender Core updates its HTTPS configuration to reference the new certificate, but the

previous certificate may remain unless it is manually deleted. Its presence alone does not indicate that it is still in use.

What to verify before removing the certificate on Windows

Before deleting the old certificate from the Windows certificate store, confirm the following:

- MetaDefender Core is operating correctly over HTTPS using the new certificate
- The active HTTPS configuration in the MetaDefender Core web console under Settings > Security > Secure connection > Details displays the expected certificate
- The HTTPS configuration references the new certificate thumbprint
- The old certificate is not referenced by any active HTTPS bindings or Windows services

If these checks are satisfied, no additional preparation is required.

Is it safe to delete the old certificate from Windows?

Yes. If the old certificate is not referenced by MetaDefender Core or any other Windows application or service, it can be safely removed directly from the Windows certificate store. No HTTPS reset, reconfiguration, or service restart is required in this scenario.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Can MetaDefender Core scan GPG encrypted files and how can I configure it to do so?

Check your version:

This article applies to MetaDefender Core and is applicable to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

Overview

When scanning files encrypted with GPG such as .txt.gpg files, MetaDefender Core may return the status "**Document is encrypted with a password"**. This can occur even when a password is provided, depending on how the file type is detected and processed.

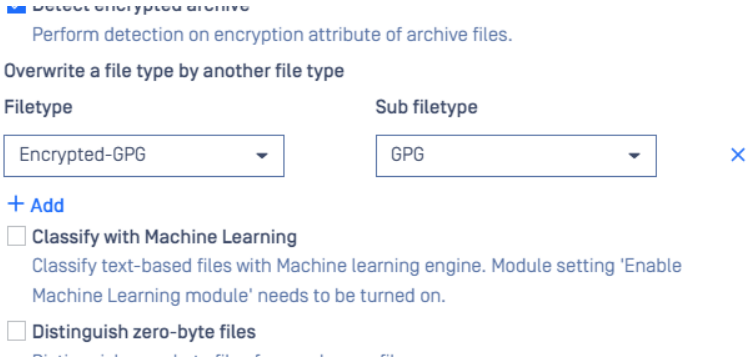
This behavior is not necessarily a limitation, but it may require a workflow configuration adjustment so the system correctly identifies and handles GPG encrypted files.

Solution

You can configure MetaDefender Core to properly recognize GPG encrypted files by overriding the file type in the workflow settings.

Follow these steps:

1. Open the MetaDefender Core web UI
2. Navigate to Workflow Management and then Workflows
3. Select the workflow you are using for scanning
4. Go to the File Type tab
5. Scroll down to the section labeled **Overwrite a file type by another file type**
6. Click Add
7. In the Filetype field, select Encrypted GPG
8. In the Sub filetype field, select GPG
9. Save the workflow



After applying this configuration, MetaDefender Core will properly detect GPG encrypted files.

Support:

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

What can I do if MetaDefender shows a high number of files that failed to scan?

Check Your Version:

This article applies to all MetaDefender V5 releases deployed on Windows or Linux systems.

Under normal circumstances, all active engines scan every file.

However, there are some situations in which scanning might be impacted and a “failed to scan” result is observed. In such situations, the behavior may be caused by a number of factors:

- Some engines failed to scan, and exceeded the scan failure threshold settings in place on the Core side.
- Core itself has a fatal problem e.g. Core service crashed.
- Due to configurations:
  - Failed the scan when sanitization timeout or failed.
  - Failed the scan when archive extraction failed.

Please Note:

MetaDefender V5 include an option that modifies the behavior of the product when a specific engine fails to scan - such as in the case mentioned above.

Here, MetaDefender Core can be configured to fail the scan if any of the engines report problems. In other words, it can toss out all partially complete scans.

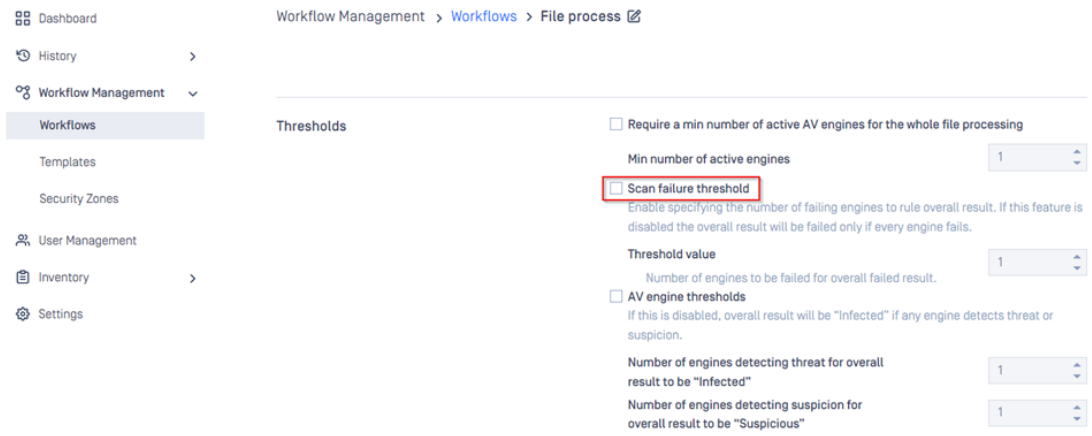
Failing to disable this option can give the illusion that MetaDefender Core is failing to scan large amounts of files.

Metadefender Core V5 Releases

To disable this option on V5, follow these steps:

1. In your web browser, navigate to the MetaDefender Core **Management Console** at http://localhost:8008/ (you may need to log in under admin privilege)
2. On the left menu, navigate to **Workflow Management** → **Workflow** to list all available rules
3. Click on a workflow you are using for the current scan to modify the setting
4. Click on the **Metascan** tab

Uncheck the option **Scan Failure Threshold** shown below.



Support:

If you have followed the instructions above but are still seeing a high number of files logged as **Failed to Scan**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a

support case or chatting with our support engineer.

What can I do when MetaDefender Core (Windows) has an insufficient disk space error?

Check Your Version:

This article applies to all MetaDefender Core (Windows) versions

While using MetaDefender Core, you may occasionally encounter a **full disk space issue**, which can cause the Core service to stop functioning. MetaDefender Core includes an **Auto Retention** mechanism designed to automatically remove older entries from the database to free up space.

However, this process works in two steps:

- **Old entries are marked for deletion** based on the configured retention policy.
- **The actual deletion is triggered automatically** at a later time.

In some cases, the disk may become full **before** the deletion process is initiated. Since the deletion also requires a minimal amount of free space to execute, this can prevent the cleanup from starting.

To manually free up disk space and allow MetaDefender Core to resume normal operation, consider the following steps to remove unnecessary files or data:

1. **Manually delete/move the log folders:**

Warning:

Recommended to stop the MetaDefender Core's services before proceeding with these steps

This method can help you free up disk space and minimize disruption to the MetaDefender Core server's operation. You can move/delete all the log folders/files located inside these locations:

- `C:\Program Files\OPSWAT\MetaDefender Core\data\logs`, **except the file "core.log"**
- `C:\Program Files\OPSWAT\MetaDefender Core\data\pg_data\log`

Info:

Moving is recommended to keep the old logs for future needs.

2. **Manually delete/move the engine folders:**

Follow this article for detailed instructions: How do I manually clean up the MD Core engine's local folder?

Warning:

Deleting MetaDefender Core Engines will require an engine update after that.

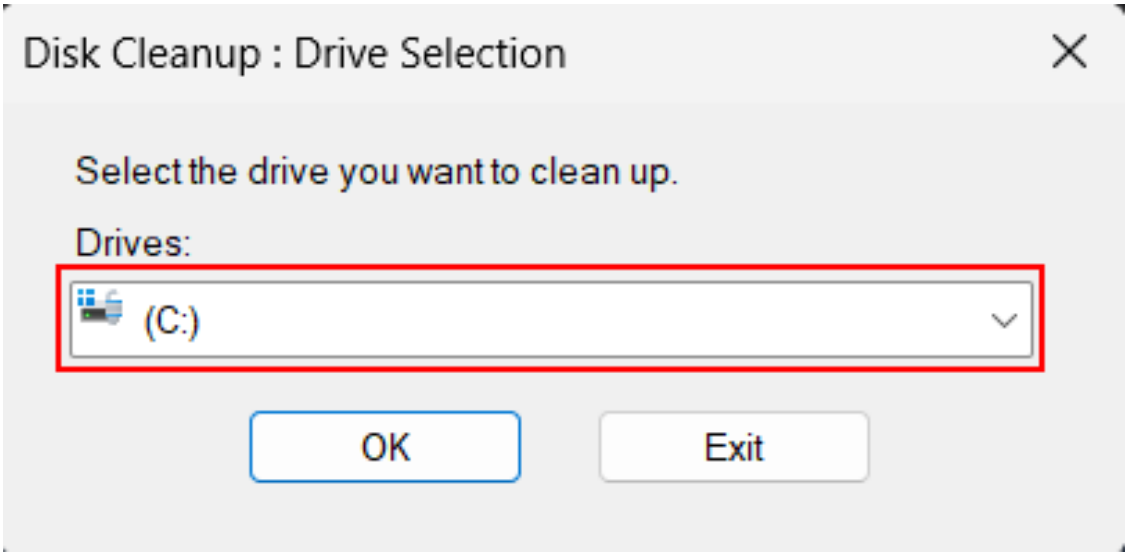
3. **Using the Windows Disk Cleanup tool:**

This method will not interfere with the MetaDefender Core installation files, but the Windows unnecessary files.

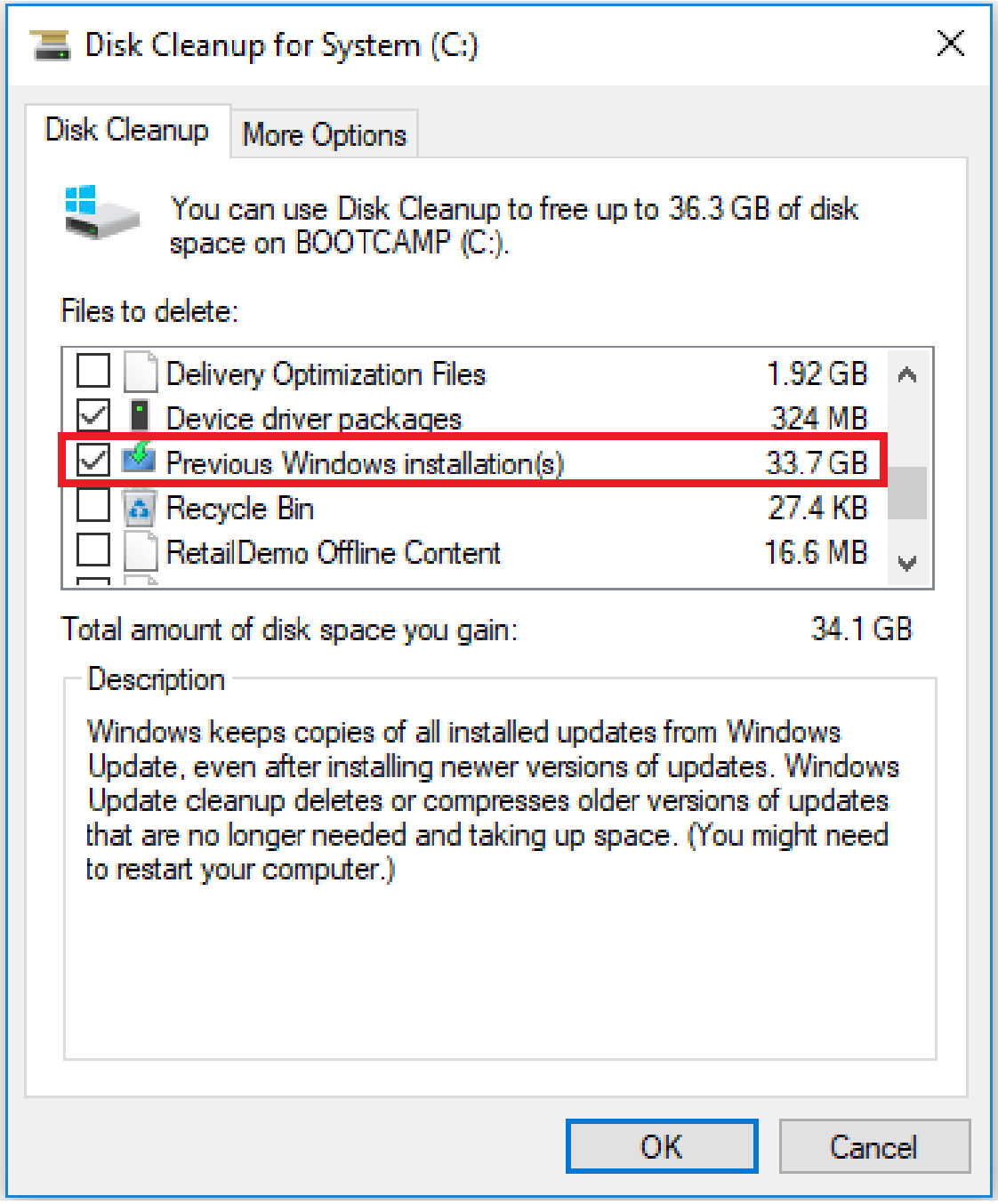
Info:

Only apply this when your install location is on the C drive.

- From Run, type `cleanmgr` then enter.
- Choose the C drive on the next pop-up.



- From the Disk Cleanup, choose Cleanup system files, confirm the C drive again.
- Select as much as possible from the list. Recommended to choose `Windows Update Cleanup`, `Previous Windows installation(s)`, `Temporary files`



Choose OK in the end to start the Windows Disk Cleanup.

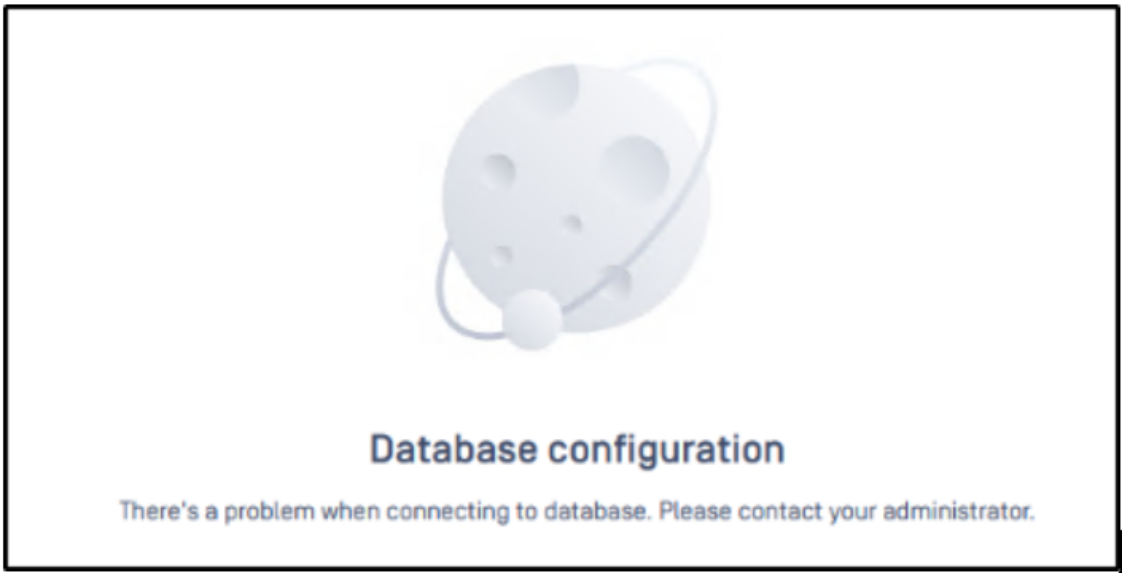
Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with our support engineer**.

Why is MetaDefender Core unable to connect to the database?

Issue

MetaDefender Core fails to connect to the database. The following is displayed when connecting to the MetaDefender Core web console:



The following error is displayed in the core.log:

> Error
>
> "[ERROR] <timestamp>:(core.postgres) Postgres issue, error='Could not load postgres's information from file', param-'config file is invalid, ..."

Cause

The `ometascan_postegres.cat` and `ometascan_postgres.dat` files have become corrupted.

Remediation Steps

1. Stop the MetaDefender Core Service

2. Backup the corrupted Files

Backup the corrupted *ometascan_postgres.dat* and *ometascan_postgres.cat* files to a different location for future reference.

- Windows: The files are located in the Core's installation folder.
- Linux: The files are located at:

FIle List · bash

```
/usr/lib/ometascan/ometascan_postgres.cat
/usr/lib/ometascan/ometascan_postgres.dat
```

3. Remove the Corrupted Files *ometascan_postgres.dat* and *ometascan_postgres.cat* files.

Delete the corrupted *ometascan_postgres.dat* and *ometascan_postgres.cat* files.

- Windows: Navigate to the Core's installation folder and delete the files.
- Linux: Run the following commands:

shell · bash

```
sudo rm /usr/lib/ometascan/ometascan_postgres.cat
sudo rm /usr/lib/ometascan/ometascan_postgres.dat
```

4. Create the Ignition File

Create an ignition file with the specified content.

- Windows: Create `C:\opswat\ometascan.conf`
- Linux: Create `/etc/opswat/ometascan.conf`

> **Note:**
> If you are restoring an existing database, you will need to provide the username and password credentials used previously. Please consult your internal change management documentation to get these credentials.

Add the following content to the file:

Ignition File

```
[dbserver]
type=your db type, 'local' for bundle postgres, 'remote' for remote postgres.
host=ip address of remote postgres. For bundle postgres, it has to be localhost.
port=port of postgres.
user=postgres's user.
password=postgres's password.
```

Example of an ignition file using bundled postgres:

Ignition File

```
[dbserver]
type=local
host=localhost
port=5432
user=mypostgresuser
password=mypostgrespassword
```

Customers who specified an instance name during initial Core installation will need to enter the correct value for instance_name in the ignition file. If not, Core will initiate a new database with the default instance name.

Example:

Ignition File · xml

```
[dbserver]
type=local
host=localhost
port=5432
user=mypostgresuser
password=mypostgrespassword
instance_name=myinstancename
```

If customers have configured a private user for their database and want to reuse this private user, they need to ensure that correct private_username and private_password are set in the ignition file.

Example:

Ignition File · xml

```
[dbserver]
type=local
host=localhost
port=5432
user=mypostgresuser
password=mypostgrespassword
private_username=myprivateusername
private_password=myprivatepassword
```

5. Re-generate `ometascan_postgres.cat` and `ometascan_postgres.dat`

Windows

1. Open Command Prompt as Administrator
2. Navigate to the MetaDefender Core's installation folder.
3. Run the following command:

CMD · bash

```
ometascan-upgrade-db.exe
```

Linux

Export:

Code · bash

```
LD_LIBRARY_PATH=/usr/lib/ometascan:$LD_LIBRARY_PATH
```

Run the upgrade database tool using:

Code · bash

```
sudo /usr/sbin/ometascan-upgrade-db
```

Change the owner and set permissions for the newly generated ometascan_postgres.cat and ometascan_postgres.dat files:

Code · bash

```
sudo chown metascan:metascan /usr/lib/ometascan/ometascan_postgres.cat
sudo chown metascan:metascan /usr/lib/ometascan/ometascan_postgres.dat
sudo chmod 777 /usr/lib/ometascan/ometascan_postgres.cat
sudo chmod 777 /usr/lib/ometascan/ometascan_postgres.dat
```

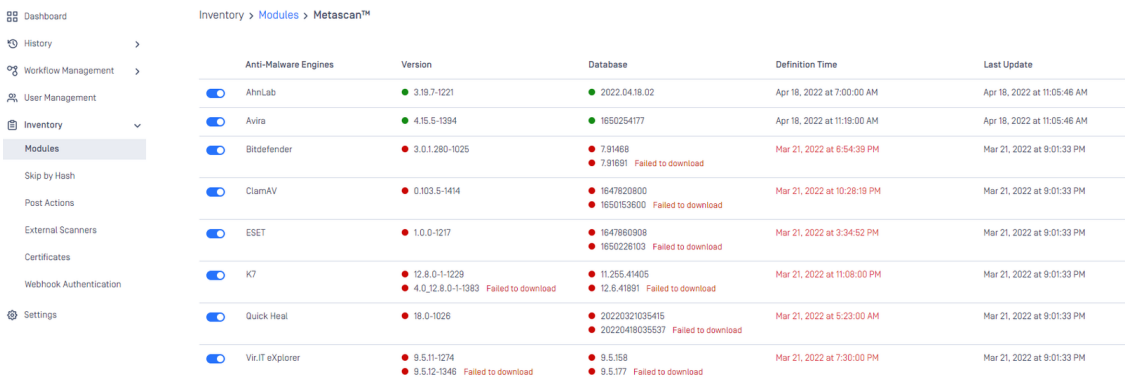
6. Start MetaDefender Core Service

Conclusion

Following these steps should resolve the issue of MetaDefender Core failing to connect to the database due to configuration files corruption. If the problems persist, please contact OPSWAT support for further assistance.

What should I do if an engine is in "failed" or "permanently failed" status?

Sometimes, during the engines downloading/deployment process, some of them may remain in "**failed**" or "**permanently failed**" status.



In this case, you can do the following:

- Check problematic engine's dependencies here : Recommended System Configuration
- Check system requirements to ensure number of CPU cores and memory met.
- Check the system's available memory where MD Core is installed, it should be available at least 3GB.
- Check MD Core's issues if any on "Notifications" e.g. disk space issue, it should be available at least 5GB.
- Check Internet connection (if using online update method) to ensure stable and avoid enigne package download

being corrupted.

- Try:
  - Disable and re-enable engine again.
  - If still not working, remove entirely both engine and its database, and then trigger update again.
- If still not working eventually after all checks above, collect MD Core support package and send to OPSWAT support.

Support:

If you have followed all of these steps and your engines are still unusable, please proceed to create a support case or chat with our support engineer.

What are the engine clean-up instructions?

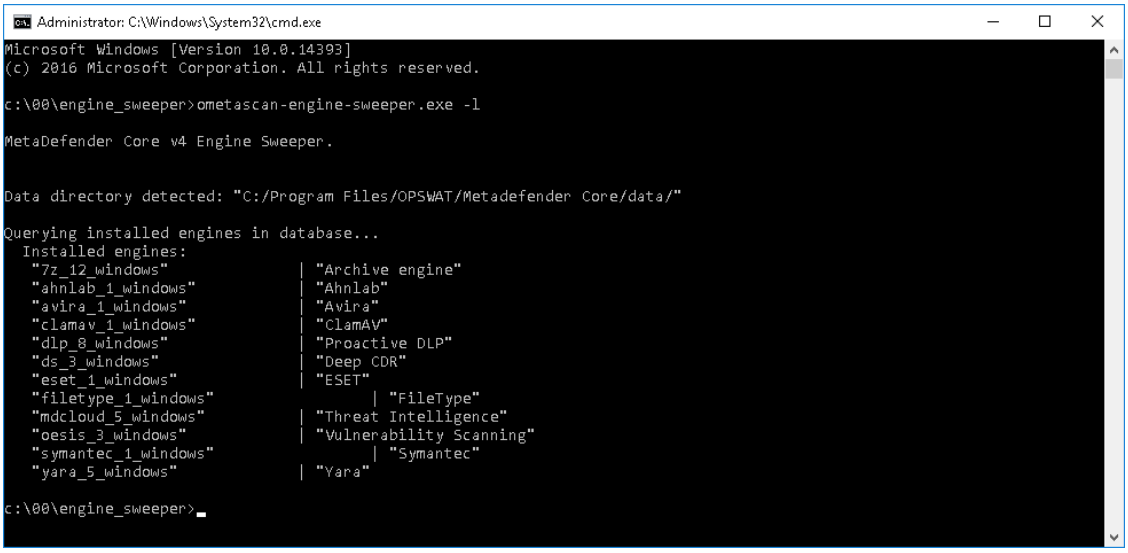
Info

This tool only works on MetaDefender Core **4.18.0 and older**.For Metadefender Core **4.19.0+**, please use the instructions found here: Engine Clean-up Tool

For MetaDefender Core 4.19.2 or newer, all engines could be removed directly on the MetaDefender Core management console UI under Modules page or you can consult this article : How to Remove Engines from MetaDefender Core using command line?

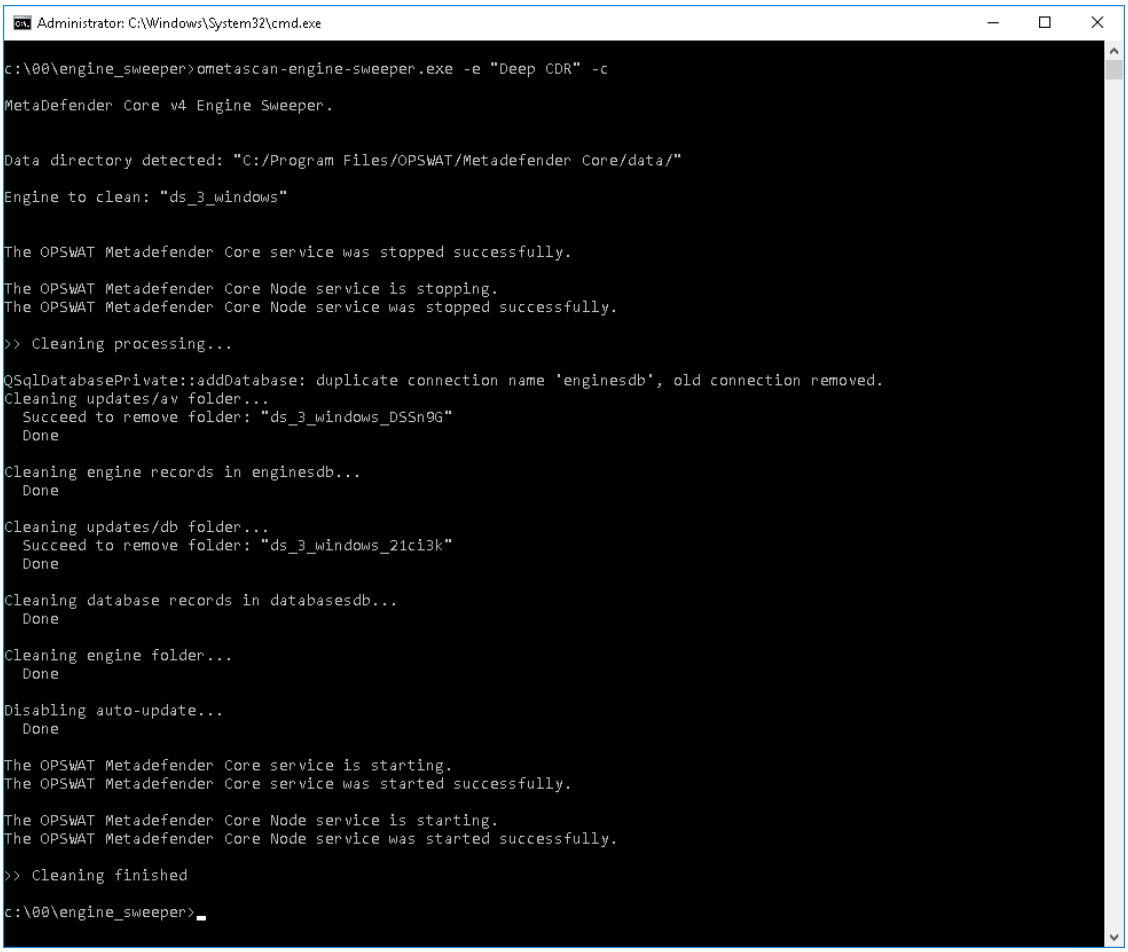
Sometimes, during the engines downloading/deployment process, some of them may remain in "**failed**" or "**permanently failed**" status. In this case, you can perform an engine clean-up by downloading the engine_sweeper tool and following the next steps:

1. Extract the engine_sweeper.zip package to a temporary location on the server where MetaDefender Core is installed
2. Open a Command Prompt window **as Administrator** and navigate to the temporary folder where you have extracted the archive
3. Run the command **ometascan-engine-sweeper.exe -l** to display a list of all installed engines



4. Run the command **ometascan-engine-sweeper.exe -e <engine name from above> -c** to clean the engine having the issues

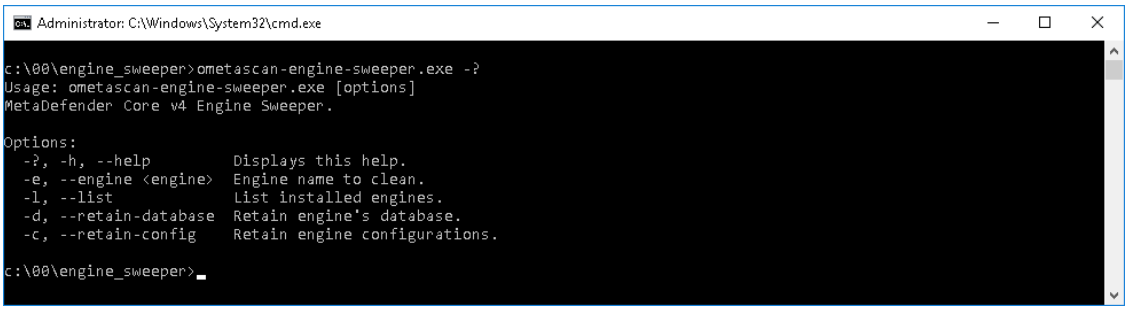
Note: During the cleanup process, the MetaDefender Core and Node services will be **restarted**, and the Updates Settings Source will be set to "**Manual**"



5. Check the engine in Inventory > Modules as it should now be grey. (e.g. Deep CDR in the following screenshot)



6. Revert your Updates Settings Source from "Manual" to your previous setting and trigger an update, for the engine to re-deploy successfully
7. More parameters that can be used with this tool can be found in the next screenshot



Support:

If you have followed all of these steps and your engines are still unusable, please see How to Create Support Package With Bundle Tools?, then please proceed to create a support case or chat with our support engineer.

Why is the scan stuck in "processing" state on WebScan UI, when the Core Processing History shows that it is already finished?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

If a file has been submitted for scan via WebUI, WebUI will make 999 queries as part of the polling process.

If file analysis progress hasn't reached 100% by the time the polling stops, the WebUI will appear to be frozen.

Remedy:

A simple webpage refresh will reset the query counter and allow the WebUI to continue fetching results.

Support:

If you have refreshed the webpage but the WebUI is still stuck in a **Processing** state, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why have the ESET scan engine failed to initialize on the hardened Linux OS?

Check Your Version:

This article applies to all MetaDefender Core V5 and V4 releases deployed on hardened Linux systems.

Pro Tip:

For customers using a hardened Linux OS, such as Hardened Amazon Linux 2, the ESET engine will fail to initialize unless certain changes are made.

The ESET scan engine require execution permission on a temp folder. However, while on the hardened system, such permissions are restricted on: **/var/tmp**, resulting in the engine's failure to initialize.

Browse MetaDefender Configuration for instructions on how to change the default temp folder (**tempdirectory**) to a folder that has execution permission.

Support:

If you have followed all of the steps above but your **ESET** engine are still failing to initialize, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why has the Symantec custom engine failed to deploy?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

To install and deploy MetaDefender Core's Symantec custom engine, certain **Registry Keys** need to be present in the registry.

If these keys do not exist, the Symantec engine installation process will time out.

To install the Symantec custom engine in MetaDefender Core, you can manually add the missing registry keys by completing the following steps:

Please Note:

Registry key names and values are case-sensitive.

1. Open the Windows registry (**regedit.exe**)
2. Navigate to **HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters**
3. Create a new **Value Name** with the type **Multi-String Value** and the name: **NullSessionPipes**
4. **Save** the changes
5. Remove the Symantec engine from your MetaDefender Core system
6. **Uninstall** the Symantec Protection Engine manually from Windows
7. Navigate to the location where the Symantec engine is located (**Program Files\Symantec**) and **Delete** this folder manually
8. Click **Update all** in the **Modules** page to reinstall the engine.

Support:

If you have followed the instructions above but are still experiencing a Symantec installation **timeout**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What are common reasons for Deep CDR scan/sanitization failures and how do I fix them?

Check Your Version:

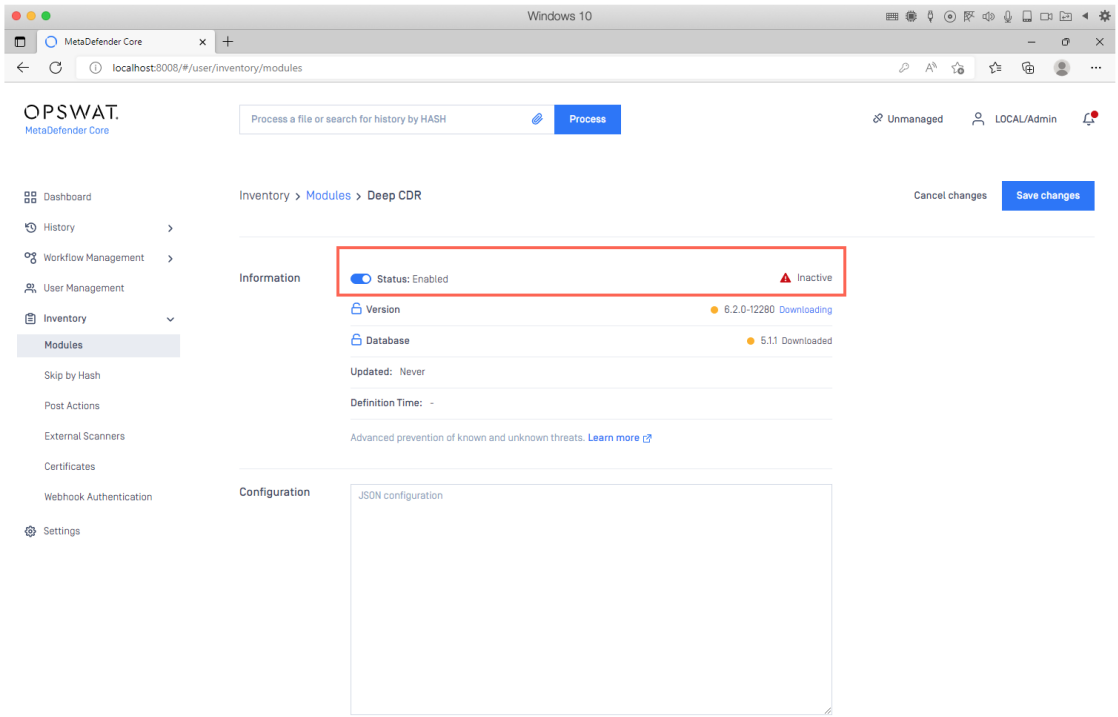
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Please Note:

In order to function, MetaDefender Core's **Deep Content Disarm and Reconstruction** feature (Deep CDR) must

be:

- included under your unique license key
- enabled via **MetaDefender Core Management Console>Inventory>Modules>Deep CDR**
- fully downloaded (engine and current database)
- listed as **Active** instead of **Inactive** as it is in the screenshot below.



Once these conditions are met, you should be ready to run all supported file types through MetaDefender's Deep CDR engine as part of your chosen Workflow. If not, and if one or more of your deep CDR scans fail, this is likely due to one of the following:

You are attempting to process a supported file type, but the version is outdated:

- For example, CDR is unable to process an Office file format created before MS Office 97 or a DWG file created with AutoCAD before version 2004.

You are attempting to process an invalid file structure:

Files that have become corrupted (damaged and unreadable) by hardware or software malfunctions or by attackers, these files will be blocked by MetaDefender Core, but they cannot be sanitized by Deep CDR. Read here for more information.

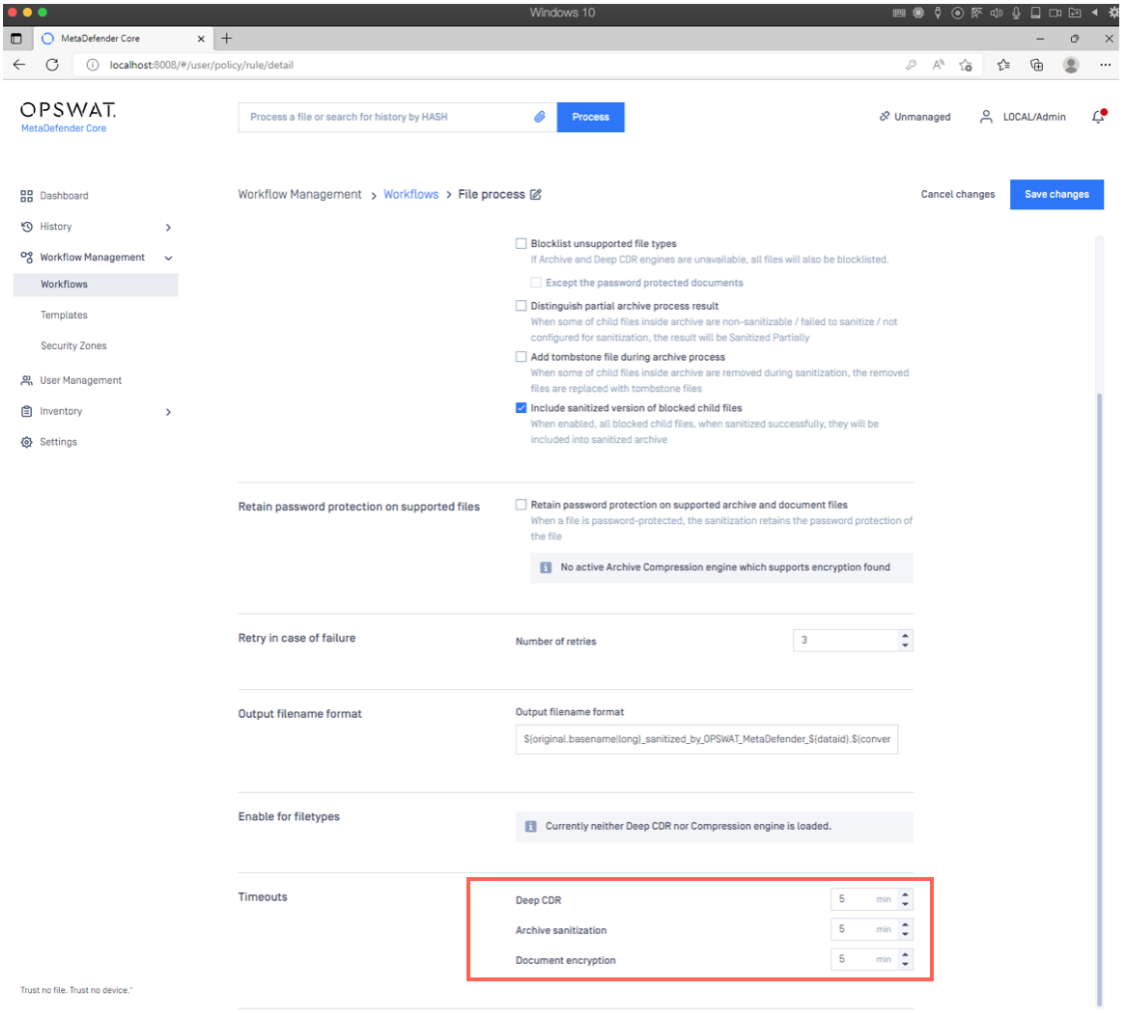
When retrieving your scan results via REST API, Deep CDR sanitization failures will be clearly classified under the **post_processing** section in your JSON output, as illustrated in the screenshot below.

```
...
"post_processing": {
  "actions_failed": "Sanitization Failed",
  "actions_ran": "",
  "converted_destination": "",
  "converted_to": "",
  "copy_move_destination": "",
  "sanitization_details": {
    "failure_category": "invalid file structure",
    "details": "Startxref not found"
  }
},
...
```


File processing has exceeded your configured settings for Deep CDR:

- Go to your **Core Management Console>Workflow Management>{Relevant Workflow}>Deep CDR** to configure settings such as Deep CDR, sanitization and encryption timeouts, illustrated in the screenshot below.

Always be sure to click **Save changes** before exiting the Workflow.



You are attempting to process a password-protected file:

- When a file is password protected or no active archive compression engine supports the encryption found, the file cannot be opened and will therefore not be sanitized by Deep CDR.

Other, unknown reasons:

- If unspecified failures occur, please reach out to the OPSWAT support team by following the instructions at the end of this article.

Support:

If you have followed the relevant instructions above but are still having difficulty **Scanning and Sanitizing Files With Deep CDR**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What do the various error messages mean in MetaDefender Core Rest API?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

For users interacting with MetaDefender Core via REST API, it is essential that you familiarize yourself with the meaning of the various coded responses and error messages that the Core might return during the course of your system administration and daily file processing flow.

Please Note:

The tables below cover possible responses to all aspects of the REST interaction process, including:

- authorization
- analysis
- batch processing
- administration
- configuration
- YARA
- engines
- stats.

It is important to note that, while messages may often feature the same codes, their meanings are usually dependent on the nature of your API request, as illustrated in the following tables.

Auth

API Request	API Response	Meaning

API Request	API Response	Meaning
POST /login	200	OK
	404	Invalid credentials
	500	Unexpected event on server
POST /logout	200	OK
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information
	500	Unexpected event on server
POST /user/changepassword	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	405	The user has no rights for this operation
	500	Unexpected event on server

Analysis

API Request	API Response	Meaning
POST /file	200	Successful file submission
	400	Invalid callback URL and/or API key
	403	Invalid user information or not allowed
	411	Content-length header is missing from the request

API Request	API Response	Meaning
	422	Body input is empty
	500	Unexpected event on server
	503	Server is too busy, scan queue is full, try again later
POST /file/sync	200	The scan is completed before connection timeout
	400	Invalid API key
	403	Unsupported Callback URL
	408	Timed out response: scan is still in progress, but the connection is timed out
	411	Content-length header is missing from the request
	422	Body input is empty
	500	Internal server error
	503	Server is too busy, scan queue is full, try again later
GET /file/{data_id}	200	Entire analysis report generated by MetaDefender Core
	500	Unexpected event on server
GET /file/{data_id}/blocked-leaves	200	Returns list of blocked leaf files
	404	Request's resource was not found
	405	The user has no rights for this operation
	500	Unexpected event on server

API Request	API Response	Meaning
GET /archive/{data_id}	200	Entire analysis report, containing all extracted files, generated by MetaDefender Core
	500	Unexpected event on server
GET /hash/{md5|sha1|sha256}	200	Get information or file
	404	Invalid hash format
GET /file/webhook/{data_id}	200	Webhooks status fetched successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	404	Request's resource was not found
	500	Unexpected event on server
GET /file/rules	200	Returns the list of available rules
	500	Unexpected event on server
GET /file/converted/{data_id}	200	Returns the sanitized content
	404	Request's resource was not found
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /file/download/{data_id}	200	Returns the sanitized or DLP processed content
	404	Request's resource was not found

API Request	API Response	Meaning
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /quarantine/{sha256}/download	200	Returns the quarantined file
	403	Invalid user information or not allowed
	404	Request's resource was not found
	500	Unexpected event on server
POST /file/{data_id}/cancel	200	Batch canceled or analysis successfully canceled
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	404	Batch not found (invalid ID) or request resource not found
	405	The user has no rights for this operation
	500	Unexpected event on server

Batch

API Request	API Response	Meaning
POST /file/batch	200	Batch created successfully
	400	Bad request (e.g. invalid header,

API Request	API Response	Meaning
		API key missing or invalid)
	403	Invalid user information or not allowed
	500	Unexpected event on server
POST /file/batch/{batchId}/close	200	Batch request successfully closed
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	404	Request's resource was not found
	500	Unexpected event on server
GET /file/batch/{batchID}	200	Batch progress paginated report (50 entries/page)
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	404	Request's resource was not found
	500	Unexpected event on server
GET /file/batch/{batchId}/ certificate	200	Signed batch result and certificate are sent back in response body (YAML format)
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed

API Request	API Response	Meaning
	404	Request's resource was not found
	500	Unexpected event on server
POST /file/batch/{batchId}/cancel	200	Batch canceled
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	404	Batch not found (invalid ID)
	500	Unexpected event on server

Admin

API Request	API Response	Meaning
GET /admin/export	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /admin/export/v2	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not

API Request	API Response	Meaning
		allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
POST /admin/import	200	Request processed successfully
	304	The configuration has not changed
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
POST /admin/import/v2	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
POST /admin/user	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)

API Request	API Response	Meaning
	403	Invalid user information or not allowed
	404	Batch not found (invalid ID)
	500	Unexpected event on server
POST /admin/role	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /admin/license/backup	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	500	Unexpected event on server
POST /admin/license/backup	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	500	Unexpected event on server
POST /admin/license/backup/	200	Request processed successfully

API Request	API Response	Meaning
nomination		
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	500	Unexpected event on server

License

API Request	API Response	Meaning
POST /admin/license/activation	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	500	Unexpected event on server
POST /admin/license	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	500	Unexpected event on server
GET /admin/license	200	Request processed successfully
	403	Invalid user information or not allowed
	500	Unexpected event on server

API Request	API Response	Meaning

Config

API Request	API Response	Meaning
PUT /admin/config/auditlog	200	Successful file submission
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/quarantine	200	The scan is completed before connection timeout
	403	Unsupported Callback URL
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/sanitize	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/scanhistory	200	Request processed successfully
	403	Invalid user information or not allowed

API Request	API Response	Meaning
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/session	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/update	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /admin/config/skip	200	A list of all skip/white/black listed hashes
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/skip	200	Put a list of all skip/white/black listed hashes to the server
	403	Invalid user information or not allowed

API Request	API Response	Meaning
	404	Request's resource was not found
	500	Unexpected event on server
POST /admin/config/skip	200	Add a list of all skip/white/black listed hashes to the server
	403	Invalid user information or not allowed
	404	Batch not found (invalid ID) or request resource not found
	500	Unexpected event on server
DELETE /admin/config/skip	200	Delete list of all skip/white/black listed hashes successfully
	403	Invalid user information or not allowed
	404	Request's resource was not found
	500	Unexpected event on server
GET /admin/config/webhook	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/webhook	200	Request processed successfully
	403	Invalid user information or not allowed

API Request	API Response	Meaning
	404	Request's resource was not found
	500	Unexpected event on server
GET /admin/config/proxy	200	A proxy list is returned
	403	Invalid user information or not allowed
PUT /admin/config/proxy	200	A proxy list is returned
	400	Bad request (e.g. invalid header, API key missing or invalid)
POST /admin/config/proxy/testconnection	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
GET /admin/config/file/sync	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Error while requesting information
PUT /admin/config/file/sync	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Error while modifying configuration

API Request	API Response	Meaning
GET /admin/config/sessioncookie	200	Request processed successfully
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
PUT /admin/config/sessioncookie	200	Request processed successfully
	403	Invalid user information or not allowed
	404	Request's resource was not found
	500	Unexpected event on server
GET /admin/config/healthcheck	200	Health check criterias to determine whether MetaDefender Core server status is healthy or not
	403	Invalid user information or not allowed
	500	Unexpected event on server
POST /admin/config/healthcheck	200	Health check criterias to determine whether MetaDefender Core server status is healthy or not
	400	Bad request (e.g. invalid header, API key missing or invalid)
	500	Error while modifying configuration

YARA

API Request	API Response	Meaning
GET /admin/config/yara/sources	200	Returns a list of YARA engine sources
	403	Invalid user information or not allowed
	500	Unexpected event on server
PUT /admin/config/yara/sources	200	Returns a list of YARA engine sources
	403	Invalid user information or not allowed
	500	Unexpected event on server
GET /yara/package	200	Request processed successfully
	403	Invalid user information or not allowed
	500	Unexpected event on server
POST /yara/generate	200	Request processed successfully
	403	Invalid user information or not allowed
	500	Unexpected event on server

Engines

API Request	API Response	Meaning
POST /admin/engine/{engineId}/pin	200	Pin engine successfully
	400	Bad request (e.g. invalid header,

API Request	API Response	Meaning
		API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
POST /admin/engine/{engineId}/unpin	200	Unpin engine successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
POST /admin/engine/{engineId}/enable	200	Enable engine successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
POST /admin/engine/{engineId}/disable	200	Disable engine successfully

API Request	API Response	Meaning
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server

Statistics

API Request	API Response	Meaning
GET /version	200	Provides product ID and version information
	500	Unexpected event on server
GET /stat/engines	200	Returns an array with all engines and their details
GET /stat/nodes	200	Returns an array with all engines and their details
	403	Invalid user information or not allowed
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /stat/activeperformance	200	Returns an active performance report on Metadefender Core
	403	Invalid user information or not allowed

API Request	API Response	Meaning
	405	The user has no rights for this operation
	500	Unexpected event on server
GET /readyz	200	The service is currently healthy
	500	Unexpected event on server
	503	The service is currently unhealthy

Support:

If you are **Encountering Errors On MetaDefender Core and Are Unable To Find a Solution Within Our Documentation Or Knowledge Base**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why are some modules not appearing in my MetaDefender Core inventory?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

If your inventory does not include all of the modules that are listed as part of your MetaDefender Core package license, this is likely indicative that the system is deployed offline (in a restricted environment) and is unable to connect to the necessary OPSWAT hosts for initial module activation and regular updates.

For offline environments, the Core deployment requires an import YAML license into the activation tab for the engines as well as engines download using MetaDefender Update Downloader to be successfully deployed.

Pro Tip:

If you have installed or if you wish to use the MetaDefender Core in an online environment, you will have to allow access to the following hosts' for accurate functioning of all included modules:

- https://activation.dl.opswat.com - this is for product activation/licensing
- https://update.dl.opswat.com - this is for fetching engine/database updates

Please Note:

IP address-based whitelisting on your firewall might fail after some time, since OPSWAT uses CDN (Content Delivery Network) to faster deliver updates around the globe, and IP addresses of edge servers might change over time.

Info:

If the internal policy requires certificate validation, the firewall must allow access to the CRL and OCSP URLs of the OPSWAT Activation and Engine Update servers' certificates, as well as their Intermediate CA (ICA) certificate. To extract the URLs that need whitelisting, please follow How to retrieve the CRL and OCSP URLs for the Server and ICA Certificates?

If the issue is network related, please keep in mind that, in an online environment where MetaDefender Core is configured to download module and scan engine update packages directly from an OPSWAT update server source, using a gigabit network speed (125MB/sec) is highly recommended to avoid potential network related issues during file downloads. The same is true for initial activation downloads.

At the very least, the requirement is a network speed of 100Mbps (12.5MB/sec).

Support:

If you have followed the instructions above but **Some Modules Are Still Not Appearing In Your MetaDefender Core Inventory**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why does my MetaDefender Core install fail?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

There are certain hardware, software and system requirements that must be met in order for your MetaDefender Core install to run smoothly through to completion.

- meeting third-party requirements
- downloading and running the official *.msi* or *.deb* installer package
- ensuring sufficient network bandwidth and host access.
- Always check the system resource requirements and release notes for known limitation first before installing / upgrading Core.
- When the installation failed for whatever reasons, it's worth looking into OS system log / event log to find a clue.
- If it does not help, then rerun the installation again with logging enabled (e.g. *msiexec /i "C:\installer\ometascan-5.2.1-1-x64.msi.msi" /L*V "C:\log\example.log"*) to capture verbose installation log, and

escalate to OPSWAT support.

Pro Tip:

Should you not be running an OS that is supported under your licenced MetaDefender Core release, you will be unable to run the MetaDefender Core *.msi* or *.deb* package installer that is covered later in this article.

To learn more, please read our article on Recommended System Configuration.

Pro Tip:

If you are using an unsupported browser, you might assume that your install has failed because you are unable to access the Management Console and use MetaDefender Core, so please be mindful of this requirement.

Allowing sufficient storage space and resources

For both Windows and Linux systems, it is highly recommended that you install MetaDefender Core on machines with SSD drives, as optimal speeds are required for seamless install and swift file processing, particularly when high throughput is expected.

Furthermore, for all MetaDefender Core V5 releases installed on Windows or Linux systems, the following CPU, RAM and Storage specs must be met:

Package	CPU cores	Free System RAM	Free Disk Space
MetaDefender Core 8	8	8 GB	16 GB
MetaDefender Core 12	16	16 GB	24 GB
MetaDefender Core 16	16	16 GB	32 GB
MetaDefender Core 20	32	16 GB	40 GB
MetaDefender Core 32	32	32 GB	120 GB

Pro Tip:

Should you not meet these minimum MetaDefender Core V.5 system requirements, you will likely encounter problems installing and/or using MetaDefender Core.

To learn more about these requirements, please read our article on Recommended System Configuration.

In addition to the above, you will also require a minimum of 4.5GB for each 1M analysis reports generated, as well as

additional space for sanitized and quarantined files, depending on the nature of your throughput; but failing to meet this requirement will not impact the install process.

Meeting third-party requirements

Even when all other requirements are met, MetaDefender Core will fail to install and/or function when certain third-party dependencies are not met.

In many cases, your Windows version or Linux distro will already include all or a few of these dependencies but, in some cases, you will need to install them.

Pro Tip:

For this reason, we recommend that you check your OS against this third-party dependency list and install any missing components on your system before you run the MetaDefender Core installer.

Third-party dependencies are as follows:

Windows Third-Party Dependencies	Linux Third-Party Dependencies
.NET framework 4.5 or above	OpenSSL
Microsoft Visual C++ Redistributable for Visual Studio 2015–2022 (version 14.38 or higher), x64 version	grep
	lib32stdc++6 (>= 4.5)
	libc6-i386 (>= 2.10)
PostgreSQL 12.x (at least 12.3) and 14.11 • Only applicable when using a pre-installed PostgreSQL server running remotely.	procps
	zlib1g
	libcurl3 (>= 7.19.7)
	libcurl4
	PostgreSQL 12 (>= 12.3) or 14 (14.11)

There are also various engine-specific third-party dependencies that should be met and, while not meeting these will impact your ability to utilize the various scan engines included under your package, will not interrupt the MetaDefender Core install process.

To learn more about engine-specific dependencies, please Read This.

Here is an example of a Windows installation of MetaDefender Core that failed due to missing third-party dependencies. The following error message shows what the user may see when the required dependencies are not installed:

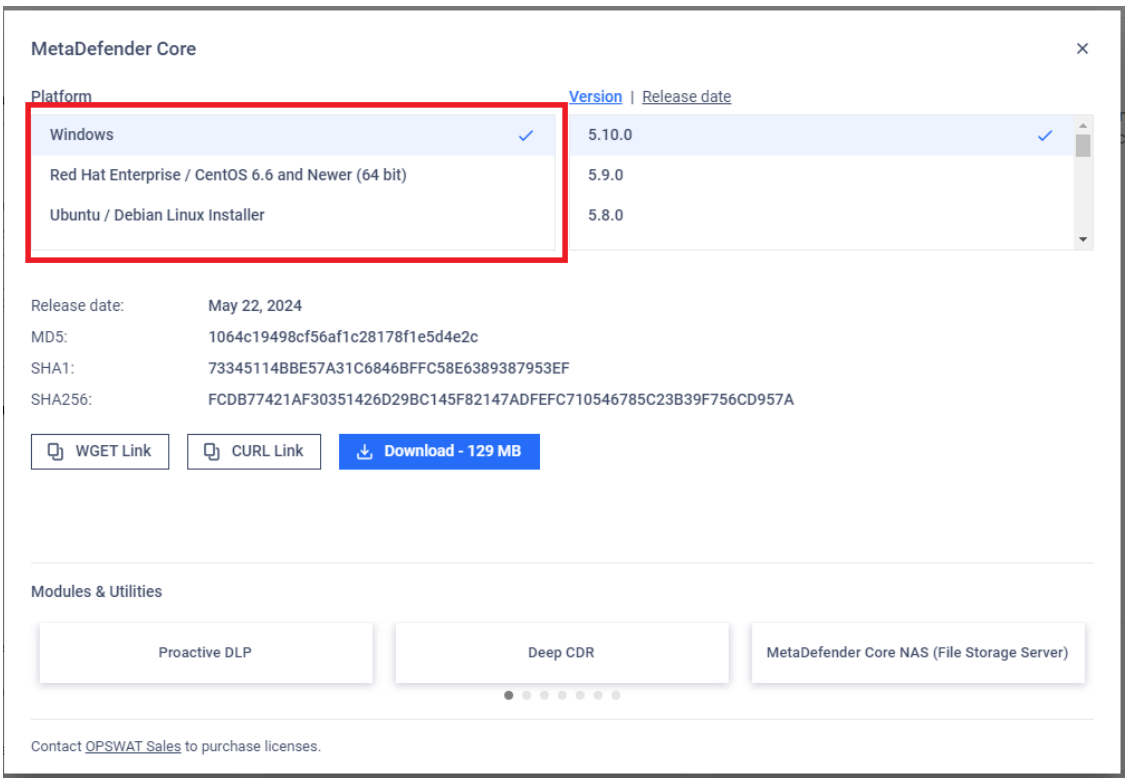
"What Does This Windows Installer Package Popup Window Means when MetaDefender Core does not successfully installed?"

Downloading and running the official *.msi* or *.deb* installer package

To install MetaDefender Core on any compatible system, you must download the relevant verified installer package directly from the My OPSWAT> Products > MetaDefender Core page.

This *.msi* (for Windows) or *.deb* (for Linux) file features both the MetaDefender Core software package and the install wizard for seamless setup and launch.

You should also ensure that you download the installer package that is associated with your OS, as illustrated in the screenshots below.



Pro Tip:

Not making these necessary adjustments to your firewall and security settings will interrupt the install and deployment process

You will be unable to launch the **MetaDefender Core Management Console** and use the product unless these hosts are allowed.

Support:

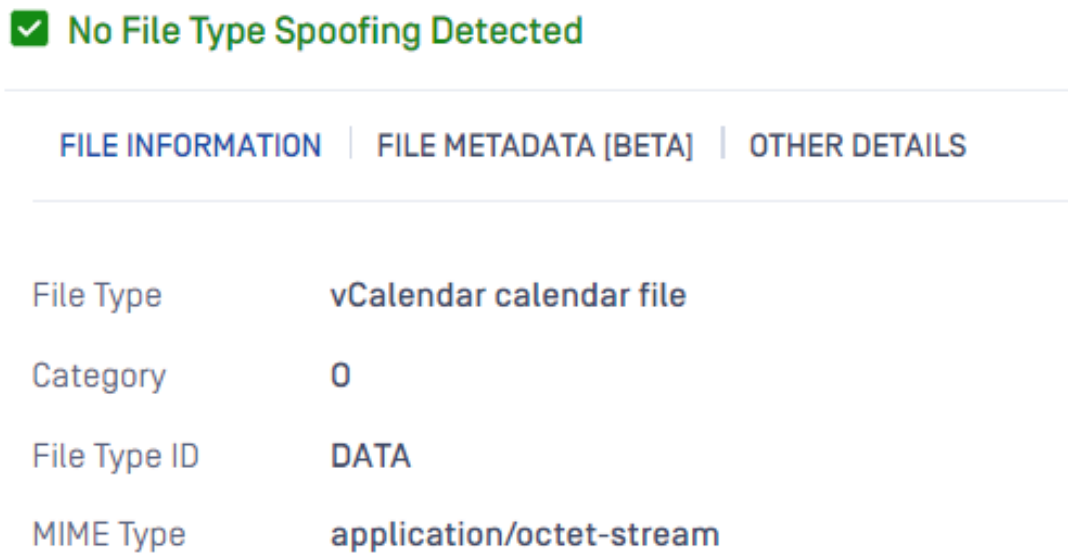
If you have followed the instructions above but are still **Unable To Install Metadefender Core on Your System**, please reach out by creating a support case or chatting with our support engineer.

Why do you sometimes see a discrepancy between the File Type and MIME Type information?

Check Your Version:

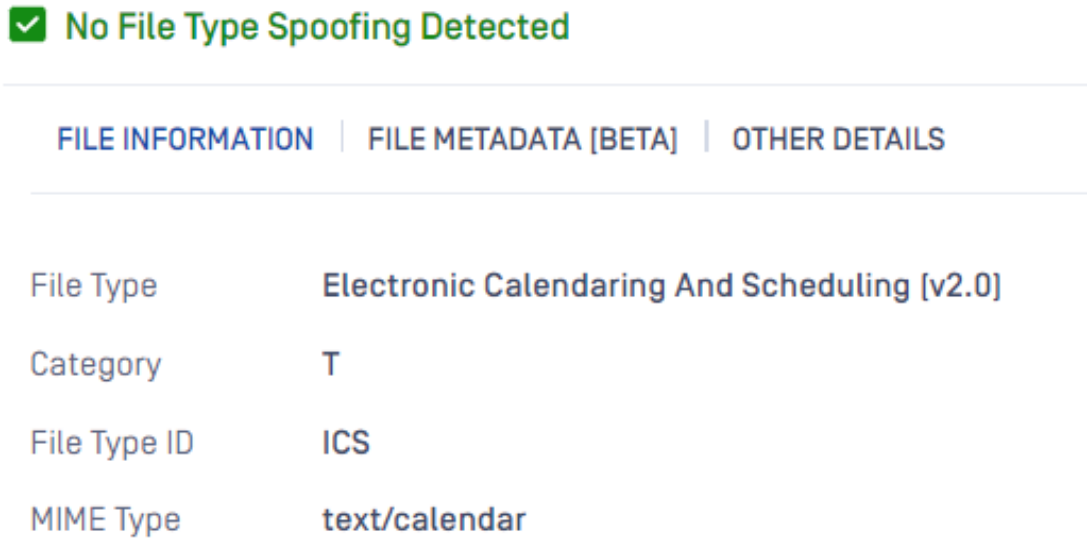
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

In certain cases, it is observed that the "File Type" and "File Type ID/MIME Type" information associated with a file do not match, as shown in the example below:



This indicates that the file type engine has identified the input file as a vCalendar file, but with some degree of uncertainty. It is possible that the input file does not strictly conform to the file specification, which may cause the file type and MIME type information to be inconsistent.

When a file follows the file specification correctly, the categories, file types, and MIME type will have the appropriate and consistent information, as demonstrated in the example below:



It is important to note that the correct identification of file types and MIME types is critical in various applications, such as content filtering, file sharing, and data management. Therefore, it is recommended to ensure that files adhere strictly to the file specification and standards to avoid discrepancies in the file type and MIME type information.

Support:

If Further Assistance is required, please follow the instructions on How to Create Support Package With Bundle Tools?, then proceed to create a support case or chat with our support engineer.

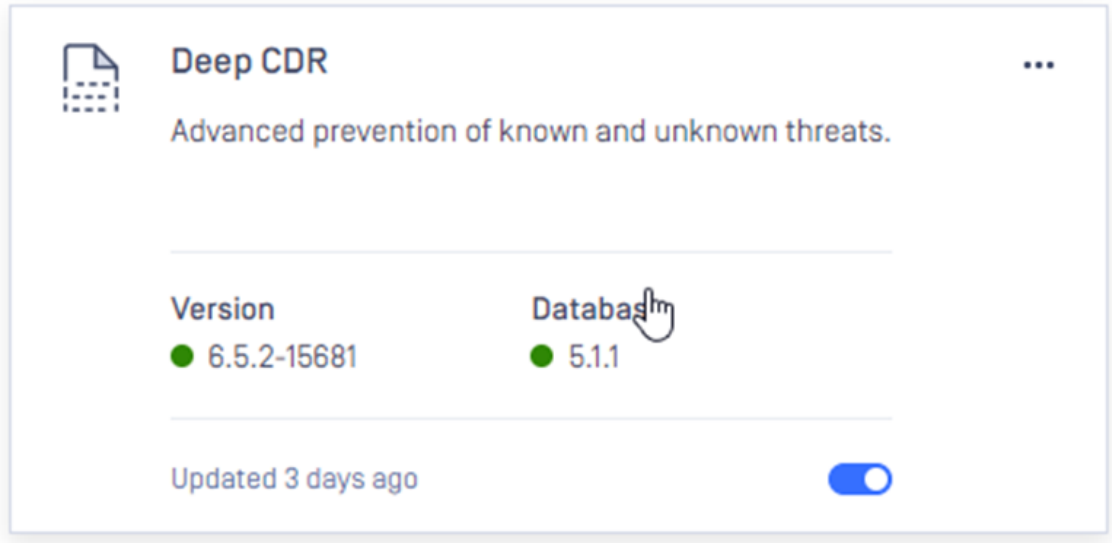
Why files are not sanitized?

Check Your Version

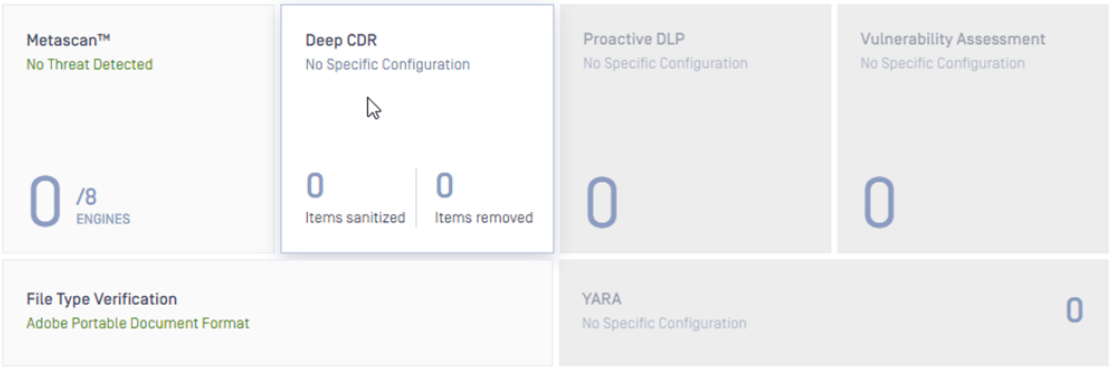
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

There are a few details too look at when the files you scan are not being sanitized:

1. Make sure that the Deep CDR engine is enabled.

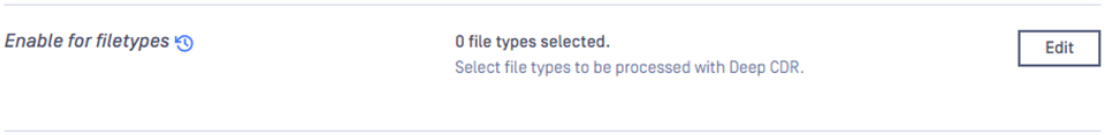


2. Make sure that the extension of the file you are scanning is enabled in the DEEP CDR module.

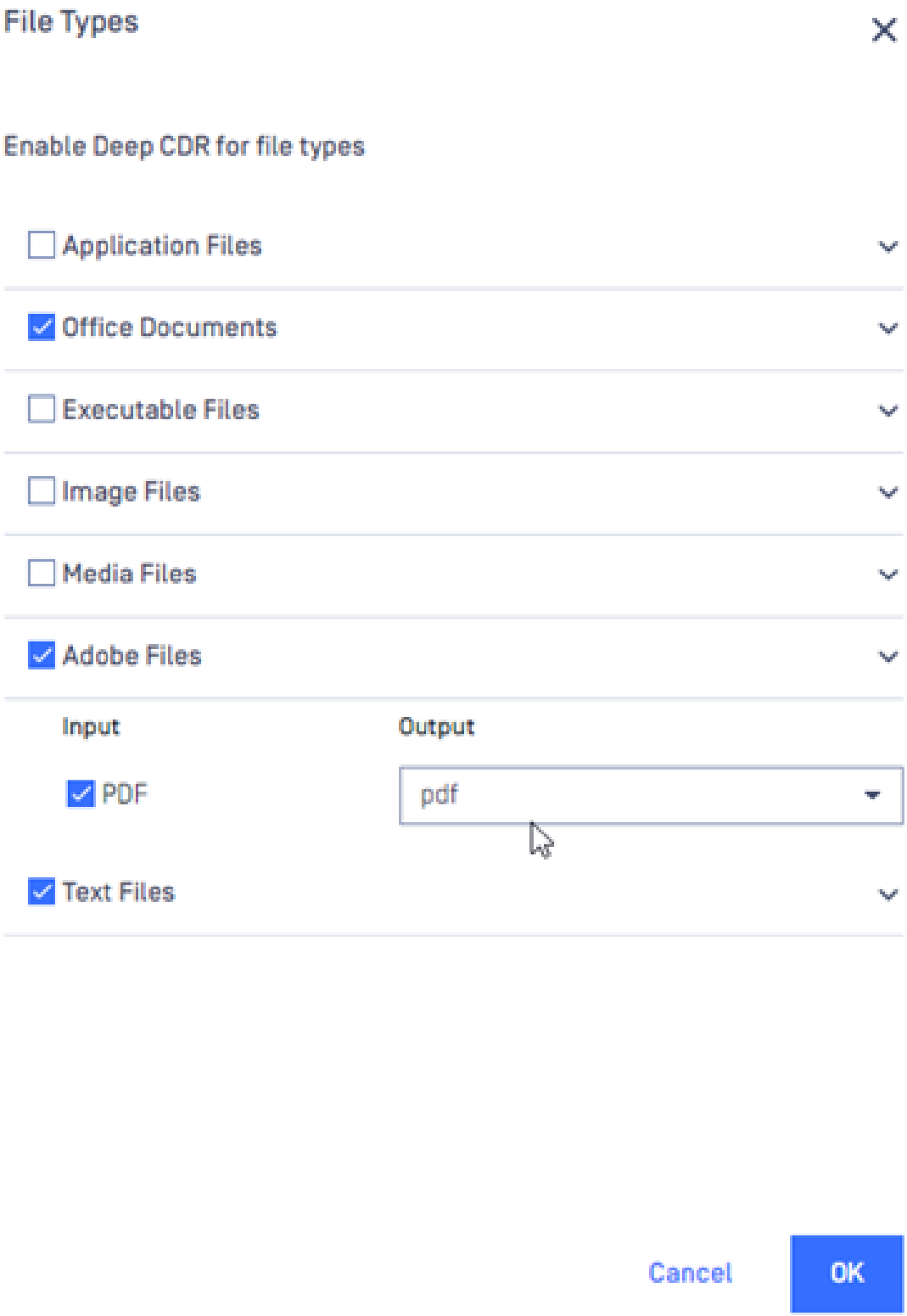


To do so, please access the MetaDefender Core WEB UI (by default http://localhost:8008) and access Workflow Management -> Workflows -> Select your workflow -> DEEP CDR tab

Here navigate to “Enable for filetypes”



Click on Edit and enable all extensions that you want to sanitize:



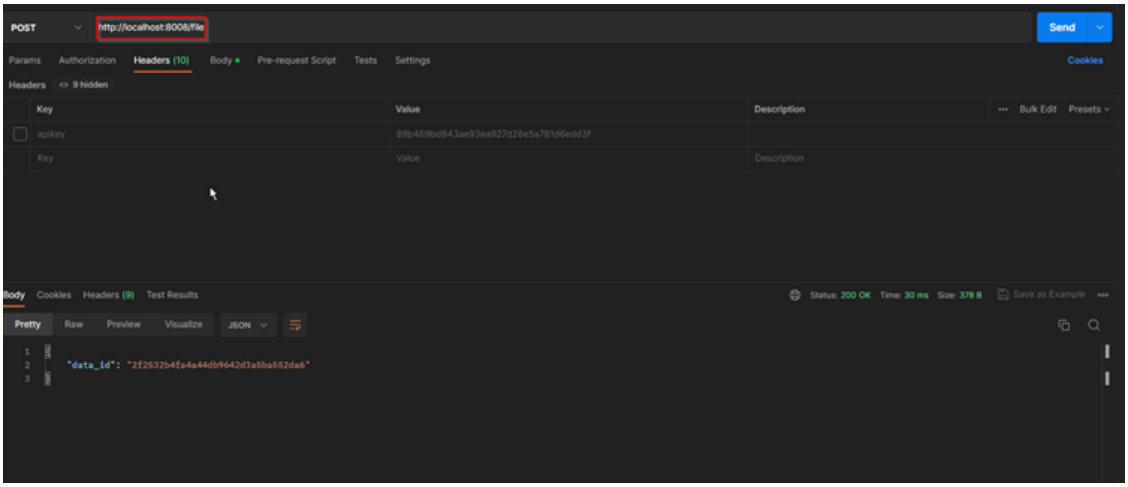
Once selected, click OK and then Save changes.

Support

If you have followed the instructions above but **Files are still not sanitized** please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why am I receiving a “400” error for API file submission?

When submitting files to a MetaDefender Core instance users might experience a 400 API error response. One of the reasons for this behavior might be related to an incomplete URL format.



Please make sure that you specify the port in your API call (default 8008 or whatever port is set-up on the deployment)

For additional information on API calls please use the documentation material below.

POST - Analyze File (Asynchronous mode)

Support

If you are a **MetaDefender Core Custom** customer, OPSWAT highly recommends that you contact OPSWAT Support for expert guidance through the upgrade process. Create a support case or chat with our support engineer to get in touch with our team.

How to Mitigate Scan Failures in MetaDefender Core Triggered by Real-Time Protection Systems?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

Summary:

This article provides insights and solutions to a prevalent issue in MetaDefender Core where scans may fail due to disruptions from real-time protection systems. The problem often manifests without generating explicit error messages, complicating diagnosis. This guide will walk you through the steps to identify and rectify this situation.

Problem Description:

Users of MetaDefender Core may experience scan failures that do not return specific error messages. A successful scan should typically display "No available scan result" in the Metascan details, and the corresponding report should state "Not Scanned."

Primary Cause:

In most cases, real-time protection systems like antivirus software delete or quarantine files uploaded to MetaDefender Core, perceiving them as potential threats. This action inhibits MetaDefender Core's ability to process these files, leading to scan failures. Notably, these failures may also show errors from engines indicating the file does not exist.

Solution:

To resolve this issue and ensure effective scanning, follow these steps:

1. Identify the real-time protection (RTP) software operational on the MetaDefender Core server.
2. Navigate to the settings of the identified RTP software.
3. Look for the “Exclusions” or “Exceptions” section in the settings.
4. Add the full installation path of MetaDefender Core, and the temporary upload path used by MetaDefender Core, to the exclusions list.
5. **Important Note:** If Windows Defender is used *both* as the local antivirus **and** as an AV engine inside MetaDefender Core, **do not exclude** `engineprocess.exe` from the real-time protection exclusion list. In that scenario, excluding the engine process may interfere with the integrated AV engine functionality.
6. In other cases (for example, when Windows Defender is used only as the local AV, not as the AV engine for MetaDefender Core), you may exclude `engineprocess.exe` along with other processes like `ometascan.exe`, `postgres.exe`, and `nginx.exe` to prevent interference by the real-time protection system.
7. Save your changes and exit the settings.
8. Conduct a test scan to confirm the resolution of the problem. For detailed instructions on adding exclusions or exceptions to your AV software, refer to the user manual or documentation provided by the software vendor.

Conclusion:

By following the provided steps and adjusting your real-time protection software settings, you should be able to address MetaDefender Core scan failures. This process ensures that files you upload for scanning are not wrongly removed by real-time protection systems, enabling successful MetaDefender Core scans. As a result, you should observe "No available scan result" in the Metascan details and "Not Scanned" in the exported report, as expected.

Support:

If the problem persists despite following the outlined steps, please open a **Support Case** with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our **OPSWAT Expert Forum**.

How to Resolve MetaDefender Core Service Start-up Failures on IPv6-Disabled Linux Systems?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Linux systems.

Issue: MetaDefender Core services fail to initialize on Linux systems that lack IPv6 support. Symptoms include unresponsive services and an inaccessible MetaDefender Core dashboard.

Cause: The underlying web server (typically Nginx) attempts to access both IPv4 and IPv6 addresses when the 'restaddress' configuration in the 'ometascan.conf' file is set to '*'. If the Linux system lacks IPv6 support or an IPv6 IP address, Nginx fails to start. The error messages in the 'ometascan.log' and 'nginx-ometascan.log' logs reflect this failure.

Here are the associated log entries indicating such errors:

ometascan.log · sql

```
[ERROR   ] 2023.07.12 07:48:01.781: (common.base) Error in Metadefender Core service, shutdown.., error='REST frontend stopped, exitCode: '1'' [msgid: 131]

[INFO    ] 2023.07.12 07:48:01.813: (core.postgres) Postgres info, info='systemctl stop ometascan-pg', param='' [msgid: 4085]

[ERROR   ] 2023.07.12 07:48:01.932: (common.base) Exception has been caught, message='REST frontend stopped, exitCode: '1'' [msgid: 833]
```

nginx-ometascan.log · sql

```
2023/07/12 07:49:07 [emerg] 9677#0: socket() [::]:8008 failed (97: Address family not supported by protocol)
```

Solution:

1. Open a terminal or SSH into your Linux system.
2. Go to the directory where the MetaDefender Core configuration file is located (/etc/ometascan/ometascan.conf).
3. Open the 'ometascan.conf' file with a text editor (e.g., vi or nano).
4. Find the line that specifies the 'restaddress' configuration (it is usually set to 'restaddress=*').
5. Change the 'restaddress' line to 'restaddress=0.0.0.0'. This action forces the MetaDefender Core services to listen on any IPv4 address set up on the server.
6. Save the changes to the 'ometascan.conf' file and close the text editor.
7. Restart the MetaDefender Core services to apply the configuration changes using the command: 'sudo systemctl restart ometascan'.

8. Confirm the successful operation of the MetaDefender Core services by checking their status with the command: 'sudo systemctl status ometascan'.

By altering the 'restaddress' line to '0.0.0.0' in the 'ometascan.conf' file, the MetaDefender Core services are set to listen on any available IPv4 address on the server. This action ensures that the services can start successfully even on Linux systems without IPv6 support.

If the issue persists or if you have further concerns, please refer to the MetaDefender Core documentation or reach out to OPSWAT support for assistance.

Support:

To contact us, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How to Understand OPSWAT's Multiscanning AV Detection Differences Compared to Other Scanners?

Check Your Version

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

There are times when OPSWAT Multiscanning antivirus detection presents results that differ from those provided by other scanner providers. This variance might lead to some questions and concerns, but it's imperative to understand that there are specific and logical explanations for these differences. In this page, we'll explore these reasons.

Keeping security and false positives in balance

One of the primary factors contributing to the difference in detection results is the configuration of OPSWAT's AV engines. We carefully choose our configurations to provide robust security while minimizing the possibility of harmless files being incorrectly flagged. Maintaining this balance is crucial to maintaining both the integrity of the system and the trust of the users.

If OPSWAT Mulitscanning engine doesn't detect a file that other scanner providers do, it's possible that adjusting the engine configurations could resolve this issue.

Engine	Configuration
AegisLab	Cloud Scan - On
ClamAV	Detect PUP/PUA - On
ESET	Extract Archive - On

Engine	Configuration
Ikarus	Extract Archive - On
Huorong	Extract Archive - On
Sophos	Cloud scan - On
Symantec	Insight Scan - On
Crowstrike ML	Change Threshold Detection to lower level

Cloud Features and Enterprise Suitability

Another significant point of differentiation is OPSWAT's approach to cloud features. The AV engines we employ are designed with enterprises in mind, focusing on robust, on-premises solutions. This approach contrasts with many other services that rely heavily on cloud-based features. While cloud-based AV solutions are popular and effective for some users, they might not always be the best fit for enterprise environments where control, privacy, and security are paramount.

Listing Known Detection Differences

Here is you can find a list of known differences in detections between OPSWAT and other scanner services. These differences can stem from a variety of factors such as licensing agreements, efforts to reduce high false positives, the use of remote servers, or choosing to utilize a different product from the same company. Each of these elements contributes to the unique way OPSWAT approaches and implements its antivirus detection strategies. Here is a list of the known differences:

Engine	Difference
Antiy	OPSWAT products don’t use their cloud service.
BitDefender Theta	According to the vendor, this is not a standard version, it is a machine-learning engine version, they don’t sell it to anyone, they only put it in VT to collect data and use it to update the standard version.
CrowdStrike Falcon	OPSWAT products don’t use their cloud feature. This feature will send the customer data to CrowdStrike server to analyze more, the detection comes from the cloud server. OPSWAT product is for enterprise customers, we do not allow sending the customer’s data to outside the organization, therefore the detection is just based on the local database and can’t detect the file. In general, the vendor will review

Engine	Difference
	the cloud detection and add it to the local database if needed, it just takes time, we don't know when.
McAfee	The comparable version in VT is Trellix ENS not the McAfee Scanner. OPSWAT products don't use Artemis detection. The Artemis detection comes from Trellix GTI Cloud detection, this detection will or won't be moved to the traditional/local database, it depends on the prevalence.
Webroot	OPSWAT products use SMD, Some providers use File Reputation.

Support:

If you need help learning more information about **OPSWAT's Multiscanning AV Detection Differences Compared to Other Scanners** feel free to create a support case or chat with our support engineer.

How to Troubleshoot Inaccessible Management Console?

Introduction:

This knowledge base article provides step-by-step instructions for checking and resolving the issue when MetaDefender Core becomes unreachable after disabling HTTPS.

Problem: You cannot access the MetaDefender Core Management Console from your browser e.g. http://localhost:8008

Symptom: After you enter the Metadefender Core Management Console address (e.g. http://localhost:8008) you get an error message (connection refused) or your browser is waiting for reply.

Steps to resolve the issue:

- Check if your computer can access the Metadefender Core IP address / domain name. Verify that you opened the firewall port on the Metadefender Core server for the Management Console.
- Check if you are using proper protocol HTTP / HTTPS.
- Check if you are using proxy for the connection.
- Check if MetaDefender Core service is still running.
- Check MetaDefender Core's Nginx web server log for a clue.

- Sometimes the browser will autofill/cache the address and even if you are entering the URL manually with HTTP instead of HTTPS, the browser will still load the HTTPS, and thus the error message. To resolve this situation, you can either clear the browser history/cookies or use a different browser / computer to test the connection.

Once browser data has been cleared, the MetaDefender Core console can be accessed using the HTTP protocol without further issues.

Support:

If you encounter any issues during the activation process or require further assistance, please contact OPSWAT support via My OPSWAT.

Our support team will be able to provide you with the necessary guidance and resolution for any license-related concerns.

How do I fix module update error “Error parsing metadescriptor, message='Could not download metadescriptor'”?

Check Your Versions:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Cause

This error occurs when there is connection issue between MetaDefender Core and the update server, such as proxy server not configured correctly, update server unreachable, or SSL issues.

Solution: If MetaDefender Core is updating from the Internet

MetaDefender Core pulls engine updates from `update.dl.opswat.com` when Internet update is configured, to avoid issue:

1. Make sure that update.dl.opswat.com is reachable from the MetaDefender Core server .
2. If a proxy is used, please make sure it’s configured correctly in Proxy Configuration.

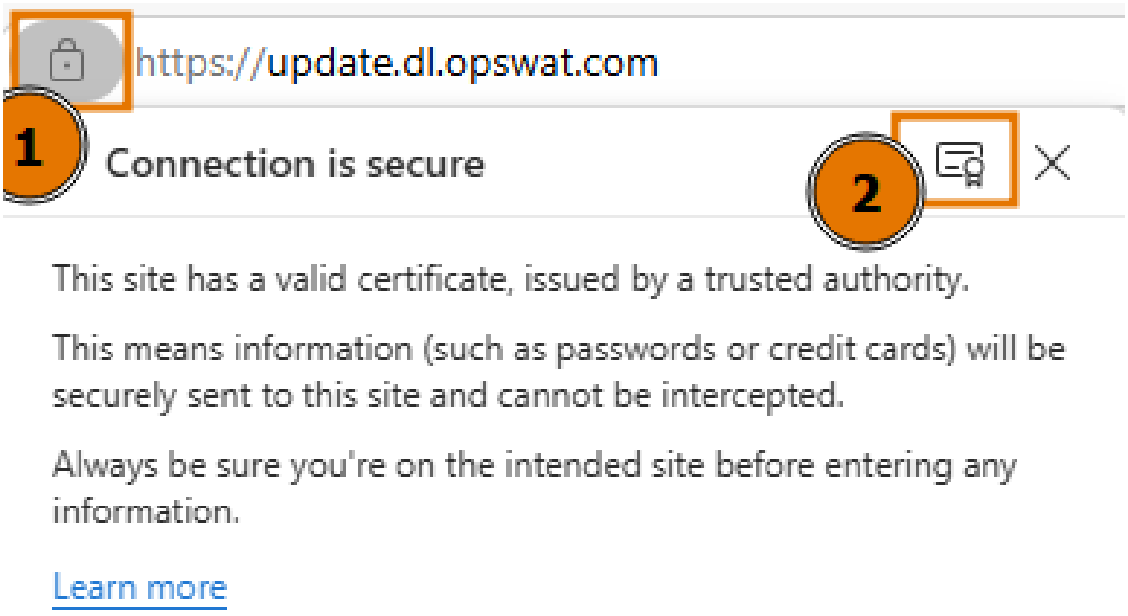
Certificate not available on the server's trusted certificate store

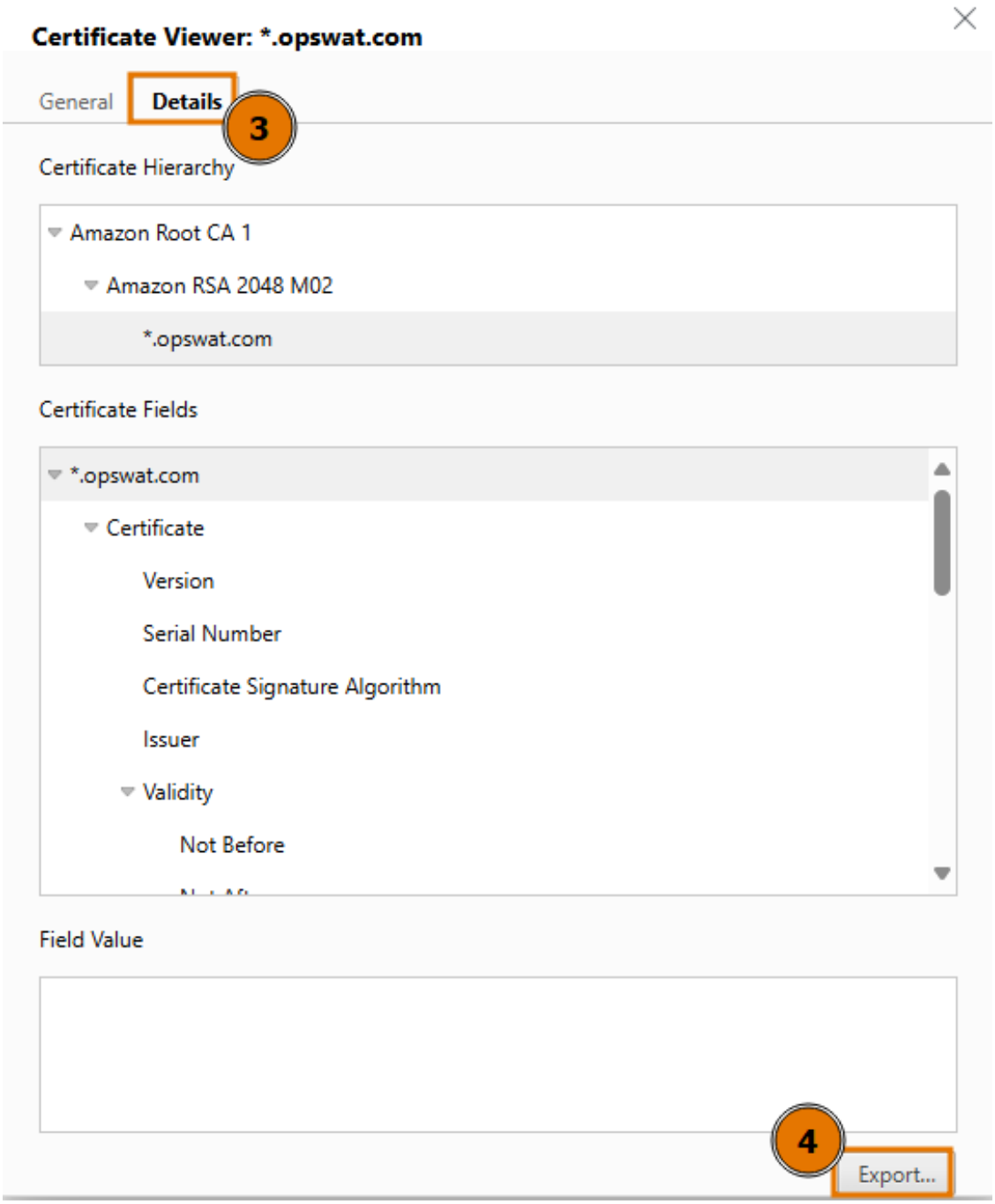
Although the **Trusted Root Certificate Authorities** store is updated frequently from Microsoft’s repository by the operating system, in some rare cases, the SSL certificate used by update.dl.opswat.com might be missing from the MetaDefender Core server’s trust store.

When the above certificate error occurs for `update.dl.opswat.com`, you might find the following in core.log:

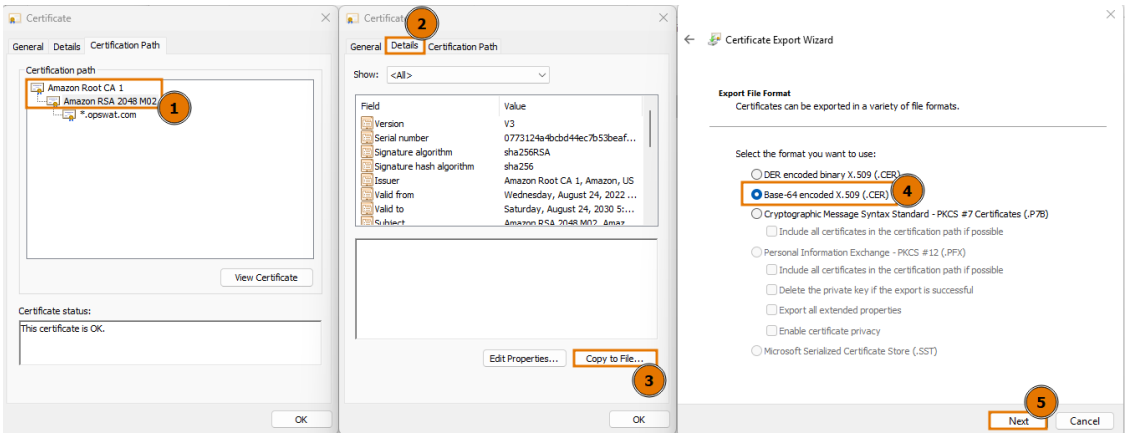
```
Downloading metadescriptor,
downloadlink=https://update.dl.opswat.com/metadescriptor...
SSL errors while downloading file
Certificate chain for the ssl error
SSL error, The issuer certificate of a locally looked up certificate could not be
found
```

To remediate this issue, manually access https://update.dl.opswat.com from a browser, and download the certificate chain:





Once the certificate chain is downloaded, export the root and intermediate certificates:



Then, manually import each certificate in the chain (including root and intermediate certificates) into the server's trust store:

- Windows:
  1. Double-click the .crt file → opens "Certificate".
  2. Click "Install Certificate...".
  3. Choose "Local Machine" (You will need Administrator rights).
  4. Choose "Place all certificates in the following store"
  5. Click Browse → select Trusted Root Certification Authorities.
  6. Click Next, then Finish.
- Debian:
  1. Copy your certificate to the trusted directory: `sudo cp mycert.crt /usr/local/share/ca-certificates/`
  2. Update CA store:`sudo update-ca-certificates`
- RedHat Enterprise Linux:
  1. Copy the certificate: `sudo cp mycert.crt /etc/pki/ca-trust/source/anchors/`
  2. Update CA Trust: sudo update-ca-trust extract

After that, restart "OPSWAT MetaDefender Core" service, and the certificate for `update.dl.opswat.com` should be trusted.

Solution: If MetaDefender Core is managed by OPSWAT Central Management

MetaDefender Core is pulling module updates from OCM, to avoid SSL issue:

1. If OCM is using a certificate for HTTPS connection, make sure that the certificate subject's common name matches the OCM server's URL.
2. The OCM server's certificate must also be trusted on the MetaDefender Core server by importing to the server's trust store.
3. The OCM server's address must be resolvable from the MetaDefender Core server.

Support:

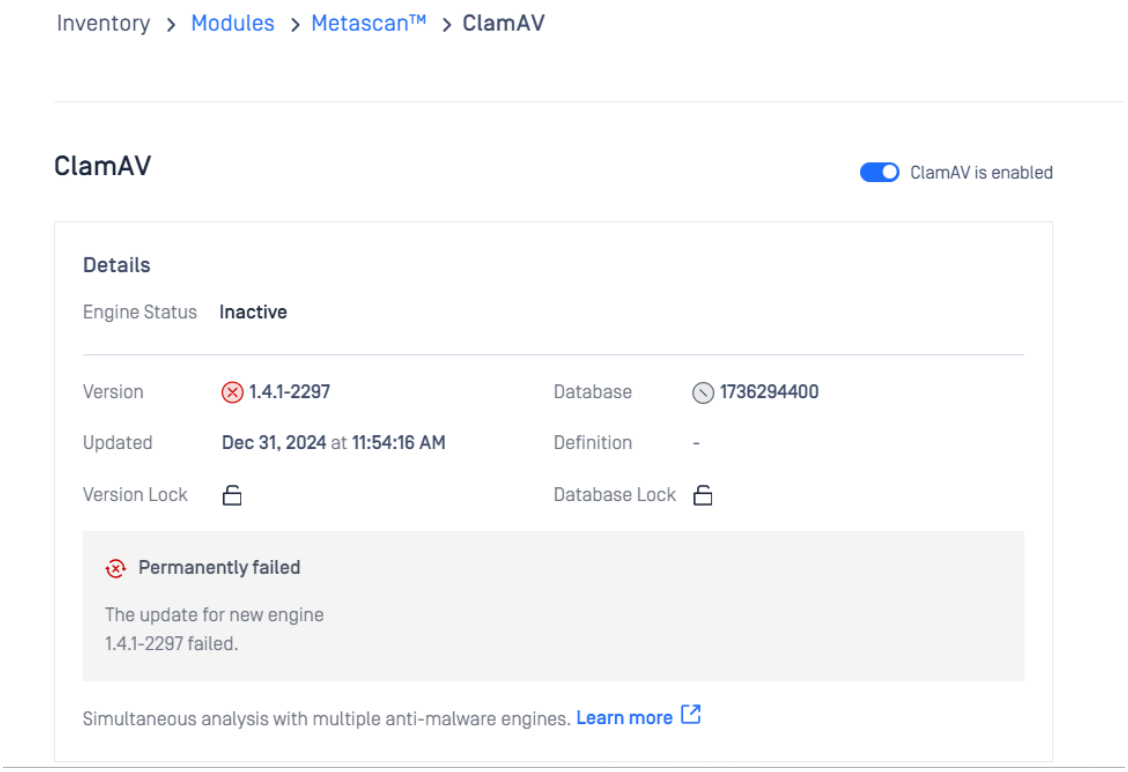
If you have followed the instructions above but are still unable to update MetaDefender Core modules with the error message "Error parsing metadescriptor, message='Could not download metadescriptor'", please follow these instructions on **How to Create Support Package With Bundle Tools?**, before **creating a support case or chatting with our support engineer**.

Why does ClamAV show "permanently failed" after enabling the

daemon log file?

Problem

After enabling the ClamAV daemon log file, you may encounter the error message: `Permanently failed`



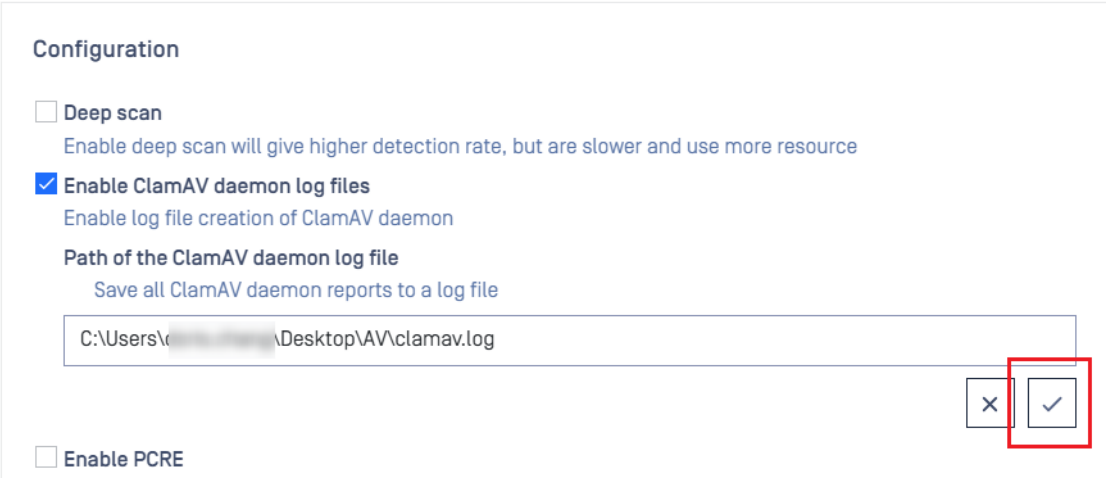
This error occurs because no valid path for the log file has been specified.

Cause:

This is intended behavior. The ClamAV engine requires a valid path for the daemon log file to function correctly. Without it, the daemon cannot initialize properly.

Steps to Resolve

1. **Specify the Destination Path and Log File Name** Select a destination path and specify a file name for the ClamAV daemon log file. **Example:** `C:\Users\user.name\Desktop\AV\clamav.log`
2. **Update Configuration** Navigate to **MetaDefender Core > ClamAV > Configuration** and update the Path of the ClamAV daemon log file with the chosen path and file name.



3. **Re-enable the ClamAV Engine** After saving the changes, disable the ClamAV engine and then re-enable it. This forces the daemon to reinitialize with the updated configuration.

Log File	Description
oms_clamav.log	Located at `C:\Program Files\OPSWAT\MetaDefender Core\data\logs\oms_clamav`. This log file is specific to the MetaDefender Core integration and provides logs for integration-related activities.
Daemon Log File	The ClamAV daemon's own log file, offering detailed information for debugging ClamAV-specific issues.

The daemon log file is particularly useful for troubleshooting issues specific to the ClamAV engine, while the `oms_clamav.log` focuses on MetaDefender Core's operations.

Support:

If you have any further questions please proceed to create a support case or chat with our support engineer.

How to troubleshoot HTTPS failure on MetaDefender Core?

Info

This article applies to all MetaDefender V5 releases deployed on Windows or Linux systems.

You are planning on setting up HTTPS or updating HTTPS certificate.

Whenever you're enabling HTTPS for MetaDefender Core, a number of issues can occur, the aim of this article is to outline some common issues and how to troubleshoot and resolve them.

Issue

After enabling HTTPS and restarting MD Core service, the management console is inaccessible.

When this happens, it is most likely caused by NGINX error. To troubleshoot this, you can view the NGINX's error logs for the specific error message, located here:

- Windows: C:\Program Files\OPSWAT\MetaDefender Core\nginx\nginx_error.log
- Linux: /var/log/nginx/error.log

Error

"SSL: error:0B080074:x509 certificate routines:X509_check_ private_key:key values mismatch"

Possible cause:

The HTTPS certificate and private key are mismatched. You can verify this by running the following commands:

Code · bash

```
# Replace "cert.crt" with the path to the certificate
openssl x509 -noout -modulus -in cert.crt | openssl md5
# Replace "key.key" with the path to the private key
openssl rsa -noout -modulus -in key.key | openssl md5
```

If the output MD5 hash values of the certificate and private key don't match up, then they will result in the SSL error above.

Remediation Steps:

To solve this issue, generate a new certificate-private key and ensure that they match each other.

1. If the MD Core management console is still inaccessible, use the HTTPS Configuration Removal Tool and access the management console via HTTP.
2. Follow the steps to enable HTTPS using the newly generated certificate-private key pair.

Error

"SSL_CTX_use_certificate("C:/Windows/Temp/ometascan/https_cert.pem") failed (SSL: error:0A00018E:SSL routines::ca md too weak)"

For other HTTPS failure, please follow instructions in this article HTTPS Failure on MetaDefender Core 5.2.0 (or newer)

Support:

If you have followed the instructions above but are still encountering **HTTPS failure on MetaDefender Core**, please follow these instructions on How to Create Support Package With Bundle Tools?, before **creating a support case or chatting with our support engineer**.

How to Remove Engines from MetaDefender Core using command line?

If you encounter issues with specific scan engines in MetaDefender Core, you may need to remove them temporarily or permanently. This article provides instructions for removing scan engines from MetaDefender Core using built-in tools for both Windows and Linux systems.

Prerequisites

Administrator access to the MetaDefender Core system.

- MetaDefender Core installed on Windows or Linux.
- Command-line access (Command Prompt for Windows, Terminal for Linux).

Removing a Scan Engine on Windows

Steps:

Open Command Prompt as Administrator.

1. Navigate to the MetaDefender Core installation directory:

```bash
cd "C:\Program Files\OPSWAT\MetaDefender Core"
```

2. Run the following command to remove the engines:

```bash
ometascan-engine-remove.exe
```

3. Verify that all the engines has been removed by checking the MetaDefender Core management interface or logs.

4. Restart MetaDefender Core service to apply the changes.

Removing a Scan Engine on Linux

Steps:

Open a Terminal session.

1. Run the following command to remove the engines:

```bash
sudo /usr/bin/ometascan-engine-remove
```

2. Verify that all the engines has been removed by checking the MetaDefender Core management interface or logs.

3. Restart MetaDefender Core to apply the changes:

```bash
sudo systemctl restart ometascan
```

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to enable debug logging for MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

Note:

To assist the OPSWAT Support team in diagnosing issues more effectively, we may require debug logs from your MetaDefender Core for a detailed analysis.

Enabling debug logging level on MetaDefender Core should only be used when necessary and advised by the OPSWAT team.

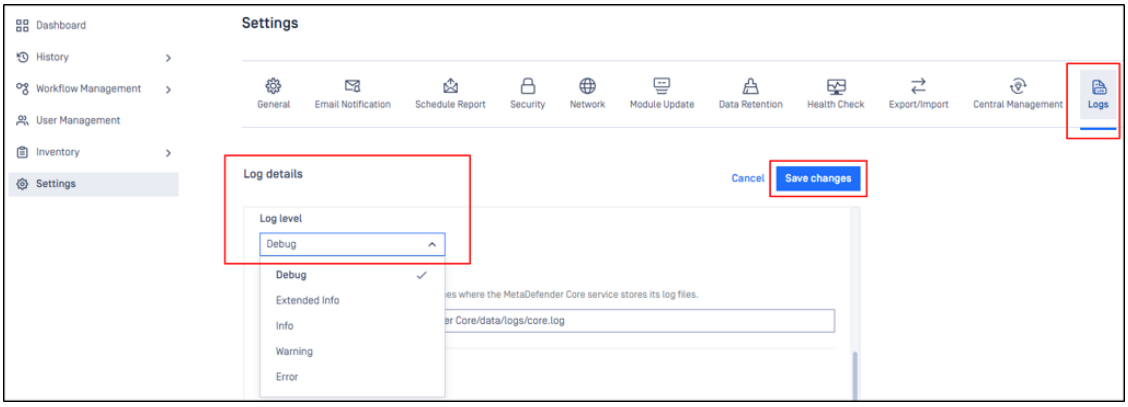
To enable debug logging for MetaDefender Core, please follow the appropriate guide based on your operating system:

Windows

The log level can be updated through the Web UI (available starting with MD Core v5.15.1) or via the Windows Registry.

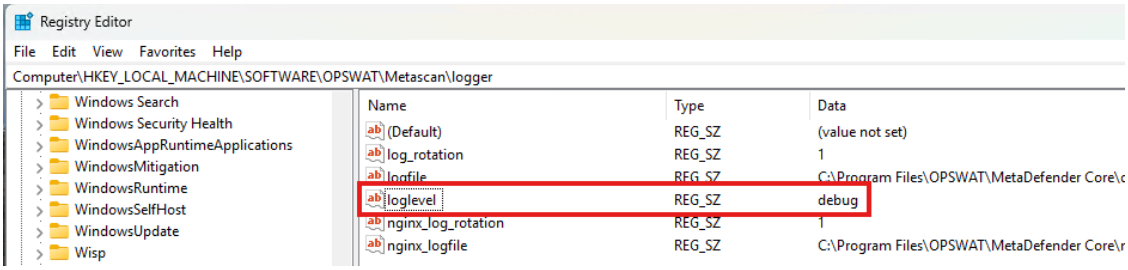
1. Web Console:

- Navigate to ***Settings*** > ***Logs*** > ***Log File*** section
- Click on the ***Edit*** button and update the ***Log level*** to ***debug***
- Save the changes (service restart is **NOT** required)



2. Windows Registry:

- Open the Registry Editor
- Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`
- Change the value of `loglevel`to `debug`
- Restart the MetaDefender Core service



Linux

The log level can be updated through the Web UI (available starting with MD Core v5.15.1) or via the ***ometascan.conf*** configuration file.

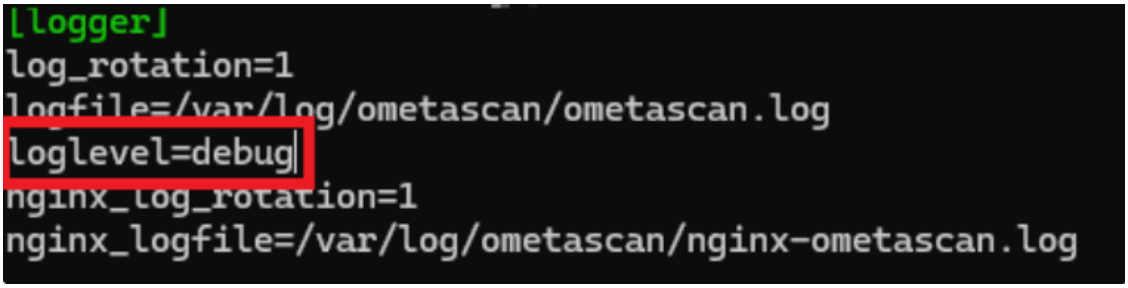
1. Web Console:

- Navigate to ***Settings*** > ***Logs*** > ***Log File*** section
- Click on the ***Edit*** button and update the ***Log level*** to ***debug***
- Save the changes (service restart is **NOT** required)

2. Configuration file:

- Open the file at `/etc/ometascan/ometascan.conf` with your preferred text editor.
- In the **[logger]** section, change the value of `loglevel` to `debug`.

- Restart the MetaDefender Core service.



Attention:

After finishing with debug logging mode, the **loglevel** should be set back to **info**. Failed to do so, the size of your log files could increase significantly.

Generating a Support Package

Once debug logging is enabled, please attempt to reproduce the issue you are experiencing. Afterward, generate a Support Package and send it to the Support team so we can properly investigate the issue.

For a detailed guide on generating a Support Package, please refer to the following URLs:

- How to Create Support Package With Web Console
- How to Create Support Package With Bundle Tools

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Why did the deployment of the Threat Intelligence Module Fail?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

One common reason for this failure is the absence of the required Microsoft Visual C++ Redistributable for Visual Studio 2015-2022 (version 14.38 or higher) on the system.

Resolution:

1. Verify Installation: Ensure that the Microsoft Visual C++ Redistributable for Visual Studio 2015-2022 (version 14.38 or higher) is installed on your machine.
2. Download and Install: If the required redistributable is missing, download and install the x64 version from the

following link: Microsoft Visual C++ Redistributable Download

3. Disable/Enable: Retry deploying the Threat Intelligence Module and verify if the issue is resolved.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Why did the deployment of the Reputation Module Fail?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

One common reason for this failure is the absence of the required Microsoft Visual C++ Redistributable for Visual Studio 2015-2022 (version 14.38 or higher) on the system.

Resolution:

1. Verify Installation: Ensure that the Microsoft Visual C++ Redistributable for Visual Studio 2015-2022 (version 14.38 or higher) is installed on your machine.
2. Download and Install: If the required redistributable is missing, download and install the x64 version from the following link: Microsoft Visual C++ Redistributable Download
3. Disable/Enable engine: Retry deploying the Reputation Module and verify if the issue is resolved.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Why is MetaDefender Core not starting after moving or deleting the certificates from the initial location?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

MetaDefender Core relies on nginx for its web interface. On startup, nginx will check the configured location for the SSL certificate, will attempt to load it into C:\Windows\Temp\ometascan and then will start and be ready to accept connections.

If at any time the certificate is deleted or removed, nginx will no longer be able to start and this will cause Core to

completely fail. The SSL configuration is not easily accessible to find the previously configured certificate and paths, making recovering this information not practical.

To resolve this, the best option will be to reset the SSL configuration and, once the Core service is successfully restarted using HTTP, the certificate can be imported back into the MetaDefender Core certificate inventory and the HTTPS configuration can be reenabled.

To do this, simply follow the steps outlined here: HTTPS Configuration Removal Tool

Please Note:

The SSL configuration reset tool will normally not affect any other part of the MD Core configuration, but best practices are to perform a configuration export before running this tool.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

Why doesn't CDR detect the JavaScript I appended to a PDF file?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Cause

If the JavaScript-containing objects are added to the PDF but the Xref (Cross-Reference) table is not updated accordingly, CDR will not detect them.

- The Xref table is crucial for referencing valid objects in a PDF.
- Objects missing from the Xref table are considered garbage data and will be removed by the CDR engine.

Solution

To ensure that JavaScript in a PDF is detected by CDR:

1. Update the Xref Table:

- When adding JavaScript, ensure that the corresponding object is correctly referenced in the Xref table.

2. Follow PDF File Structure Specifications:

- The JavaScript should be stored in a properly structured and up-to-date object within the PDF.

Conclusion

For JavaScript to be recognized in a PDF by CDR, it must be placed in a valid object that is correctly indexed in the Xref table and complies with the PDF file structure specifications.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why Deep CDR is not Sanitizing Files Inside a ZIP?

The .gif files are expected to be sanitized and converted to .jpeg format using Deep CDR rules. While this process worked for individual .gif files, when the files were inside a ZIP archive, the ZIP file returned with the original .gif files instead of the converted .jpeg files.

The issue was due to a missing configuration setting in the MetaDefender Core. Specifically, the "Enable archive sanitization" option was not activated, and the appropriate archive file types were not selected under "Enable for archive compression filetypes."

Solution:

To resolve this issue, follow these steps:

1. Navigate to the MetaDefender Core UI.
2. Go to the Workflows section and select the relevant workflow.
3. Access the Compression tab.
4. Ensure that "Enable archive sanitization" is turned on.
5. Under "Enable for archive compression filetypes," select all relevant archive file types.

Once these settings are correctly configured, Deep CDR should properly sanitize files inside ZIP archives as expected.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

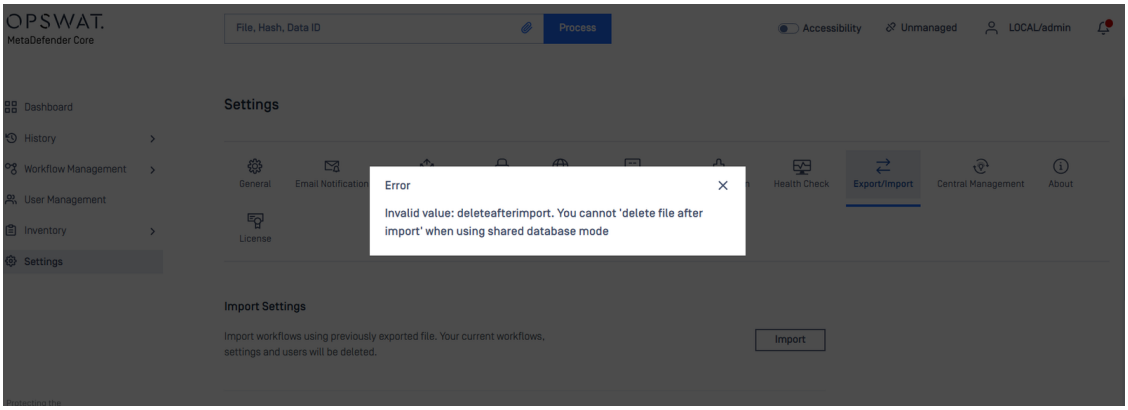
Why does the “deleteafterimport” error appear when importing an MD Core configuration from standalone mode to shared database mode?

Support:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

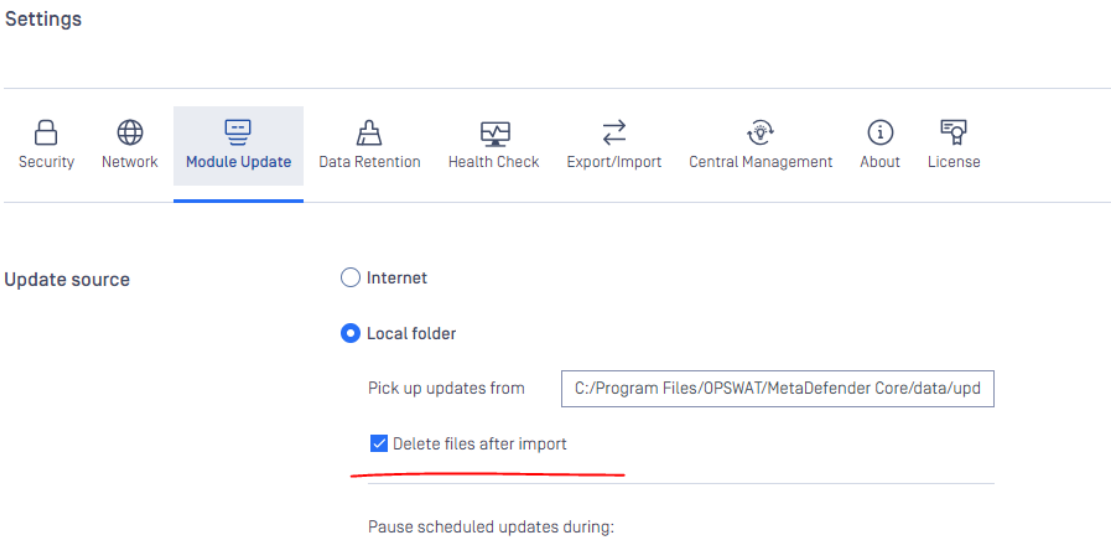
Issue: An error message, **"deleteafterimport"**, appears when a user imports an MD Core configuration from a standalone database mode instance to an MD Core instance using shared database mode.

Note: Officially, importing configurations between standalone and shared database modes is **not supported** due to potential differences in configuration settings. Some configurations may exist only in standalone mode but not in shared DB mode, and vice versa. Additionally, the use cases for these modes differ, making configuration discrepancies expected.



Cause:

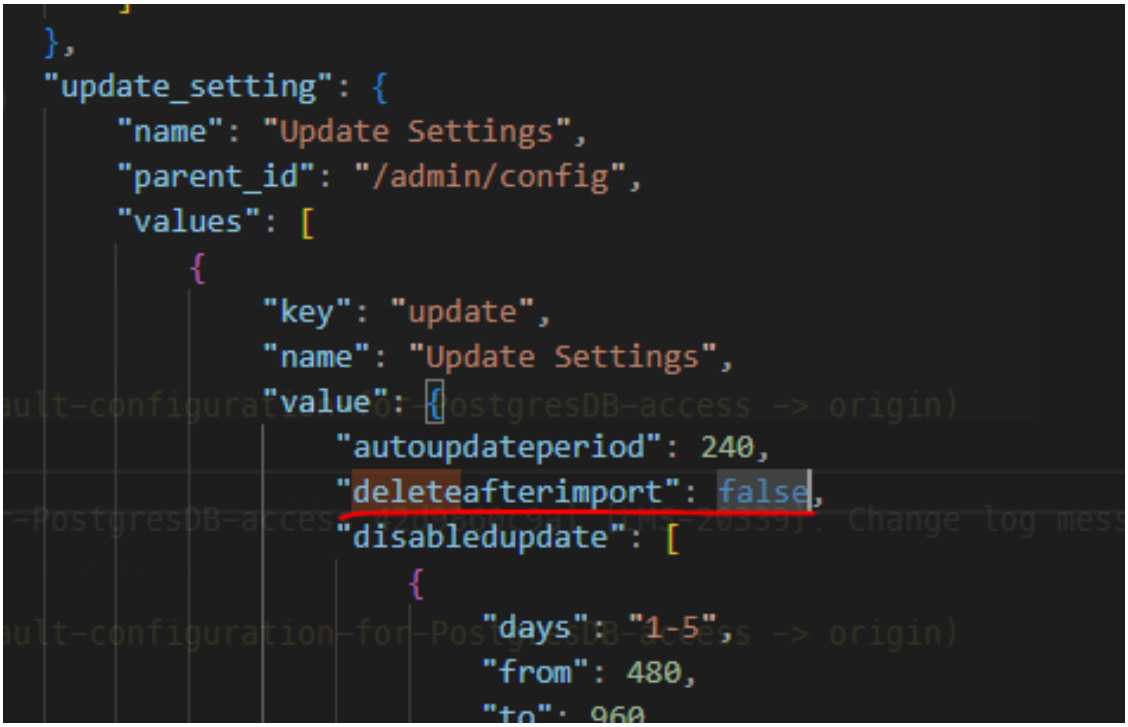
The `deleteafterimport` configuration is not valid for **Shared Database Mode**. This option specifies that once **MD Core** completes updating the package from the configured folder, it will automatically delete the files in that folder. However, this behavior is not supported in **Shared Database Mode** due to its centralized data management structure.



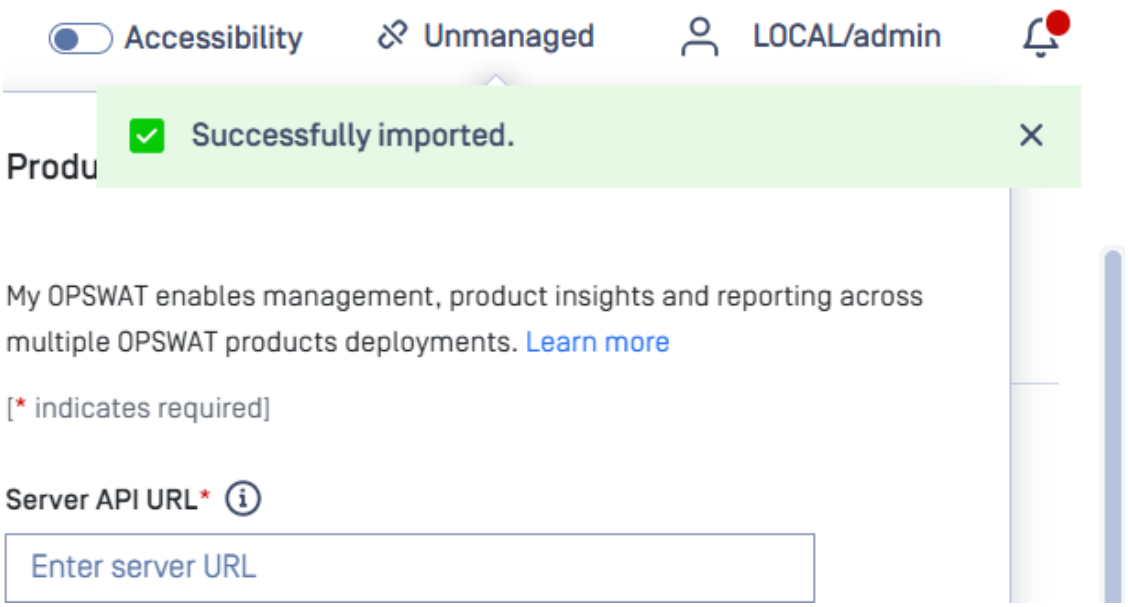
But in shared db mode we can have multiple MD Core point to the same location (shared folder/network) this option is always disabled in shareddb, because if not it will delete all files other MD Core cannot have the files failed to updates.

Temporary solution:

- Open the **settings_export.json** file located in the exported ZIP file.
- Modify the specified field "**deleteafterimport**" from **true** to **false**.
- Recompress the files into a ZIP archive.
- Import the updated ZIP file to MD Core.



- Ensure that the configuration is imported successfully.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What is the meaning of action_ran and action_failed in JSON scan result?

Check Your Version:

This article applies to all MD Core from V4.18 releases deployed on Windows and Linux systems.

Understanding `actions_ran` and `actions_failed` in MetaDefender Core Responses

When you submit a file to *MetaDefender Core (MD Core)* for scanning and sanitization, it may return two specific fields in the JSON response:

- *actions_ran*
- *actions_failed*

These fields indicate which actions were performed successfully and which ones encountered issues during the sanitization process.

`actions_ran`: Actions Completed Successfully

This field lists all actions that *successfully ran* on the submitted file.

- Values are returned as a string, with multiple actions separated by a vertical bar `|`.
- An *empty string* indicates that *no action was successfully executed*.

Possible Values

Sanitized:*Deep CDR__ARC Compression Sanitization*_**Sanitized Partially:**_

This can occur when:

- Some nested files failed sanitization.
- Some file types were excluded from sanitization (based on configuration).
- The archive contains blocked file types. (*Only shown if the "Distinguish Partial Archive Process Result" option is enabled in Archive Sanitization settings.)*

Sanitization Skipped : The file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.

`actions_failed`: Actions That Failed or Were Not Completed

This field lists actions that ***failed or were interrupted*** during processing.

- An *empty string* means *no failure occurred* during sanitization.

Possible Values:

Sanitization Failed: Sanitization was attempted but did not succeed.

Sanitization Timed Out: The sanitization process exceeded the *standard timeout limit* set for the action.

Sanitization Global Timeout Exceeded: The entire scan process exceeded the *global timeout* defined in the Workflow configuration:

- *Workflow > [Workflow name] > Timeouts > Enable global timeout*

Sanitization Canceled: The sanitization process was *manually canceled* by a client or user while in progress.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why can't I sanitize HEIC files?

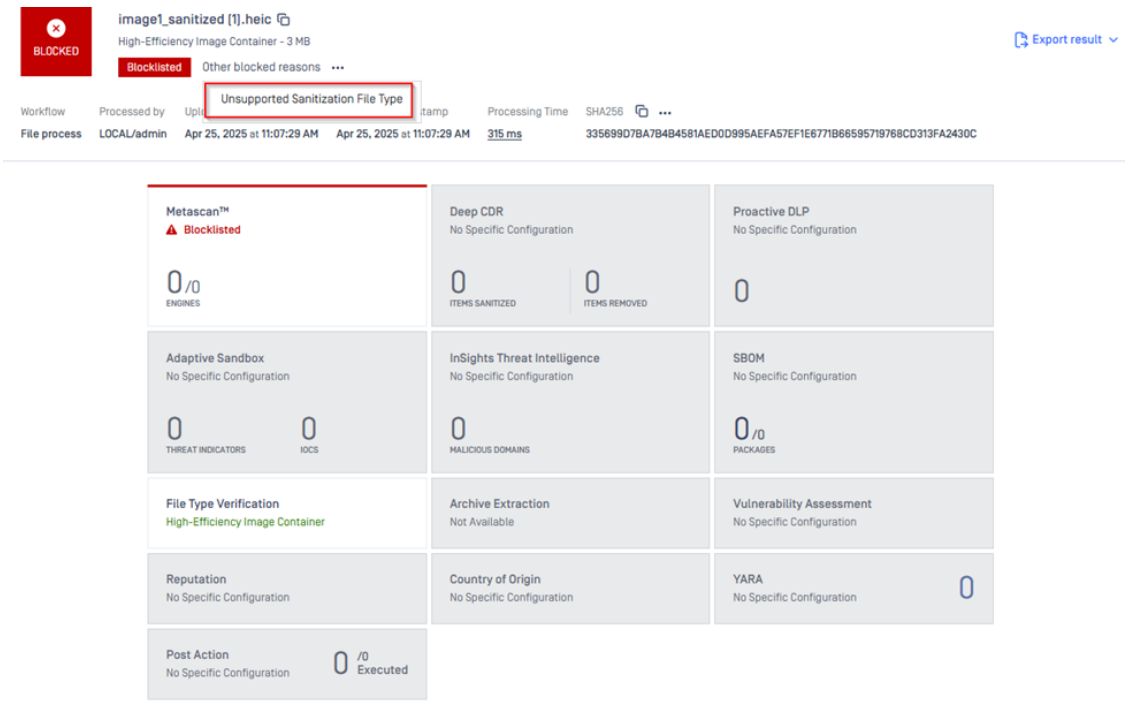
Issue:

You may encounter issues when attempting to sanitize HEIC image files using MetaDefender Core with Deep CDR enabled.

Symptoms include:

- HEIC files are blocklisted with the reason: "Other reasons"
- When viewing additional details, the message "Unsupported Sanitization File Type" is shown
- This occurs even though HEIC is listed as a supported file type on the Deep CDR Supported File Types documentation page

Please refer to the screenshot below for more details on the error message related to the failed scan of a HEIC file:



Solution:

Support for sanitizing HEIC files is available on-demand. The HEIC file format is based on x256 encoding, it requires a special license from a 3rd Party library to process.

If you are seeing them blocked due to unsupported file type errors, it's likely because your current license does not include this feature.

To resolve this:

- Please contact OPSWAT Customer Support or your Sales representative to request an update to your license key that enables HEIC sanitization.

Once the license is updated, HEIC sanitization should function as expected.

Support:

If you encounter any issues during the activation process or require further assistance, please contact OPSWAT support via My OPSWAT.

Our support team will be able to provide you with the necessary guidance and resolution for any license-related concerns.

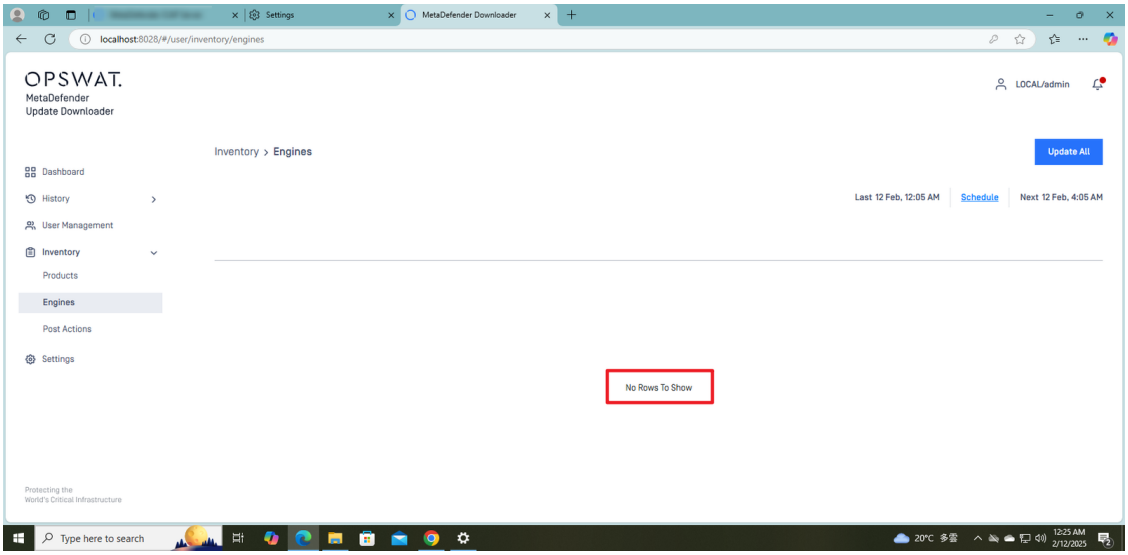
Why Am I Getting "No Rows to Show" When Using a Proxy Without Authentication in Browser Settings?

Problem

When you set the proxy without authentication in Edge or Chrome, it typically applies to the system proxy settings. However, the MetaDefender Update Downloader relies on WinHTTP instead of WinINet, meaning it does not inherit

the browser's proxy settings.

As a result, you may encounter the error: "No rows to show."



Solution

Since the MetaDefender Update Downloader currently does not provide a UI option to configure proxy settings, you need to set the WinHTTP proxy manually using the `netsh` command:

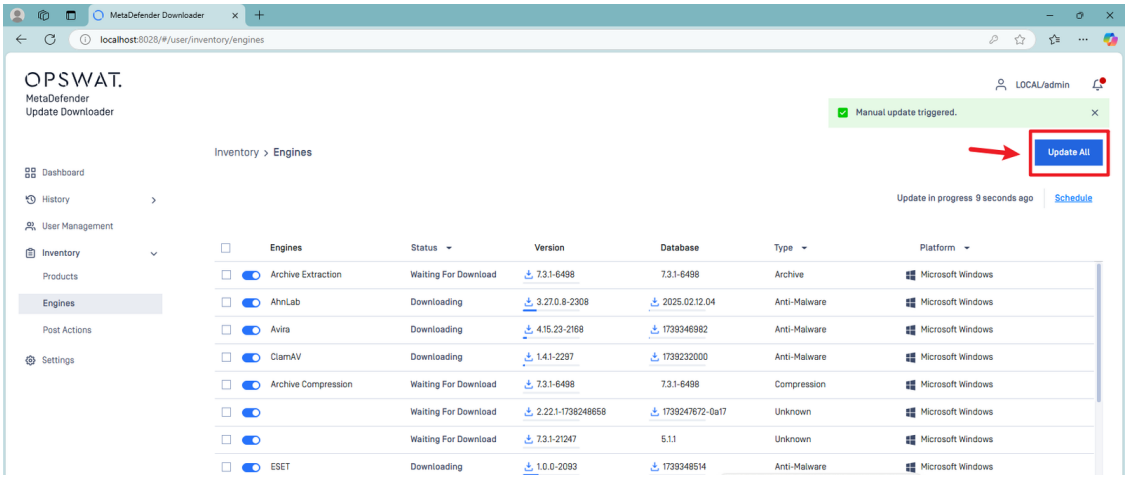
1. Open Command Prompt as Administrator:

- Press `Win + R`, type `cmd`, and press `Ctrl + Shift + Enter`.

2. Set the WinHTTP Proxy:

- Run the following command, replacing `<proxy_ip>:<port>` with your actual proxy details: `netsh winhttp set proxy <proxy_ip>:<port>`
- Example: `netsh winhttp set proxy 192.168.1.100:8080`

3. Refresh the Update Downloader web console and click "Update All."



4. Check if the page now displays correctly.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How Does TLS 1.0/1.1 Deprecation Affect Compatibility Across OPSWAT Products?

Check Your Version:

This article applies to all MetaDefender Email Gateway Security (EGS), MetaDefender Core (MD Core), MetaDefender Sandbox, MetaDefender MFT deployed on both Windows and Linux systems

Issue

A customer experienced issues sending emails to a specific recipient using MetaDefender Email Gateway Security (EGS) version 5.7.4. While receiving emails worked without issue, outgoing messages to this customer consistently failed. The recipient organization confirmed that their infrastructure only supports **TLS 1.1**, which is incompatible with OPSWAT's updated TLS policies.

The customer requested guidance on enabling TLS 1.1 on various OPSWAT products (EGS, MDCORE, Sandbox) or implementing exceptions for specific IP addresses or DNS names to restore mail flow to the affected domain.

Root Cause

As of version **5.7.0 and later**, OPSWAT has **fully deprecated support for TLS 1.0 and TLS 1.1** across its product line. This deprecation aligns with modern security standards and is intended to protect the integrity and confidentiality of communications handled by OPSWAT products.

Affected Products and TLS Support

Product	TLS 1.0/1.1 Support	TLS 1.2+ Support	Notes
MetaDefender Email Gateway (EGS)	❌ Not supported	✅ Supported	No exceptions possible by IP/DNS
MetaDefender Core (MDCORE)	⚠️ Possible via config, **not recommended**	✅ Supported	Can be enabled via ssl.conf.mdcore, but discouraged
MetaDefender Sandbox	❌ Not supported	✅ Supported	Follows EGS policies
MetaDefender MFT	❌ Not supported	✅ Supported	TLS 1.1 fully deprecated

Configuration Exceptions

EGS

No exceptions can be made within **EGS** for specific IP addresses or domains to enable TLS 1.0/1.1. This restriction is a security safeguard designed to enforce consistent secure communication across all environments.

MD Core

While MD Core may technically allow for legacy protocol configuration changes (via ssl.conf.mdcore), **OPSWAT does not support or recommend** enabling deprecated TLS versions due to known vulnerabilities and risk factors.

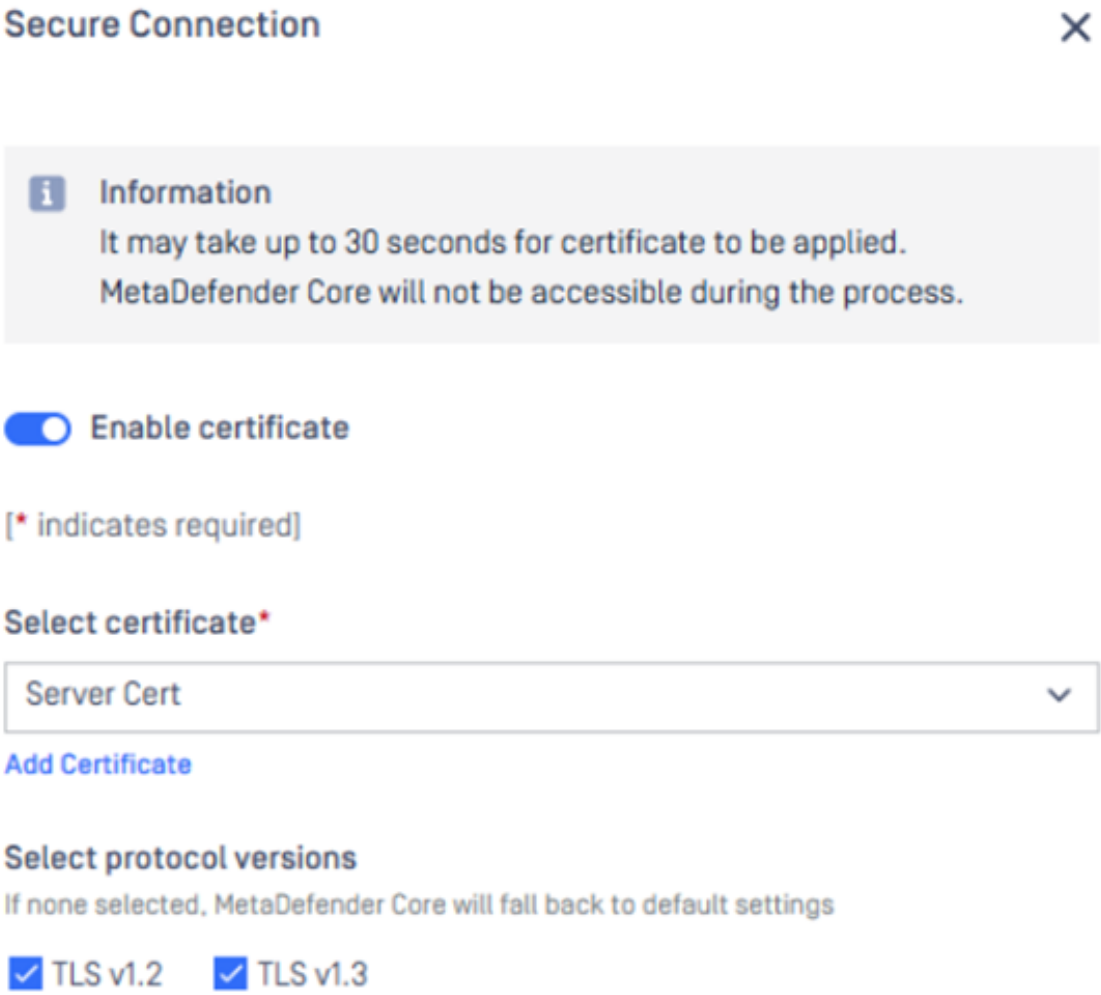
SSL customizations are easy to manage in MD Core v14+. We recommend using the new Secure Connection Settings shown below instead of the conf file. For best security please stick with a **minimum** of TLSv1.2.

Recommendation

If a partner or customer is unable to upgrade from TLS 1.1 due to legacy system limitations, the following alternatives are recommended:

1. E**ngage the partner organization** to encourage the adoption of TLS 1.2 or newer protocols.
2. **Implement an SMTP relay or intermediate mail gateway** that can downgrade/upgrade TLS between the OPSWAT environment and the legacy system (acting as a secure bridge).
3. **Separate the affected communication route** from OPSWAT-secured email flow if compliance/security policies allow for temporary workarounds.

Since **v5.14.0**, MetaDefender Core supports configurable TLS versions via the **web console**. Only **TLS 1.2 and 1.3** are available. We recommend configuring TLS settings through the web console instead of the config file to reduce errors and ensure better manageability.



Conclusion

OPSWAT prioritizes strong encryption and secure communication protocols. TLS 1.0 and 1.1 are no longer supported, and exceptions to this rule are not possible within most OPSWAT products, including Email Gateway Security.

Customers needing assistance in architecting alternative solutions or facing similar compatibility concerns are encouraged to reach out to OPSWAT Support for tailored guidance.

Related Documentation:

- OPSWAT SSL Configuration Guide
- MetaDefender Email Gateway Release Notes

Support:

If Further Assistance is required, please proceed to log a support case or chat with our support engineer.

Why can small file changes bypass the Antivirus detection?

When you open a file in a text editor and **add even a single character**, you **modify the file's binary content**. This changes the **file's checksum (hash)** — a fingerprint used by antivirus software to recognize known malware. Many traditional antivirus engines rely heavily on **static signatures**, which are based on specific patterns or hashes.

So, when you add, for example, a space:

- The file's checksum (e.g., its MD5 or SHA256 hash) changes.
- Signature-based AV engines no longer recognize it as matching a known malicious file.
- Unless the antivirus is doing **behavioral analysis**, **deep scanning**, or **structure-aware parsing** (looking inside the file in a smarter way), it won't flag it anymore.

In short:

The AV was looking for an *exact match* of the malicious file, and your tiny edit broke that match.

Important:

- Modern, better AVs and EDR (Endpoint Detection and Response) systems often go beyond simple signature matching — they can **analyze the file's structure, detect exploits**, or **simulate its behavior**.
- But many basic AV engines are still very "hash-dependent."

Heuristic detection is *specifically* designed to catch exactly these kinds of tricks, where a simple file change (like adding a space) would fool basic signature-based detection.

Here's how it helps:

- **Heuristics** don't rely only on exact file hashes or fixed byte patterns.
- Instead, they **analyze the structure, behavior, and content patterns** of the file to see if it "acts suspiciously," even if it's slightly modified.
- For example, in a PDF, heuristics might look for:
  - Hidden JavaScript code
  - Unusual embedded objects (like an executable inside the PDF)

- Exploit patterns targeting vulnerabilities (like old Adobe Reader flaws)
- Abnormal structures (e.g., heavily obfuscated streams)

So **even if you change the file slightly** (adding a space, changing a few bytes), **a good heuristic engine** would still flag it if the **malicious core behavior or structure** is present.

Example:

- Signature detection: "This exact MD5 hash = known bad."
- Heuristic detection: "This PDF has JavaScript that automatically runs shellcode — suspicious, even if it's new or slightly altered."

However, heuristics aren't perfect:

- They can sometimes **miss** very well-obfuscated threats.
- They also sometimes generate **false positives** (marking legit files as suspicious).

For a comprehensive list of engines that support heuristic capabilities, please refer to: Anti-malware capabilities

Info:

Note that AI/heuristic capabilities may be disabled by default and must be manually enabled via the engine's advanced settings, as outlined at: Configuring engine advanced settings

To enhance detection capabilities, we recommend complementing MetaDefender's Multiscanning technology, which leverages 30+ leading anti-malware engines, with additional layers of protection such as our Deep Content Disarm and Reconstruction (Deep CDR) engine and Sandbox engine for adaptive threat analysis.

Support:

If **Further Assistance** is required regarding this topic, please proceed to log a **support case or chat with a support engineer**.

How do I verify if MetaDefender products are sending out syslog messages to the syslog server?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

After setting up syslog configuration, there are a few ways to verify that the MetaDefender product is sending syslog:

1. Check on the syslog server if it's receiving syslog messages.
2. Set up packet capture on the server running MetaDefender product to see if syslog messages are being sent out on the configured syslog port and protocol (default is UDP port 514).

If there are no messages received on the syslog server, or if packet capture doesn't show any traffic sent on syslog port, you may need to check your syslog configuration for the MetaDefender product to make sure it's correct, and/or check network connection between the MetaDefender server and the syslog server.

To set up and analyze packet capture, please follow one of the following methods:

Method 1: Use a packet capture program

You can use a dedicated program to capture packets from the server, such as Wireshark.

Method 2: Use Windows built-in tool

Packet Monitor (pktmon) is a Windows built-in tool that you can use to capture packets. To use it, follow these steps:

1. In Powershell or Command Prompt, add the pktmon filter for syslog port: `pktmon filter add -p 514`
2. (Optional) List the current filters: `pktmon filter list`
3. Start packet capture, specifying that the entire packet should be logged with the argument "-pkt-size 0": `pktmon start --capture --pkt-size 0`
4. After performing some actions on the MetaDefender product that send syslog messages, stop the capture: `pktmon stop`
5. The capture file "PktMon.etl" (by default will be saved in C:\Windows\System32) can be analyzed using tools like Microsoft Network Monitor or Wireshark.
6. (Optional) Export the capture to a human-readable format (the "output.txt" file will be in the same directory as "PktMon.etl": `pktmon format PktMon.etl -o output.txt`

Method 3: Use Linux built-in tool

Tcpdump is a Linux built-in tool for packet capture. Run the following command in terminal to capture traffic on UDP port 514:

- `sudo tcpdump -I any udp port 514`

This command will display the captured packets in real-time in your terminal. If you want to save the captured packets to a file analysis, run the following command:

- `sudo tcpdump -i any udp port 514 -w capture_file.pcap`

To read and analyze the saved file, run the following command:

- `tcpdump -r capture_file.pcap`

Log message format

If the monitoring/analytics solution is configured to only ingest/forward log messages formatted in **Common Event Format (CEF)**, the log format will need to be configured to CEF as well in MetaDefender Core.

Symptom: if the monitoring/analytics solution is configured to only ingest logs in CEF, you may notice that MetaDefender Core is sending out syslog (when analyzing packet capture), but nothing shows up on the syslog server.

Remediation: follow the MetaDefender Configuration documentation to configure MetaDefender Core log message format to CEF, and it will send out syslog messages formatted in CEF as well.

- Windows:
  - Path: `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`
  - Key: cef
  - Type: string value
  - Value: true
- Linux:
  - Path `/etc/ometascan/ometascan.conf`
  - Content:

```
[logger]
cef=true
```

Support:

If you are having issues or require further assistance regarding Verifying syslog configuration on MetaDefender Core, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I fix files being blocked due to unexpected "File type mismatch" verdict?

Check Your Version:

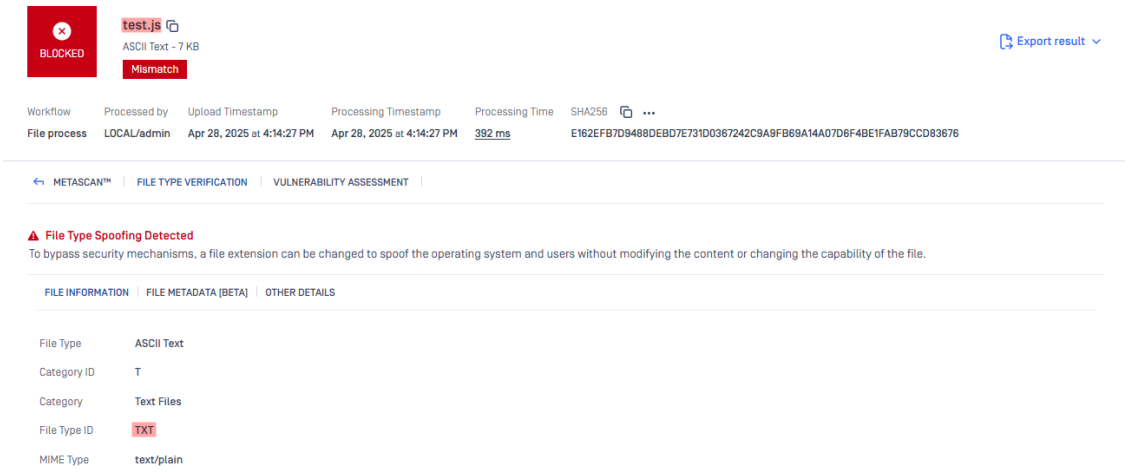
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Behavior

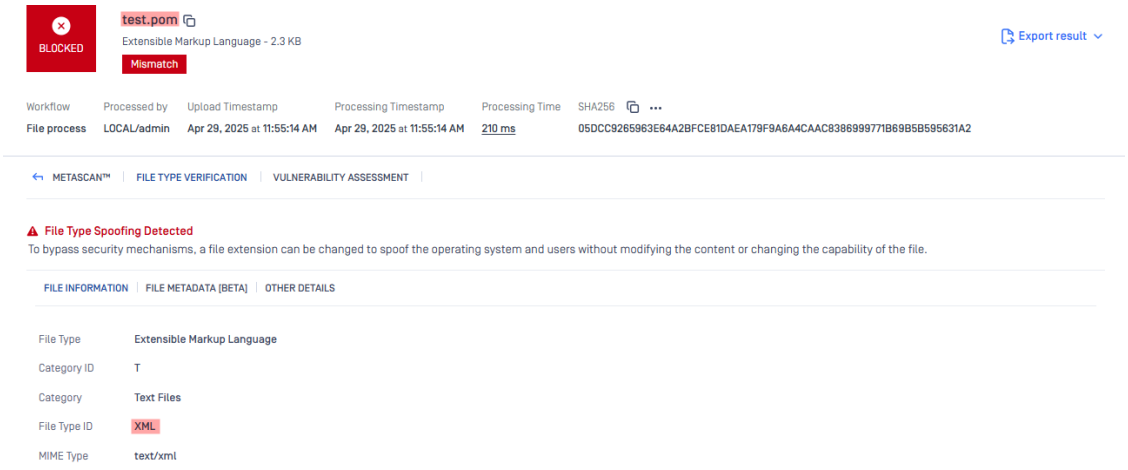
When File type engine's "Detect file type mismatch" option is enabled in your workflow, you may experience some

files (such as JavaScript, Project Object Model) being blocked by File type engine due to "File Type Spoofing detected".

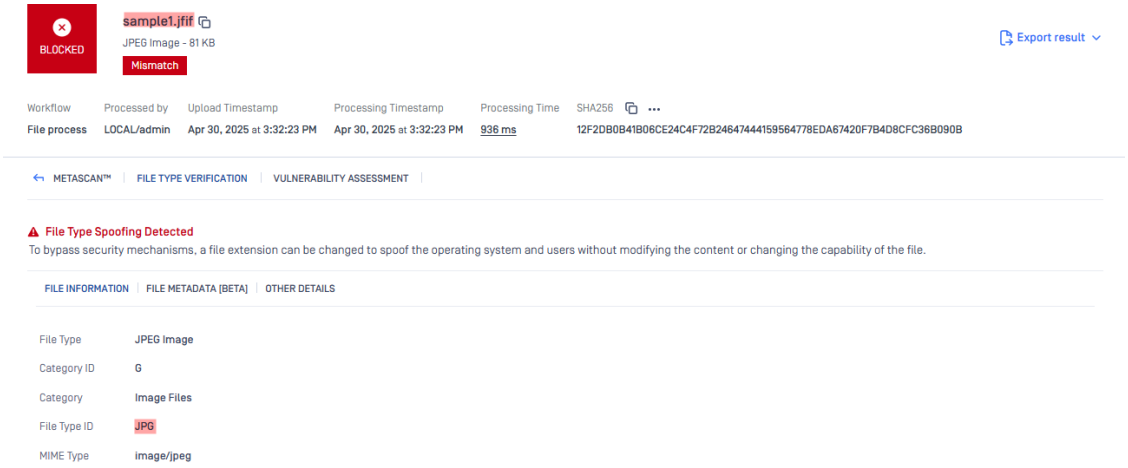
- Example 1: JavaScript source file being detected as TXT (plaintext) and blocked



- Example 2: Project Object Model file being detected as XML and blocked



- Example 3: JPEG File Interchange Format image (JFIF) being detected as JPG and blocked



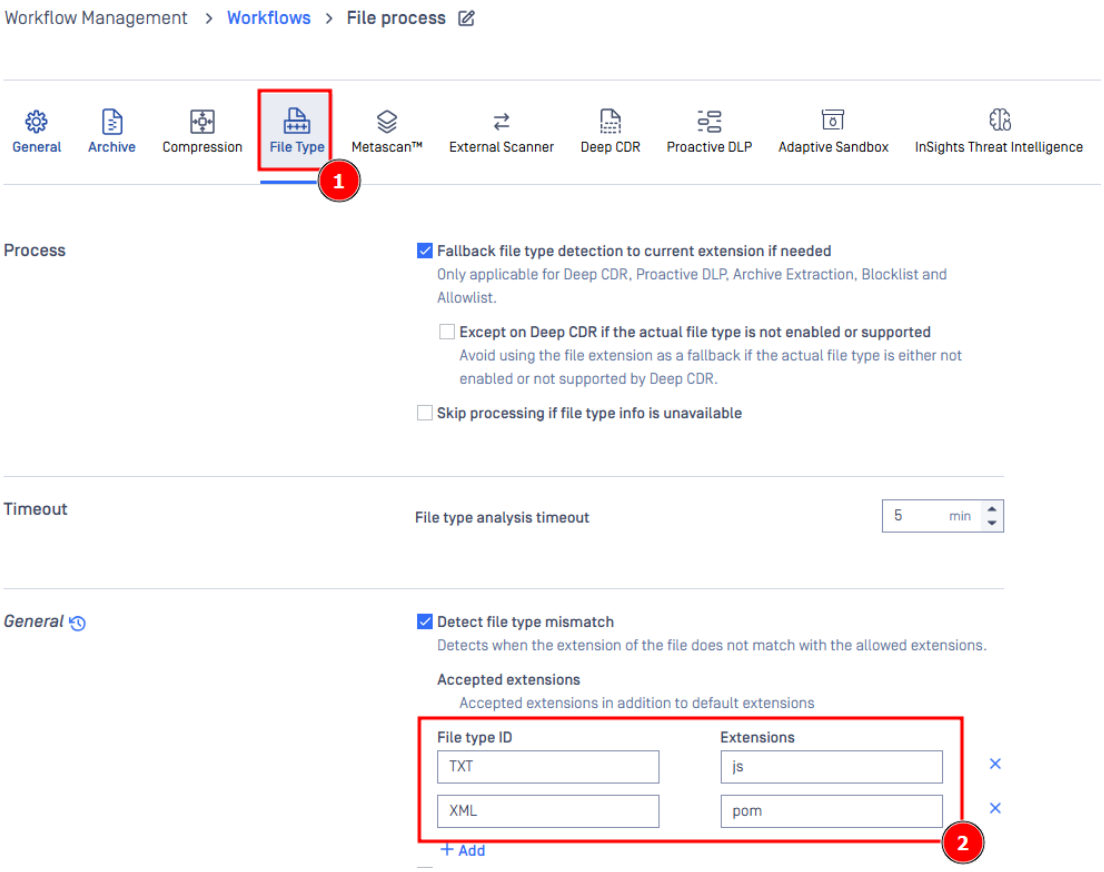
In these examples, the underlying file type used by JavaScript, POM, JFIF files are plain text, XML, and JPEG respectively, even though their names have the extensions .js, .pom, and .jfif. Thus, that's the type detected by File type engine.

Remediation

To avoid blocked verdict due to File type mismatch, there are 2 configurations you can use.

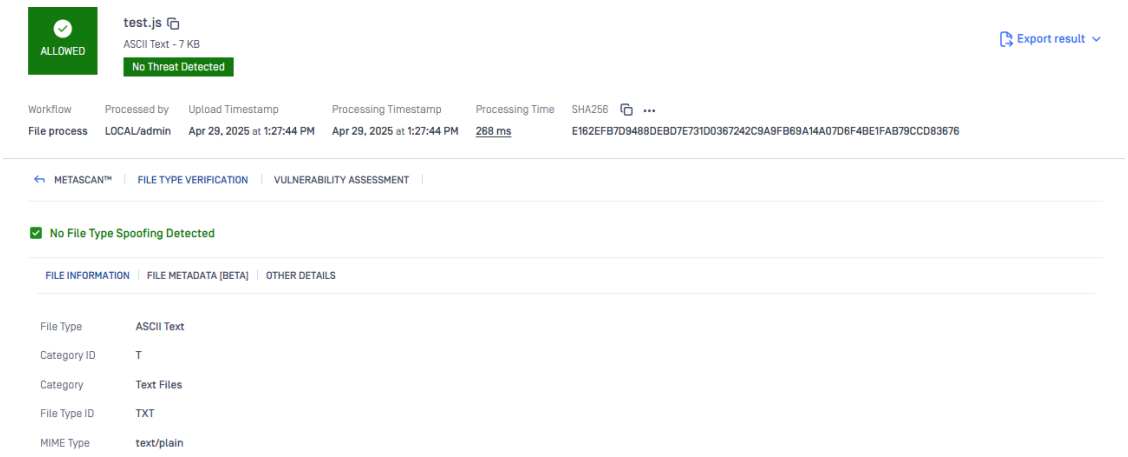
1. Add the filename's extension to the list of accepted extensions for that file type:

- **Note :** more details can be found on the article on Customize mismatch detection settings.
- In workflow settings, go to "File Type", and add the following in "Accepted extensions":
- File type ID: the file type detected by the engine (e.g. TXT, XML, JPG)
- Extensions: the extension in the file's name that you want to allow (e.g. js, pom, jfif):



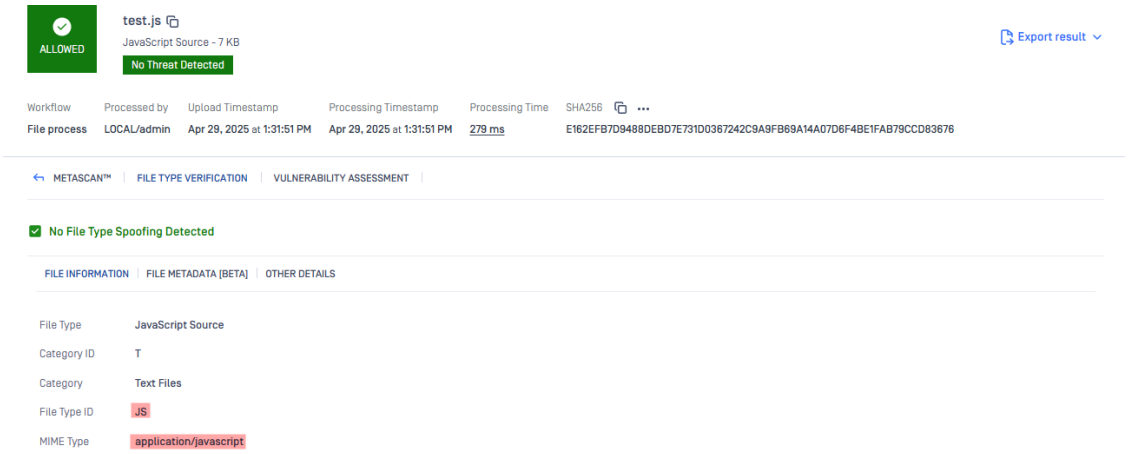
-

- After making this configuration, the JavaScript source file is still detected as TXT, but it is no longer blocked due to mismatch:



2. In the case of text files, enable File type engine's machine-learning module to classify the file type based on its contents:

   - More details about this feature can be found at Text detection with AI.
   - In Inventory > Modules > File Type, check 'Enable Machine learning module".
   - In workflow settings, go to "File Type" and enable "Classify with Machine Learning".
   - File type engine is now able to detect that the source file is JavaScript, and no longer blocked due to mismatch:



Warning:

The machine learning text detection feature is disabled by default, and unlike file signature matching, AI-based relies on heuristics and is not guaranteed to be precise.

In cases where the machine-learning feature isn't able to correctly detect the file type of text files, **it's generally preferred to use option 1 to customize the mismatch detection setting**.

Support:

If you have followed the instructions above but are still unable to fix source code files being blocked due to "File type mismatch", please follow these instructions on **How to Create Support Package With Bundle Tools?**, before **creating a support case or chatting with our support engineer**.

Why are MetaDefender Core engines permanently failed?

Introduction:

This knowledge base article provides details regarding the Core AV engines

Details:

We found that the issue is coming from an environmental activity which deleted the /var/tmp folder after 30 days, thus preventing MD Core's internal tasks to update the engines because the oms_testfile is deleted. We have performed the below steps to resolve the issue:

- we have checked the current configuration of the automatic cleanup task with below command:

```
cat usr/lib/tmpfiles.d/tmp.conf
```

- We have then proceed with modifying the file using below command:

```
sudo vim usr/lib/tmpfiles.d/tmp.conf
```

- Inside the config file, we have disabled the following parameters by adding a comment sign "#" in front of them:

`# v /tmp 1777 root root 10d ->` deletes the folder /tmp after 10 days

`# v /vat/tmp 1777 root root 30d ->` deletes the folder /var/tmp after 30 days -> Here MD Core creates it's temporary files and these files and folders should not be deleted during functionality

- after this activity has been completed, save the document, using the command

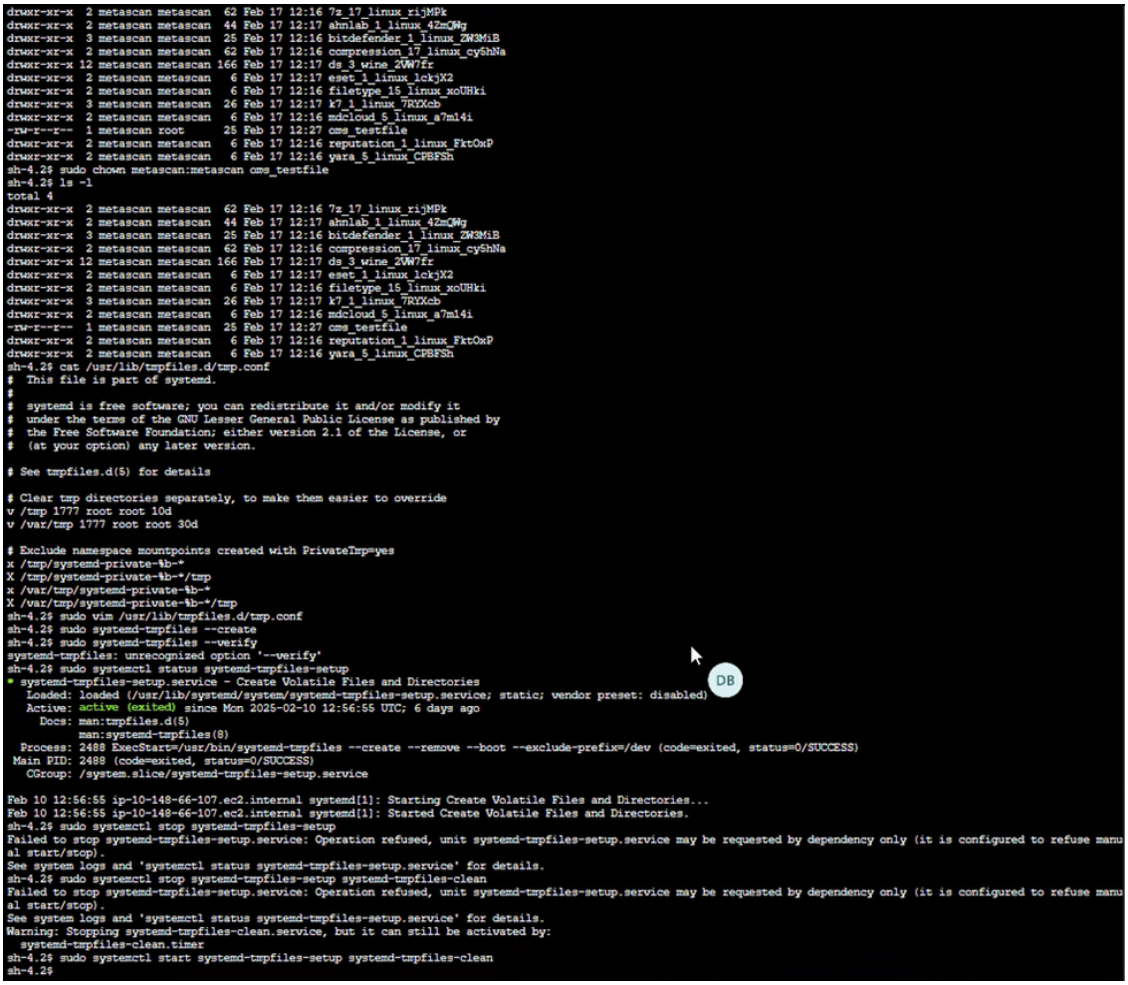
```
:wq
```

- We have implemented these changes by restarting the cleanup service using the below command

```
systemctl stop systemd-tmpfiles-setup.service systemd-tmpfiles-clean.timer
```

```
systemctl start systemd-tmpfiles-setup.service systemd-tmpfiles-clean.timer
```

Once the above activities have been completed, the cleanup services should not affect MD Core functionality.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

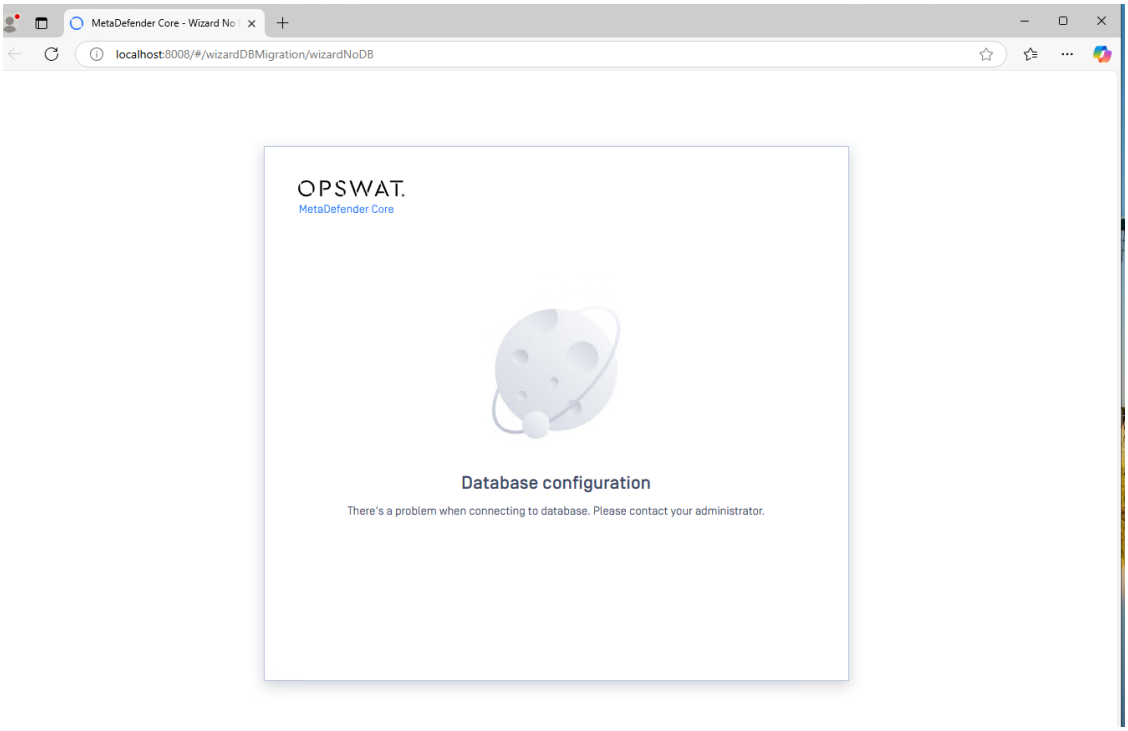
Why Does MetaDefender Core Fail to Connect to the Database After a System Restart?

Check Your Version:

This article applies to all MetaDefender Core version releases deployed on Windows.

Scenario

When MetaDefender Core is configured to use a remote PostgreSQL database, the service may fail to reconnect to the database after the operating system restarts, even if the PostgreSQL server is running. As a result, you may see an error dashboard when trying to access the MetaDefender Core web UI.



When checking MD Core logs,

```
[INFO   ] 2025.04.29 16:05:37.428: (common.base) ==== START ====, version='5.14.0' [msgid: 356]
[INFO   ] 2025.04.29 16:05:38.029: (core.postgres) Postgres info, info='Temp directory: ', param='c:\windows\temp' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.033: (core.postgres) Postgres info, info='Current hba config version = 3, expected = 3', param='' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.035: (core.postgres) Postgres info, info='Postgres private user is not defined', param='' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.036: (core.postgres) Postgres info, info='Database name', param='metadefender_core_twbst1' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.036: (core.postgres) Postgres info, info='Postgres private password is not defined', param='' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.036: (core.postgres) Postgres info, info='Generating private password for the first time', param='metadefender_core_twbst1' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.037: (core.postgres) Postgres info, info='Checking for database upgrade', param='' [msgid: 4085]
[INFO   ] 2025.04.29 16:05:38.049: (core.postgres) Postgres info, info='Can't create connection', param='' [msgid: 4085]
```

We can see the MetaDefender Core service could not reach the Database.

```
[INFO   ] 2025.04.29 16:05:38.049: (core.postgres) Postgres info, info='Can't create connection', param='' [msgid: 4085]
```

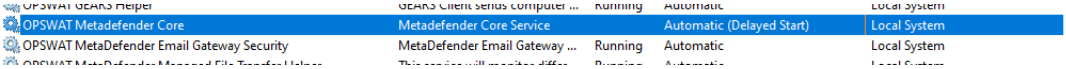
The root cause

The root cause is that MetaDefender Core service starts immediately after the operating system boots, before the network is fully initialized. As a result, the MetaDefender Core service is unable to connect to the PostgreSQL server during startup. This leads to the dashboard displaying a "could not connect to the database" message.

When the service is restarted manually, MD Core reconnects to the PostgreSQL server. By that time, the network is fully established, allowing the connection to succeed. This is why the service functions normally after a manual restart.

The resolution/workaround

The recommended workaround is to configure the MetaDefender Core service to start with the "Delayed Start" option.



This option allows the MetaDefender Core service to wait until the network has initialized successfully before starting service.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What do you do if your engine fails to update?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5releases deployed on Windows and Linux systems.

If the engine in your OPSWAT MetaDefender Core fails to update and you are unsure of the product version, you can follow these steps:

1. **Check Update Configuration**:

- Ensure that the update configuration is set correctly. Updates can be configured to **Internet** (Automatic), **Local** (Automatic), or **Manual**. For more details, refer to the Engines Update Configuration.

2. **Use Update Downloader**:

- If your MetaDefender Core server is deployed offline, you will need to use the **MetaDefender Update Downloader** utility to download the engine updates definition updates.
- Navigate to **Product Downloads** in My OPSWAT > select MetaDefender Core > locate the **Modules & Utilities** section > download the **Update Downloader**.
- After downloading, configure the update downloader to fetch updates and apply them to the offline MetaDefender Core system. For more information, follow the instructions in the Update Downloader server configuration file.

3. **Apply Updates Manually**:

- Upload the update packages through the MetaDefender Core **Management Console**, follow the instructions in the Engines Update Configuration.

4. **Verify Internet Connectivity**:

- if not using MD Core in air-gap or offline mode, ensure that the server has proper internet connectivity to access the update servers. Allow access to `https://update.dl.opswat.com` for fetching engine/database updates.

5. **Upgrade to the Latest Version**:

- If the issue persists, consider upgrading to the latest version of MetaDefender Core. Visit **My OPSWAT** to download the latest version. Upgrading ensures you benefit from the latest features and bug fixes.

These steps should help resolve the issue with engine updates in MetaDefender Core.

> **Support:**
> If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What is causing my Scans to be slow on MetaDefender Core?

Slow scan performance in MetaDefender Core can be caused by several factors:

- **Older MetaDefender Core Version:** It is best practice to upgrade to the latest version since the latest version has the latest bug fixes and performance improvements.
- **MetaDefender Core Configuration**: Settings such as deep scans, heuristic analysis, and archive scanning can impact performance. More thorough scans require more resources and time.
  - Please note that only certain AV engines have Heuristic analysis. The Heuristic scan is a method of detecting potential threats by analyzing the behavior, structure, and characteristics of files rather than relying solely on known malware signatures. This approach allows the AV engine to identify new, unknown, or modified threats that may not yet be included in traditional signature databases.
  - AV engines which have this feature enabled may be slower than others.
- **Number of Engines**: The number of antivirus engines configured can affect performance. While more engines provide better security, they can slow down the scanning process.
- **File Characteristics**: The type and size of files being scanned (for example: executables, archives, multimedia files) can influence performance, as some file types are more complex to process.
  - Examples of file types that can affect the performance of MetaDefender Core during scanning include:
    - **Executables**: These files can be complex to process due to their potential to contain malicious code or embedded resources.
    - **Archives**: These files, such as ZIP may contain multiple nested files, which increases the processing complexity and time required for scanning.
    - **Multimedia files**: These files, like videos or images, can vary significantly in size and complexity, impacting the scanning process depending on their characteristics.
- **Deep CDR and Proactive DLP**: These features can impact the scanning process in MetaDefender Core. Deep

CDR processes files by disarming and reconstructing them, which involves decompression, pixel processing, and reconstruction, potentially adding complexity to the scanning process. Proactive DLP, on the other hand, checks for sensitive data in files and can redact or watermark the data, which also adds processing steps.

- **Concurrent Scans**: The number of simultaneous scans being performed can divide the machine's resources and impact overall performance.
  - To improve performance when dealing with concurrent scans in MetaDefender Core, you can consider the following steps:
    - **Optimize Background Processes**: Suspend any nonessential background processes and services running on the operating system, as they can consume resources and impact scanning performance.
    - **Adjust Concurrent Scans**: The number of simultaneous scans can divide the machine's resources. Reducing the number of concurrent scans or optimizing the integration method can help improve performance.
    - **Set Power Profile to Performance**: Configure the system's power profile to "Performance" settings to avoid CPU throttling and ensure maximum processing power.
    - **Upgrade Hardware**: If possible, use systems with higher CPU cores (e.g., 32-48 cores) and sufficient memory to handle high workloads efficiently.
    - **Configure Parallel Scans**: Increase the parallel number of scans running per engine using the `parallelcount` setting. This allows better utilization of CPU resources for engines supporting multi-threaded scans.
    - **Increase Queue Size**: If the queue is full of incoming scan requests, increase the maximum queue size using the API call `PUT /admin/config/scan` with the appropriate configuration. Or update the queue by logging in to MD Core UI console > click on Settings > click on General tab
    - **Use the Latest Version**: Always ensure you are using the latest version of MetaDefender Core for optimal performance and to avoid legacy stability issues.
    - **Optimize Network**: For network scanning, ensure a minimum of 1 Gbps network speed, with 5 Gbps recommended, to avoid potential network-related issues.

- **Integration Method**: How MetaDefender Core is integrated into other applications (e.g., via APIs) can also affect performance, especially if the integration is not optimized.
  - An example of how the integration method can affect performance is when MetaDefender Core is integrated into other applications via APIs. If the integration is not optimized, it can lead to inefficiencies in processing, such as increased response times or reduced throughput. For instance, the location of the client application (remote or local) and the use of system caching or engine-level caching can significantly influence performance.

- **Real-Time Protection Systems**: Real-time protection software on the MetaDefender Core server may interfere with scans by deleting or quarantining files, which can disrupt the scanning process.
  - To address this issue navigate to the settings of the real-time protection software.
  - Locate the 'exclusions' or 'exceptions' section in the settings.
  - Add the full installation path of MetaDefender Core to the exclusions list

- **Engine-Specific Factors**: Some engines, like ClamAV, are known to have slower performance due to their design.

Adjusting engine-specific settings, such as limiting file size or disabling certain features, can help mitigate this.

It is recommended to review these factors and optimize the configuration to improve scan performance. As stated previously, always ensure you are using the latest version of MetaDefender Core for the best performance and stability.

After confirming that you've optimized the configuration to the best that you can and scans are still being processed slowly, please submit a support ticket to OPSWAT Support.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What do I do if my modules disappear from the Core UI?

If you are noticing that module(s) are missing from the Core UI there are a few different things that you can check prior to submitting a support ticket:

1. **MetaDefender Core Cannot Reach the Update Server:** The update server for MetaDefender Core is https://update.dl.opswat.com, which is used for fetching engine and database updates. To ensure that MetaDefender Core can reach the Update Server, please review the following troubleshooting steps below:
    1. **Check Internet Connection**: Verify that the MetaDefender Core server has a stable internet connection. This is necessary for connecting to the update server.
    2. **Allowlist Required URLs**: Make sure the following URLs are accessible from your environment:
        - **https://activation.dl.opswat.com** (for license activation)
        - **https://update.dl.opswat.com** (for engine and database updates)
    3. **Firewall Configuration**: Verify that the firewall is not blocking the required URLs or ports. If the server is in a restricted environment, configure the proxy settings in MetaDefender Core to allow access to the update and activation servers.
    4. **Offline Updates**: If the server is offline, you can use the **MetaDefender Update Downloader** utility to download updates on a system with internet access and then transfer them to the offline server. The utility can be downloaded from **My OPSWAT Portal** under Produt Downloads > click on **MetaDefender Core** > under **Modules & Utilities** click on **Update Downloader for Offline Environment**
    5. **Check System Notifications**: Verify if there are any notifications in the MetaDefender Core interface indicating issues such as disk space or memory availability, as these can also affect updates.
2. **Restart the Services:** Sometimes, restarting the MetaDefender Core services can resolve issues with missing modules. Restart the services through operating system's services manager or by using the command line.
3. **Clear Browser Cache**: If you are accessing the MetaDefender Core user interface through a web browser, clear your browser cache and refresh the page. Cached data can sometimes cause display issues.
4. **Reinstall MetaDefender Core**: If the above steps do not resolve the issue, consider reinstalling MetaDefender Core. Ensure you have back up of your configuration and data before proceeding with a reinstallation.

1. Have a backup version (snapshot, recovery point,...)
2. Import/Export your configuration settings.
3. Export processing history as needed.

Contact OPSWAT Support:

If the problem persists, reach out to OPSWAT support for further assistance. In the ticket please provide the following details:

- Details about your environment
- The version of MetaDefender Core you are running
- Details about any specific errors you are receiving and relevant screenshots
- Support Package. This will allow us to check logs and check for any specific errors. Please review our documentations on How to Create Support Package With Bundle Tools?
- License key. This will allow us to confirm and verfiy which modules should be available to you based on the purchased package.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why Is Core Receiving HTTP 403 or Timeout Errors when downloading engine updates from OCM?

Issue

You may see log messages similar to the following in your Core system:

Code · html

```
[WARNING] 2025.04.10 18:23:40.925: (common.communication) Error while downloading file,
url='QUrl("http://XXXXX:9000/console/core/metadescriptor?
version=5.13.3&deployment=XXXXX&serial=0")', http_status='0', error='UnknownNetworkError',
errorDetails='Connection timed out' [msgid: 485]
[ERROR  ] 2025.04.10 18:23:40.925: (common.update) Error parsing metadescriptor, message='Could
not download metadescriptor' [msgid: 282]
[WARNING] 2025.04.10 18:23:51.796: (common.communication) Error while downloading file,
url='QUrl("http://XXXXX:9000/console/core/metadescriptor?
version=5.13.3&deployment=XXXXX&serial=0")', http_status='403', error='ContentAccessDenied',
errorDetails='Error transferring http://XXXXX:9000/console/core/metadescriptor?... - server
replied: Forbidden' [msgid: 485]
```

Cause

These messages usually mean that Core is having trouble connecting to OCM (OPSWAT Central Management). It may be caused by:

- Network timeout or proxy issues
- Port 9000/tcp being blocked
- Restrictions in your firewall or proxy

Solution

Please check the following:

1. **Proxy or Network Settings** Make sure your network allows Core to connect to the OCM server. If you're using a proxy, verify that it allows traffic to the OCM address on port **9000/tcp**.
2. **Firewall Rules** Ensure port **9000/tcp** is open between Core and OCM. This is required for downloading updates and descriptors.
3. **More Info on Ports** For a full list of ports used by OCM, see the documentation here: Recommended System Requirements - OPSWAT Central Management v7 - Legacy.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What do I do when licensed engines are not detected despite active licenses?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems

If you're seeing the error message "**No active licensed engines found**" in MetaDefender Core, follow the steps below to identify and resolve the issue:

1. Check License Configuration

- Confirm that the activated license key includes the necessary scanning engines.
- MetaDefender Core downloads and enables engines based on your license key.
- If the required engines are not included, you may need a different license key.
- Contact the OPSWAT support team for help with licensing validation or replacement.

2. Verify Engine Activation in the Console

- Open the MetaDefender Core console.
- Navigate to: **Inventory > Modules > Metascan**.
- Look for any activation or installation errors related to the engines.
- If you see issues, check the logs in the MetaDefender Core pod for more detail.

3. Ensure Sufficient System Resources

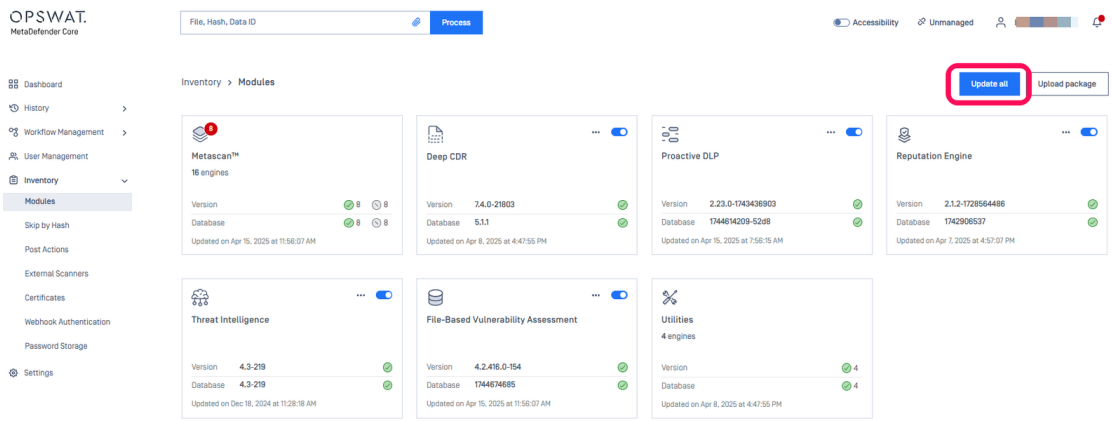
- Verify that your system meets the minimum resource requirements:
  - **CPU**: Sufficient cores for all licensed engines.
  - **Memory**: At least **3 GB available**.
  - **Disk Space**: At least **5 GB available**.
- Inadequate resources can prevent engines from initializing properly.

4. Check Internet Connection (if using online updates)

- A stable internet connection is required to download engine packages.
- Network interruptions or timeouts can result in incomplete or corrupted downloads.

5. Re-enable or Reinstall Engines

- Try disabling and then re-enabling the affected engines.
- If the issue persists, try the following steps:
  - Remove the engine and its associated database.
  - Deactivate the license, then reactivate it.
  - Navigate to the **Inventory** tab > **Modules**, and click the **Update All** button.



6. IKARUS Engine Specific Configuration

- If using the **IKARUS engine**, it requires shared memory configuration to function properly.
- For **Docker**, add the following flag:

Code · docker

```
--shm-size=1gb
```

For **Kubernetes (K8s)** deployments, include this in your YAML:

Code · yaml

```
spec:
  volumes:
    - name: dshm
      emptyDir:
        medium: Memory
  containers:
    - volumeMounts:
        - mountPath: /dev/shm
          name: dshm
```

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why do I receive errors when I upload the Config file in MetaDefender Core?

Summary

Uploading a Configuration File in MetaDefender Core (MD Core) may result in errors if there are mismatches between versions or platforms, or if file paths do not align with the current environment. This article outlines common causes and provides steps for resolution.

Issue

Users may encounter errors when attempting to upload a Configuration File to MD Core. Usually, the most common error message is related to an invalid pickupfolder path.

Cause

There are several potential reasons for these upload errors:

1. **Version Mismatch:** The Configuration File may not be compatible across different versions of MD Core. Changes in core functionality and configuration structure between versions can prevent successful file import.
2. **Platform Mismatch:** Configuration Files generated on one platform (e.g., Windows) may not be compatible with another (e.g., Linux). Attempting to upload a configuration from a Windows instance to a Linux-based MD Core

can lead to incompatibility errors.

3. **Incorrect File Paths:** One of the most frequent errors involves the pickupfolder path. This error typically occurs when the path in the Configuration File does not match the installation directory of the target MD Core system.

Resolution

1. **Ensure Version Compatibility:**

- Confirm that both the source and target MD Core instances are running the same version.
- If upgrading/downgrading the Configuration File manually, review version-specific documentation to adjust settings accordingly.

2. **Match Platforms:**

- Only use Configuration Files on MD Core instances with the same underlying operating system (e.g., Linux-to-Linux, Windows-to-Windows).
- Avoid cross-platform transfers to prevent compatibility issues.

3. **Correct the Pickupfolder Paths:**

- Open the Configuration File in a text editor.
- Locate the pickupfolder entry.
- Update the path to reflect the actual directory used by your MD Core installation.
  - Example: Change e:\MD Core\pickup to C:\Program Files\MD

Additional Notes

- Always back up your current configuration before making any modifications.
- Use caution when editing configuration files manually to avoid syntax errors or misconfigurations.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to access MetaDefender Core when SSO fails?

Check Your Version:

This article applies to all MetaDefender Core version 5.0 and newer releases deployed on Windows and Linux systems.

Overview

When Single Sign-On (SSO) is configured in **MetaDefender Core**, there may be situations where login attempts using SSO fail due to configuration errors, identity provider issues, or connectivity problems. To ensure continued access to the system for administrative purposes, MetaDefender Core provides a **backup login page** that bypasses the SSO flow.

This article outlines the issue and provides instructions for accessing the backup login page to perform diagnostics or disable SSO if needed.

Problem Description

Users are unable to access MetaDefender Core via SSO authentication. The following symptoms may be observed:

- Redirect loops or timeouts when attempting to log in
- HTTP 500 Internal Server Errors
- Login page not loading
- No access to UI for corrective actions

Solution: Access the Backup Login Page

MetaDefender Core includes a **backup login page** which allows direct access using local credentials (i.e., those stored in the Core database) — bypassing the SSO mechanism.

How to Access:

Replace <mdcore-host> with your actual MetaDefender Core hostname or IP:

https://<mdcore-host>:8008/#/public/backuplogin

For example: https://localhost:8008/#/public/backuplogin

Use This Page To:

- Log in with a local administrator account
- Access system settings
- Review or modify SSO configuration
- Revert to local authentication if needed

How to Disable SSO (If required)

Once logged in via the backup login page:

1. Go to **User Management** **Directories**.
2. Disable SSO temporarily by toggling the SSO name off.
3. Save the changes.

Best Practices

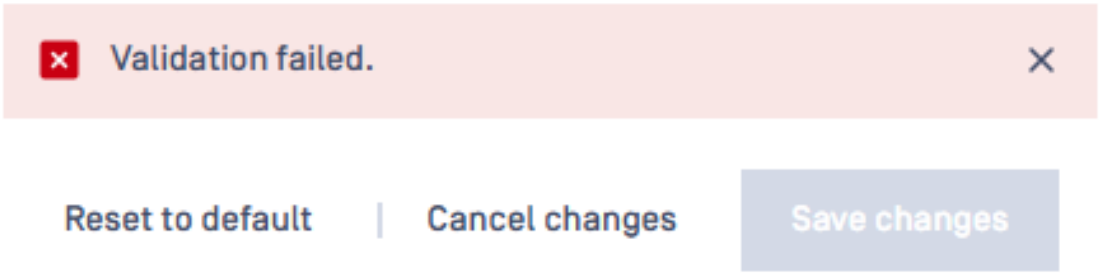
- Test SSO settings in a staging environment before production deployment.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why am I seeing error message "Validation failed" after saving Sandbox module settings?

In the Sandbox module settings, clicking "**Save changes**" triggers an error message—"**Validation failed**"—which appears in the top-right corner of the screen.



This behavior is expected in MetaDefender Core versions prior to 5.7.1. It occurs because the content within the **ChatGPT prompt** exceeds the character limit supported by the Sandbox engine.

The issue has been resolved in MetaDefender Core version 5.7.1 and later.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What Are the Default Credentials for MetaDefender Core on Azure Marketplace VM?

Check Your Version:

This article applies to OPSWAT MetaDefender Core instances deployed on Azure virtual machines using the MetaDefender Core image available in the Azure Marketplace.

Issue:

After deploying a test virtual machine in Azure using the OPSWAT MetaDefender Core image, you are unable to access the MetaDefender Core web console because you do not know the default username and password.

Resolution:

The default login credentials for the OPSWAT MetaDefender Core Azure Marketplace image are:

- **Username:** `admin`

- **Password:** `instance-id` of the deployed Azure VM

How to Retrieve the Instance ID

You can obtain the instance ID (used as the password) using either of the following methods:

Option 1: PowerShell Script in Azure Cloud Shell

Code · html

```
$GetVM = Get-AzVM
Foreach ($vm in $GetVM)
{
    $vmId = ""
    $vmId = $vm.VMid
    $vmIdList += $vmId + "rn"
}
Write-Output $vmIdList
```

Make sure to run the script in the same subscription where the MetaDefender Core VM is deployed.

Option 2: Azure Resource Explorer Portal

In **Azure Resource Explorer** (Resource Explorer), you can navigate to:

1. **Microsoft.Compute** > **virtualMachines**.
2. Select the specific **VM**.
3. Look for the `"vmId"` under the `"properties"` section.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why does the MetaDefender Core service not start?

Check Your Version:

This article applies to all MetaDefender Core version 5.0 and newer releases deployed on Windows and Linux systems.

MetaDefender Core Service Fails to Start (Windows & Linux)

If the MetaDefender Core service fails to start, it may be due to a variety of system or configuration-related issues. This article outlines common causes and troubleshooting steps for both **Windows** and **Linux** environments.

Symptoms

- MetaDefender Core does not start after installation or reboot.
- The service is visible but stuck in a "Starting" or "Stopped" state.
- Log files show initialization errors.
- The web UI is unreachable on the configured port.

Troubleshooting Steps

1. ✅ **Check Service Status**

Windows:

- Open **Services (services.msc)**.
- Locate **MetaDefender Core**.
- Right-click → **Start** (or Restart).
- If the service does not start, check the **Windows Event Viewer** and the logs located at: C:\Program Files\OPSWAT\MetaDefender Core\data\logs\Core.log

Linux:

- Run the following command:

sudo systemctl status ometascan

- Check logs using:

sudo journalctl -u ometascan

tail /var/log/ometascan/ometascan.log

2. **Configuration issues**

Registry Issues (Windows)

If the Windows registry was manually edited or configured incorrectly, the service may fail to start.

Resolution:

- Whenever changing the configuration in the registry, please make sure that the key names and the values are according to the documentation found in the link below: MetaDefender Configuration
- Any typos or wrong configuration may prevent the service start
- If edits were made, restore from backup or compare the registry under:

HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan

- Ensure values such as InstallationFolder, loglevel, and other critical keys are valid.
- Check for any typo's in the keys names or values
- Be careful for Keys dependencies, such as when syslog is configured, syslog_level is also required, or when nginx_logfile is configured, nginx_log_rotation is also required.

- The rest port is configured on a free port. If the port is in use by a different application, the service might fail as well.

Configuration (Linux)

If the MetaDefender Core configuration was manually edited or configured incorrectly, the service may fail to start. The configuration file for the server is located in /etc/ometascan/ometascan.conf

Resolution:

- Whenever changing the configuration in the ometascan.conf, please make sure that the names and the values are according to the documentation found in the link below: MetaDefender Configuration
- Any typos or wrong configuration may prevent the service to start
- If edits were made, restore from backup or compare the configuration file ometascan.conf
- Check for any typo's in the names or values
- Be careful for Keys dependencies, such as when syslog is configured, syslog_level is also required, or when nginx_logfile is configured, nginx_log_rotation is also required.
- The rest port is configured on a free port. If the port is in use by a different application, the service might fail as well.

3. **SSL Certificate or Key Issues**

MetaDefender Core may be configured to use SSL. If the certificate or private key path is invalid or the files are missing, the service will not start.

Common errors:

- File not found
- Permission denied
- Invalid certificate format

⚠ For production deployments, ensure the certificate is valid and trusted.

Solution:

For Windows:

Please navigate to the MetaDefender Core installation folder, by default: C:\Program Files\OPSWAT\MetaDefender Core and run the ometascan-reset-ssl-conf.exe tool **as Administrator** to disable the SSL configuration.

For Linux:

Please run the script /usr/sbin/ometascan-reset-ssl-conf **with root privileges**

Once done, please try starting the service again.

4. **Temporary Folder Not Accessible**

MetaDefender Core relies on a configured temporary folder for file processing. If this directory is misconfigured or inaccessible, the service will fail.

Error message:

[ERROR] <timestamp>: (common.base) Error in Service, shutdown.., error='Extension setting up failed: extension: 'ekm', **rc: '-612'**, error: '" [msgid: 8]

Default paths:

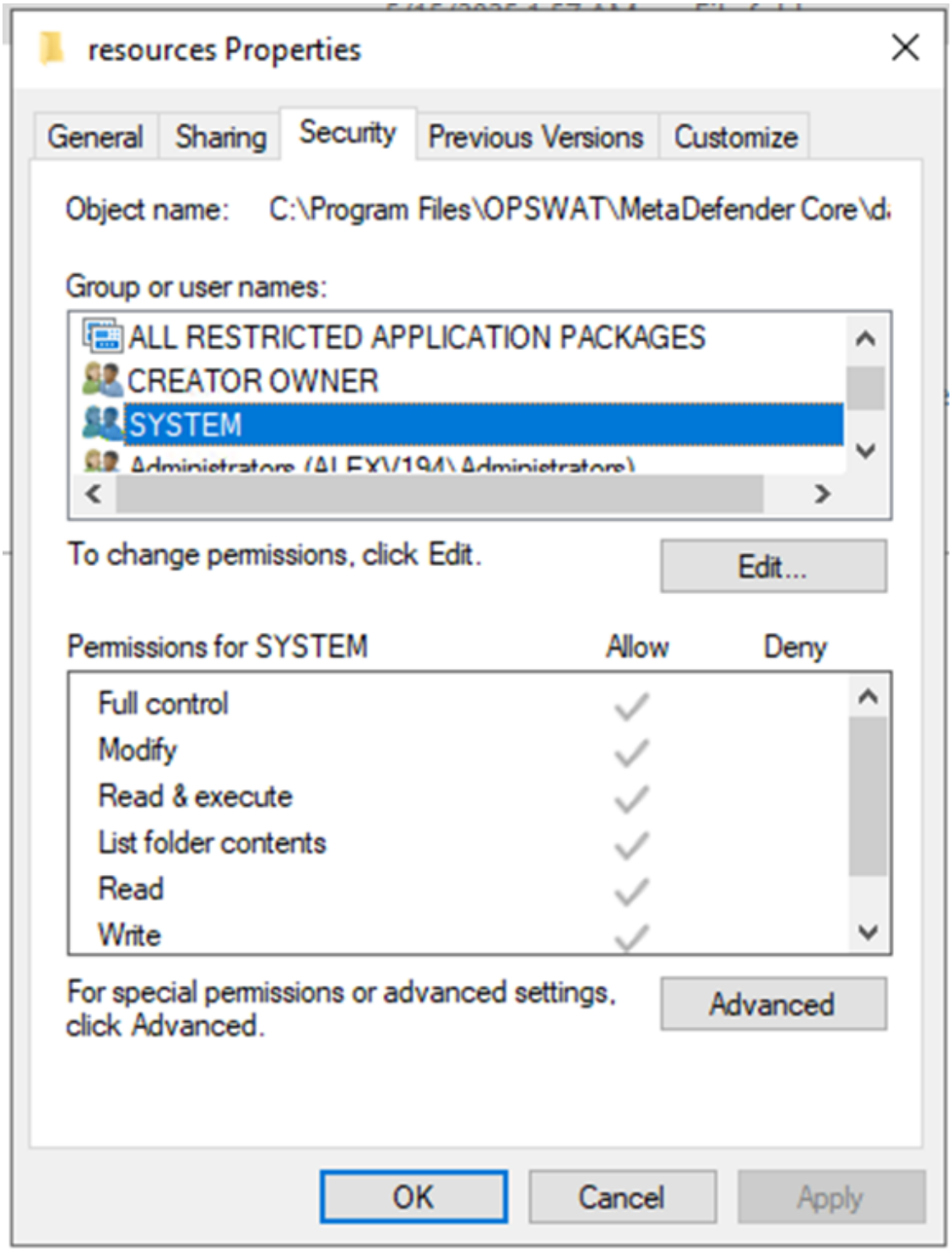
- **Windows**: C:\Program Files\OPSWAT\MetaDefender Core\data\resources\
- **Linux**: /var/tmp/ometascan/resources

Solution

Windows:

First of all, please check the mentioned at point 2 regarding the current path for the Resources folder. If no entry is configured, then the default location will be in the MetaDefender Core installation Folder\data\resources

Please check the folder permissions:



System group should have full permissions.

If the permissions are not configured as above, please configure the correct permissions and then try starting the service once again.

Linux:

First of all, please check the correct path configured at point 2 for the temporary folder. By default, this will be in the path /var/tmp/ometascan/resources and the permissions should be assigned to user metascan with the permissions mentioned below:

```
alex@alex-191:/var/tmp/ometascan/resources$ ls -lh
total 4,0K
-rw-r--r-- 1 metascan metascan 25 apr 28 17:01 oms_testfile
```

If the permissions are different, please use the commands below to set the correct permissions:

sudo chmod 755 /var/tmp/ometascan/resources

sudo chown metascan:metascan /var/tmp/ometascan/resources

5. **IPv6 is Disabled on the Linux System**

In some Linux systems, IPv6 maybe disabled or not supported, in which case startup will fail. The workaround is to force MetaDefender Core service to use IPv4.

For more details on this issue, please refer to How to Resolve MetaDefender Core Service Start-up Failures on IPv6-Disabled Linux Systems?

Contacting Support

If you are unable to resolve the issue, please collect the following information:

- Please create a support package using the steps provided in the below links:
  - How to Create Support Package With Web Console
  - How to Create Support Package With Bundle Tools?

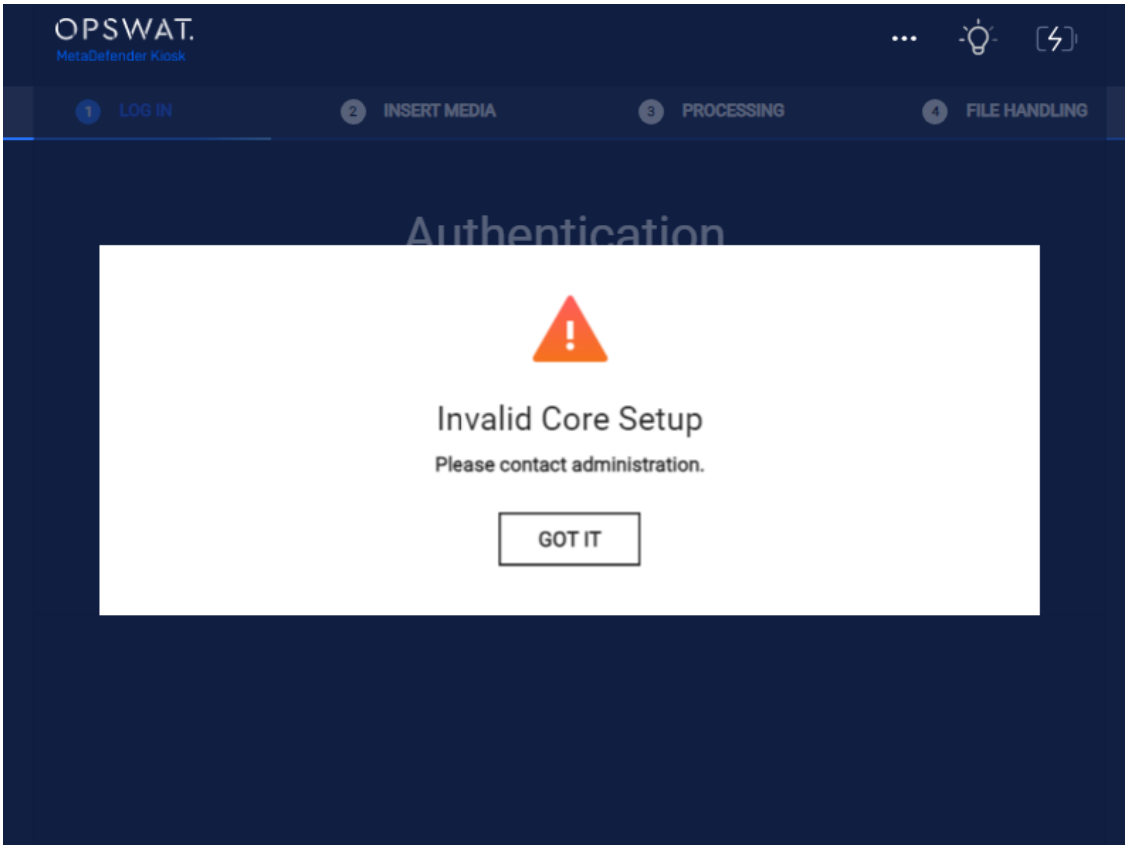
Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why does Kiosk show "Invalid Core Setup"?

Check Your Version:

This article applies to all MetaDefender Kiosk releases deployed on Windows.

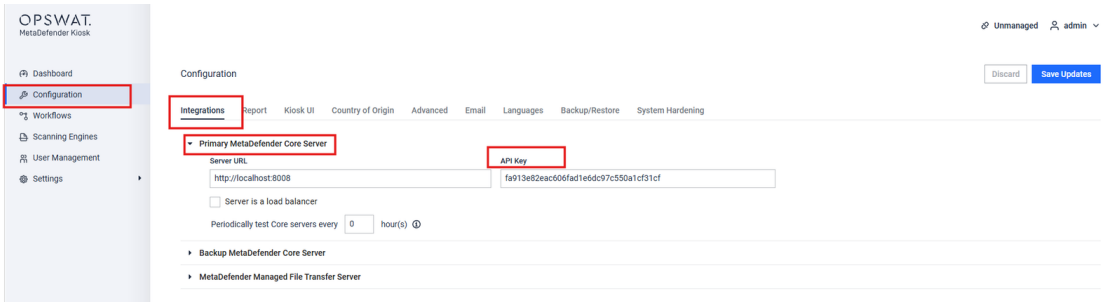


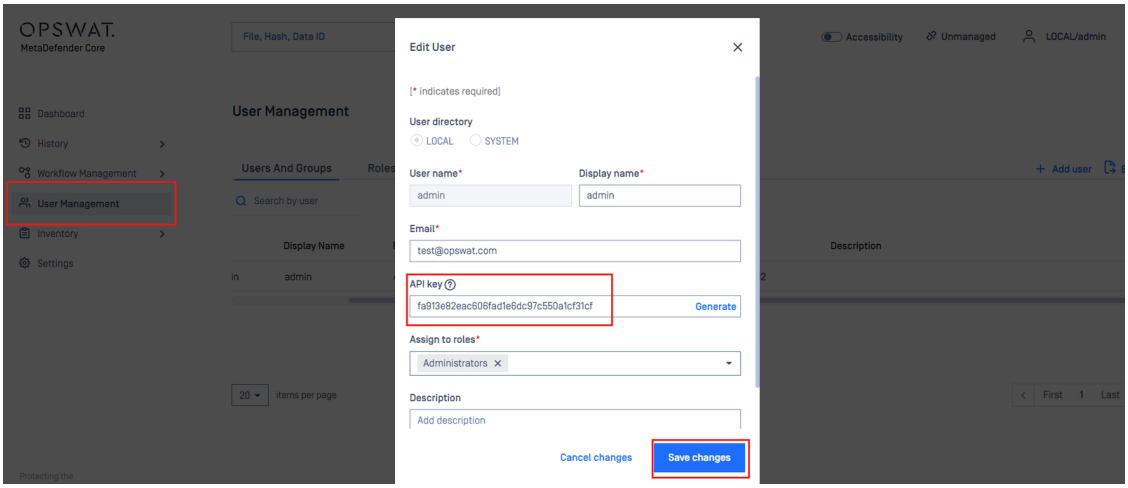
Introduction

There are many different reasons why you may view the “Invalid Core Setup”, please review the below:

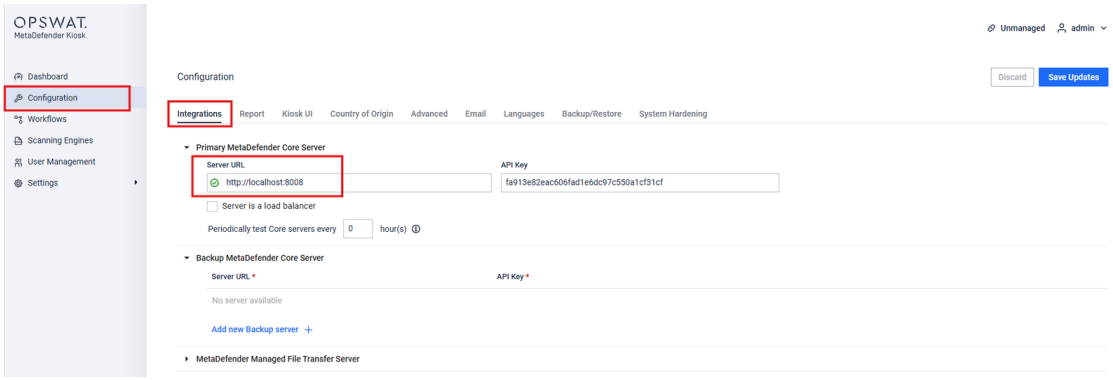
1. **Incorrect API key**

Navigate to Kiosk > Configuration > Integrations > Primary MetaDefender Core Server. Compare the API key with the admin local user on MetaDefender Core





2. **Incorrect MetaDefender Core "Server URL"**



*

Open the MetaDefender Core Website in the Kiosk machines Browser and copy the URL and paste it here, ending with the port

*

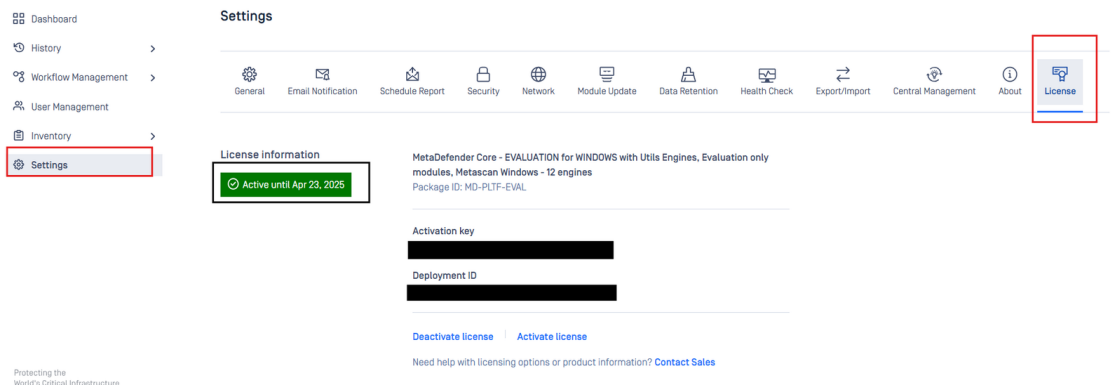
If MetaDefender Core has a certificate enabled for HTTPS, please use the https://<FQDN>:8008

Example: https://certificateSAN:8008

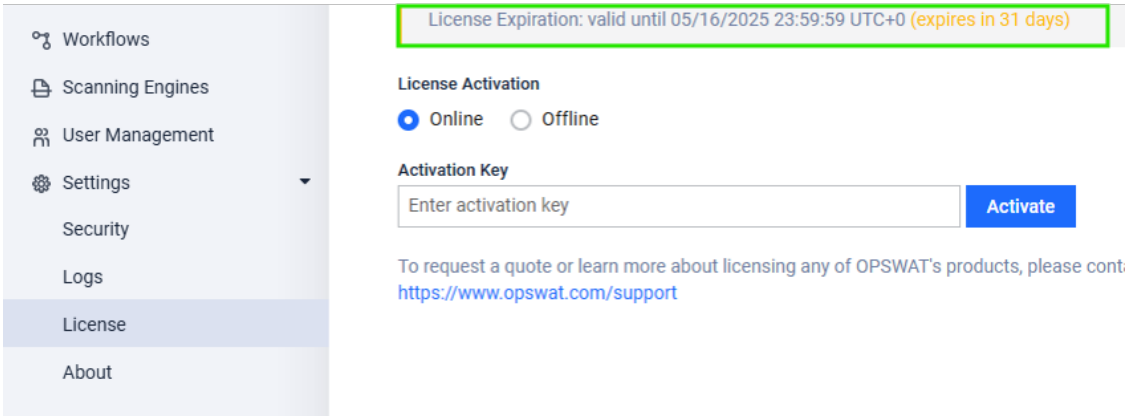
3. **Unlicensed MetaDefender Core or MetaDefender Kiosk**

Confirm that both the MetaDefender Core and MetaDefender Kiosk are properly licensed.

MetaDefender Core:



MetaDefender Kiosk:



How to activate Kiosk offline: Managing License Information

4. **Network connectivity issue between MetaDefender Core and MetaDefender Kiosk**

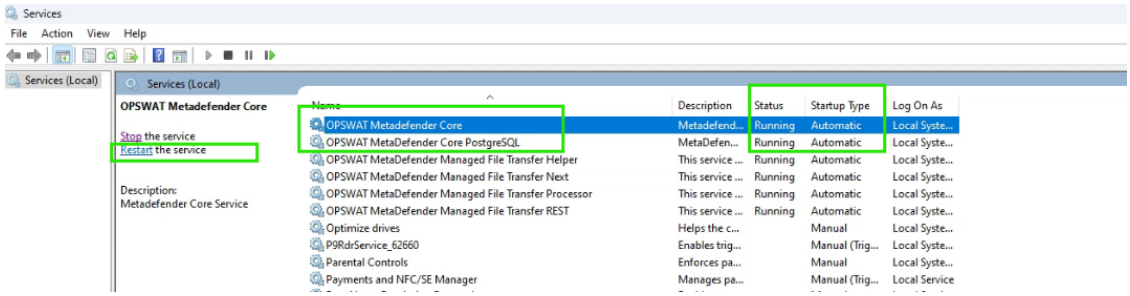
Check for network issues that might prevent communication between the Kiosk and Core. Consult with your networking team if necessary. Example: Firewall, proxy, low bandwidth

5. **Expired Certificate** If the certificate on the MetaDefender Core machine is expired, update and install it correctly on the Kiosk's Windows OS inside the Local Machines Trusted Root Certificate Store.

> *
>
> Open the MetaDefender Core website from the Kiosk machine after performing the change

6. **MetaDefender Core service is not running**

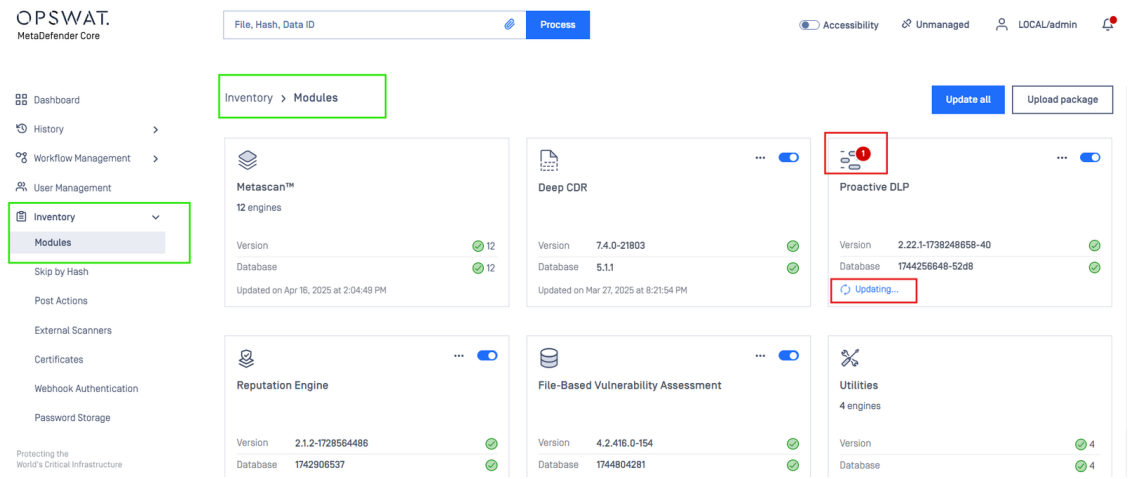
Verify that the MD Core services are running properly. Restart them if necessary.



7. **MetaDefender Core Modules inactive/permanently failed**

If required engines are still updating, please wait until they are complete.

If they are missing dependencies, check our "Third Party Dependencies" to resolve



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why is my file rejected even though it appears to be under the configured size limit?

Check Your Version:

This article applies to MetaDefender Core v4.x and 5.x.

You may encounter a situation where a file upload is rejected by MDCore even though the file appears to be within the allowed size limit. This commonly stems from differences in how operating systems and applications report file sizes.

Root Cause

MDCore enforces file size limits using **decimal units** (base-10), which is standard for most web applications and cloud services. However, many operating systems—particularly Windows—use **binary units** (base-2) when reporting file sizes. This discrepancy can lead to misunderstandings.

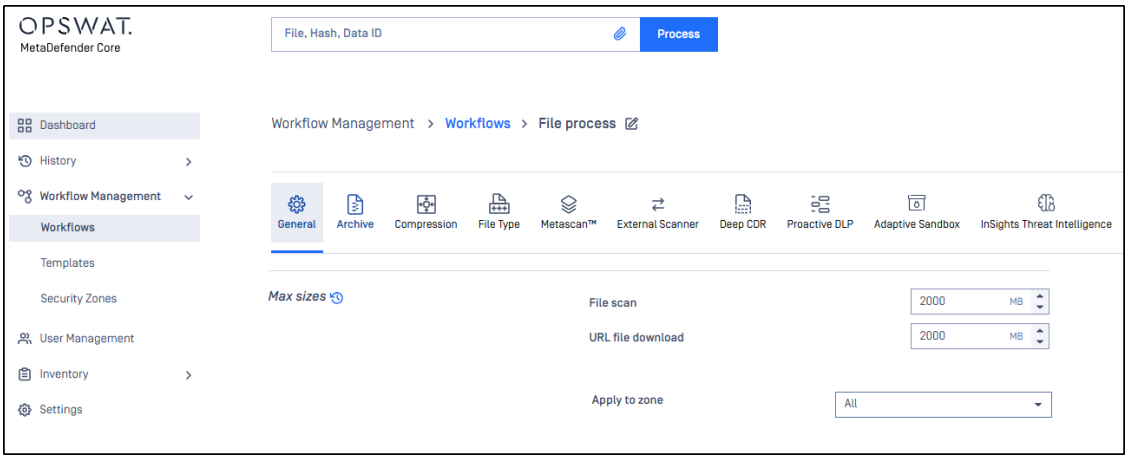
Comparison of Unit Systems

Unit System	1 MB in Bytes	Used By
Decimal	1,000,000 bytes	HDD/SSD manufacturers, cloud storage services, web browsers, some Linux distros
Binary	1,048,576 bytes	Windows File Explorer, older macOS, most programming languages

Example

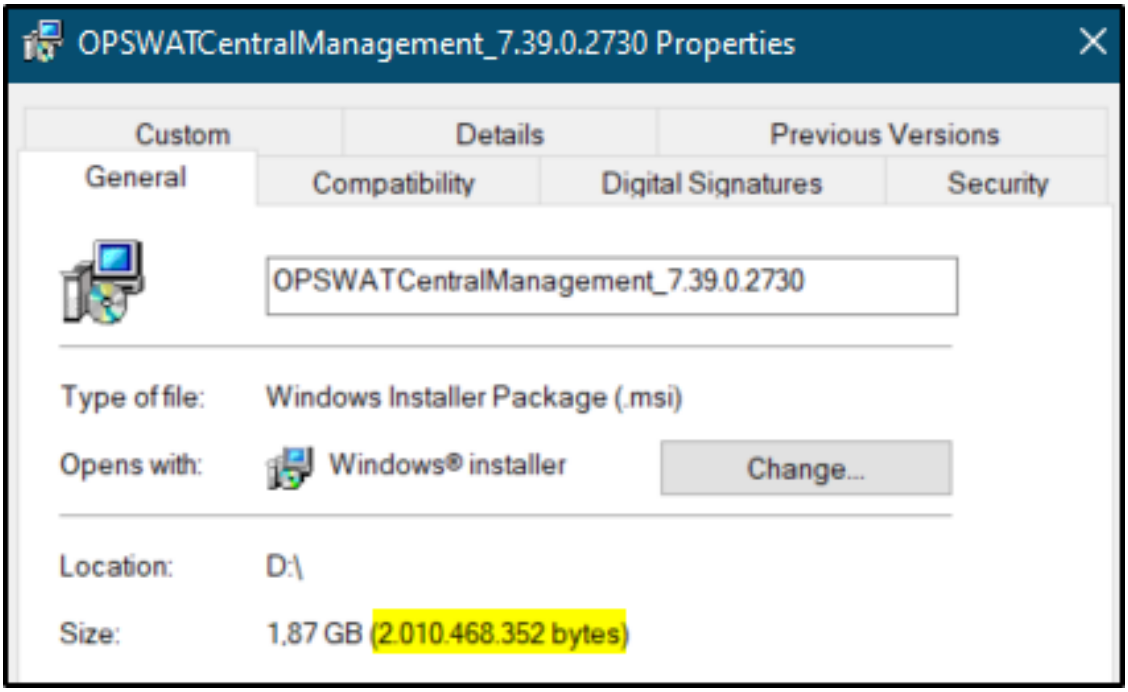
Suppose your MDCore upload limit is configured as **2000 MB (decimal)**:

- **2000 MB (decimal) = 2,000,000,000 bytes**



If a file appears in Windows Explorer as **1.87 GB**, you might assume it's under the limit.

However, when measured in bytes, the actual size is **2,010,468,352 bytes**



This exceeds the 2,000,000,000-byte limit set in MDCore, causing the file to be rejected.

Resolution

To prevent this issue:

- Always verify the **actual byte size** of files before uploading.
- If using Windows, right-click the file **Properties** Check the **Size in bytes**, not just the "Size" shown in MB/GB.
- Consider increasing the configured upload limit slightly to account for rounding inconsistencies, if appropriate.

Support:

If you still have questions regarding the details in this KB, please create a support case or chat with one of our support engineers.

How Can I Solve a Database Indexing Issue?

While troubleshooting MetaDefender Core, you may encounter the following error in the logs:

Failed to delete records, reason='ERROR: index "request_1_pkey" contains unexpected zero page at block 2760'.

invoke failed, out='QMap(("error", QVariant(QString, "ERROR: index "request_1_pkey" contains unexpected zero page at block 2760"

This error indicates an issue with the **PostgreSQL index** on the request_1_pkey table.

Root Cause

The error typically occurs when an index in the **PostgreSQL** database becomes corrupted or unreadable. In this case, the request_1_pkey index is damaged, which prevents MetaDefender Core from performing certain operations.

Solution

To resolve the issue, you need to **rebuild (reindex)** the corrupted index using a PostgreSQL command.

Step-by-Step Instructions

1. Open Command Prompt

2. Navigate to the PostgreSQL Binaries Folder

cd "C:\Program Files\OPSWAT\MetaDefender Core\postgres\bin"

Adjust the path if MD Core is installed in a different location.

3. Connect to the MetaDefender Core Database

Run the following command to access PostgreSQL:

psql.exe -U <your_postgres_username> -d metadefender_core_8gyrxm

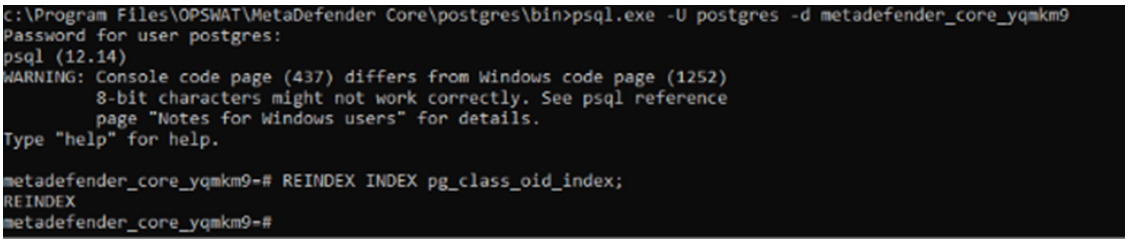
- Replace <your_postgres_username> with the correct PostgreSQL username used during installation.
- If prompted, enter the PostgreSQL password.

4. Run the Reindex Command

Once connected to the database prompt enter:

Reindex index scan.request1 pkey;

Please see the following example below:



Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

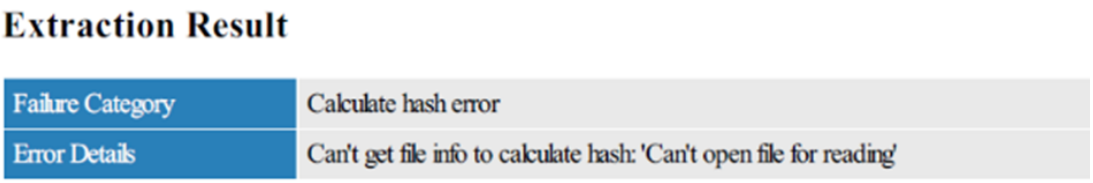
Why Does MetaDefender Core Show a 'Calculate hash error' When Scanning Malicious Files?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

Issue

When testing the upload of a malicious file to MetaDefender Core, the system unexpectedly allows the file and displays the following error during scanning:



Extraction Result

Failure Category	Calculate hash error
Error Details	Can't get file info to calculate hash: 'Can't open file for reading'

Archive Extraction: Extraction error Extraction Result Failure Category: Calculate hash error Error Details: Can't get fileinfo to calculate hash: 'Can't open file for reading'

Root Cause

The error occurs because local antivirus software on the system is detecting and handling the malware file before MetaDefender Core can access it. As a result, MetaDefender Core is unable to read the file and generate the necessary hash for scanning.

This typically happens when Real-Time Protection (RTP) is enabled on the local antivirus, causing the file to be quarantined or blocked immediately after upload.

Solution

- **Verify file integrity**
  - Confirm that the file you submitted is complete and accessible.
  - Re-upload the file if needed and check if the error re-occurs.
- **Check anti-virus / real-time protection interference**
  - If you are using any anti-virus or endpoint protection on the same machine as MetaDefender Core, check whether the real-time protection has locked or removed the file before the hash calculation.
  - Consider temporarily disabling real-time protection for testing, or exclude the upload/temp folders of MetaDefender Core from the real-time scan.
  - Note: Do **not** exclude `engineprocess.exe` if Windows Defender (or any AV) is used both as the *local AV* and as an *AV-engine inside MetaDefender Core*. (See related KB for exclusion guidelines.)
- **Check folder permissions and storage capacity**
  - Ensure the service account running MetaDefender Core has read/write access to the upload folder and temp directory.
  - Verify there is sufficient free disk space in the partition that holds the upload/temp folder.
  - If the file is large or part of an archive, ensure it can be extracted and processed.
- **Review Core logs for additional details**
  - On Windows: check `C:\Program Files\OPSWAT\MetaDefender Core\logs\*` for messages around the

"Calculate hash error".

- On Linux: check `/var/log/metadefender-core/` (or equivalent) for more context.

- **Retest the submission**
  - After applying the above checks/remediations, resubmit the file for scanning.
  - If the error disappears, you have resolved the root cause.
  - If the error persists, collect the upload file, the logs (with debug if possible) and open a support case with OPSWAT.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why MD Core is Not Reachable?

Issue Summary: MetaDefender Core may become unreachable due to nginx failing to start properly after a restart or shutdown.

Error Logs: In nginx_error.log, you may see the following entry:

[error] 10904#14248: OpenEvent("Global\ngx_reopen_3536") failed (2: The system cannot find the file specified)

Root Cause: This issue occurs when nginx fails to clean up its temporary files during a shutdown or restart. On subsequent startups, these leftover files can interfere with nginx's ability to launch, preventing MD Core from becoming reachable.

Remediation Steps:

1. **Stop the Service:**

- Open Services (services.msc) and stop **OPSWAT MetaDefender Core**.

2. **Clean Up Temporary Files:**

- Navigate to: C:\Windows\Temp\ometascan
- Delete all files inside this directory. Files you may find include:
- ometascan-nginx.pid
- nginx.conf
- Other temporary files

3. **Restart the Service:**

- Start the **OPSWAT MetaDefender Core** service again.

Additional Notes:

- Ensure you have administrator privileges to delete the files.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why am I getting "invalid structure" or "archive corrupted" errors when extracting archives in Core?

Check Your Version:

This article is applied to Core releases deployed on Windows and Linux systems.

Issue: Core encounters errors such as "invalid structure" or "archive corrupted" during archive extraction.

Resolution: The issue might stem from the archive file itself being corrupted, rather than a problem with Core's extraction engine. To verify this, follow the steps below.

Steps to Diagnose:

- Test the archive using the standalone 7Zip application on a Windows system.
- Attempt to extract the archive using 7Zip.
- If 7Zip also fails to extract the file and reports similar errors, the archive is likely corrupted.

Example of 7Zip extraction issue message:



Conclusion: If 7Zip reports errors, the problem lies within the archive file and not with Core. You will need to obtain a valid, uncorrupted archive.

Support:

If 7Zip reports errors, the problem lies within the archive file and not with Core. You will need to obtain a valid, uncorrupted archive.

Why are my script files being detected as ASCII TXT?

Check Your Version:

This article applies to all MetaDefender Kiosk V4 releases deployed on Windows systems.

Overview:

The FileType engine supports detection of common script file types such as BASH, JavaScript (JS), PowerShell (PS1), and Python (PY). However, due to the textual nature of these files, they may occasionally be identified as generic ASCII TXT files.

Why Misclassification Occurs: Script files are fundamentally plain text and often lack unique signatures or structural characteristics that distinguish them from general text files. This makes accurate identification inherently difficult using conventional detection methods.

Enhancing Detection with Machine Learning: To improve detection accuracy for script files:

1. **Enable the Machine Learning module** in the FileType engine settings.
2. **Update your workflow** according to the steps outlined here: Text detection with AI

Limitations and Expectations: Even with advanced machine learning models, perfect detection is not guaranteed. Due to the flexible and diverse nature of script files:

- Some script files may still be misclassified.
- False positives and false negatives are possible and expected.

Continuous Improvement and User Contribution: Our team actively refines the AI models to improve accuracy. If you encounter misclassifications:

- Collect the misidentified samples.
- Open a support case with our team and attach the collected samples.

This feedback helps us fine-tune the models for better alignment with real-world use cases.

Conclusion: While detection of script files can be challenging, enabling AI enhancements and collaborating with support can significantly improve accuracy over time.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

How to Resolve SSL/TLS Errors with MetaDefender Core Webhook Callbacks?

Check Your Version:

This article applies to MetaDefender Core 5 releases deployed on Windows, Linux.

Issue:

When the webhook callback functionality is invoked, it may fail due to SSL/TLS errors. The core.log file will contain an error similar to the following:

(core.webhook) SSL/TLS failed, issue='The issuer certificate of a locally looked up certificate could not be found, No certificates could be verified'

Resolution:

Windows

1. Obtain the webhook server's SSL certificate chain (root CA, intermediate CA, and server certificate).
2. Open the Microsoft Management Console (mmc.exe).
3. Add the Certificates snap-in for the local computer.
4. Navigate to `Trusted Root Certification Authorities > Certificates`.
5. Import the root, intermediate, and server certificates into the `Trusted Root Certification Authorities` store.
6. Restart the OPSWAT MetaDefender Core service:

- OPSWAT MetaDefender Core

After the service restart, the trusted root certificates will be exported to a certificate located in the MetaDefender Core installation folder Data folder (by default `C:\Program Files\OPSWAT\MetaDefender Core\data`).

Linux

By default, MetaDefender Core loads root CAs from the following locations:

- `/etc/ssl/certs/`
- `/usr/share/ssl/`
- `/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem`

Steps to import the certificates:

Debian/Ubuntu

1. `sudo cp -f <cert_file> /usr/local/share/ca-certificates/`
2. `sudo update-ca-certificates`
3. `sudo systemctl restart ometascan`

CentOS/RHEL

1. `sudo cp -f <cert_file> /etc/pki/ca-trust/source/anchors/`

2. `sudo update-ca-trust enable`
3. `sudo update-ca-trust extract`
4. `sudo systemctl restart ometascan`

Steps to Reproduce:

Trigger a webhook callback to confirm that the SSL/TLS error occurs. Check the core.log for the SSL/TLS error.

Prevention:

Ensure the webhook server's SSL certificate chain is trusted by the operating system by installing the intermediate or root certificate authorities in the local certificate store.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

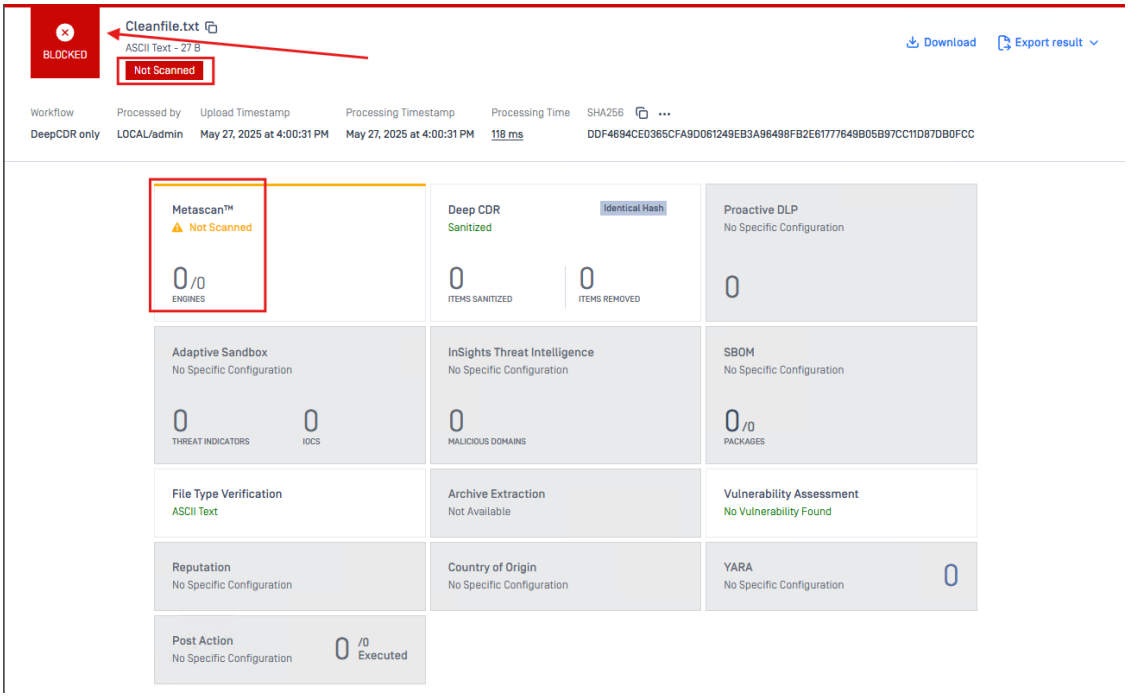
Why does MetaDefender Core show "Blocked" with a scan result of "Not Scanned" when no malware is detected?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows/Linux systems.

Overview

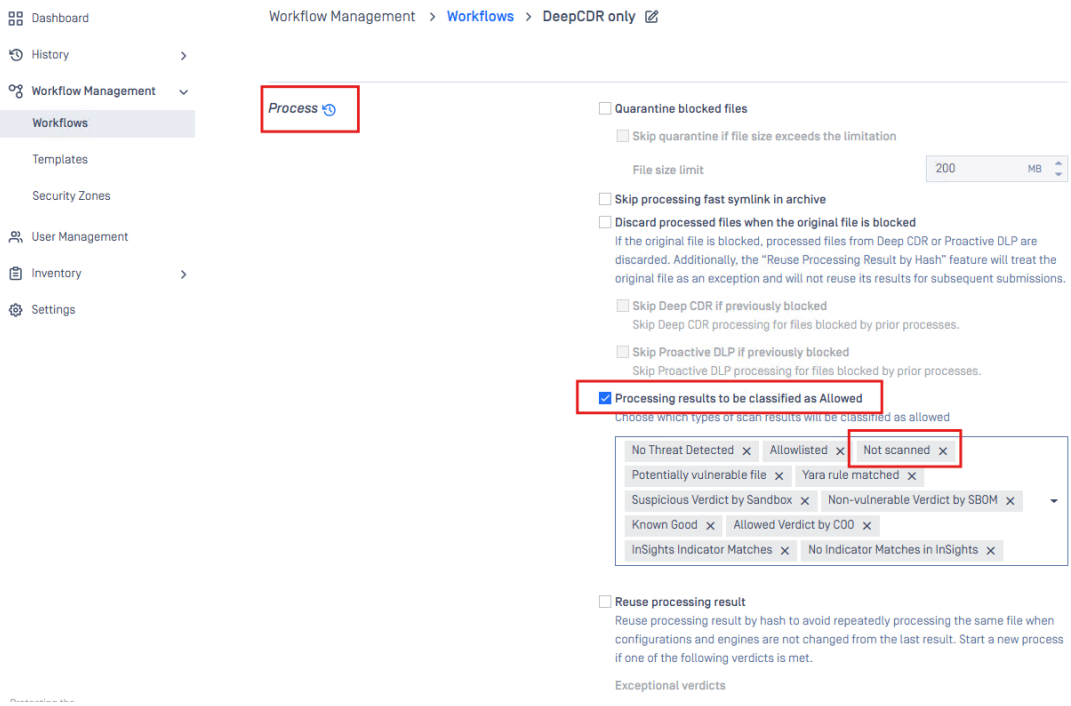
This article explains the reason why files scanned through MetaDefender Core receive a "**Blocked**" status with a scan result of "**Not Scanned**" even when no malware is detected and suggests a solution.



Reason

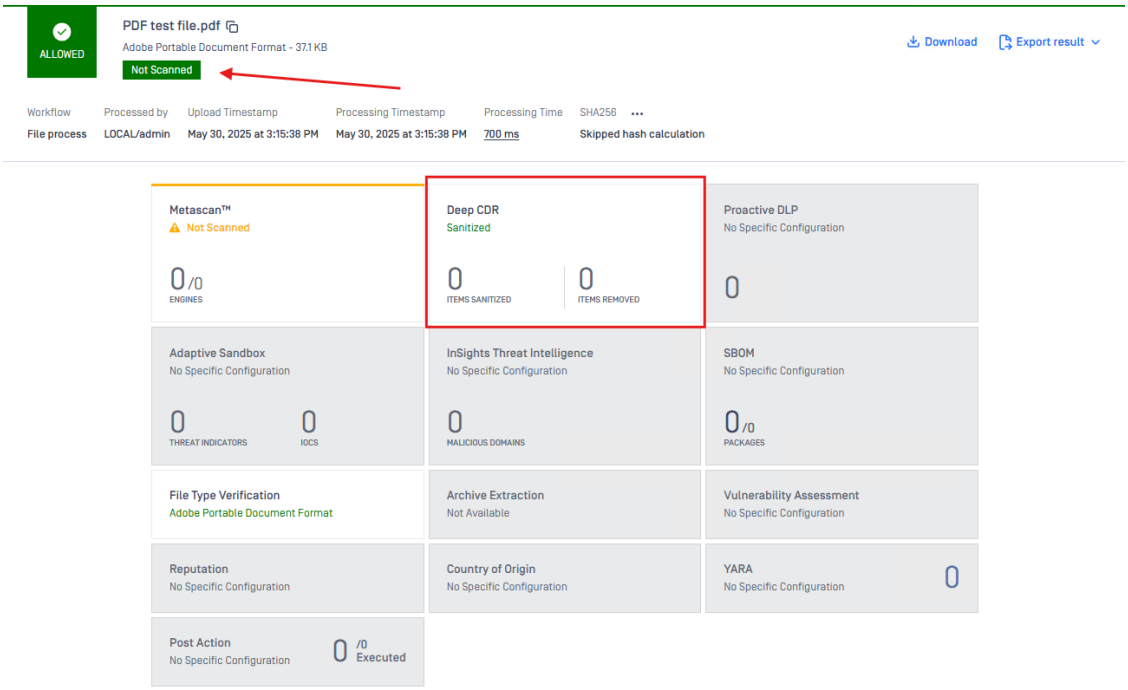
The issue occurred because no antivirus engine was available to scan the file. As a result, the scanning session was marked as “**Not Scanned**” This is expected behavior when no AV engine participates in the scanning process.

Solution

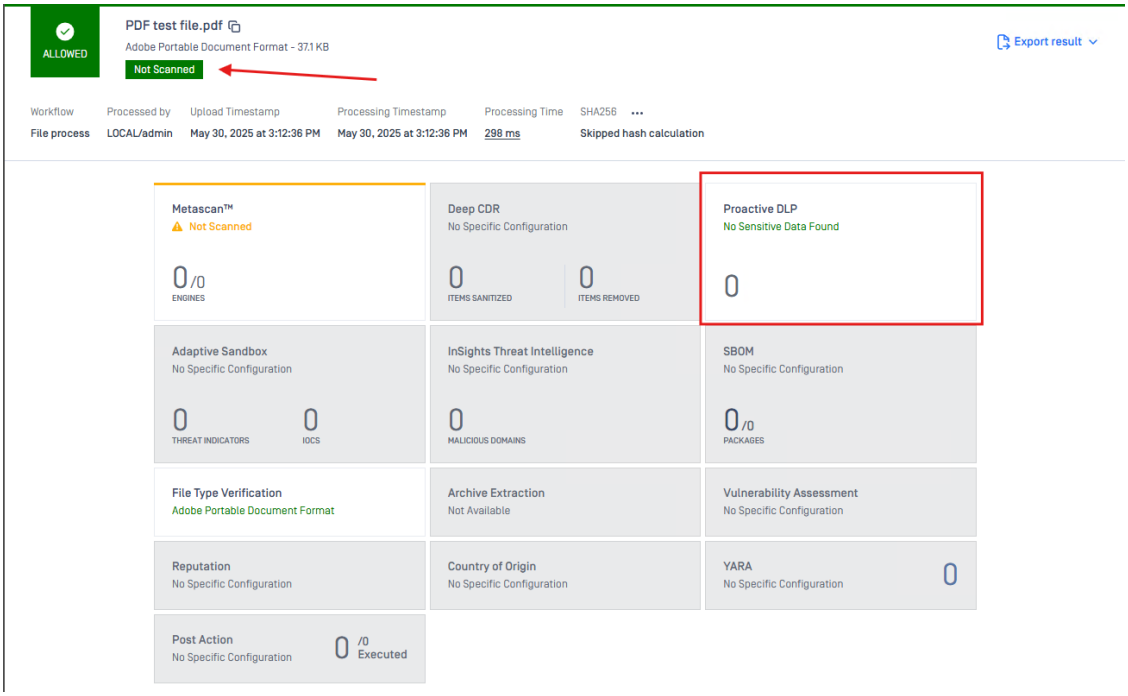


After adding the **“Not Scanned”** scan result option to the enabled **Processing result to be classified as Allowed** setting under the **Process** section in the **General** tab of the workflow, files with a “Not Scanned” scan result will now be classified as *Allowed* instead of *Blocked*.

Option 1: Example CDR



Option 2: Example DLP



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I remediate MetaDefender Core using too much resource while sanitizing TIFF/TIF image file?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue

When processing TIFF/TIF image files, you may notice some behaviors such as:

- Deep CDR sanitization of the TIFF/TIF image takes a long time.
- High resource usage (CPU, memory, disk space).
- Other scan requests getting timed out while waiting in queue.

Cause

TIFF/TIF image files can have many pages (multi-page TIFF files). When the Deep CDR engine processes them, it needs to extract and handle each page. While the original size of the image may be small, the size of the decompressed image can be quite big, which contributes to the increased resource usage and sanitization time.

For example:

- A TIFF file with size of 2.35 MB, containing 30 pages.
- Each page's dimension is 2496 x 3507, meaning that the total decompressed (raw) size could be about 250 MB.

Remediation

You can set some limitations to TIFF/TIF image processing, to avoid using up all the system's resources and slowing down other processing requests.

In Workflow > Deep CDR > File type handling > Image:

- **TIFF Size Limit (MB):** This limits the raw decompressed size of TIFF image that Deep CDR can handle (Deep CDR Image Limit Configurations).
- **Restrict processing of big TIFF files in single thread:** Any TIFF file with a decompressed image size greater than or equal to 128MB will be processed in single-thread mode. Files with a decompressed image size of 128MB or less will continue to be processed in multi-thread mode.
- **Deep Sanitization:** By turning this off, CDR will only validate file structure and check for any tailing data after the end of the file and remove that part. This is faster, but the tradeoff is less secure compared to the having this option on, which is the default.

If enough system memory is available, you can increase the **Image Memory Limit (MB)** in Deep CDR's engine settings to get more of the processing done on memory rather than on disk.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Why are PDF documents being flagged as false positives by VirusBlokAda?

Issue:

PDF files, including official letters or documents with standard formatting, are sometimes flagged as suspicious by the VirusBlokAda engine

Cause:

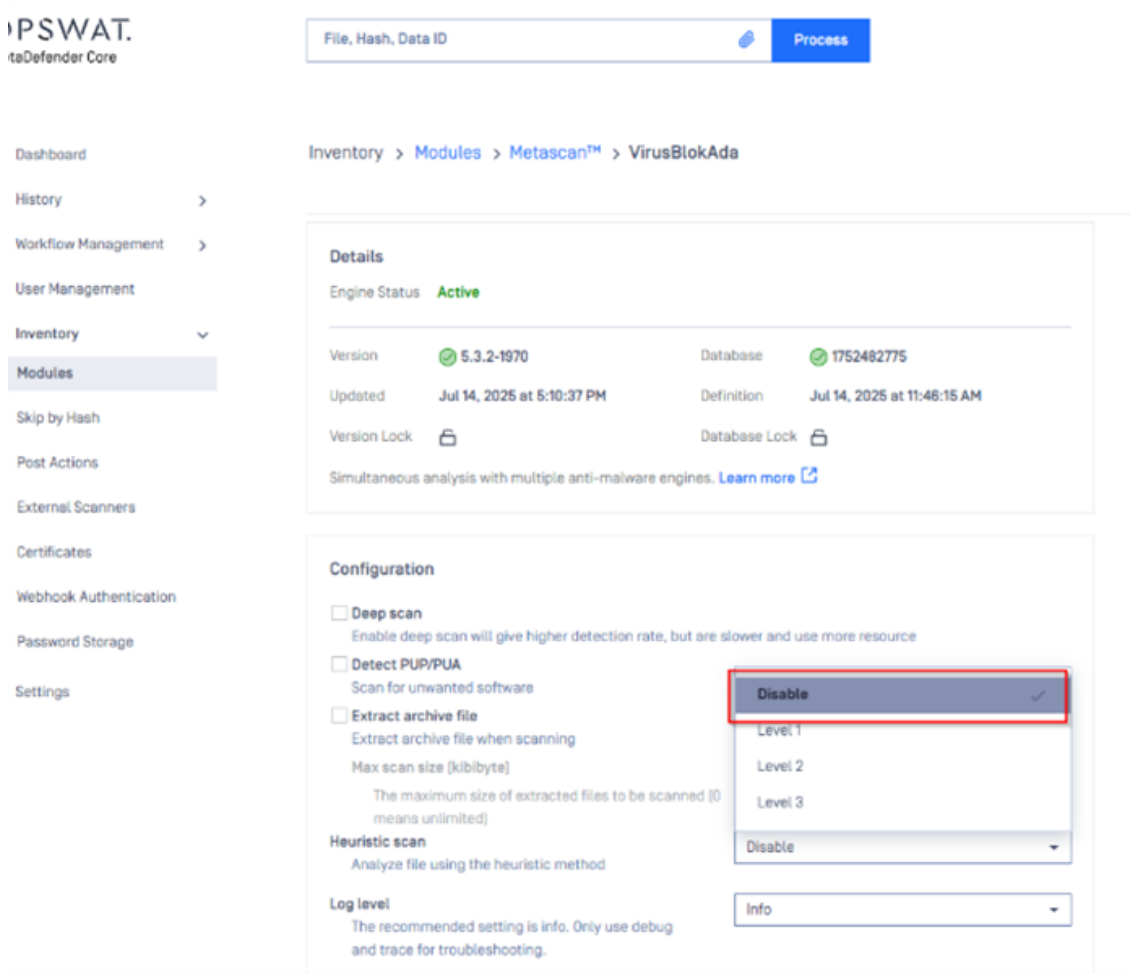
The issue is often related to the **Heuristic Scan** feature, which can mistakenly classify normal PDF structures as threats. **Resolution / Workaround:**

1. Verify that the flagged PDF is from a trusted source.

2. If confirmed safe, adjust the VirusBlokAda settings:

- Disable **Heuristic Scan**, or
- Set **Heuristic Scan Level** to **Level 1** to reduce unwanted detections.

Note: Disabling or lowering heuristic scanning may reduce detection sensitivity, so only adjust this setting if false positives are persistent.



Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why did my scanned image fail with "exceeded file size" error?

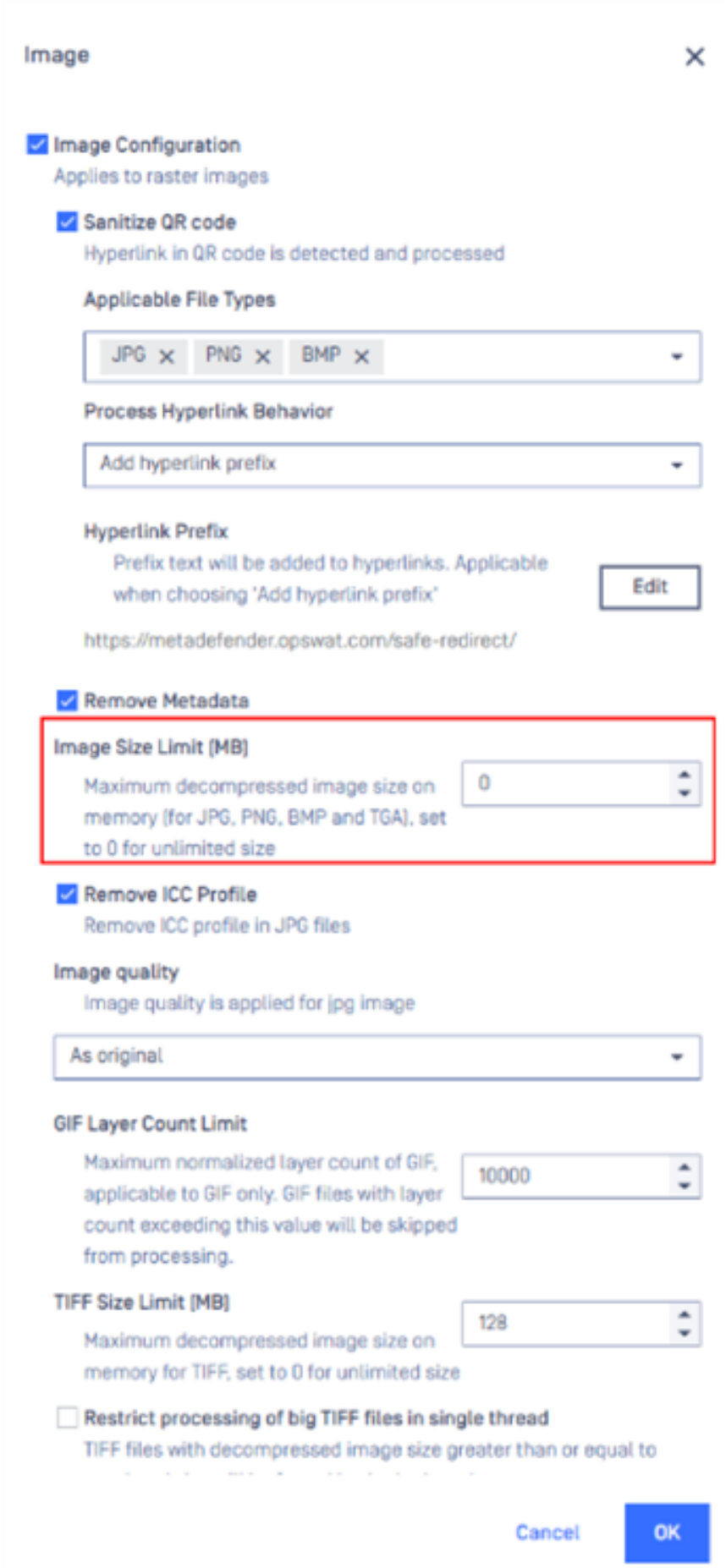
Issue: When scanning a large JPG image, the following error appears in the logs:

sanitized: FAILED

result: "_{"output_file_path":"","process_details":{"details":"Decompressed image size would be 1314979770 bytes, exceeding size limit of 268435456 bytes.","failure_category":"exceeded file size_"}

Cause: The decompressed image size (≈1.3 GB) exceeds the configured Image Size Limit (256 MB by default).

Resolution: Update the Image Size Limit configuration to allow processing of larger images.


Image
Image Configuration
Applies to raster images
Sanitize QR code
Hyperlink in QR code is detected and processed
Applicable File Types
JPG
PNG
BMP
Process Hyperlink Behavior
Add hyperlink prefix
Hyperlink Prefix
Prefix text will be added to hyperlinks. Applicable when choosing 'Add hyperlink prefix'
Edit
https://metadefender.opswat.com/safe-redirect/
Remove Metadata
Image Size Limit (MB)
Maximum decompressed image size on memory (for JPG, PNG, BMP and TGA), set to 0 for unlimited size
0
Remove ICC Profile
Remove ICC profile in JPG files
Image quality
Image quality is applied for jpg image
As original
GIF Layer Count Limit
Maximum normalized layer count of GIF, applicable to GIF only. GIF files with layer count exceeding this value will be skipped from processing.
10000
TIFF Size Limit (MB)
Maximum decompressed image size on memory for TIFF, set to 0 for unlimited size
128
Restrict processing of big TIFF files in single thread
TIFF files with decompressed image size greater than or equal to
Cancel
OK

Notes:

- Always set the limit slightly higher than the failing image size to avoid repeat issues.
- Increasing this value may require additional memory and disk resources—adjust system capacity as needed.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why isn't an attached file inside a PDF being converted to a different format during Deep CDR sanitization?

When sanitizing a PDF that contains an attached file (for example, an XLS attachment), you may notice that the attached file is sanitized but not converted into a different file type (e.g., XLS XLSX), even if your policy specifies a conversion.

This is expected behavior due to how recursive sanitization works in Deep CDR:

- Attached or embedded files are treated as components of the parent PDF, rather than standalone inputs.
- Deep CDR is designed to preserve the original file structure and format of embedded content when sanitizing.
- As a result, the attached file is sanitized but is not subject to type conversion rules (e.g., XLS XLSX, DOC DOCX, etc.).

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why is MetaDefender Core Failing to Connect via Proxy?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue:

MetaDefender Core requires a correctly configured proxy to establish internet connectivity for:

- Downloading engine packages
- Licensing the product
- Other

If the proxy configuration is incorrect, MetaDefender Core may fail to connect to the internet, even if a proxy server

has been added.

MetaDefender Core uses its own proxy settings and does not rely on the operating system configuration.

The proxy entry must be defined using only IP or domain name and port.

Proxy URLs with prefixes (e.g., http:// or https://) are not supported.

When the proxy is incorrectly configured, you may see warnings in the log file:

- Log file location: `<Core installation folder>\data\logs\Core.log`
- Example warning message: `[WARNING] date / time (common.communication) Error while downloading file, url='', http_status='0', error='ProxyNotFoundError', errorDetails='Unknown error' [msgid: 485]`

Resolution:

Open MetaDefender Core UI → Settings → Network.

Review the Proxy Server configuration.

✅ Correct format:

- `proxy.company.com port 8080`
- `10.10.10.10 port 8080`

❌ Incorrect format:

- http://proxy.company.com `port 8080`
- https://10.10.10.10 `port 8080`

If your proxy entry includes `http://` or `https://`, remove the prefix.

Save the settings and retry the proxy connection.

Prevention:

If MetaDefender Core cannot reach the internet via proxy:

- Ensure the proxy server is entered as IP or domain + port only.
- Avoid using http:// or https:// prefixes.
- Check the MetaDefenderCore.log for related warnings (ProxyNotFoundError).

Correcting the format should resolve proxy connectivity issues.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why is unused data removed in Deep CDR resulting in empty

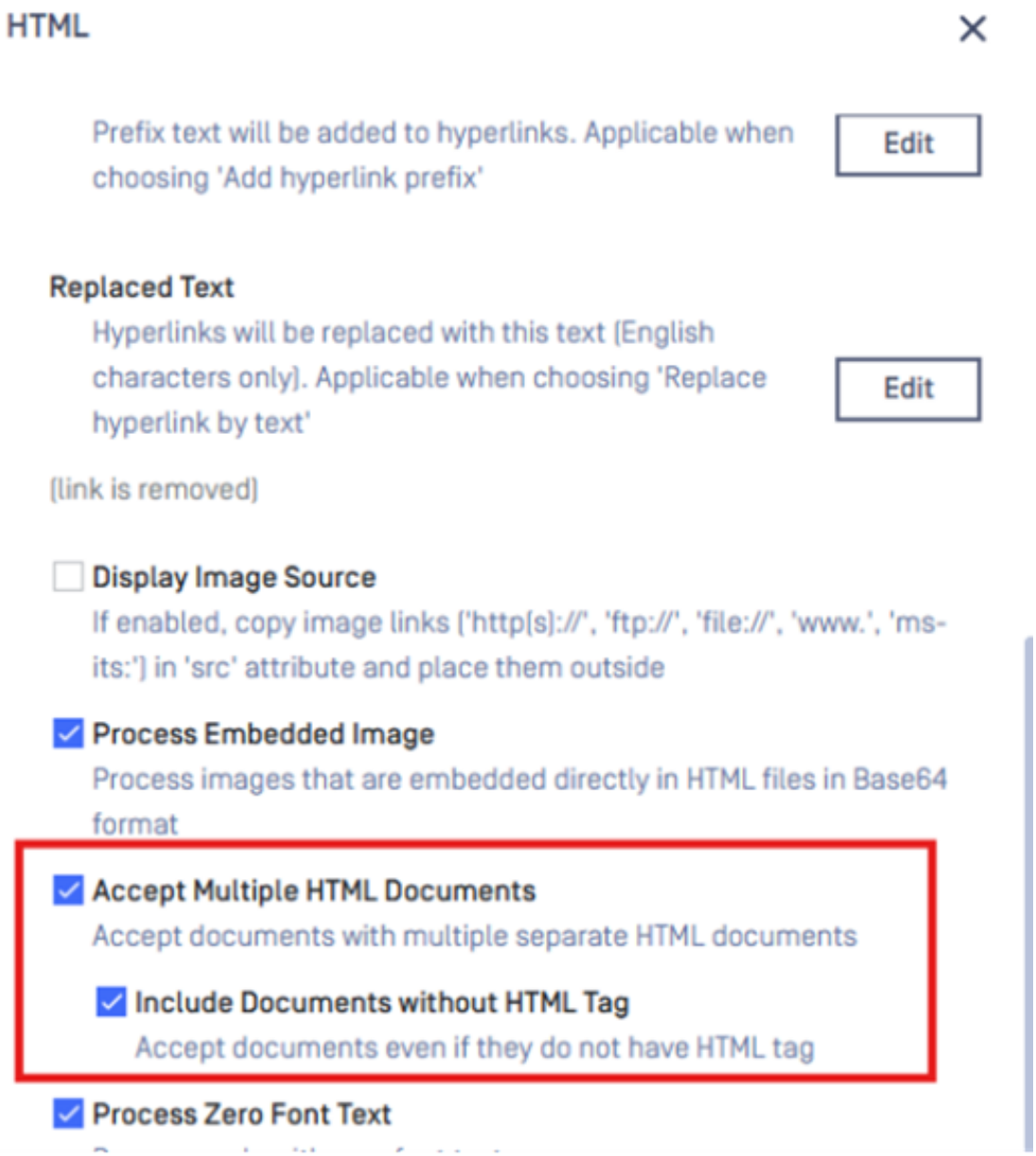
emails being received?

Issue description:

The end user reports receiving empty emails from a specific client. Analysis of the scan results indicates that the associated .HTML file contained 'unused data,' which was sanitized during processing.

Solution:

To ensure proper sanitization in the Deep CDR engine, enable the Include Documents Without HTML Tag option. This setting is required for files with invalid or missing HTML structures, allowing the engine to process them correctly.



We are already aware of this scenario and have provided this configuration to help users bypass the issue.

Support:

If you encounter any issues or require further assistance, please contact OPSWAT support via My OPSWAT at: https://my.opswat.com/home

Our support team will be able to provide you with the necessary guidance and resolution for any license-related concerns.

Why are MetaDefender Core and PostgreSQL services shutting down unexpectedly?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows.

Issue: MetaDefender Core and PostgreSQL services shut down unexpectedly due to the deletion or inaccessibility of the temporary directory `C:\Windows\temp\ometascan\`. This is likely caused by a scheduled system reboot or automated cleanup task that removed or cleared the temporary folder.

Root Cause: The root cause was the deletion or inaccessibility of the temporary directory `C:\Windows\Temp\ometascan\` used by nginx. Investigation and identification of the scheduled task/script that triggered the shutdown and the missing temp folder issue, confirmed by log analysis and event viewer data, was key to the solution.

Resolution: Ensure that the temporary directory `C:\Windows\Temp\ometascan\` is not deleted or cleared by any scheduled tasks, cleanup tools, antivirus software, or group policies.

Prevention:

- Ensure that the temporary directory `C:\Windows\Temp\ometascan\` is not deleted or cleared by any scheduled tasks, cleanup tools, antivirus software, or group policies.
- Review and modify scheduled tasks or scripts running to prevent them from removing or interfering with the MetaDefender Core temporary folder.
- Exclude the MetaDefender Core installation path (`C:\Program Files\OPSWAT`) and temporary directory (`C:\Windows\Temp`) from antivirus and security software scans or cleanup routines.
- Monitor and verify that the MetaDefender Core and PostgreSQL services start normally after system reboots or scheduled restarts.
- Collect and review Windows Event Viewer logs related to Task Scheduler to identify any tasks that might affect the service.
- Follow OPSWAT's recommended system configuration guidelines and antivirus exclusion instructions available at: Recommended System Configuration

Workarounds: Restart the MetaDefender Core service manually if it fails to start automatically, ensuring it is run under an account with appropriate permissions (e.g., Administrator).

Steps to Resolve:

- Review and modify scheduled tasks or scripts running to prevent them from removing or interfering with the

MetaDefender Core temporary folder.

- Exclude the MetaDefender Core installation path (`C:\Program Files\OPSWAT`) and temporary directory (`C:\Windows\Temp`) from antivirus and security software scans or cleanup routines.
- Monitor and verify that the MetaDefender Core and PostgreSQL services start normally after system reboots or scheduled restarts.
- Collect and review Windows Event Viewer logs related to Task Scheduler to identify any tasks that might affect the service.
- Follow OPSWAT's recommended system configuration guidelines and antivirus exclusion instructions available at: Recommended System Configuration
- If the temporary folder is missing, investigate what process or script is removing it and disable or adjust that process accordingly.
- Restart the MetaDefender Core service manually if it fails to start automatically, ensuring it is run under an account with appropriate permissions (e.g., Administrator).
- Keep the MetaDefender Core updated and monitor logs for any recurring errors.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why can't I access the MetaDefender Core user interface after performing an upgrade?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

After performing an upgrade of MetaDefender Core, you may encounter issues accessing the user interface.

Scenario 1: System or Configuration-Related Issues

If the MetaDefender Core service fails to start or crashes, it could be due to various system-level or configuration-related problems. These may include missing dependencies, permission issues, or service conflicts.

Refer to this support article: Why does the MetaDefender Core service not start? for a detailed list of common causes and troubleshooting steps for both Windows and Linux environments.

Scenario 2: SSL Certificate Issue After Upgrade

Another common cause, especially after an upgrade, is that SSL certificate settings are not retained. If HTTPS was previously enabled with a custom certificate, the upgrade process may not preserve that configuration. As a result, the UI may fail to load.

To resolve this, refer to the steps for checking local access and resetting the SSL configuration.

Note:

This section addresses a specific case where the MetaDefender Core UI becomes inaccessible after an upgrade, due to a lost SSL certificate configuration.

- For general UI access issues unrelated to upgrades or SSL, see Troubleshooting Inaccessible Management Console.
- For detailed steps on using the SSL configuration reset tool, see Resetting HTTPS Configuration.

Here's how you can resolve the issue:

Step 1: Check local access

1. Open your browser.
2. Try accessing the MetaDefender Core UI through one of the following addresses:
   - http://localhost:8008
   - https://localhost:8008

If successful, you can re-add your certificate directly from the UI.

Step 2: Reset SSL configuration

If you still cannot access the UI:

1. Navigate to the MetaDefender Core installation folder:

```
C:\Program Files\OPSWAT\MetaDefender Core\
```

2. Locate and run the tool with **administrator/root** privileges:

```
ometascan-reset-ssl-conf.exe
```

3. This will remove the existing certificate configuration.
4. Re-add your certificate through the MetaDefender Core UI.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

Why is ESET detecting legitimate software as a Potentially Unsafe Application (PUA/PUP)?

Check Your Version:

This article applies to all MetaDefender Core releases on Windows and Linux with **ESET antivirus licensed**.

Issue:

ESET may classify certain legitimate software tools as **Potentially Unsafe Applications (PUA/PUP)**.

These tools—such as remote access utilities, keyloggers, and password crackers—are commonly used for system administration but can be misused for malicious purposes.

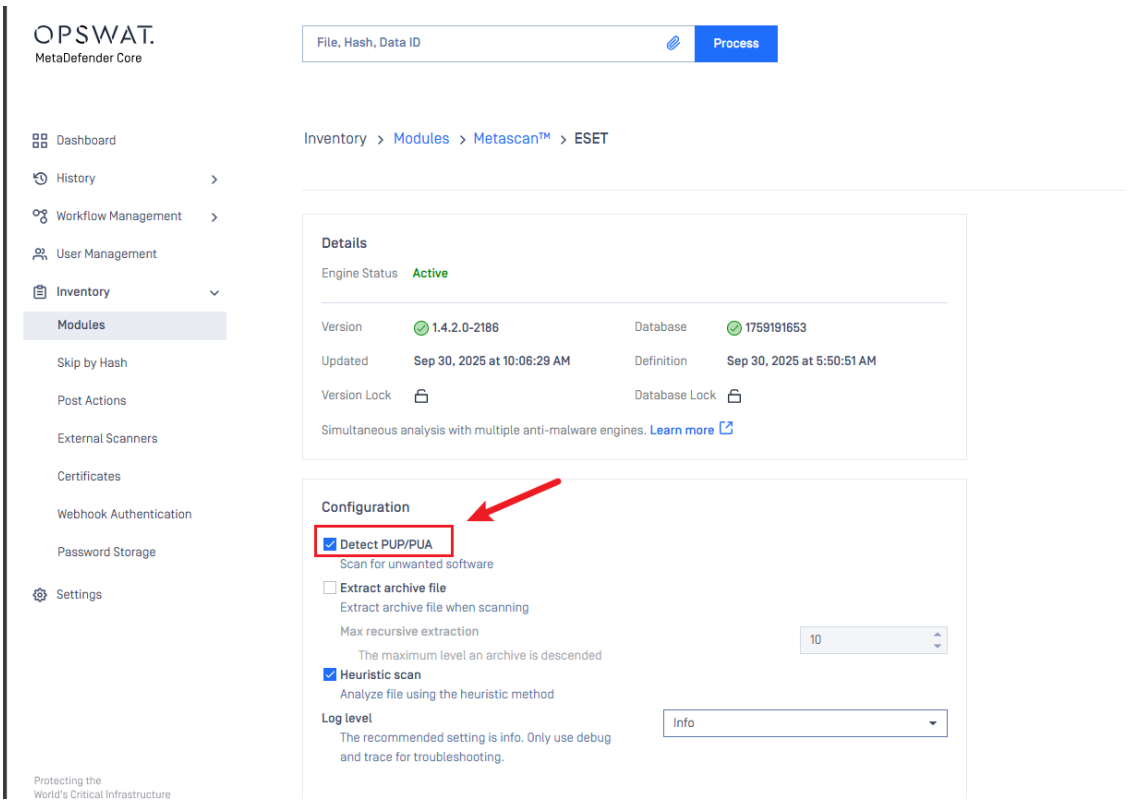
To mitigate risk, ESET flags them under the **Win32/PrcView** category and similar classifications.

Note:

This classification does not indicate the software is malicious, only that it has the potential to be misused.

If you're experiencing excessive false positives, you can disable PUA/PUP detection in ESET by following the steps below.

1. Go to **Inventory > Modules > Metascan > ESET**.
2. Disable "**Detect PUP/PUA**".



Workarounds:

Disabling PUA/PUP detection may reduce protection against potentially risky tools. As an alternative, you can **keep detection enabled** and **add detected files to the local whitelist**.

Refer to How to locally whitelist a file using allowlist for instructions.

Related Articles:

- How to locally whitelist a file using allowlist
- ESET Glossary | ESET Online Help (External link)

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why can't I configure the SMTP server with authentication set to "None" in MetaDefender Core 5.15.2?

Check Your Version:

This article is applied to MetaDefender Core deployments on Windows and Linux, version 5.15.2.

Issue

When configuring an SMTP server in **MetaDefender Core 5.15.2**, users may encounter an error if the authentication method is set to **None**.

Sample error message in Core log:

```
(common.workflow) Cannot login mail server with given options,
id='lms::workflow::RestWorkflowExecutor(0x2b80c0a7f20, name = "WF-127.0.0.1:64159")',
block='VerifyEmailBlock', error='Failed to send AUTH PLAIN command. Response code: 504' [msgid:
3523]
```

This issue affects **MetaDefender Core version 5.15.2** when setting up an email server with **Authentication Method: None**.

Solution

The issue was **addressed in MetaDefender Core version 5.16.0**. Please **upgrade to version 5.16.0 or later** to successfully configure SMTP servers using **Authentication Method: None**.

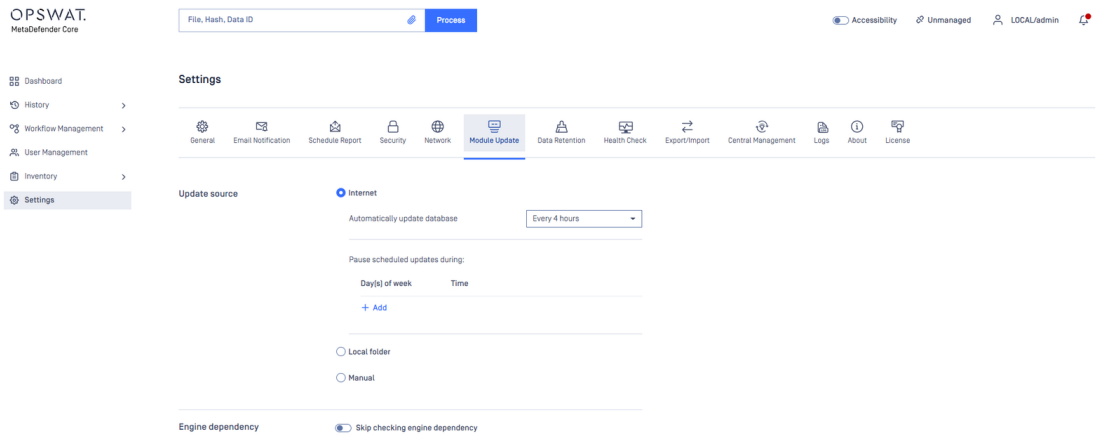
Note:

The issue **does not affect other SMTP authentication methods** such as **AUTH PLAIN** or **AUTH LOGIN**. If you are unable to schedule upgrade, you may use one of these authentication methods as a temporary workaround.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why are my Engines not initializing/updating over the Internet or OPSWAT Central Management On-Premises



MetaDefender Core offers several methods for updating its engines, providing flexibility to accommodate different deployment environments and security requirements. The available update methods include:

1. **Internet Update** MetaDefender Core connects directly to the **OPSWAT Update Server** (`https://update.dl.opswat.com`) to automatically initialize and update its modules.
2. **Local Folder Update** Module packages can be placed in a designated local directory, from which MetaDefender Core automatically detects and applies updates.
3. **Manual Update** Modules can be manually uploaded via the **Inventory** → **Modules** section within the MetaDefender Core management interface.

When **MetaDefender Core** is integrated with **OPSWAT Central Management (CM)**, a **fourth update option** becomes available, replacing the Internet Update method:

4. **Central Management Update** Engines are downloaded and deployed directly through **OPSWAT Central Management**, allowing centralized administration and streamlined maintenance across multiple MetaDefender Core instances.



Why Are My Engines Not Downloading/Updating via Internet or Central Management?

When selecting either the **Internet** or **Central Management (CM)** update option, it is crucial to ensure that direct connectivity to the respective update source is fully established.

- For **Internet updates**, MetaDefender Core must be able to connect to: `https://update.dl.opswat.com`
- For **Central Management updates**, the **Central Management** server itself must have access to the OPSWAT update server (`https://update.dl.opswat.com`) to retrieve and distribute engine updates to connected MetaDefender Core instances.

If this connectivity is blocked or restricted — due to firewall rules, proxy configuration issues, or DNS resolution problems — the engines will fail to initialize or update, resulting in errors such as the following:

- **When connected to OCM:**

Code

```
[WARNING] 2025.10.14 05:04:11.131: (common.communication) Error while downloading file,
url='QUrl("http://<OCM_URL>/console/core/package/
<REDACTED>&deployment=<DEPLOYMENT_ID>&key=<API_KEY>&serial=1")', http_status='400',
error='ProtocolInvalidOperationError', errorDetails='Error transferring http://<OCM_URL>/
console/core/package/<REDACTED>&deployment=<DEPLOYMENT_ID>&key=<API_KEY>&serial=1 - server
replied: ' [msgid: 485]
```

- **When downloading directly from Internet:**

Code

```
[INFO   ] 2025.10.14 05:07:28.934: (common.update) Downloading metadescriptor,
downloadlink='https://update.dl.opswat.com/metadescriptor?version=<REDACTED>' [msgid: 5758]
[WARNING] 2025.10.14 05:07:28.934: (common.communication) Error while downloading file,
url='QUrl("https://update.dl.opswat.com/metadescriptor?version=<REDACTED>")', http_status='0',
error='HostNotFoundError', errorDetails='Host update.dl.opswat.com not found' [msgid: 485]
[ERROR  ] 2025.10.14 05:07:28.949: (common.update) Error parsing metadescriptor, message='Could
not download metadescriptor' [msgid: 282]
```

Additional references:

What links, target-services or target host-IPs need to be allowed for Metadefender Core to function accurately?

What do I do if my modules disappear from the Core UI?

What do you do if your engine fails to update?

Conclusion

Maintaining proper network connectivity is essential for ensuring smooth and consistent engine updates in MetaDefender Core. Whether updates are performed directly from the Internet or managed centrally through OPSWAT Central Management, access to `https://update.dl.opswat.com` must be verified and unrestricted.

By confirming network reachability, validating proxy settings, and ensuring correct DNS resolution, administrators can prevent update failures, maintain optimal engine performance, and ensure MetaDefender Core remains fully up to date with the latest threat intelligence and scanning capabilities.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why am I getting an "Internal server error" when logging into MetaDefender Core via SAML SSO

Check Your Version:

This article applies to MetaDefender Core V4 and V5 deployed on Windows and Linux.

Issue: Authentication failure when logging into MetaDefender Core via the SAML SSO authentication method.



Error Details: Error: "Internal server error"

This error occurs while signing into MetaDefender Core using SSO method.

Reason: This error occurred due to an incorrect system time on the MetaDefender Core machine, which caused the session to expire before the validity period.

Resolution:

Method 1: If the machine is connected to the internet.

1. Right-click the clock on the Windows taskbar and select **Adjust date and time**.
2. In **Date & time**, turn **On** the **Set time automatically** toggle.
3. If needed, turn **On Set time zone automatically** (or choose the correct time zone from the **Time zone** dropdown).
4. Scroll down and click **Sync now** under **Additional settings** → **Sync your clock** to force a manual sync.

Method 2: If the machine is not connected to the internet.

1. Right-click the clock on the taskbar **Adjust date and time**.
2. Turn **Off Set time automatically**.
3. Click **Change** under **Set the date and time manually**.
4. Enter the correct **Date** and **Time**, then click **Change**.
5. Then go to MetaDefender Core and restart the service.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why is HTTPS enabled in the UI, but the MetaDefender Core URL doesn't change to HTTPS?

Check Your Version:

This article applies to all MetaDefender Core version 5.x on Windows Bases

Scenario

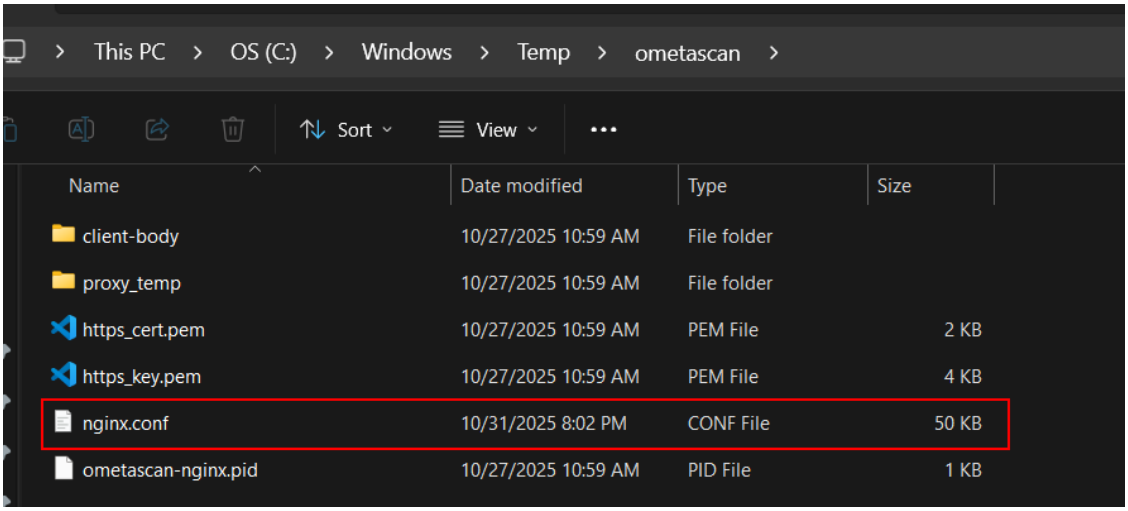
When you enable **HTTPS** for the MetaDefender Core UI to enhance security, you may notice that although the UI indicates HTTPS is enabled, the Core URL still appears as **HTTP**. This Knowledge Base article provides detailed instructions to resolve this issue and ensure the MetaDefender Core UI is properly accessible over HTTPS.

Root Cause

The **MetaDefender Core Web Server** uses **NGINX** as its underlying web server. Its configuration file is stored at:

C:\Windows\Temp\ometascan\nginx.conf

This file serves as the master configuration for the NGINX process.

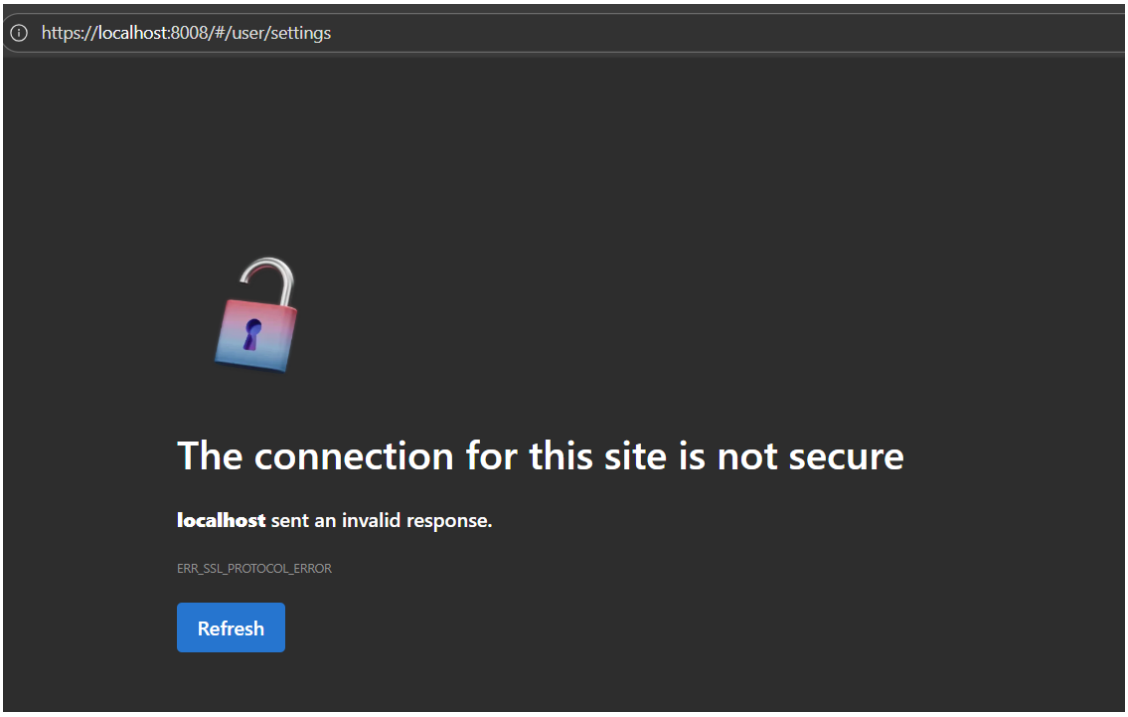


In certain environments—particularly those with **Real-Time Protection (RTP)** enabled—the configuration file may be automatically deleted. Once NGINX has successfully started, it loads the configuration into memory, so the removal of the file does not immediately affect the running web service.

However, if the `nginx.conf` file is missing, any subsequent configuration changes (such as enabling or disabling HTTPS) will not take effect, since NGINX cannot reload or update its settings without this file.

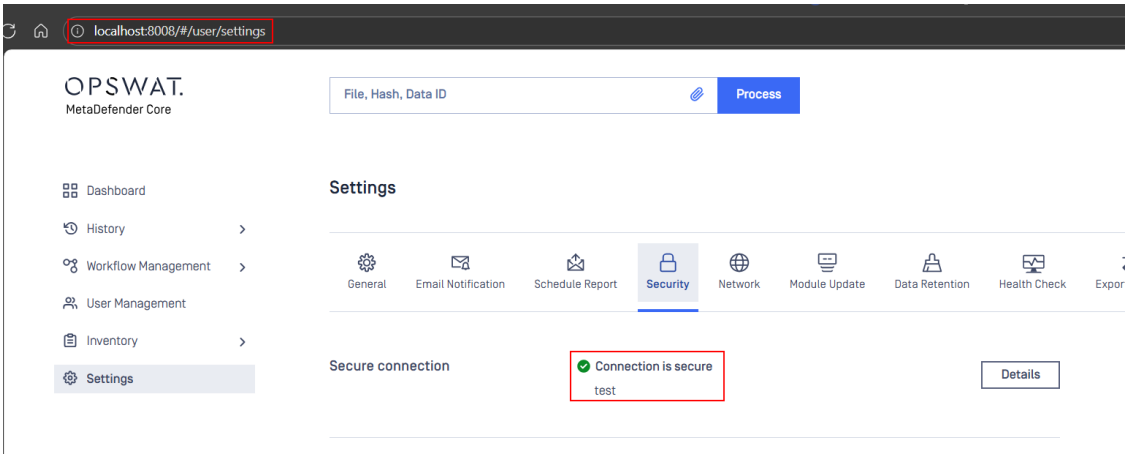
Symptom

After enabling **HTTPS** through the MetaDefender Core UI, you may encounter an error page when attempting to access the web interface. This indicates that the browser's request did not reach the web server, suggesting that the **NGINX web service** is not properly serving HTTPS traffic or its configuration is missing or invalid.



When you try to access the **MetaDefender Core UI** using **HTTP** instead of **HTTPS**, the interface loads successfully. However, upon reviewing the **HTTPS settings** in the UI, you can see that the **HTTPS option is already enabled**, as shown below.

This indicates that HTTPS has been configured, but the Core instance continues to serve the web interface over HTTP due to missing or invalid NGINX configuration files.



Resolution

Navigate to the following directory:

`C:\Windows\Temp\ometascan`

If the **ometascan** folder does not exist or is empty, there is no need to worry — this can be automatically regenerated by restarting the MetaDefender Core service.

Follow the steps below:

1. **Stop** the **MetaDefender Core** service.
2. **Start** the service again.

During startup, MetaDefender Core checks for the `ometascan` temporary folder.

- If the folder or its configuration files are missing, the system will automatically recreate them and regenerate the required **NGINX configuration files**.
- Once the process completes, the web service will resume normal operation, and **HTTPS** will be automatically re-enabled.

Note:

If you notice that RTP keeps deleting the **nginx.conf** file in the folder **"C:\Windows\Temp\ometascan"**, you can add this folder to the RTP exclusion list to prevent the file from being removed.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer at Home - My OPSWAT .

Why does the MetaDefender Core Deployment ID change after rebooting a Hyper-V VM?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Hyper-V.

Issue:

After rebooting a MetaDefender Core virtual machine hosted on Hyper-V, the Deployment ID may change unexpectedly. This behavior is typically caused by the VM's network adapters being configured with **dynamic MAC addresses**.

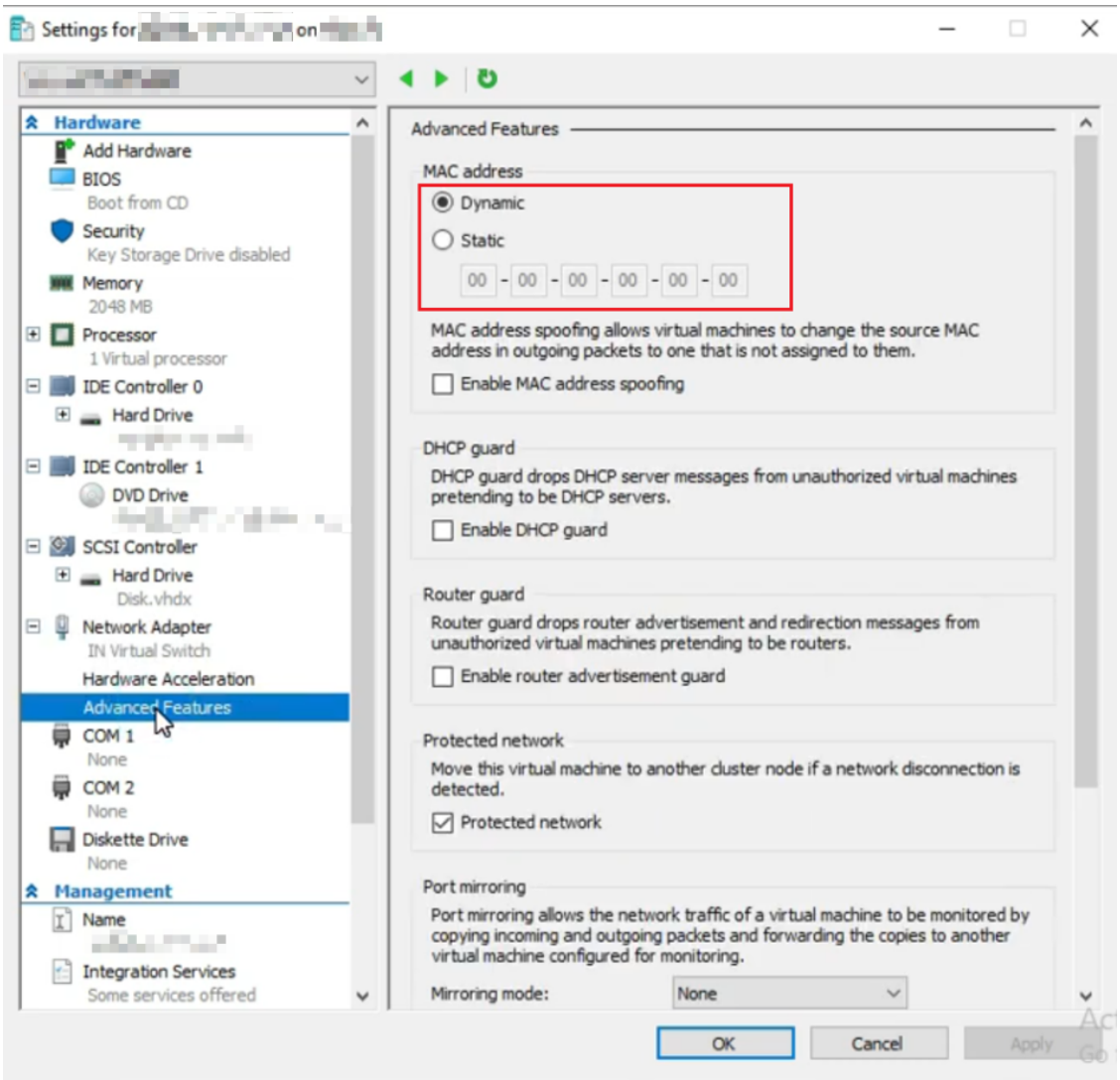
In Hyper-V environments, dynamic MAC addresses can be reassigned during a reboot, resulting in a different MAC address.

Since MetaDefender Core generates its Deployment ID based on hardware identifiers such as the MAC address, any change to the MAC address will lead to a new Deployment ID being created.

How to Check MAC Address Settings in Hyper-V

To verify whether your VM is using dynamic or static MAC addresses:

1. Open **Hyper-V Manager** on the host machine.
2. Right-click the virtual machine in question and select **Settings**.
3. In the left pane, expand the VM name and select **Network Adapter**.
4. Click on **Advanced Features**.
5. Check the **MAC Address** setting — if it is set to **Dynamic**, it may change after a reboot.



If your VM has multiple network adapters, repeat the steps for each one.

Recommended Solution

To prevent the Deployment ID from changing after a reboot, configure the MAC address of each network adapter to **Static**. This ensures that the MAC address remains consistent across reboots, maintaining the same Deployment ID.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why are MetaDefender Core Syslog Messages Flagged as Non-Compliant with certain syslog standards?

Check Your Version:

This article applies to MD Core version 5.16.0 and older

Issue:

Syslog messages sent from MetaDefender Core are flagged in Wireshark as “Message conforms to neither RFC 5424 nor RFC 3164; trailing data appended.” This raises a question regarding whether MetaDefender Core syslog output complies with RFC 3164 or RFC 5424 standards.

Symptoms:

When capturing syslog packets from MetaDefender Core, user may observe the following:

```
[Expert Info (Note/Protocol): Message conforms to neither RFC 5424 nor RFC 3164; trailing data appended] Example: Syslog message: DAEMON.INFO: 2025-06-24T03:35:53Z localhost.localdomain MSCL [156240] Send additional configs, configs='{"SysLog": … }' The syslog message includes a <pri> header (e.g., <30>) followed by timestamp and hostname, but deviates from the strict structure defined in the RFCs.
```

Resolution:

Syslog messages in current versions of MetaDefender Core (≤ 5.16.x) may not fully comply with certain syslog format standards.

Enhanced support for fully compliant syslog messages is planned for upcoming versions.

Note:

There is no immediate workaround. The syslog output can still be parsed or processed by most syslog collectors as long as they are configured to handle non-strict message formats.

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

Why is the "The engine is not longer licensed with the current

product key" message displayed for Threat Intelligence after license renewal?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue: *After applying new activation keys for license renewal, the customer sees the message on the Inventory > Modules > Threat Intelligence screen: "The engine is not longer licensed with the current product key." The customer is unsure if this message will automatically change after the license start date (October 1, 2025) or if further action is needed.*

Resolution: If the customer requires Threat Intelligence functionality, please contact the Sales team to discuss licensing options.

If the customer does not require Threat Intelligence, they can ignore the message or remove the Threat Intelligence engine module from their MetaDefender Core installation.

Workarounds: If the customer does not require Threat Intelligence, they can ignore the message or remove the Threat Intelligence engine module from their MetaDefender Core installation.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Are Embedded Files Recursively Processed and Reported during CDR?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Summary:

Issue

During a CDR-enabled process, a file may contain other embedded or internal objects (for example, documents within archives, or images containing embedded files). It may be unclear whether these extracted components are also processed by CDR and whether the results are shown in the scan report.

Cause

While CDR can reconstruct and sanitize embedded content, recursive sanitization and detailed reporting depend on configuration settings and supported file types. If certain options are disabled, internal or embedded files may not be

sanitized or reported individually.

Resolution

To ensure that embedded or internal files are recursively processed and reported during CDR processing:

1. **Enable Recursive Scanning for Embedded File Types** Verify that recursion of embedded file types is enabled in the configuration. This allows CDR to sanitize internal files..
2. **Enable Detailed Sanitization Reporting** Turn on the **"Include Sanitisation Details / Processed Objects"** option. This ensures that the scan history includes detailed information about the sanitization actions and results of internal components.
3. **Confirm Supported File Types** Make sure the file types being processed support recursive scanning and sanitization. Some formats may not support deep recursion or detailed reporting. If a file type is not supported, CDR will act based on the "Remove embedded object" configuration to decide whether to remove the embedded file or not

When all of these settings are correctly configured, the CDR process will:

- Recursively scan embedded or internal files.
- Include the results and sanitization details of those components in the scan history report.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why am I getting an error when installing MetaDefender Core on Linux in an air-gap environment?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Linux.

Issue:

- When installing the MetaDefender Core package on a Linux machine via the `dnf`, `yum`, or `apt` command in an air-gap environment, you may encounter an error.

```
[root@ ~]# sudo yum install ometascan-5.17.0-1.x86_64.rpm
Updating Subscription Management repositories.

This system is registered with an entitlement server, but is not receiving updates. You can use subscription-manager to assign
 subscriptions.

Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)                    0.0  B/s |   0  B     00:00
Errors during downloading metadata for repository 'rhel-9-for-x86_64-appstream-rpms':
  - Curl error (6): Couldn't resolve host name for https://cdn.redhat.com/content/dist/rhel9/9/x86_64/appstream/os/repodata/re
pomd.xml [Could not resolve host: cdn.redhat.com]
Error: Failed to download metadata for repo 'rhel-9-for-x86_64-appstream-rpms': Cannot download repomd.xml: Cannot download re
podata/repomd.xml: All mirrors were tried
```

```
root>14:27:45 $ sudo apt install ometascan_5.17.0-1_amd64.deb
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package ometascan_5.17.0-1_amd64.deb
```

Main cause:

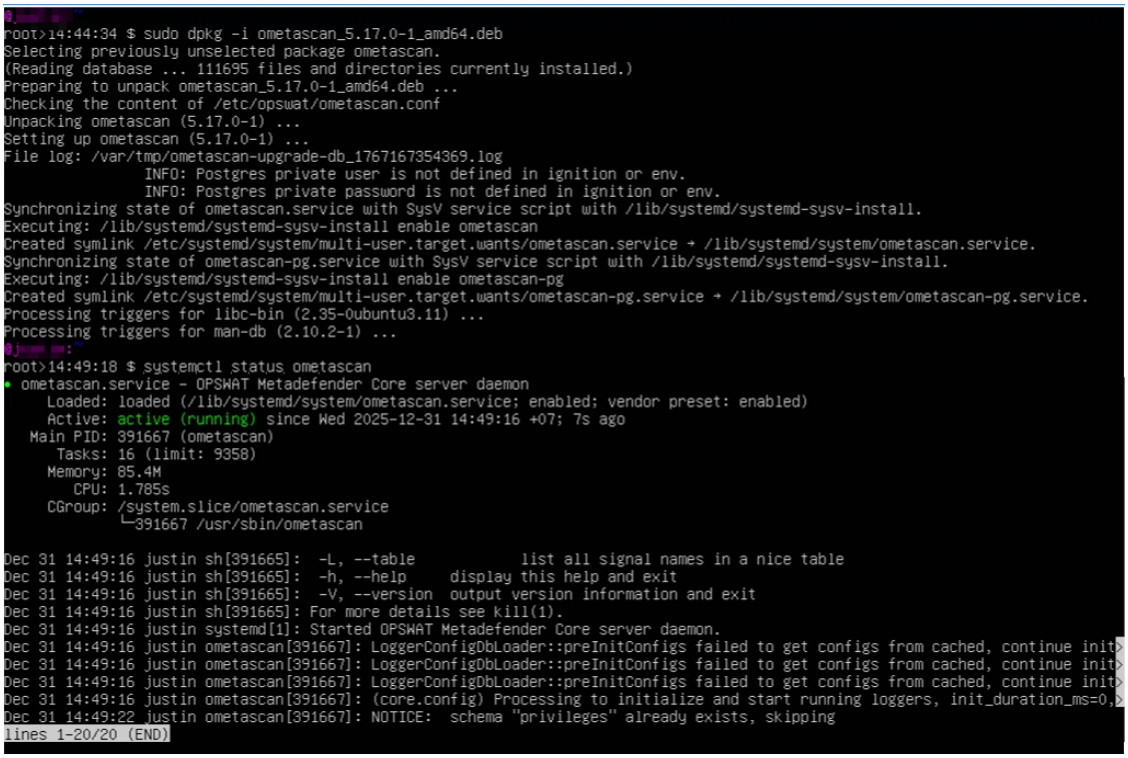
- When performing an installation via a package manager (`dnf`, `yum`, `apt`), the system attempts to reach out to a package repository to obtain information about the package being installed. In an air-gap environment, if a local repository is not configured, the download or upgrade process via the package manager will fail.

Resolution:

- Install the package using a local installation method, such as the native `rpm` or `dpkg` command.

```
[root@ ~]# sudo rpm -ivh ometascan-5.17.0-1.x86_64.rpm
Verifying...                          ################################# [100%]
Preparing...                          ################################# [100%]
Updating / installing...
   1:ometascan-5.17.0-1               ################################# [100%]
File log: /var/tmp/ometascan-upgrade-db_1767165062908.log
               INFO: Postgres private user is not defined in ignition or env.
               INFO: Postgres private password is not defined in ignition or env.
Created symlink /etc/systemd/system/multi-user.target.wants/ometascan.service → /usr/lib/systemd/system/ometascan.service.
Created symlink /etc/systemd/system/multi-user.target.wants/ometascan-pg.service → /usr/lib/systemd/system/ometascan-pg.servic
e.
[root@ ~]# systemctl status ometascan
● ometascan.service - OPSWAT Metadefender Core server daemon
     Loaded: loaded (/usr/lib/systemd/system/ometascan.service; enabled; preset: disabled)
     Active: active (running) since Wed 2025-12-31 14:11:03 +07; 18s ago
   Main PID: 4261 (ometascan)
      Tasks: 68 (limit: 100224)
     Memory: 156.3M
        CPU: 2.416s
     CGroup: /system.slice/ometascan.service
             ├─ 4261 /usr/sbin/ometascan
             ├─10235 "nginx: master process /usr/lib/ometascan/nginx -p // -c //var/run/ometascan/nginx.conf"
             ├─11205 "nginx: worker process"
             ├─11206 "nginx: worker process"
             └─11207 "nginx: worker process"

Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_sbom_license_risk_instanceid_hour_license_risk_ke>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_sbom_license_risk_instanceid_hour_license_risk_ke>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_reputation_error_verdict_instanceid_hour_verdict_>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_coo_error_verdict_request_type_instanceid_hour_ve>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_blocked_verdict_by_user_instanceid_hour_verdict_i>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_scan_performed_by_user_instanceid_hour_verdict_id>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_insightsti_error_verdict_request_type_instanceid_>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  identifier "warehouse_fact_insightsti_blocked_ioc_instanceid_hour_verdict_id>
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  extension "btree_gin" already exists, skipping
Dec 31 14:11:10 justin ometascan[4261]: NOTICE:  relation "warehouse_fact_sbom_license_risk_instanceid_hour_license_risk_k" a>
lines 1-24/24 (END)
```



Support:

If the problem still persist, please proceed to **create a support case or chat with our support engineer.** for futher assistant.

Why are Encrypted Office Documents with the same extension categorized differently from unencrypted documents?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Question:

MetaDefender Core categorizes Office documents with the same extension (.docx, .xlsx, etc.) differently.

Some are identified as "Encrypted Microsoft Office Document," while others are identified as "Microsoft Excel Workbook" or "Encrypted Microsoft Excel Sheet." This article aims to provide clarifications about the logic behind this categorization.

Clarification:

MetaDefender Core detects encrypted Microsoft Office files differently based on their encryption type and file format version. Here's a breakdown, using Microsoft Excel document as an example:

- **Unencrypted and non-password-protected .xlsx** files are detected as "Microsoft Excel Workbook."
- **Encrypted .xlsx** files are detected as "Encrypted Microsoft Office Document" due to password protection or

Microsoft Office Document Information Protection.

- In MS Office 2007 and later (docx, xlsx, pptx): encrypted files share the **same structure** across Word, Excel, and PowerPoint, so the general term “Encrypted Microsoft Office Document” is used.
- For more information on this file type, please refer to Can MetaDefender Core process Microsoft Purview Information Protection Encrypted Files?

- **Older encrypted Excel files .xls** are detected as "Encrypted Microsoft Excel Sheet."
  - In MS Office 2003 (doc, xls, ppt): encrypted files have **distinct structures** for Word, Excel, and PowerPoint, so file type engine can clearly return “Encrypted Microsoft Excel Sheet.”

Action Required:

No action or configuration change is required on the user's side, as this is the expected behavior of MetaDefender Core.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Why does MetaDefender Core skip quarantining files with the same hash?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

After scanning a known malicious file:

- The first instance of the file is successfully quarantined.
- A second instance of the same file (from another device or scan) is blocked, but not added again to the quarantine repository.
- Users may wonder why the second file does not appear in quarantine storage.

Resolution:

No action is required. The system’s skipping of duplicate quarantine entries is:

- Expected by design.
- Safe, since blocking and handling still occur as configured.
- Reduce disk space consumption.

As a result, there is no supported configuration to change this quarantine behavior.

Best practice

- Keep hash calculation enabled in workflows
- Do not modify quarantine behavior unless directed by OPSWAT Support
- Validate failures by checking that blocked files are removed from the source device

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

How to create a sample self-signed certificate with SAN using OpenSSL

Check Your Version:

This article is applied to Web Browsers (Chrome, Firefox, etc.) releases deployed on Windows Linux and macOS systems. It is required to use OpenSSL verasion 3.x to execute these steps.

Issue:

Modern web browsers require a Subject Alternative Name (SAN) in SSL/TLS certificates. The Common Name (CN) field, historically used for hostname verification, is deprecated due to compatibility and security concerns. Browsers reject certificates lacking SAN with errors like: “Certificate is not valid for the name.”

Affected Version/Platforms: Chrome 58 and later, Firefox, and other modern web browsers.

Resolution: To create an sample SSL/TLS certificate for testing purposes with a Subject Alternative Name (SAN), follow these steps:

1. **Create the** `san.cnf` Configuration File:

Code

```
[ req ]
default_bits       = 2048
prompt             = no
default_md         = sha256
distinguished_name = dn
req_extensions     = req_ext

[ dn ]
CN = yourdomain.com

[ req_ext ]
```

```
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = yourdomain.com
DNS.2 = www.yourdomain.com
DNS.3 = api.yourdomain.com
```

2. **Generate the CSR and Private Key:**

Code

```
openssl req -new -nodes -out request.csr -newkey rsa:2048 -keyout private.key -config san.cnf
```

3. **(Optional) Verify SAN Entries:**

Code

```
openssl req -in request.csr -noout -text | grep DNS
```

4. **Create the Self-Signed Certificate:**

Code

```
openssl x509 -req -in request.csr -signkey private.key -out certificate.crt -days 365 -
extensions req_ext -extfile san.cnf
```

Prevention: In case there are issues with a certificate provided from a CA, testing with a self signed certificate can help with the troubleshooting process.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

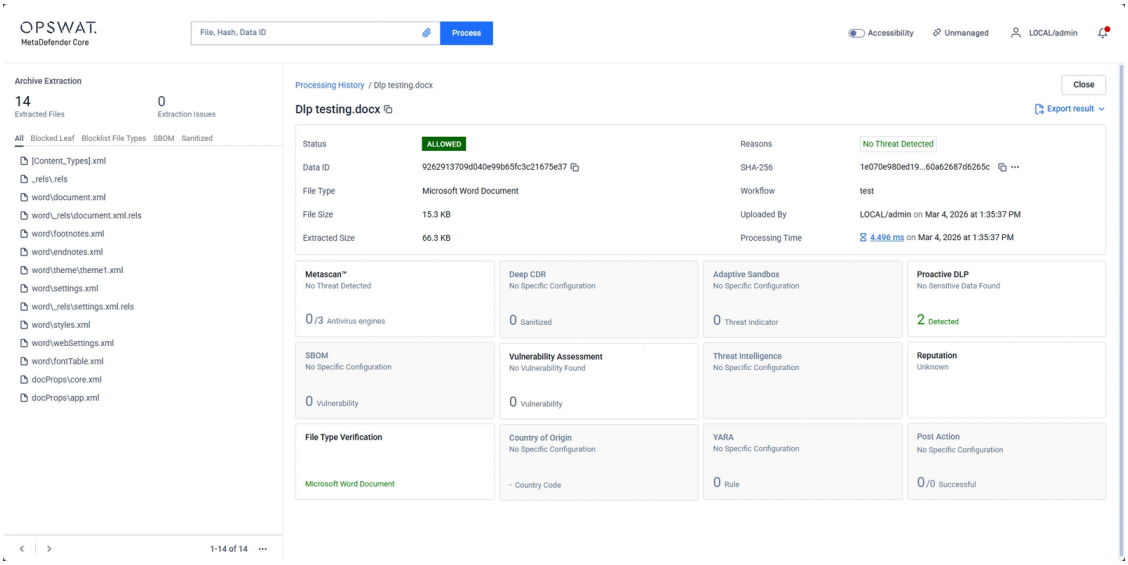
Why is MetaDefender Core Allowing Files with Sensitive Data? (Proactive DLP Threshold Settings)

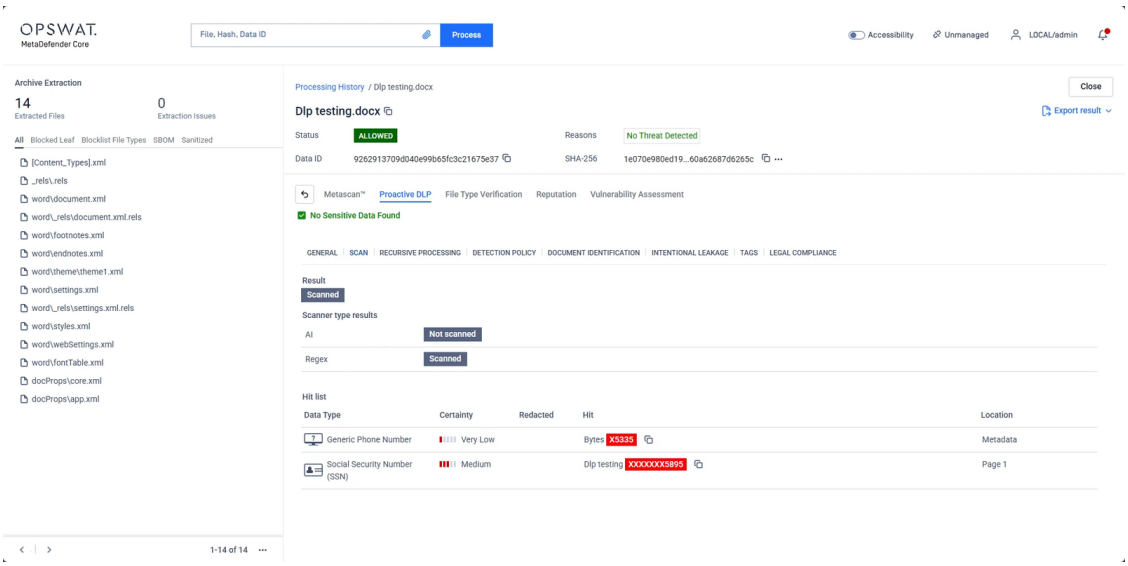
Check Your Version:

This article applies to all releases of MetaDefender Core deployed on Windows and Linux operating systems where the Proactive DLP feature has been purchased and enabled.

Issue:

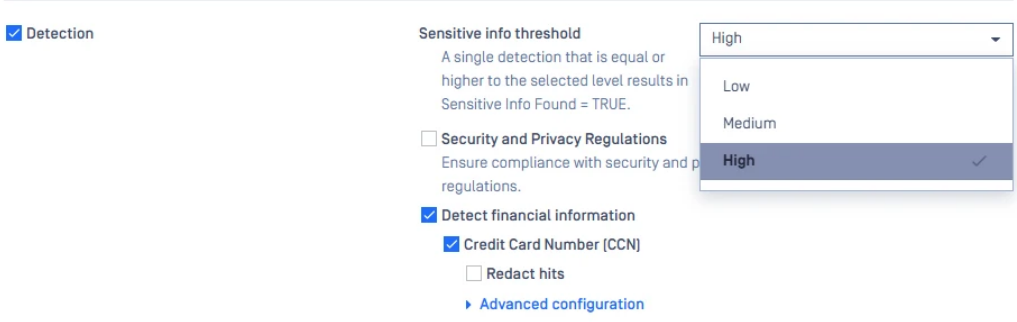
In MetaDefender Core, a file may contain sensitive information (such as SSNs or phone numbers) but still receive an "ALLOWED" status. This typically occurs due to the **Sensitive Info Threshold settings**.





Understanding the Threshold Logic

The "Sensitive Info Threshold" acts as a filter for the severity and certainty of detections. The engine evaluates hits based on a scale (e.g., Low to High).



The Rule:

A file is only flagged as "Sensitive Data Found" if at least one detection meets or exceeds the selected threshold level.

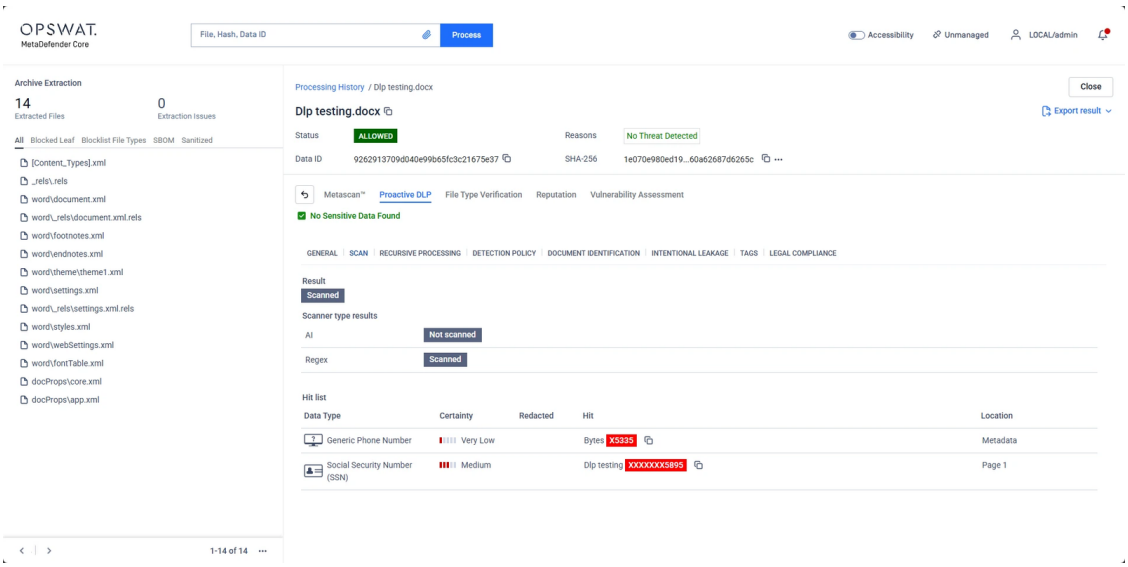
The Scenario:

If your threshold is set to High, but the Proactive DLP engine identifies hits with Very Low or Medium certainty, the system will conclude that no actionable sensitive data was found.

Analysis of Scan Result

Looking at the example file *Dlp testing.docx*:

- Hit 1 (Phone Number): Detected with Very Low certainty.
- Hit 2 (SSN): Detected with Medium certainty.



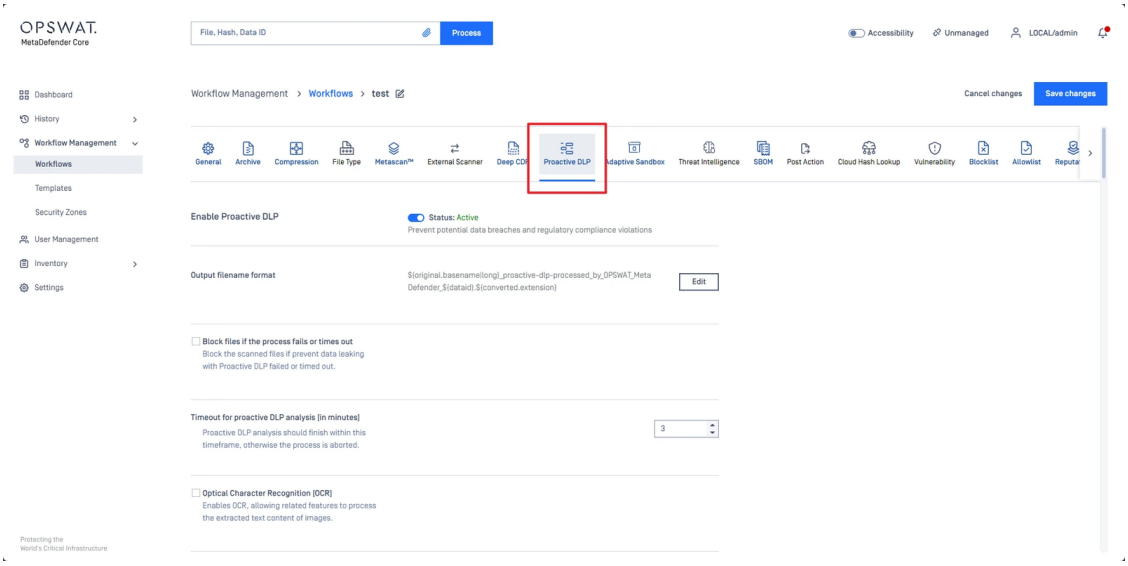
Result:

Because neither hit reached the "High" threshold (assuming that is your current configuration), the final status remains "No Sensitive Data Found" and the file is ALLOWED.

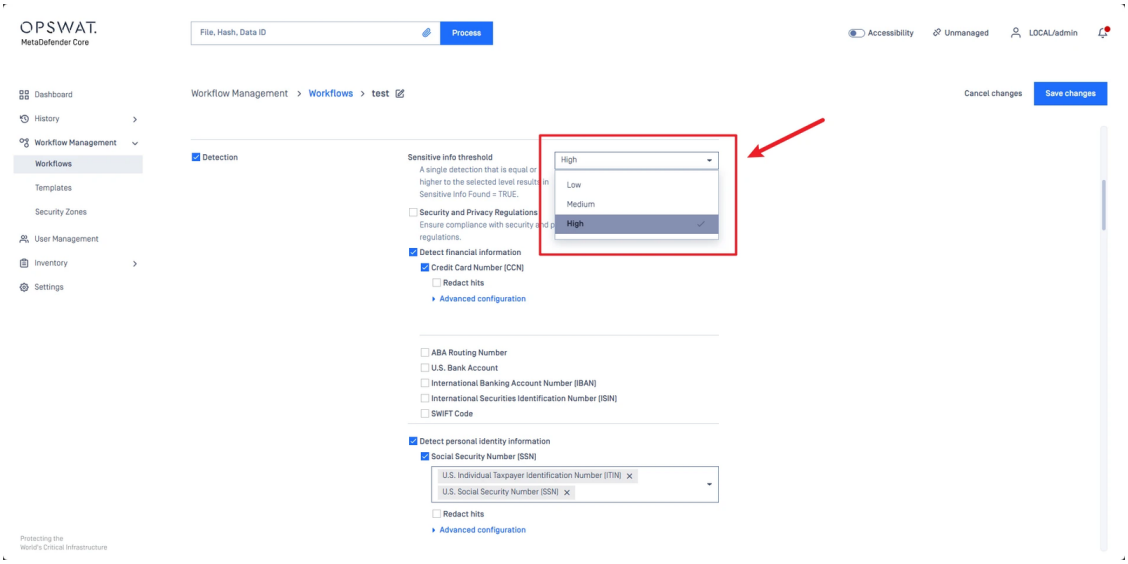
Resolution:

If you want these files to be blocked in the future, you should:

1. Navigate to your Workflow Management settings.
2. Locate the Proactive DLP configuration.



3. Under **Detection** > Lower the **Sensitive Info Threshold** to Medium or Low to capture less certain matches.



Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

What to do when my MetaDefender Core instance failing to load with error 'Extension setting up failed: extension: 'ekm', rc: '-610"?

Check Your Version:

This article is applied to MetaDefender Core releases deployed on Windows systems.

Issue

MetaDefender Core instance fails to load with the error message: `Extension setting up failed: extension: 'ekm', rc: '-610'`.

Root Cause The issue occurs because the Archive Engine stores temporary files in the `resources` folder and fails to remove them under certain conditions. These residual files can prevent MetaDefender Core from starting correctly.

Affected Versions/Platforms

- MetaDefender Core: Versions prior to **5.11.0**
- Archive Engine: Versions prior to **7.1**

Workaround To restart the MetaDefender Core instance successfully, you must remove the leftover Archive Engine folders located in the Core's `resources` directory.

Steps:

1. Stop all MetaDefender Core services.
2. Navigate to: `C:\Program Files\OPSWAT\MetaDefender Core\data\resources`
3. Delete **all** files and subfolders related to Archive Engine. The folders might be named:

- 7z_XX_windows_XXXXXX
- compression_XX_windows_XXXXXX

4. Start the MetaDefender Core service.

Recommendation To prevent this issue from recurring:

- Upgrade the **Archive Engine** to version **7.1** or later, where this cleanup issue is fixed.
- Upgrade **MetaDefender Core** to version **5.11.0** or later, where additional handling was implemented to enhance stability.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

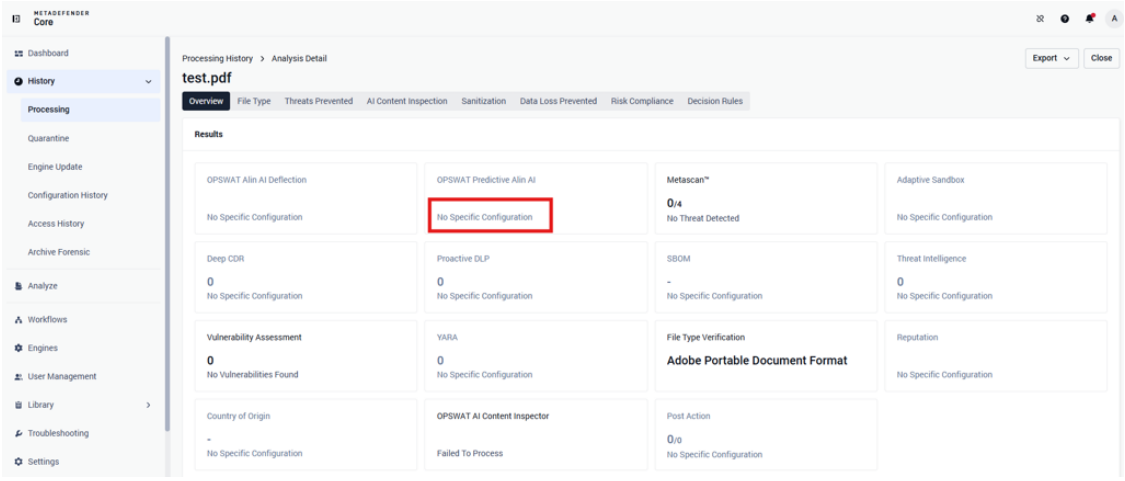
Why am I getting "We couldn’t analyze this file because no specific configuration is applied" with an OPSWAT Predictive Alin AI license?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue:

Users may encounter the error message **"We couldn't analyze this file because no specific configuration is applied"** when attempting to analyze files, even when an OPSWAT Predictive AI license is being used.

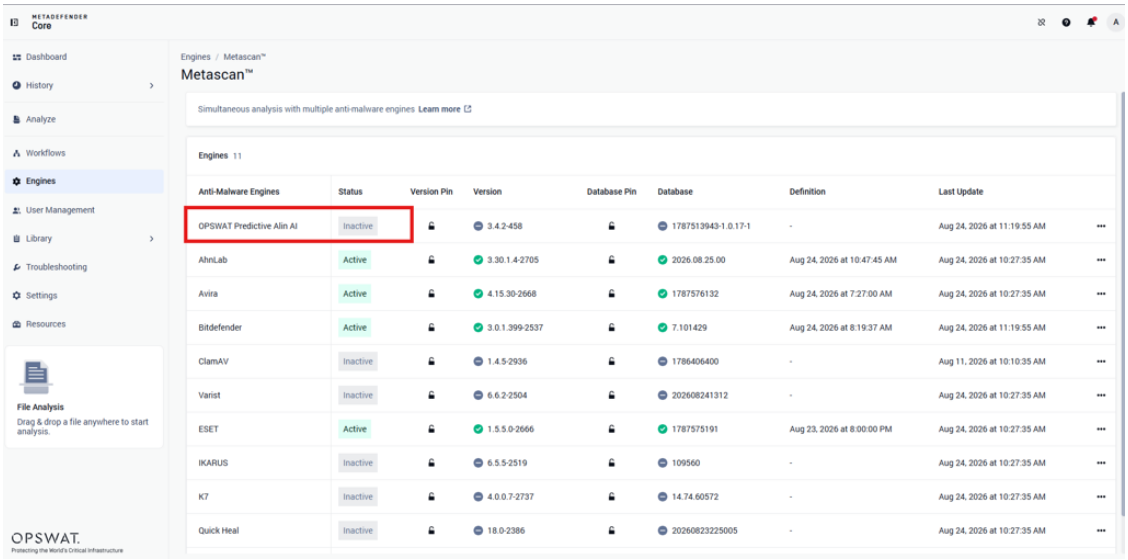


Cause:

The OPSWAT Predictive Alin AI is not activated/enabled under **Inventory > Modules > Metascan™**

Steps to Resolve:

1. Navigate to **Inventory > Modules > Metascan™**.
2. Ensure that **OPSWAT Predictive Alin AI** is enabled.



Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why am I getting a "File not found, invalid path or access" error

when using the 'filepath' header in the RestAPI on Linux?

Check Your Version:

This article applies to MetaDefender Core All MetaDefender Core releases deployed on Linux.

When using the RestAPI to send a file under the header ‘filepath', the error 'File not found, invalid path or access’ is returned.

Code

```
# curl -X POST http://MDCORE-HOST:8008/file -H "apikey: xxxxxxxxxxxxx" -H "filepath:/home/
admin/test.txt"  -H "rule:xxxxx"
{"err":"File not found, invalid path or access."}
```

Even though the file exists and is accessible when traversed manually.

Code

```
# ls -la /home/admin/test.txt
-rwxrwxrwx. 1 root root 5 Nov 12 09:39 /home/admin/test.txt

# namei -l /home/admin/test.txt
f: /home/admin/test.txt
dr-xr-xr-x root  root  /
drwxr-xr-x root  root  home
drwxr-x--- admin admin admin
-rwxrwxrwx root  root  test.txt
```

Main Cause:

MetaDefender Core runs as the ‘metascan' user. When a request is made with the ‘filepath’ header, the 'metascan’ user requires explicit permission to access the file, beyond the permissions granted to normal or root users.

Code

```
# sudo -u metascan namei -l /home/admin/test.txt
f: /home/admin/test.txt
dr-xr-xr-x root  root  /
drwxr-xr-x root  root  home
drwxr-x--- admin admin admin
                      test.txt - Permission denied
```

Resolution:

The `metascan` user needs at least 'Execute' (x) permission at each folder/file level to access the file. Adjust the permissions on the directory level to grant 'execute' permission. For example:

Code

```
# chmod +x /home/admin/

# sudo -u metascan namei -l /home/admin/test.txt
f: /home/admin/test.txt
dr-xr-xr-x root  root  /
drwxr-xr-x root  root  home
drwxr-x--x admin admin admin
-rwxrwxrwx root  root  test.txt
```

Then, verify again

Code

```
# curl -X POST http://MDCORE-HOST:8008/file -H "apikey: xxxxxxxxxxxxx" -H "filepath:/home/
admin/test.txt"  -H "rule:xxxxx"
{"data_id":"95690812c3594522a18ce2b3c8a2ae7b"}
```

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Can MetaDefender Core process Base64 Content?

Yes, MetaDefender Core is capable of decoding and processing Base64 Encoded files.

It is possible to send a Base64 Encoded string using the API Call features.

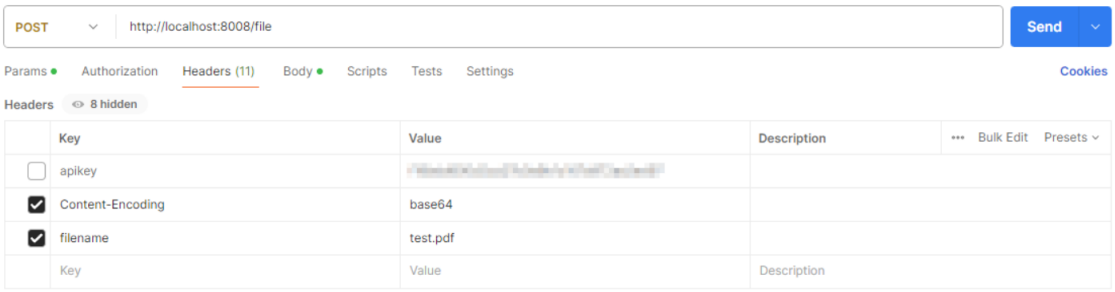
For example, we took the following PDF File and encoded it:

image

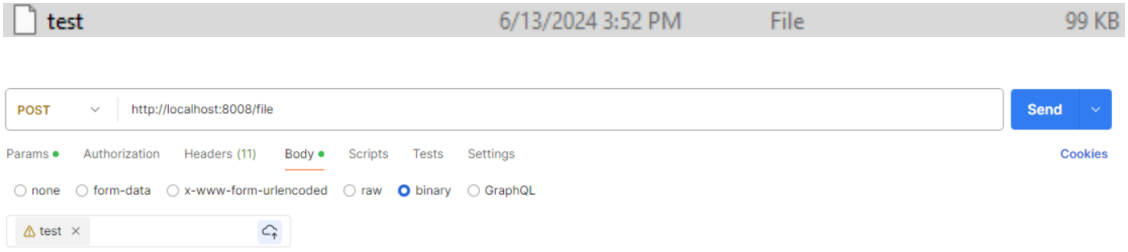
Resulting in a string:

JVBERi0xLjcNCiW1tbW1DQoxIDAgb2JqDQo8PC9UeXBlL0NhdGFsb2cvUGFnZXMgMiAwIFIvTGFuZyhlbikgL1N0cnVjdFRyZWVSb290IDI4IDAgUi9NYXJrSW5mbzw8L01hcmtlZCB0cnVlPj4vTWV0YWRhdGEgNTQgMCBSL1ZpZXdlclByZWZlcmVuY2VzIDU1IDAgUj4……

Using Postman, we prepared a POST message to <protocol>://<md_core_address>:8008/file URI, enabling Content-Encoding header, with the base64 as the given value.



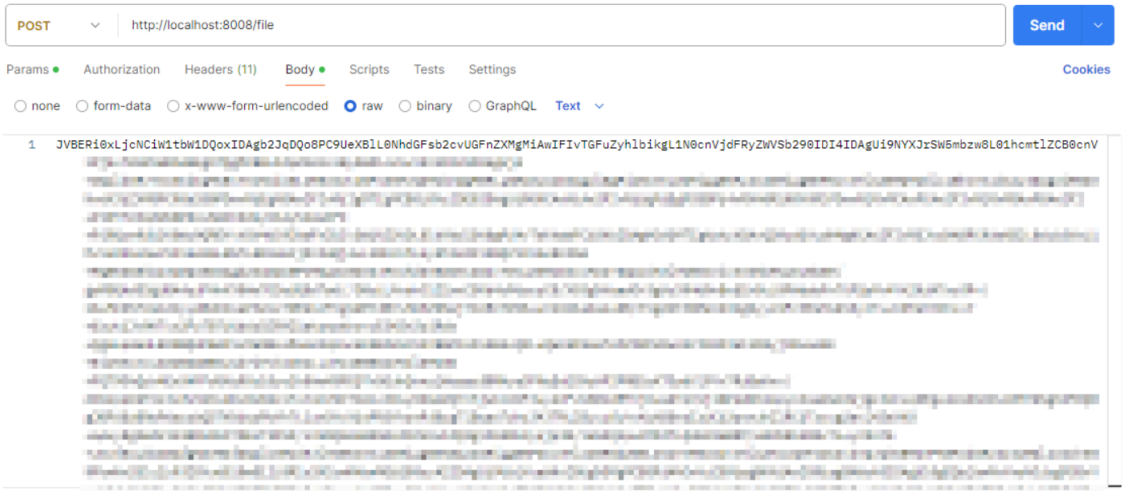
And pasted the Base64 Encoded string in an empty file which we uploaded as binary:



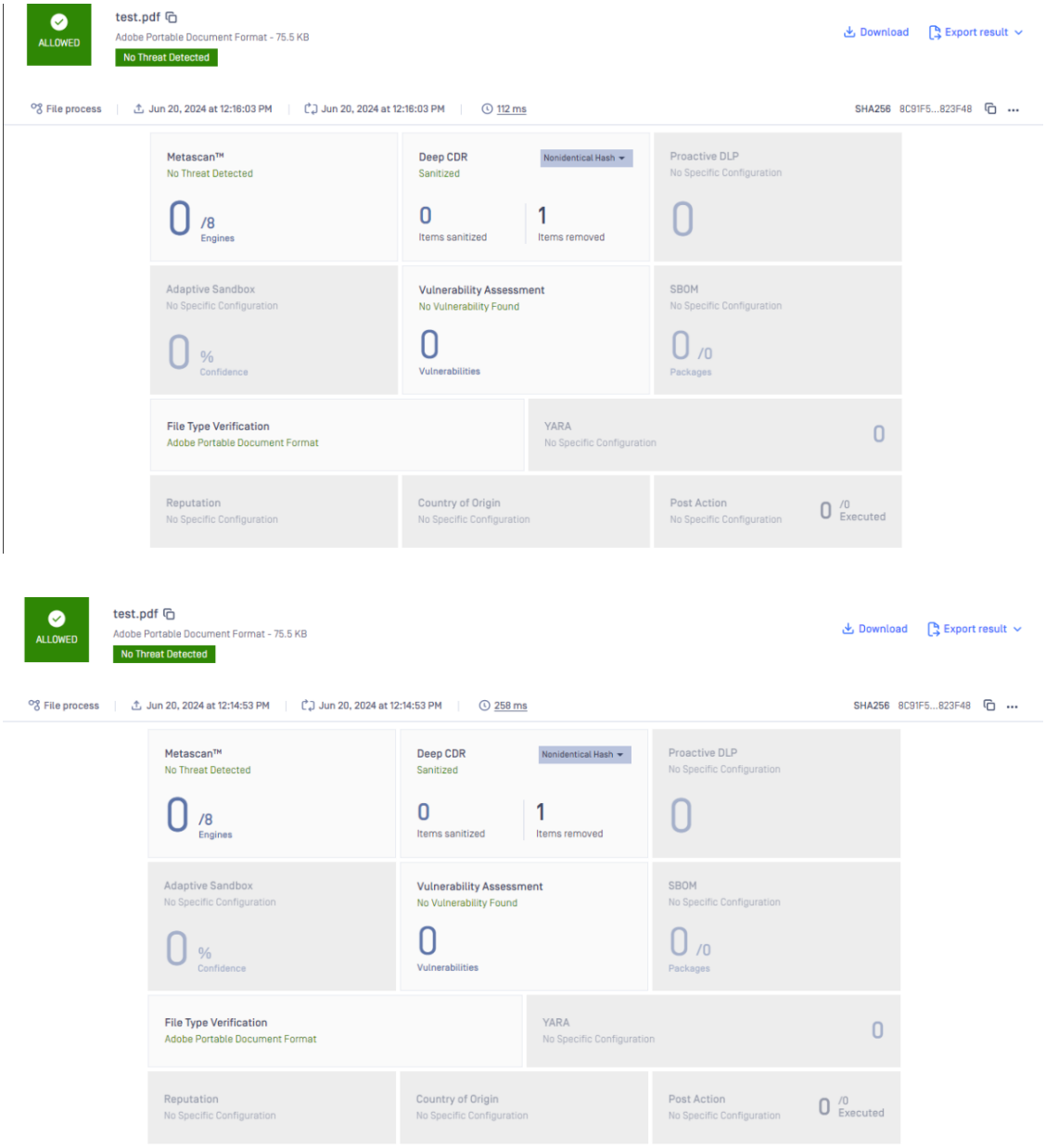
After which we have sent the message and obtained a result:



Additionally, we can send the string as a Raw Body base64 text string:



Checking the MetaDefender Core's History, we have found that the sent file has been correctly decoded, for both methods:



We can check the HASH against the original file and see they are all the same:



MetaDefender Core on K8s: Persisting Quarantine, DLP, and CDR Output Across Pod Restarts

Check Your Version:

This article applies to MetaDefender Core All MetaDefender Core releases deployed on Linux.

Problem Summary

When MetaDefender Core runs on Kubernetes without persisting output directories, files produced by quarantine, DLP, and Content Disarm and Reconstruction (CDR) are lost after pod restarts, reschedules, or scale-in/out events. This occurs because ephemeral container filesystems do not survive lifecycle changes.

What Not to Do

Avoid manually mounting internal Core application paths such as `/opt/ometascan/core_data/var/lib/ometascan/`. While it may appear to work initially, it can cause instability, misconfiguration, and operational issues over time. Do not rely on the `DATA_DIR` variable; it is not supported for this purpose and will be removed in the future.

Recommended Approach

Use a dedicated, persistent storage path exposed to the pod via a PersistentVolumeClaim (PVC), and instruct MetaDefender Core to place its output there using a new environment variable `STORAGE_PATH`. This cleanly separates app internals from persistent data and works reliably across pod restarts.

Design Overview

- Create or reference a PVC sized for your expected workload (quarantine, DLP artifacts, sanitized output, SBOMs).
- Mount the PVC into the MetaDefender Core pod at a path reserved for outputs (example: `/home/metascan/`).
- Set the environment variable `STORAGE_PATH` to the same path (example: `/home/metascan/`).
- Core organizes data by pod instance under this directory, ensuring all output persists beyond pod lifecycle events.

Benefits

- Prevents loss of quarantine, DLP, sanitized, and SBOM files after pod restarts.
- Avoids brittle mounts into internal app paths.
- Simplifies troubleshooting and auditing by keeping pod-specific subfolders.

High-Level Implementation Steps

1) Plan Capacity and Retention

- Estimate daily volume of quarantined, DLP, sanitized, and SBOM artifacts.
- Align PVC size and storage class performance (IOPS/throughput) to SLA and retention goals.
- Plan a rotation/cleanup policy.

2) Create or Select a PersistentVolumeClaim

Provision a PVC using your Kubernetes platform's standard process. At minimum define:

- Storage class (backed by your CSI driver)
- Requested capacity
- Access mode compatible with your deployment pattern (e.g., ReadWriteOnce or ReadWriteMany as your storage supports)

Details vary by platform; use your provider's documentation for PVC creation and binding.

3) Mount the PVC in the Pod and Set STORAGE_PATH

In your deployment (YAML/Helm/Operator), add:

- A volume sourced from your PVC
- A persistentDir and storage_name `/home/metascan/` (examples) under core components
- An environment variable `STORAGE_PATH=/home/metascan/`

Code

```
pvc:
  - name:  opswat-mdcore-volume   # The name of the PVC.
    enabled: true
    labels:
      app: mdcore-opswat          # Key-value pairs for categorizing and selecting resources.
    annotations:
    accessModes:                  # Specifies the desired access modes for the volume. (Common
modes ReadWriteOnce, ReadOnlyMany and ReadWriteMany)
    - ReadWriteOnce
    resources:
      requests:                   # The amount of storage requested.
        storage: 100Gi
    storageClassName: local-path  # The name of the StorageClass to use for dynamic
provisioning. If omitted, the default StorageClass is used.
    volumeMode: Filesystem        # Defines whether the volume is presented as a filesystem
(Filesystem) or a raw block device (Block).

core_components:
  md-core:
    name: md-core
    env:
      - name: STORAGE_PATH
        value: /home/metascan    # A path inside MDCore pod
    persistentDir: /home/metascan
    storage_name: opswat-mdcore-volume
```

Notes:

- Provided sample values in the above. Use values that are valid for your current system.

- Do not mount to internal app directories; use a clean path like `/home/metascan/`.

What You Should See After Startup

When the pod is running with this PVC attached, the `STORAGE_PATH` directory will include subdirectories for each pod instance name as well as shared artifacts. For example (the following are ls commands run from inside a running pod):

Code

```
root@md-core-6554dc5f5c-6cmlg:/# ls /home/metascan/
docker_upgrade_db.log  md-core-6554dc5f5c-6cmlg  md-core-6554dc5f5c-8vnkb  md-core-6554dc5f5c-
ww26l  sharedb
root@md-core-6554dc5f5c-6cmlg:/# ls /home/metascan/md-core-6554dc5f5c-6cmlg/
dlp  quarantine  sanitized  sbom_report
root@md-core-6554dc5f5c-6cmlg:/# ls /home/metascan/md-core-6554dc5f5c-6cmlg/sanitized/a8/4b/
a84bcc759e6bfac668b9aa7b46d70b243c26f2561cdfd0d474a78653633d4982
/home/metascan/md-core-6554dc5f5c-6cmlg/sanitized/a8/4b/
a84bcc759e6bfac668b9aa7b46d70b243c26f2561cdfd0d474a78653633d4982
root@md-core-6554dc5f5c-6cmlg:/#
```

This reflects:

- Per-pod folders for running and previously deleted pods—data persists across restarts.
- Directories for DLP output, quarantine, sanitized files, and SBOM reports under each pod's folder.

Summary:

Use `STORAGE_PATH` to direct Core's output to a PVC-backed directory (e.g., `/home/metascan/`). Do not mount internal app directories or use the `DATA_DIR` variable. This approach ensures quarantine, DLP, sanitized, and SBOM files persist across pod restarts.

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer

Why do some multipart uploads fail with error code 500?

Check Your Version:

This article applies to MetaDefender Core releases versions 5.17.1 and earlier deployed on Windows and Linux systems.

1. Issue description

When uploading large files via API, using /file/multipart requests with chunks that are multiple of 1MB, those chunks fail with error code 500.

2. Steps to confirm issue

In the custom script used to split the uploaded file in to chunks, verify the configured size for the split chunks. The upload fails when the chunk size is a multiple of exactly 1MB (i.e. 1MB, 7MB, 20MB etc).

The Core logs output a [WARNING] containing (core.workflow) Empty data chunk received, stop data transfer and NGINX shows a code 500 response for the upload.

3. Solution

The correct and persistent fix is udpating MetaDefender Core to version 5.18.0 or later.

4. Workaround

If updating MetaDefender is not feasible at the time, a temporary workaround is adapting the uploading script to split the files in chunks that are not equal to a multiple of 1MB, for example adding or substracting 1KB from the chunk size.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

How to Remove Infected Child Files from Archives in MetaDefender Core

Use case

Core scans an archive or container file, and one or more of the child files inside the archive are detected as infected. The desired result is to return a modified archive where the infected child file has been removed, rather than returning the original archive with the infected child file(s) sanitized.

Resolution

To remove infected child files from an archive, configure the workflow's Compression and Deep CDR settings:

Workflow area	Setting	Value
Deep CDR	Enable Deep CDR	Status: Active
Compression	Enable archive sanitization	Enabled
Compression	Enable for archive compression filetypes	Select the required archive/ container types

Workflow area	Setting	Value
Compression	Skip Malicious Items	Enabled; configure the verdicts that should be treated as malicious
Compression	Include sanitized version of blocked child files	Disabled
Compression	Add tombstone file during archive process / archive sanitization	Optional, recommended for keeping track of infected files
Deep CDR	Enable for filetypes	Optional: enable for supported child file types if clean child files should be sanitized

Why does MetaDefender Core fail to trust a self-signed certificate in Kubernetes?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Kubernetes versions listed here.

When MetaDefender Core connects to an HTTPS application, it validates the SSL/TLS certificate presented by the remote endpoint. If that application uses a self-signed certificate, or a certificate issued by an internal Certificate Authority that MetaDefender Core does not trust, the connection can fail during certificate validation.

A typical error looks similar to this:

Code

```
[WARNING] 2026.07.14 10:15:28.608: (common.communication) SSL error,
error='The certificate is self-signed, and untrusted',
errorCode='"The certificate is self-signed, and untrusted"' [msgid: 497]
```

This article explains why the error occurs and how to configure MetaDefender Core running in Kubernetes to trust the required certificate by mounting it into the container from a Kubernetes Secret.

Use the correct certificate for the trust relationship. If the remote application is truly self-signed, use that certificate. If the remote application was issued by a private or internal CA, trust the relevant CA certificate or chain instead of trusting only the leaf certificate.

Symptoms

You may observe one or more of the following:

- MetaDefender Core cannot connect to an HTTPS application that uses a self-signed or privately issued certificate.
- Logs show an SSL/TLS validation failure stating that the certificate is self-signed and untrusted.
- Connectivity tests to the same endpoint succeed only when certificate validation is bypassed outside MetaDefender Core.

Resolution

To make MetaDefender Core trust the certificate in Kubernetes, store the certificate in a Kubernetes Secret and mount it into the MetaDefender Core pod as a file.

1. Select the correct certificate

Choose the certificate based on how the remote endpoint was issued:

- **Self-signed endpoint certificate:** use that self-signed certificate.
- **Certificate issued by an internal or private CA:** use the applicable CA certificate, which may include the root CA and one or more intermediate CA certificates.

The remote application should also present the full chain during the TLS handshake where applicable.

2. Prepare the certificate file

Ensure the certificate file meets these requirements:

- PEM format
- `.crt` file extension
- Contains only the public certificate data
- Does not include the private key

A PEM certificate normally looks like this:

Code

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

In the example below, the file name is `selfsigned.crt`.

3. Create the Kubernetes Secret

Create a Secret in the same namespace where MetaDefender Core is deployed:

Code

```
kubectl create secret generic selfsigned \
  --from-file=selfsigned.crt=./selfsigned.crt \
  --namespace <mdcore-namespace>
```

In this command, `selfsigned.crt=./selfsigned.crt` means:

- `selfsigned.crt` before the equals sign is the key name stored in the Secret.
- `./selfsigned.crt` after the equals sign is the local file path.

You can verify the Secret contents with either of the following commands:

Code

```
kubectl get secret selfsigned \
  --namespace <mdcore-namespace> \
  -o jsonpath='{.data}'
```

Code

```
kubectl get secret selfsigned \
  --namespace <mdcore-namespace> \
  -o json | jq -r '.data | keys[]'
```

The output should include `selfsigned.crt`.

4. Mount the Secret into the MetaDefender Core pod

Update the MetaDefender Core Helm values to define an extra volume sourced from the Secret:

Code

```
extraVolumes:
  - name: selfsigned
    secret:
      secretName: selfsigned
      items:
        - key: selfsigned.crt
          path: selfsigned.crt
```

Then mount that file into the container:

Code

```
extraVolumeMounts:
```

```
  - name: selfsigned
    mountPath: /etc/ssl/certs/selfsigned.crt
    subPath: selfsigned.crt
    readOnly: true
```

Complete example:

Code

```
extraVolumes:
  - name: selfsigned
    secret:
      secretName: selfsigned
      items:
        - key: selfsigned.crt
          path: selfsigned.crt

extraVolumeMounts:
  - name: selfsigned
    mountPath: /etc/ssl/certs/selfsigned.crt
    subPath: selfsigned.crt
    readOnly: true
```

Additional mount required for LDAP/AD directory connections: The mount path `/etc/ssl/certs/selfsigned.crt` is sufficient for most MetaDefender Core HTTPS features. However, the **Add directory** feature with type **LDAP/AD** reads certificates from the system CA bundle file rather than individual `.crt` files in the directory. If you use LDAP/AD directory integration with a self-signed or internally issued certificate, you must also mount the certificate into the system CA bundle path. See Step 4a below.

4a. Additional mount for LDAP/AD directory connections

MetaDefender Core's LDAP/AD directory feature validates certificates against the system CA bundle file rather than scanning individual certificate files in `/etc/ssl/certs/`. If you configure an LDAP/AD directory that uses a self-signed or privately issued certificate, the certificate must also be present in the system CA bundle.

The relevant CA bundle path depends on the container base image:

Container base image	CA bundle path
CentOS / RHEL (primary for MetaDefender Core)	`/etc/pki/tls/certs/ca-bundle.crt`
Debian / Ubuntu	`/etc/ssl/certs/ca-certificates.crt`

The Secret key name, the `items[].key` value, the `items[].path` value, and the `subPath` value must match the

intended mounted filename. If they do not match, Kubernetes can fail the mount with an error such as `references non-existent secret key`.

5. Apply the Helm change

Apply the updated values using the Helm upgrade command used in your environment. Example:

Code

```
helm upgrade <release-name> <chart-name> \
  --namespace <mdcore-namespace> \
  --reuse-values \
  -f values.yaml
```

If the values were updated but the pod was not recreated yet, restart the deployment:

Code

```
kubectl rollout restart deployment/<mdcore-deployment> \
  --namespace <mdcore-namespace>
```

Then monitor rollout progress:

Code

```
kubectl rollout status deployment/<mdcore-deployment> \
  --namespace <mdcore-namespace>
```

6. Verify the certificate mount

After the new pod starts, confirm the file exists in the container.

Code

```
kubectl get pods --namespace <mdcore-namespace>
```

Code

```
kubectl exec \
  --namespace <mdcore-namespace> \
  <mdcore-pod-name> \
  -- ls -l /etc/ssl/certs/selfsigned.crt
```

The file should be present at `/etc/ssl/certs/selfsigned.crt`.

7. Test the connection again

Retry the MetaDefender Core connection to the remote HTTPS application. After the certificate is mounted correctly and the pod is running with the updated configuration, the previous SSL error should no longer appear.

You can also test directly:

Code

```
kubectl exec \
  --namespace <mdcore-namespace> \
  <mdcore-pod-name> \
  -- curl -v https://<application-hostname>:<port>/
```

Use the same hostname and port that MetaDefender Core uses. This is important because TLS hostname validation depends on the server name used for the connection.

Important considerations

- **Virtual machine deployments:** if MetaDefender Core is installed directly on a VM instead of Kubernetes, add the certificate to the operating system trust store, restart the MetaDefender Core service, and test again.
- **Use of Secrets vs ConfigMaps:** a public certificate or CA certificate does not normally contain sensitive material, so a ConfigMap can also be technically valid for distribution. A Secret may still be preferred to align with internal security policy or deployment standards.
- **subPath behavior:** mounting a single file with `subPath` preserves the rest of the existing `/etc/ssl/certs` directory. This avoids replacing the whole directory with only Secret contents.
- **Certificate updates:** when a Secret-backed file is mounted using `subPath`, updates are not automatically reflected inside an already running container. Restart the pod or rollout the deployment after certificate changes.
- **LDAP/AD directory trust path:** the LDAP/AD directory feature in MetaDefender Core reads from the system CA bundle file, not from individual files in `/etc/ssl/certs/`. If you use LDAP/AD with a self-signed or internal CA certificate, follow Step 4a to mount the certificate into the CA bundle path.
- **Private key handling:** do not include the certificate's private key in the trust file. Trust configuration requires only the public certificate or CA certificate.

FAQ: Can I use a ConfigMap instead of a Secret for the certificate?

> *Yes, for a public certificate or CA certificate, a ConfigMap is generally technically sufficient because the content is not normally confidential. However, many organizations standardize on Secrets for certificates, and MetaDefender Core can use either pattern if the file is mounted correctly.*

FAQ: What certificate should I use if the remote server certificate was issued by an internal CA?

> *Use the relevant CA certificate or certificate chain rather than trusting only the leaf server certificate. In many environments that means the root CA certificate and, if applicable, one or more intermediate CA certificates.*

FAQ: Why does the LDAP/AD directory connection still fail after mounting the certificate?

*The **Add directory** feature with type **LDAP/AD** reads the system CA bundle file, `/etc/pki/tls/certs/ca-bundle.crt` on CentOS/RHEL or `/etc/ssl/certs/ca-certificates.crt` on Debian, rather than individual `.crt` files in `/etc/ssl/certs/`. Ensure the certificate is appended to the CA bundle as described in Step 4a. Restart the pod after making changes.*

FAQ: Will the certificate update automatically if I replace the Secret?

Not reliably when the file is mounted via `subPath`. Restart or recreate the pod after updating the Secret so the new certificate file is picked up.

FAQ: What if MetaDefender Core is installed on a virtual machine instead of Kubernetes?

Install the certificate into the operating system trust store used by that host, restart the MetaDefender Core service, and then test the HTTPS connection again.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

How can I fix a failed or looping MetaDefender Core module update in MyOPSWAT Central Management v10 by clearing the package cache?

Check Your Version:

This article applies to all MyOPSWAT Central Management v10 releases deployed on Windows systems.

When should you use this?

- Modules under **Settings** → **Update Package** → **Modules** fail, stall, or loop.
- You want My OCM v10 to re-download **all module packages** from scratch.

What this does?

Clears OCM's local **package cache** and restarts the service so OCM repopulates it with fresh packages. Your OCM configuration and product inventory remain unchanged.

Steps required:

1. Change the update source to **Manual**Open Settings → Update Package → Modules → Source → set it to manual → click Save
2. **Stop the service**Open **Services** → find **My OPSWAT Central Management** → right-click **Stop**. *(Run Services as*

Administrator.)

3. **Clear cached packages**Delete the contents of:`C:\Program Files\OPSWAT\MOCM\data-agent\fusion-updater` `C:\Program Files\OPSWAT\MOCM\data-agent\mdcore`*(Delete files inside only; keep the folder.)*
4. **Start the service**In **Services**, right-click **My OPSWAT Central Management** → **Start**.*(Run Services as Administrator.)*
5. Change the update source to **Internet**Open Settings → Update Package → Modules → Source → set it to Internet → click Save
6. **Trigger module updates**After saving the settings, within the same page, press the **Update Now** button.*(The modules list might still be populated with the old versions even after deleting the cached files, but they will get updated once finishing this step.)*
7. **Verify**Confirm all module packages download and show **Up to date** status.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with one of our support engineers.

Why does custom detection for FileType not work?

Check Your Version: This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

The custom detection is only supported from File Type version 7.4 and later

Issue

Custom detection for a filetype is enabled, but the file is still detected as Unknown (Data).

Possible causes and solutions

1. **Incorrect path to the custom XML rule file**Verify that the path to your custom rule is correct at **Inventory > Modules > Utilities > FileType**, section **Enable custom detection**
2. **Settings not saved**After enabling custom detection and adding rule(s), ensure you click **Save** in the MetaDefender Core UI.
3. **Filetype engine not restarted**Restart the FileType engine for changes to take effect. You can do this by disabling and then re-enabling the FileType engine in the UI.
4. **Invalid or incorrect custom rule**Review your XML rule file to confirm that the syntax and matching criteria are correct. Check these documents for more details.
    1. Custom detection - MetaDefender Core
    2. How to add custom detection for a file type? - MetaDefender Core

5. **Built-in rule taking priority**

- A file may match both a custom rule and a built-in FileType rule
- To ensure the custom rule is used, assign it a high confidence score. Please see more details at Custom detection - MetaDefender Core
  - The “detection score” can be found in the JSON scan result: `filetype_info.file_info.likely_type_ids.score`

Support: If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why REST API File Submission with downloadfrom Header Fails When Using RAMDISK for Temp Directory?

Check Your Version: This article applies to all MetaDefender Core releases deployed on Windows and Linux systems after 5.11.1.

Issue:

When configuring a RAMDISK as the temporary directory using IMDisk, file submissions through the browser interface function correctly; however, submissions via the REST API fail due to issues with the `downloadfrom` header.

Error Details:

The following error is logged:

> *[WARNING] 2025.11.03 20:50:22.994: (common.workflow) Workflow block ended with exception, block='ErrorRestResponse', workflow_id='', error='Cannot send message to Frontend' [msgid: 97]*

You may encounter the message 'Cannot send message to Frontend' in `core.log`, while the REST API request returns an error: `“err”: “File download rejected due to insufficient disk space”`.

Resolution:

Until a fix is available, consider using a physical disk path for the temporary directory instead of a RAMDISK when submitting files via the REST API with the `downloadfrom` header.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why do I get the "The cert file is not valid" or "the file does not exist" when adding my cert?

Check Your Version:

This article applies to all MetaDefender Core releases after v5.8.0 on both Windows and Linux systems.

When adding a certificate to MetaDefender Core, you may encounter the error: “The cert file is not valid or the file does not exist.” This error can occur for a variety of reasons, often related to file type mismatches, missing passphrases, permission issues, or problems with the certificate’s content and chain. By understanding the common causes, you can more efficiently troubleshoot and ensure your certificate is properly recognized by MetaDefender Core.

Possible Causes & Solutions

1. Incorrect File Type or Mismatch

- Verify you’re using the correct SSL file types (e.g., .crt for certificate and .key for private key).
- Attempting to upload a PFX bundle (or other certificate formats) will cause an error.
  - MetaDefender Core requires a Base64-encoded file that contains only the leaf certificate, ideally without additional chain or encoding variations.
- Ensure the certificate and key match—mismatched pairs can cause validation errors. You can verify via OpenSSL hashes

2. Missing Passphrase File (if applicable)

- If your private key is encrypted, you must supply the passphrase—typically via a separate passphrase file.
- There’s an optional "Password File" field during certificate upload. If omitted for an encrypted key, it will cause failure. OPSWAT

3. Location & Filesystem Permissions (Linux)

- Confirm that both .crt and .key files are correctly placed in /etc/ometascan/nginx.d.
- They must be owned by the metascan user, with read/write permissions for that user.
- If these files are missing, inaccessible, or have incorrect ownership/permissions, it results in the “not valid or does not exist” error.

4. Certificate Content & Chain Issues

- Ensure the certificate’s **Subject** or **Subject Alternative Name** contains the server’s FQDN or IP.

- If it's a wildcard certificate, it must match the server's domain.
- Confirm that the full certificate chain (intermediate and root CAs) is trusted on the system—missing or untrusted chain members can break the validation process.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why are MetaDefender Core AppSec Engines Failing to Load After RHEL 8 to RHEL 9 Upgrade?

Applies to MetaDefender Core 5.x on **RHEL 9** after in-place upgrade from **RHEL 8**, especially hardened builds where engine paths may inherit `noexec`.

Summary

Post-upgrade, some Core servers fail to start AppSec engines. The `engines` directory may be empty or engines fail self-tests with errors like:

- `Wait for hello message timeout`
- `failed to map segment from shared object`

Primary cause: the filesystem hosting engines is mounted with `noexec`. Contributing factors: missing RHEL 9 libraries and SELinux denials.

Symptoms

- Engines don't load / self-tests fail.
- `engines` folder not repopulating after updates or manual copy.
- Errors as above in UI/logs.

Diagnosis

1. **Mount options**

Code · bash

```
ENGINES_DIR=/var/lib/ometascan/engines
findmnt -T "$ENGINES_DIR"
grep -nE ' /var |/ometascan|/engines' /etc/fstab
```

If you see `noexec`, engines can't run.

2. **SELinux status/denials**

```bash
getenforce
ausearch -m avc -ts recent | tail -n 50
```

3. **RHEL 9 deps present?**

```bash
rpm -q libnsl krb5-libs libicu openssl-libs zlib compat-openssl11 || true
```

Resolution

A. Remove `noexec` from engines filesystem

- Edit `/etc/fstab` to drop `noexec` for the partition backing engines (e.g., `/var`).
- Remount:

```bash
sudo mount -o remount,exec /var
findmnt -T /var/lib/ometascan/engines
```

If policy requires `noexec` on `/var`, move engines to a dedicated `exec` mount and bind-mount back.

B. Install required libraries

```bash
sudo dnf install -y libnsl krb5-libs libicu openssl-libs zlib compat-openssl11
```

C. Address SELinux (only if AVCs present)

- Temporary for troubleshooting:

```bash
sudo setenforce 0   # revert with: sudo setenforce 1
```

- Permanent policy:

Code · bash

```
ausearch -m avc -ts recent | audit2allow -M mdcore-engines
sudo semodule -i mdcore-engines.pp
sudo restorecon -Rv /var/lib/ometascan/engines
```

Verification

- `engines` directory repopulates.
- Engine self-tests pass; logs show successful startup/handshake.
- No new SELinux AVCs; mount for engines shows `exec`.

Best Practices

- Keep engines on a filesystem mounted with `exec`; bind-mount if `/var` must be `noexec`.
- Preinstall RHEL 9 runtime libraries on Core nodes.
- Validate SELinux contexts/policies on hardened builds.

Support:

If Further Assistance is required, please proceed to log a [support case or chat with our support engineer at https://my.opswat.com/support.

What is the difference between "fail" and "cancelled" scan result codes in MetaDefender Core?

Check Your Version: This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue: Customers are sometimes confused about the difference between the "fail" and "cancelled" scan result codes in MetaDefender Core scan results.

Resolution: Here's a breakdown of the two scan result codes:

- **Fail:** This means the scan completed, but a problem was detected. This could be due to malware being found or a processing failure during the scan.
- **Cancelled:** This means the scan was stopped before it could finish. Therefore, no conclusive result is available.

For further information, please refer to the official documentation: Description of scan result codes

Support: If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why Is a MetaDefender Core Node Stuck in Starting State in a Shared-Database Cluster?

Info:

- This article applies to MetaDefender Core in **shared-database (multi-node)** deployments running versions **prior to 5.21.0**.

Issue

In a multi-node MetaDefender Core cluster sharing a single PostgreSQL database, a node may remain stuck in **Starting** state for an extended period, potentially hours, after being restarted. The issue is caused by a database deadlock between the restarting node's one-time startup privilege-grant operation and the background statistics refresh job (`warehouse.refresh`) running on other active nodes.

Symptoms

- **Node stuck at "Starting":** The restarted Core node does not transition to a healthy or running state. The web UI and REST API remain unavailable on that node.
- **Extended startup duration:** The node may remain in Starting state for minutes to hours, depending on the load on the other nodes.
- **No scanning impact on other nodes:** File scanning on the remaining active nodes continues normally. The issue is isolated to the restarting node.
- **Deadlock errors in PostgreSQL logs:** Repeated `ERROR: deadlock detected` entries show contention between the startup `GRANT` statement and `warehouse.refresh()`.

Evidence

PostgreSQL log pattern

Code · sql

```
ERROR:  deadlock detected
DETAIL:  Process <PID_A> waits for ShareLock on transaction <TXN_1>; blocked by process <PID_B>.
         Process <PID_B> waits for ShareLock on transaction <TXN_2>; blocked by process <PID_A>.
         Process <PID_A>: GRANT ALL PRIVILEGES ON SCHEMA pg_catalog TO <user>; COMMIT; ...
```

```
Process <PID_B>: SELECT * FROM warehouse.refresh(<instance_id>::SMALLINT);
```

Core application log pattern

Code

```
CEF:0|OPSWAT|MSCW|<version>|common.base|==== START ====|2|OMSversion=<version> OMSmsgid=356
CEF:0|OPSWAT|MSCW|<version>|core.postgres|Postgres info|2|OMSinfo=Checking for database upgrade
OMSmsgid=4085
```

The node logs `==== START ====` but never progresses to a ready state. The deadlock repeats on each retry cycle until the database happens to cancel the `warehouse.refresh` process instead of the startup process.

Root Cause

When a MetaDefender Core node starts up, it performs a one-time database initialization as part of its startup sequence. At the same time, other active nodes in the cluster continuously run background maintenance jobs to keep internal statistics and dashboards up to date.

In versions prior to 5.21.0, these two operations can conflict when they run at the same time, causing the database to cancel the startup transaction. The node then retries, but under sustained cluster activity, the conflict can repeat — keeping the node stuck in Starting state.

This is a known limitation of the shared-database architecture in affected versions and is resolved by upgrading to 5.21.0 or later

Resolution

Upgrade to **MetaDefender Core 5.21.0** or later, which resolves the startup-versus-refresh deadlock at the application level. No manual workaround is needed after upgrading.

> Core 5.21.0 introduces significant UI/UX changes. Plan accordingly.

Prevention

- Schedule node restarts during low-load windows to minimize the chance of the background refresh job running concurrently with the startup grant.
- Monitor scan queue depth on remaining nodes before restarting a node. Sustained high queue depth increases the deadlock probability.

> **Support:**
>
> If you require further assistance, please follow these instructions on How to Create Support Package? before creating a support case or chatting with our support engineer.

Why is MetaDefender Core slow and returning NGINX 499 errors?

Info:

- This article applies to all MD Core.

Issue

Customers may experience severe performance degradation in MetaDefender Core, characterized by frequent request timeouts and NGINX 499 errors. This typically occurs when a client application generates a high volume of login requests without corresponding logouts or session expiration.

Symptoms

- **NGINX 499 Errors:** Client applications close the connection after waiting for a response, typically at a 15-second threshold.
- **Request Latency:** Backend processing for simple data fetches exceeds normal thresholds due to database overhead.
- **Repeated Timeouts:** The same `data_id` may fail multiple times as the client retries the timed-out operation.

Evidence

1. **NGINX Log Analysis**

   The NGINX logs show the client terminating the connection (499) after exceeding a 15-second threshold. This delay occurs specifically during the request to fetch scan results, even though the backend processing for the file itself was successful.

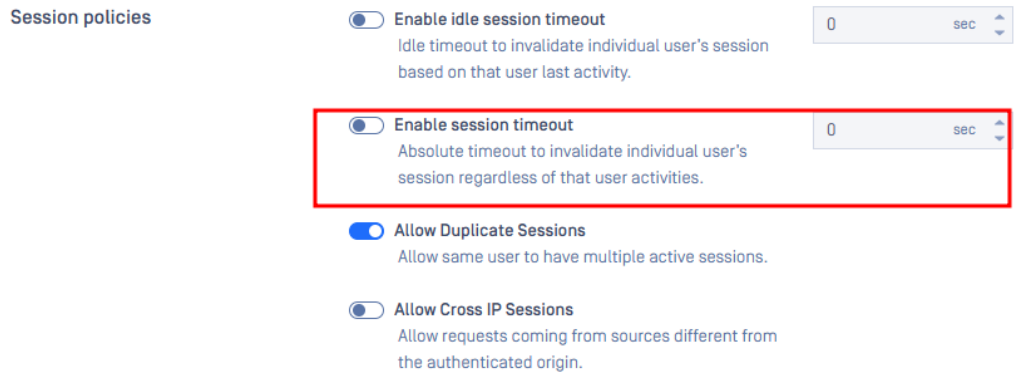
   Example Log:

   Code

   ```
   XX.XXX.XX.XX - - [11/Dec/2025:22:38:04 +0000] "GET /file/f8613d4e9e9247b4be5673e50895fd28
   HTTP/1.1" 499 0 "-" "Apache-HttpClient/5.2.3 (Java/17.0.16)" 15.012
   XX.XXX.XX.XX - - [18/Dec/2025:16:22:45 +0000] "GET /file/73e0ff8426d64bea97ce1101ad31e58e
   HTTP/1.1" 499 0 "-" "Apache-HttpClient/5.2.3 (Java/17.0.16)" 14.995
   XX.XXX.XX.XX - - [18/Dec/2025:16:23:01 +0000] "GET /file/73e0ff8426d64bea97ce1101ad31e58e
   HTTP/1.1" 499 0 "-" "Apache-HttpClient/5.2.3 (Java/17.0.16)" 15.012
   ```

2. **Session Management Settings**

   The issue is linked to the following configuration where session timeouts are disabled:



3. **Login/Logout Imbalance**

   There is a massive discrepancy between the number of sessions created and those explicitly closed.

- **Total Login Requests:** Approximately **1,000,000**.
- **Total Logout Requests:** Only **6,765 matches** found in the logs.
- **Log Evidence of Discrepancy:** The logs illustrate a high volume of file processing requests (logins) compared to very few successful logouts:
- **Login/Session Activity:** `... "POST /file HTTP/1.1" 200 57 ...`
- **Logout Example:**

```
Code
... "POST /logout HTTP/1.1" 200 40 "-" "Apache-HttpClient/5.2.3 (Java/17.0.17)" 0.125
... "POST /logout HTTP/1.1" 200 40 "-" "-" 0.156
```

Root Cause

With session timeouts disabled, session IDs never expire and are stored forever. Over several days of continuous operation, the high volume of login requests causes the number of session IDs to increase indefinitely. This creates a significant bottleneck during session validation, causing a slowdown for every request that requires a session check—most notably `GET /file/data_id`.

Resolution

To resolve the bottleneck and restore performance, session management must be configured to allow for the purging of old data.

Step 1: Enable Session Timeouts

Re-enable automatic session cleanup in the MetaDefender Core Management Console:

- Go to **Settings > Security > Session Settings**.

- Set **Session Timeout** to a functional value (e.g., 60 minutes).
- Set **Absolute Session Timeout** to a functional value (e.g., 1440 minutes / 24 hours).

Step 2: Update Client Integration

Ensure the client application follows best practices:

- **Reuse Session IDs:** Do not call `POST /login` for every request. Login once and reuse the ID for the duration of the session.
- **Explicit Logout:** Call `POST /logout` when the application has finished its task.

Step 3: Service Restart (If Necessary)

If performance does not improve immediately after changing settings, restart the MetaDefender Core service to force a refresh of the session cache and trigger the initial cleanup of the oversized session table.

Prevention

Ensure the client application follows best practices:

- **Reuse Session IDs:** Do not call `POST /login` for every request. Login once and reuse the ID for the duration of the session.
- **Explicit Logout:** Call `POST /logout` when the application has finished its task.

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

I enabled Secure API in MetaDefender Core but didn't save the API key. How do I reset it?

Check Your Version: This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Summary:

When you enable the Secure API setting, MetaDefender Core displays the API key once and warns that it cannot be retrieved again. That warning is accurate but narrow: you cannot recover the exact value that was shown, but you are not locked out.

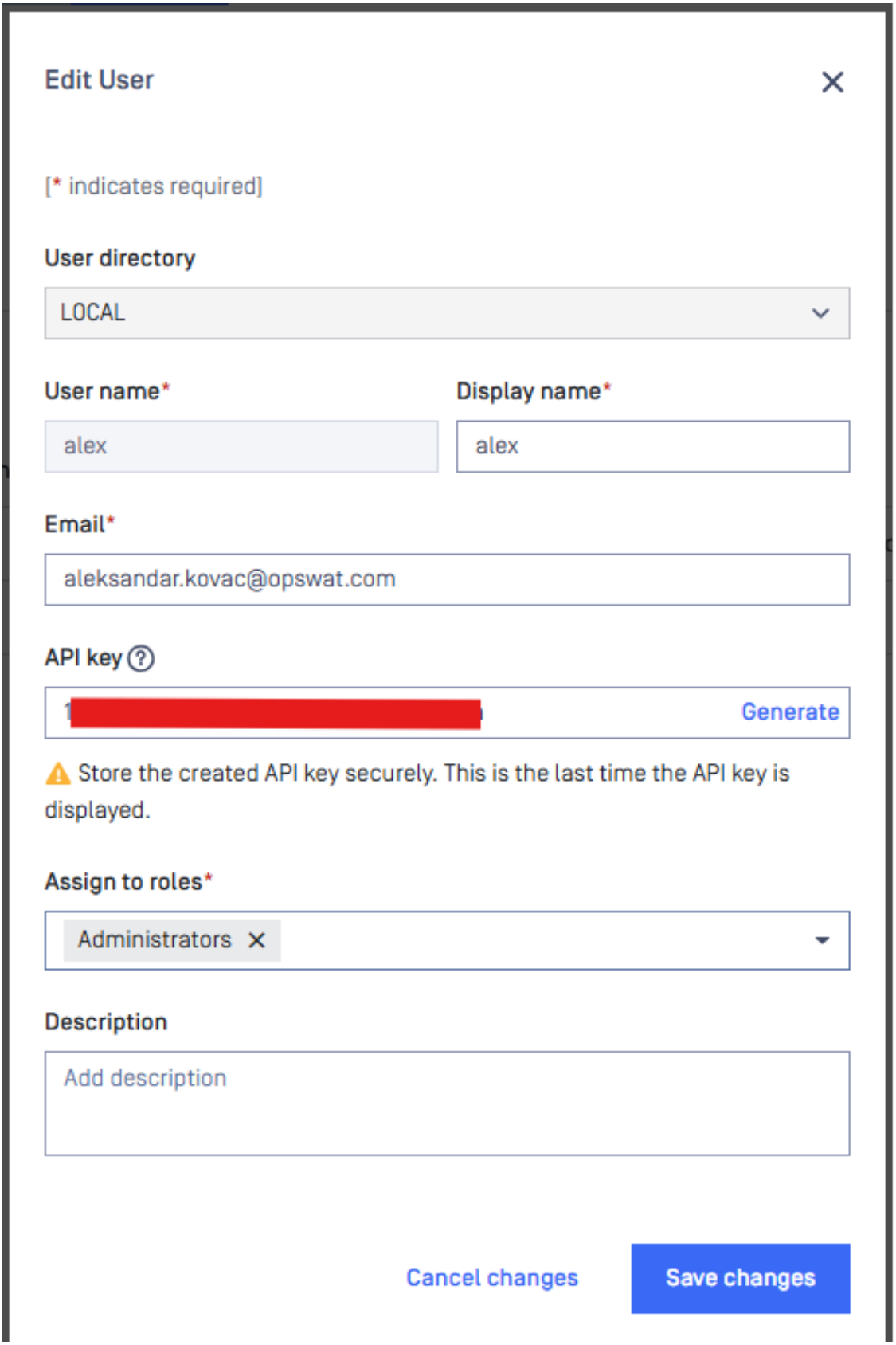
The API key is bound to a user account, so you can simply generate a new key for that user. Generating a new key

replaces the old one, and the lost value becomes irrelevant.

Resolution

To generate a new API key:

1. Open the MetaDefender Core console and go to User Management > Users and Groups.
2. Select the user account you wish to manage the API key for.
3. In the API key section click Generate.
4. Click Save changes for the new key to take effect.
5. Copy the new key and store it somewhere safe before leaving the page.



The previous key is invalidated as soon as the new one is saved, so any integration still using the old value will need to be updated with the new key.

Support:

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

Why does the MetaDefender Core Management Console display a blank page after an upgrade?

Check Your Version:

This article applies to MetaDefender Core All MetaDefender Core releases deployed on Windows and Linux.

Phenomenon

After upgrading MetaDefender Core, users may encounter a completely blank page when accessing the Management Console.

This issue is commonly reported immediately after an upgrade, leading users to believe that the upgrade has failed. However, in many cases, the MetaDefender Core services have started successfully and the issue is limited to the client browser.

Typical symptoms include:

- The Management Console displays a completely blank page.
- Core services are running normally.
- The upgrade completes successfully.
- Browser Developer Tools (F12 → Console) may show JavaScript syntax errors similar to:

```bash
Uncaught SyntaxError: Unexpected token '{'
```

Main Cause

The Management Console of newer MetaDefender Core versions uses modern JavaScript features that require a supported, up-to-date web browser.

If the browser version does not meet the supported browser requirements, the browser may fail to parse the Management Console JavaScript, preventing the application from loading and resulting in a blank page.

Although the issue is first observed immediately after an upgrade, the upgrade itself is typically successful. The problem is caused by browser compatibility rather than a failure of the MetaDefender Core server.

For the supported browser requirements, please refer to:

Recommended System Configuration - MetaDefender Core

Resolution

Step 1. Verify that MetaDefender Core is running

Confirm that the required MetaDefender Core services have started successfully.

On Windows

1. Open **Services** (`services.msc`).

2. Verify that the following services are in the **Running** state:

- **OPSWAT MetaDefender Core**
- **OPSWAT MetaDefender Core PostgreSQL**

On Linux

Run the following commands:

Code · powershell

```
systemctl status ometascan
systemctl status ometascan-pg.service
```

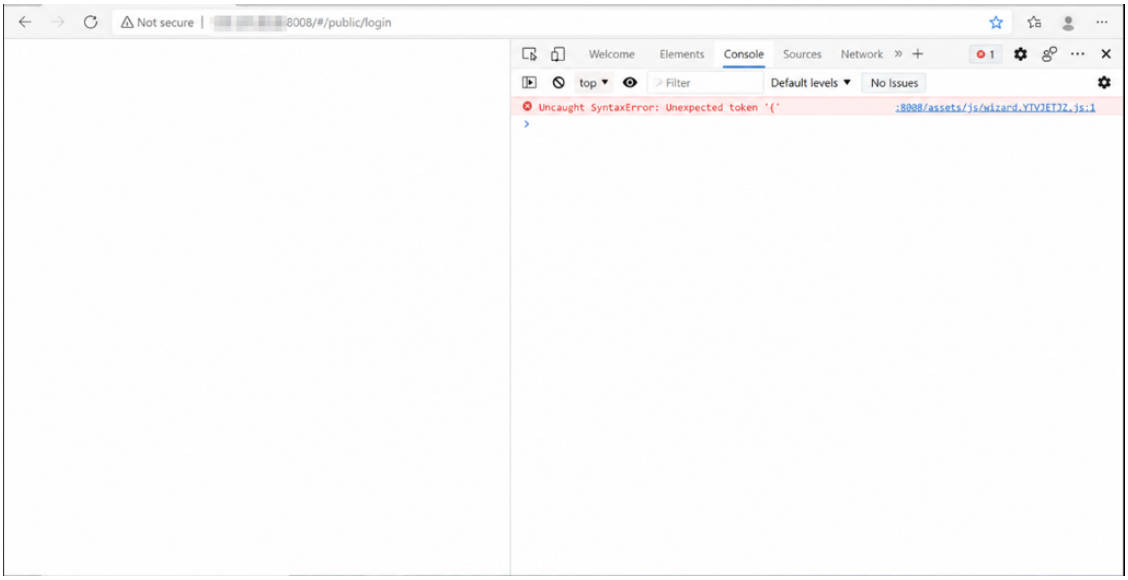
Verify that both services are **active (running)** before proceeding.

> **Note:** After starting or restarting MetaDefender Core, the services may require approximately **2–3 minutes** to fully initialize. If the Management Console is accessed while the services are still starting, a blank page may be displayed temporarily. This is expected behavior.
>
> Please wait a few minutes for the services to fully start, then refresh the Management Console. If the blank page persists, proceed with the browser troubleshooting steps below.

Step 2. Check the browser console

Open the browser Developer Tools by pressing **F12** and navigate to the **Console** tab.

If JavaScript syntax errors similar to the following are displayed:



this indicates that the browser may not support the JavaScript features required by the current Management Console.

Step 3. Verify the browser version

Confirm that the browser version satisfies the supported browser requirements documented by OPSWAT.

If the browser is outdated:

- Update Microsoft Edge or Google Chrome to the latest supported version.
- If the environment is offline, install the latest supported browser version using the vendor's offline installer.

Step 4. Perform a browser refresh

After updating the browser:

- Perform a hard refresh (**Ctrl + Shift + R**).
- If necessary, test using an InPrivate/Incognito window.
- If available, verify the behavior from another supported browser.

Notes

- This issue can easily be mistaken for an upgrade failure because it first appears immediately after upgrading MetaDefender Core.
- Browser-related display issues after an upgrade may also be caused by stale browser cache. If the browser version is supported, performing a hard refresh or testing from another supported browser is recommended as part of the initial troubleshooting process.

If Further Assistance is required, please proceed to log a support case or chat with our support engineer.

Why am I getting the error “No available rule is present for scanning” in MetaDefender Core?

Check Your Version

This article applies to MetaDefender Core All MetaDefender Core releases deployed on Windows and Linux.

Phenomenon

File submissions fail with the following error, even though the expected workflow is visible in the MetaDefender Core Management Console:

```
No available rule is present for scanning.
```

Main Cause

The incoming scan request does not match an available workflow. This may be related to an incorrect workflow/rule name or restrictions that prevent the request from using the workflow, such as role permissions, User-Agent conditions, or Security Zone settings.

Resolution

Step 1. Verify the workflow name

Navigate to:

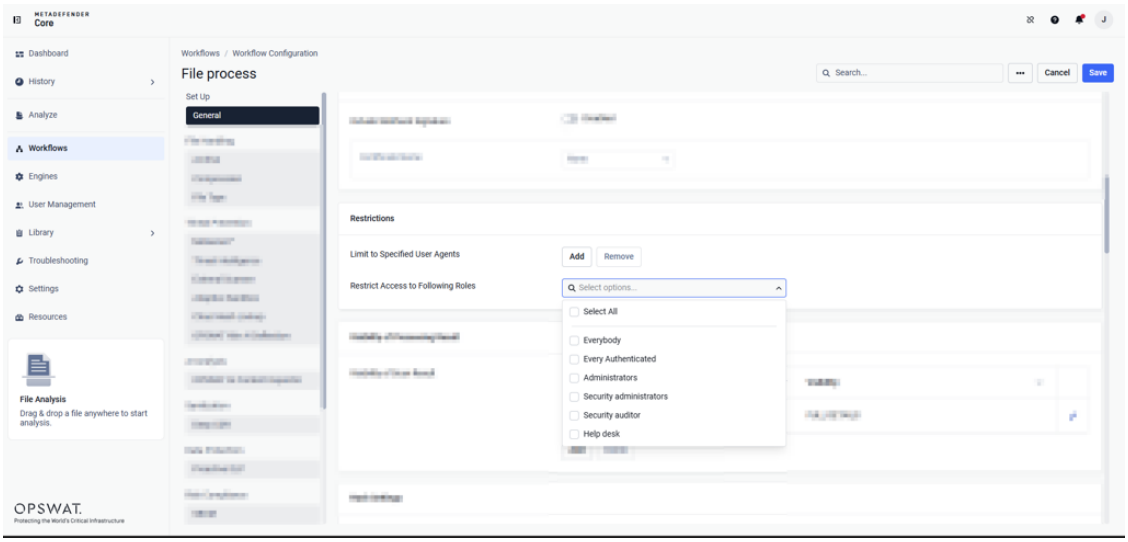
MetaDefender Core Workflow Management Workflows

Confirm that the expected workflow exists. If the application specifies a workflow using the `rule` header, ensure its value matches the workflow name exactly, including capitalization.

Step 2. Review workflow restrictions

Open the workflow and review **Restrict access to following roles**, along with any other configured restrictions.

Confirm that the application’s scan requests meet these conditions, and adjust the relevant settings where necessary.



Important: Removing restrictions can broaden access to the workflow. Any changes should follow your organization's security requirements. If restrictions are temporarily relaxed for testing, restore the appropriate access controls afterward.

Step 3. Test and compare the requests

Submit the same test file using Postman or `curl`, with the same workflow and authentication details as the application.

If direct submission succeeds but the application still fails, compare the endpoint, request headers, and authentication information. A successful direct API test does not, by itself, confirm a defect in the application or SDK.

Refer to the MetaDefender Core File Analysis API documentation.

After applying the changes, retry scanning through the original application and confirm that the error no longer occurs.

> If Further Assistance is required, please proceed to log a support case or chat with our support engineer.

Fixing 'cellimages.xml does not exist' Error in XLSX Files (WPS Office)

Overview

Some XLSX files created or edited with WPS Office fail in strict validators or CDR/sanitization pipelines with the error: "Invalid File Structure: Relationship file cellimages.xml does not exist." This occurs when the package references xl/cellimages.xml but the file is missing.

Symptoms

- Security or sanitization tool logs: "Invalid File Structure: Relationship file cellimages.xml does not exist"

- File was saved with WPS Office, or shows WPS-specific content types/namespaces
- Excel may open the file normally or prompt to repair

Why it happens

XLSX is a ZIP of XML parts. Affected files declare and reference a cell image map but omit the actual part, creating a dangling reference.

Typical signs inside the package:

- [Content_Types].xml contains an Override for /xl/cellimages.xml with a WPS content type
- xl/_rels/workbook.xml.rels contains a relationship of type http://www.wps.cn/officeDocument/2020/cellImage targeting cellimages.xml
- The physical file xl/cellimages.xml is absent

Quick fix (recommended): Open in Excel and re-save

1. Open the original XLSX in Microsoft Excel.
2. If prompted to repair, accept the repair.
3. File > Save As, and save a new copy (e.g., filename_fixed.xlsx).
4. Submit or use the newly saved file. Most workflows will now pass.

Why this works: Excel will remove or correct the dangling relationship so the package becomes structurally valid.

Manual fix (advanced): Add a minimal xl/cellimages.xml

Use this if you cannot access Excel or need a deterministic repair for a single file.

1. Make a working copy of the XLSX file. Rename the copy from .xlsx to .zip.
2. Extract the ZIP. Navigate to the xl directory.
3. Create a new file named cellimages.xml inside xl with the minimal valid content below.
4. Repack: Select all extracted root items ([Content_Types].xml, _rels, docProps, xl, etc.), ZIP them (so these items are at ZIP root), then rename back to .xlsx.
5. Open the repaired file in Excel to verify no warnings; then submit it to your pipeline.

Code · bash

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cellImages xmlns="http://www.wps.cn/officeDocument/2020/cellImage"></cellImages>
```

This minimal part satisfies the existing WPS Office relationship and content type without adding images or changing data.

Verification

1. Open the file in Microsoft Excel. Expect no repair prompts.
2. Run it through your CDR/sanitization or validation flow. The previous structural error should be gone.

Prevention

- Prefer generating and final-saving XLSX files with Microsoft Excel or OOXML-compliant libraries when strict validation is required.
- If using WPS Office, update to the latest version and validate outputs in a staging environment before distribution.
- Avoid post-processing that removes parts from xl/ while leaving references in relationships or [Content_Types].xml.

Troubleshooting / FAQ

Excel still shows a repair dialog after the fix

There may be additional structural issues (e.g., broken drawing or sharedStrings references). Repeat the inspect-and-repair process for any missing parts Excel mentions, or use Excel's Save As to regenerate the package.

Can I just remove the relationship instead of adding cellimages.xml?

Yes, removing the dangling relationship and its content type entry can also resolve the error. However, ensure that no other parts depend on it. Adding a minimal part is often simpler and safer for WPS-linked cell images.

Is data altered by this fix?

No. The minimal cellimages.xml contains no images or data; it only restores structural consistency. Always keep a backup before editing.

How do I automate the repair?

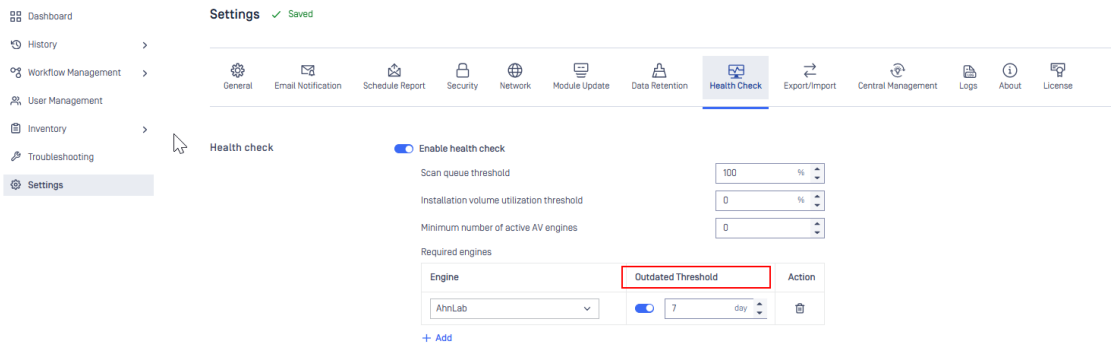
Use a script that unzips the XLSX, checks for WPS cellImage references in [Content_Types].xml and xl/_rels/workbook.xml.rels, writes a minimal xl/cellimages.xml if missing, then re-zips. Libraries like Python zipfile + lxml, or .NET System.IO.Packaging/Open XML SDK can help.

If Further Assistance is required, please proceed to log a support case or chat with our support engineer.

What does the "outdated" field under engines in the health check API indicate?

Check Your Version: This article applies to MetaDefender Core versions that include the health check API (`GET /readyz`).

The MetaDefender Core health check API (`GET /readyz`) returns an `outdated` field for each engine. This field indicates whether the engine's installed definition database is older than the **Outdated Threshold** configured in the Health Check settings.



How the field works

The `outdated` field is a staleness indicator based on the age of the installed definitions. It does **not** compare the installed definitions against the latest definitions published by OPSWAT.

- **No Outdated Threshold configured:** Core always reports `"outdated": false`, regardless of how old the installed definitions are.
- **Outdated Threshold configured:** `"outdated": false` means only that the installed definitions have not exceeded the configured threshold. It does not confirm that the engine is using the latest definitions available from OPSWAT or the antivirus vendor.

Example

If an engine update failed yesterday but the installed definitions are still within the configured threshold, the health check API continues to report:

Code · json

```
"outdated": false
```

Recommendation

Do not rely on the `outdated` field alone to confirm that an engine is running the latest available definitions. To verify that engines are up to date, see **How to check whether AV engines are running the latest update from the OPSWAT update server?**

Support: If further assistance is required, please log a support case or chat with our support engineers.

What is the meaning of the Stall detected messages when downloading engines on MetaDefender Core?

Check your version

This article applies to all MetaDefender V5 releases deployed on Windows or Linux system

Symptoms

Engine or database updates fail to complete, and the MetaDefender Core log records repeated "Stall detected" entries for a download, followed by the download being cancelled and the partial package removed. The entries look similar to this:

Code · xml

```
[INFO   ] (common.communication) Stall detected for download, url='.../eset_1_windows-
database-1691385165.zip' [msgid: 601]
[INFO   ] (common.communication) Stall detected for download, url='.../eset_1_windows-
database-1691385165.zip' [msgid: 601]
[WARNING] (common.communication) Download failed because of stall, url='.../eset_1_windows-
database-1691385165.zip' [msgid: 602]
[WARNING] (common.communication) Error while downloading file, error='OperationCanceledError',
errorDetails='Operation canceled' [msgid: 485]
[ERROR  ] (common.update) Download error occurred, removing downloaded package [msgid: 838]
[INFO   ] (common.update) Package has been marked failed [msgid: 1097]
[INFO   ] (common.update) Package folder was removed [msgid: 1786]
```

The key indicators are the repeated `Stall detected for download [msgid: 601]` messages and the subsequent `Download failed because of stall [msgid: 602]`.

Root cause

A "stall" means MetaDefender Core opened the connection to the OPSWAT update server but stopped receiving data for long enough that the transfer was considered inactive. After several stall detections, MetaDefender Core cancels the download, discards the incomplete package, and marks the update as failed.

This is a network condition, not a fault in MetaDefender Core.

The connection to `update.dl.opswat.com` is being obstructed or interrupted so that data cannot flow through reliably. Common causes include:

- A proxy, firewall, or SSL/TLS inspection device that buffers, throttles, or interrupts the connection
- An intrusion prevention system (IPS) or deep packet inspection rule that slows or resets the session
- Bandwidth shaping or rate limiting on the path to the internet
- General packet loss, latency, or an unstable route to the OPSWAT content delivery network

Resolution

Verify and stabilize network connectivity between MetaDefender Core and the OPSWAT update server.

1. Confirm the server can reach the OPSWAT update endpoint over HTTPS (port 443):
   - `update.dl.opswat.com`
2. If a proxy is used, confirm it is configured correctly in MetaDefender Core and that it is not buffering or timing out large transfers.
3. Ask your network team to check whether any firewall, proxy, SSL/TLS inspection, IPS, or content-filtering device is inspecting, throttling, or resetting traffic to `update.dl.opswat.com`. Where possible, exclude the OPSWAT update domains from inspection and throttling.
4. Review bandwidth or rate-limiting policies that could interrupt sustained downloads, especially for the larger engine packages.
5. Retry the engine update after the network path has been verified.

Support:

If you need further assistance, please contact OPSWAT Support through the OPSWAT Portal or your assigned support representative.

How do I recover MetaDefender Core pods failing with HTTP 403 “Access denied” because of an API key mismatch?

Check your version

This article applies to MetaDefender Core deployed on Kubernetes using the `mdcore-api-key` Secret. Before proceeding, confirm that your deployment uses this Secret and that the pod logs show repeated HTTP 403 “Access denied” errors as described below.

When to use this

Use this KB when:

- MetaDefender Core is deployed on Kubernetes and the pods were previously running normally.
- The issue started after a recent change, such as an upgrade, restore, redeployment, Secret rotation, Helm/GitOps sync, or database/PVC reuse.
- Pod logs show repeated `403 Access denied` errors.
- You want to recover the existing deployment safely without recreating MetaDefender Core, PostgreSQL, or persistent volumes.

Common log messages include:

- `Warning: Setup proxy failed. Response (403): {"err":"Access denied"}`
- `Warning: License activation failed. Response (403): {"err":"Access denied"}`
- `Info: The license failed to deactivate via the Core service (403): {"err":"Access denied"}`

> The proxy warning appears only if a proxy is configured.

What this means

This usually means the API key inside Kubernetes does not match the API key stored in the MetaDefender Core database.

In short:

- The pod starts with the API key from the `mdcore-api-key` Secret.
- MetaDefender Core checks that key against the key stored in its database.
- If they do not match, Core returns HTTP 403.
- The pod may fail health checks and restart before you can fix it.

Quick recovery steps

1. Confirm the pod logs show `403 Access denied`.
2. Temporarily remove only the `livenessProbe` and `startupProbe` so the pod stays running while you repair it. Save each probe definition before removing it so you can add the probes back later.
3. Port-forward to the pod and log in to the MetaDefender Core web interface.
4. Choose one of the two repair options:
   - **Option A:** First read the API key value from the Kubernetes `mdcore-api-key` Secret. Then go to the MDCore web console and update the administrator API key there to match the Secret value.

- **Option B:** If Option A does not work, log in to the MDCore web console, create a new API key there, then patch the Kubernetes Secret with the new key.

5. After completing Option A or Option B, run the non-interactive `kubectl exec` verification command to confirm the right API key now exists in the database.
6. If verification succeeds, restart the rollout and wait for the pods to come online.
7. If Option A verification fails, use Option B, verify again, restart the rollout, confirm the pods are healthy, add the probes back, and restart the deployment if required.

Step-by-step

1. Confirm the error

Check pod status:

```
Code

kubectl get pods -n <namespace>
```

Check logs:

```
Code

kubectl logs -n <namespace> deployment/md-core --since=30m
```

If you see repeated `Response (403): {"err":"Access denied"}`, continue with this procedure.

> If the logs show connection refused, DNS, timeout, or PostgreSQL authentication errors instead, this KB is not the right fix.

2. Temporarily disable restart probes

Save only the current probe definitions, then remove the probes temporarily. This avoids restoring a stale full Deployment manifest later.

```
Code

kubectl get deployment md-core -n <namespace> -o
jsonpath='{.spec.template.spec.containers[0].livenessProbe}' > liveness.json
kubectl get deployment md-core -n <namespace> -o
jsonpath='{.spec.template.spec.containers[0].startupProbe}' > startup.json
```

Then remove only the `livenessProbe` and `startupProbe` manually. The liveness probe can restart the pod, and

removing the startup probe gives you time to complete the repair before Kubernetes starts enforcing the liveness check.

Check that both probes are defined before you remove them. If either output is empty, do not run the corresponding remove command.

Code

```
kubectl get deployment md-core -n <namespace> -o
jsonpath='{.spec.template.spec.containers[0].livenessProbe}{"\n"}
{.spec.template.spec.containers[0].startupProbe}{"\n"}'
kubectl patch deployment md-core -n <namespace> --type=json -p='[{"op":"remove","path":"/spec/
template/spec/containers/0/livenessProbe"}]'
kubectl patch deployment md-core -n <namespace> --type=json -p='[{"op":"remove","path":"/spec/
template/spec/containers/0/startupProbe"}]'
```

Wait for the rollout:

Code

```
kubectl rollout status deployment/md-core -n <namespace>
```

Only leave the probes disabled long enough to recover the deployment. Re-enable them after the Secret is fixed.

3. Connect to the pod

Find the new pod:

Code

```
kubectl get pods -n <namespace>
```

Port-forward to it:

Code

```
kubectl port-forward -n <namespace> pod/<mdcore-pod-name> 8008:8008
```

Then open one of these in your browser:

- `http://127.0.0.1:8008`
- `https://127.0.0.1:8008` if TLS is enabled

4. Choose the repair option

There are two repair options. Start with **Option A**: use the API key value that is already stored in Kubernetes and update MDCore to match it. Use **Option B** only if Option A does not work.

Option A: Update MDCore with the API key from the Kubernetes Secret

Use this option first. The goal is to take the API key value currently stored in the Kubernetes Secret and update the MDCore administrator API key in the web console so MDCore and Kubernetes match. If this verification fails because the Kubernetes Secret contains an invalid API key, continue with Option B.

1. Get the current API key value from the Kubernetes `mdcore-api-key` Secret.
2. Log in to the MetaDefender Core web interface with an administrator account.
3. Go to **User management** → **Users**, open the administrator user, and update the API key so it matches the value from the Kubernetes Secret. MetaDefender Core API keys are 36 characters long.
4. Log in to the MDCore pod and verify the API key with `curl` using the command below.
5. If the `curl` verification returns `200`, continue to Step 5.

Code

```
# Get the current API key from the mdcore-api-key Secret first.
MDCORE_API_KEY="$(kubectl get secret mdcore-api-key -n <namespace> -o jsonpath='{.data.value}' |
base64 -d)"
printf 'Kubernetes Secret API key: %s\n' "${MDCORE_API_KEY}"
# Find the MDCore pod and verify the key with a non-interactive exec.
MDCORE_POD=$(kubectl get pod -n <namespace> -l app=md-core -o
jsonpath='{.items[0].metadata.name}')
kubectl exec -n <namespace> "$MDCORE_POD" -- \
  curl -s -o /dev/null -w '%{http_code}\n' -H "apikey: $MDCORE_API_KEY" http://localhost:8008/
stat/nodes
```

Expected result: `200`. If the command returns `200`, continue to Step 5. If it returns `403` or another failure because the Secret value is invalid, use Option B to create a new MDCore API key and patch the Kubernetes Secret.

Option B: Create a new MDCore API key and patch the Kubernetes Secret

Use this option only if Option A does not work. In this path, create a new API key in the MDCore web console, then update Kubernetes so the Secret contains the same new key.

1. Log in to the MetaDefender Core web interface with an administrator account.
2. Go to **User management** → **Users**, open the administrator user, and generate or reset the API key.
3. Copy the new 36-character API key.
4. Patch the existing `mdcore-api-key` Kubernetes Secret with the new key.
5. Log in to the MDCore pod and verify the API key with `curl` using the command below.
6. If the `curl` verification returns `200`, continue to Step 5.

Code

```
read -rsp "New MetaDefender Core API key: " MDCORE_API_KEY
echo
MDCORE_API_KEY_B64="$(printf '%s' "${MDCORE_API_KEY}" | base64 | tr -d '\n')"
kubectl patch secret mdcore-api-key -n <namespace> --type=merge -p "{\"data\":{\"value\":
\"${MDCORE_API_KEY_B64}\"}}"
# Find the MDCore pod and verify the key with a non-interactive exec.
MDCORE_POD=$(kubectl get pod -n <namespace> -l app=md-core -o
jsonpath='{.items[0].metadata.name}')
kubectl exec -n <namespace> "$MDCORE_POD" -- \
  curl -s -o /dev/null -w '%{http_code}\n' -H "apikey: $MDCORE_API_KEY" http://localhost:8008/
stat/nodes
```

Expected result: `200`. If the command returns `200`, continue to Step 5.

Do not paste API keys into tickets, screenshots, or shared notes. Base64 is not encryption.

If the `mdcore-cred` Secret was rotated too, UI login may also fail. Fix administrator login first, then continue with the API key repair.

After completing Option A or Option B, test the `curl` command internally from the MDCore pod. This verifies that the right API key now exists in the database. If the request returns the expected successful response, proceed with restoring the Deployment, restarting the rollout, waiting for pods to come online, and re-enabling health checks.

5. Restart the pod so it reads the repaired key

After completing Option A or Option B, restart the deployment so the pod reads the updated Secret:

Code

```
kubectl rollout restart deployment/md-core -n <namespace>
kubectl rollout status deployment/md-core -n <namespace>
```

If your environment is managed by Argo CD, Flux, or an external secret manager, make sure the same valid API key is also updated in the source of truth. Otherwise the managed value may be put back automatically.

6. Restore the original probes

Add the saved probe definitions back to the Deployment. Do not use `kubectl rollout undo` here: the restart in Step 5 creates a new revision, so undo would roll back only to the probe-less revision instead of the original probe configuration.

Run:

Code

```
kubectl patch deployment md-core -n <namespace> --type=json \
  -p="[{\"op\":\"add\",\"path\":\"/spec/template/spec/containers/0/livenessProbe\",\"value\":
$(cat liveness.json)}]"
kubectl patch deployment md-core -n <namespace> --type=json \
  -p="[{\"op\":\"add\",\"path\":\"/spec/template/spec/containers/0/startupProbe\",\"value\":
$(cat startup.json)}]"
kubectl rollout status deployment/md-core -n <namespace>
```

Support:

If you need further assistance, please contact OPSWAT Support through the OPSWAT Portal or your assigned support representative.

What temporary folder do Custom Engines use ?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Currently, there are six **Custom Engines** which use **C:\Windows\temp** as a temporary folder, instead of the **Metadefender Core** folder.

These Custom Engines are:

- Filseclab
- Huorong
- Windows Defender
- Netgate
- Systweak
- Emsisoft

Support:

For further assistance regarding **Custom Engine** temporary folders, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What PDF file annotations are Supported/Non-Supported by

MetaDefender Core's Deep CDR?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core Deep CDR's annotation support is clarified in the table below:

Supported	Non-Supported
Link, File Attachment, 3D, and *Widget*.	Text, FreeText, Line, Square, Circle, Polygon, PolyLine, Highlight, Underline, Squiggly, StrikeOut, Stamp, Caret, Ink, Popup, Sound, Movie, Screen, PrinterMark, TrapNet, Watermark, and Redact.

The relationship between MetaDefender Core's configurations and annotations is outlined in the table below:

Configuration	Annotation
Remove Hyperlink	Remove Link annotation
Remove Embedded Object	Remove File Attachment and 3D annotation
Remove Form	Remove Widget annotation

Support:

For further assistance regarding **MetaDefender Core Deep CDR**'s supported/non-supported **PDF Annotations**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What does "Potentially Vulnerable File" result mean?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Potentially Vulnerable Files are files associated with vulnerable components or applications identified by OPSWAT's File-Based Vulnerability Assessment technology.

Pro Tip:

Our File-Based Vulnerability Assessment technology allows IT administrators to:

- Check certain types of software for known vulnerabilities before installation
- Scan systems for known vulnerabilities when devices are at rest
- Quickly examine running applications and their loaded libraries for vulnerabilities.

This technology correlates vulnerabilities to software components, product installers, firmware packages, and many other types of binary files, which are collected from a vast community of users and enterprise customers. To learn more, Read This.

Support:

If you require assistance regarding **Potentially Vulnerable File** results or our **File-Based Vulnerability Assessment** technology, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What CAB files are Supported/Non-Supported by MetaDefender's Deep CDR?

Check Your Version:

This article applies to all MetaDefender Core V5 and V4 releases deployed on Windows or Linux systems.

Please Note:

It is important to note that, among the archive files with a **.cab** filename extension, there are actually two distinct file formats, one of which is supported by Metadefender's Deep CDR, and one of which is not.

The differences between the two **.cab** formats are outlined below.

Type ID	Description	Support to extract
CAB	Microsoft Cabinet Archive	**YES**
InstallShieldCAB	InstallShield Cabinet Archive	**NO**

1. Microsoft Cabinet, Signature: MSCF

- **Cabinet** (or **CAB**) is a Microsoft Windows archive-file format that supports lossless data compression and embedded digital certificates that are used for maintaining archive integrity.

- **Cabinet** files have **.cab** filename extensions.
- **Cabinet** files are recognized by their first 4 bytes: **MSCF**.
- *Cabinet** files were originally known as **Diamond** files.

2. InstallShield Cabinet, Signature: ISC

- **InstallShield CAB** files are also archive files, but they are used by the installer program, InstallShield for its own proprietary archiving formats. Hence, they are distinct from **Microsoft Cabinet** files.
- **InstallShield CAB** files also have .**cab** file name extensions.
- **InstallShield CAB** file headers are not the same as **Microsoft CAB** files, so they are incompatible and cannot be manipulated or edited with the programs that are made to interact with the standard **Cabinet** format - including MetaDefender Core's Deep CDR.
- There are actually numerous other (Non-Microsoft) installers that use the **.cab** extension for their proprietary archiving formats, but **InstallShield CAB** files are the most common.
- *InstallShield CAB** files are recognized by their first 3 bytes: **ISC**.

Support:

For further guidance regarding supported and non-supported **Cabinet** files, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What are the MetaDefender Core Security Policies and how do they work?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

As an overall simplified explanation of the topic, file processing on MetaDefender Core is orchestrated via workflow management.

- Any file processing request must be tied to a specific "workflow rule" whereas all product configurations and rules are defined.
- Each "workflow rule" must be tied to specific "template" and "security zones":
  - A "template" is literally a template that one or many "workflow rule" can be based on, which means, inherit all configurations' values that "template" defines to use for that "workflow rule". Of course, you can customize configuration values for "workflow rule" after inheriting configurations' values from "template". Similar relational concept like users & groups, or files & folders permission.
  - A "security zones" is created as network filtering rule based IP address range for that corresponding "workflow

rule". That means, any client source's address going out of that IP address range, that client is not allowed to use that "workflow rule" for file processing.

The term **Security Policies** refers to three highly configurable objects that work in combination to accomplish MetaDefender's primary security function.

These objects are:

1. **Workflows**
2. **Security Zones**
3. **Templates**

Workflows

Definition:

A **Workflow** is the object that each file interacts with directly when being processed by MetaDefender.

I.e, each file is processed through one (and only one) of the user-defined **Workflows**.

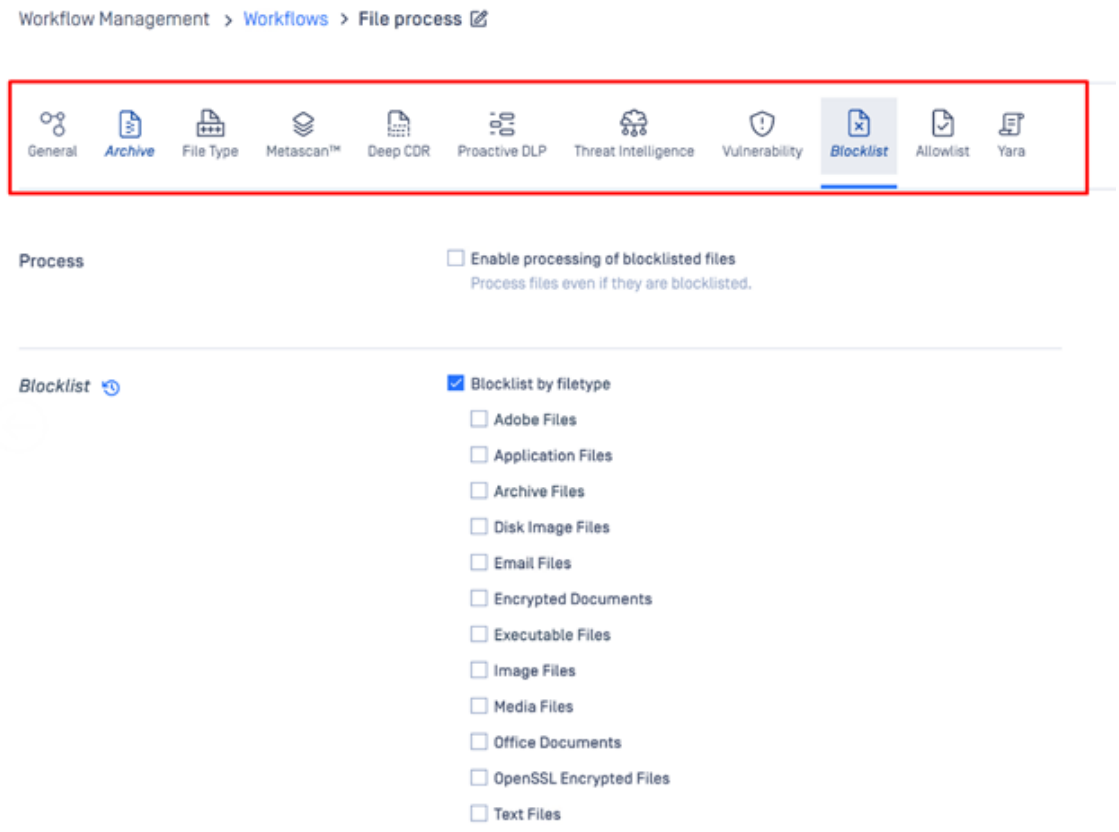
The workflow is identified by its name.

- It defines eligibility parameters to use it (i.e. whether a client is in the proper **Security Zone** and/or the actual logged-in user is in the specified **Role** and/or the client has provided the required **user_agent**).
- It inherits processing characteristics (i.e. whether to scan files with the malware engines, if and how to use data sanitization, if and how to extract archives, etc.) from a **Workflow** that gets assigned to it.
- It also allows direct assignment of processing characteristics that override the characteristics of the **Workflow**.

You create a **Workflow** by giving it a name and assigning a **Security Zone** and a **Template** to it. You can also assign specific processing characteristics to it.

- A file's eligibility to be processed by the **Workflow** is determined by the filtering parameters under the **General** tab.
- If all the required parameters are matching, the processing actions performed on that file are then determined by the specific processing characteristics set on the **Workflow** such as:
  - **General** tab
  - **Archive** tab
  - **Filetype** tab
  - **Metascan** tab
  - **Deep CDR** tab
  - **Proactive DLP** tab
  - **Threat Intelligence** tab
  - **Vulnerability** tab

- **Blocklist** tab (as shown in the screenshot below)
- **Allowlist** tab
- **Yara** tab



- Workflows can be reordered using **drag & drop**.

Please Note:

A file that is eligible to be processed by more than one **Workflow** will still only get assigned to one **Workflow** (the assignment logic is described below).

A file that is not eligible for any **Workflow** will not be processed.

Security Zones

Definition:

A **Security Zone** is the object that defines a network or a set of networks (as defined by IP masks).

Only files with a source location in that network (or set of networks) are eligible to be routed to a **Workflow** that is assigned to that **Security Zone**.

Templates

Definition:

A **Template** is the object where you define a set of process actions (and associated properties) such as malware scanning, sanitization, archive handling, etc.

The **Template** does not get applied directly to the file. Instead, the **Workflow** is associated with one or more **Templates**, and it is the **Workflow** that gets applied to the file.

The Template can be thought of as a template of process settings.

- By assigning the **Workflow** to a **Template**, the **Workflow** inherits the **Template** settings for each field that has not been directly populated on the **Workflow**.

Templates that are included out-of-the-box with each MetaDefender Core V5 installation are:

1. **Default**
2. **Skip Images**
3. **Executables Only**

- These **Workflow Templates** cannot be altered or deleted, but they can be copied to custom **Workflows** that can then be edited.

Please Note:

Only the three Workflows mentioned above will be migrated when you upgrade from MetaDefender Core V4 to V5.

Support:

If you have any issues navigating MetaDefender Core's **Security Policies**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Is Metadefender Core compromised while scanning files?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems.

No, it is not.

Writing a malicious file to the disk does not mean it can do harm since, to activate malware, the malicious files must

first be interpreted. MetaDefender Core does not interpret or run the files that are written to the disk - it only uses them as bitstreams.

Additional “Temporary processing files” name is randomized to avoid accidentally trigging the files, we are taking this approach to minimize the risk that malicious files could be triggered while processing. It follows that Metadefender Core should not be compromised during scans.

Support:

If you believe that MetaDefender Core has somehow been compromised during a scan, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What is the Queue Mechanism on Metadefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core is equipped with a robust queue mechanism designed to efficiently manage high-volume data processing tasks, such as file scanning. This article outlines the core concepts of the queue mechanism, its behavior, and configuration parameters.

Overview of the Queue Mechanism

MetaDefender Core organizes its file processing tasks using a queue system. This allows the platform to handle incoming data in a controlled and orderly manner.

- **Processing Threads (Workflows):** Each item in the queue is processed by a separate workflow, or thread, on the system.
- **Processing Order:** By default, items are handled on a **first come, first served** basis. While this ensures fairness, it does **not** guarantee a predictable processing time for each item or the overall queue. To help align file processing with business priorities, MetaDefender Core allows you to set processing priorities based on the workflow in use. This can be configured under the "**General**" tab of each workflow.
- **Archive File Handling:** By default, MetaDefender Core reserves **25% of the total queue size** specifically for archive processing. This cap applies to **all archives in the queue collectively**, not per archive. For example, submitting a single archive will only occupy up to 25% of the total queue, regardless of the archive's internal content. Starting with MetaDefender Core v5.14.1, the value is configurable up to 75% of the total queue. This setting can be adjusted from the **Settings > General** tab. This design ensures that:
  - Additional file submissions are not blocked by resource-heavy archive extraction.
  - There’s capacity for **parallel processing** across different file types.

Queue Size

- **Default Size:** The default queue size is **500 items**. Starting with MetaDefender Core v5.11.0, the value can be set directly in the UI via the **Settings > General** tab.
- **Monitoring Queue Status:** You can check the number of queued items in the **Dashboard** > **System Health** > **Current Processing Slots** tab by using the REST API endpoint `GET /stat/nodes`

Summary

The queue mechanism in MetaDefender Core v5 is designed to balance performance and stability under high data loads. With controlled archive handling and configurable queue size, the system ensures efficient resource allocation and throughput.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

How does the Post Actions feature work?

Check Your Version:

This article applies to all MetaDefender Core releases subsequent to V4.8.0, including all MetaDefender Core V5 releases, deployed on Windows systems.

Disclaimer

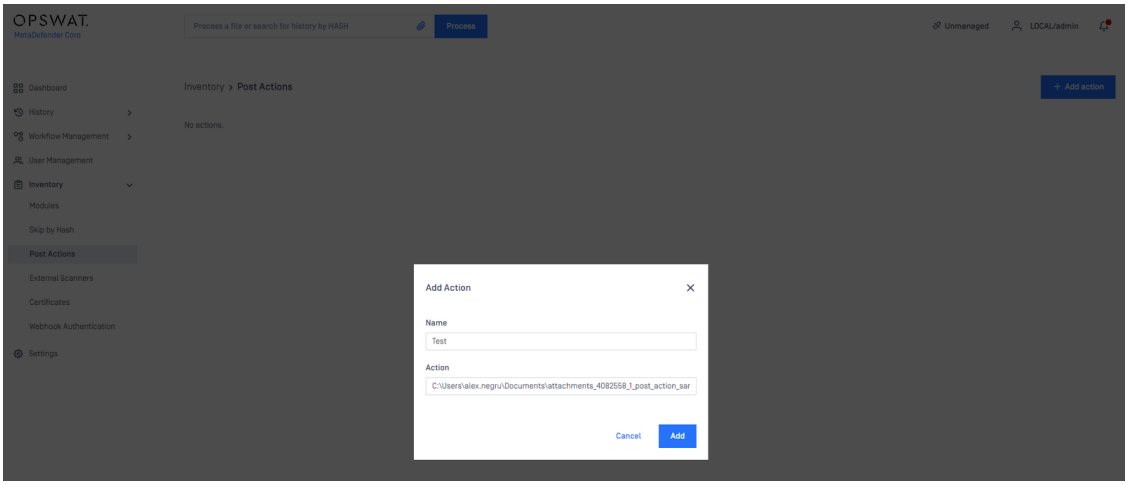
The following sample script is provided for illustrative purposes only and is not guaranteed to be functional in a production environment. Please be aware that script only works Windows platforms.

MetaDefender Core V4.8.0 and all subsequent releases include the **Post Actions** feature:

- This feature allows you to define a **Post Action** which is then invoked through a command line executable or script.
- This executable/script will be called following each scan.
- Additional documentation for this feature can be found here: External Scanners and Post Actions.

The sample in the images below is a Powershell script. For this script to work properly, we must call the Powershell executable via the **Inventory>Post Actions** tab in the MetaDefender Core Management Console, as follows:

Once on the **Post Actions** screen, we click **Add Action** before entering the chosen name and full path to the executable, as illustrated in the screenshots below.





You will need to specify the location from where Powershell is running in your system, followed by:

- **ExecutionPolicy Bypass**
- **-File <PathToYourScriptFile>**

We created a sample Powershell script that sorts the files according to their results - either **Allowed** or **Blocked**.

- The script is called after the scan is finished.

Input:

1. The current scan results **JSON** from **STDIN**.
2. The full path to the currently scanned file as the last argument on the command line.

Output:

The script has 5 possible return values:

- **0** - success
- **1** - **JSON** parse error, the script was unable to parse the expected **JSON** from **STDIN**
- **2** - copy error, the file copy failed
- **3** - file path of the currently scanned file is invalid
- **4** - the destination path of either **allowed**, **blocked** or both is invalid.

Pro Tip:

The script itself can be found and downloaded from the following link:

post_action_sample.ps1

Support:

If you have any issues navigating MetaDefender Core's **Post Actions** feature, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How does the External Scanners feature work?

Check Your Version:

This article applies to all MetaDefender Core releases subsequent to V4.8.0, including all MetaDefender Core V5 releases, deployed on Windows or Linux systems.

Disclaimer

This sample script is provided for illustrative purposes only and is not guaranteed to be functional in a production environment.

MetaDefender Core V4.8.0 and all subsequent releases include the **External Scanners** feature:

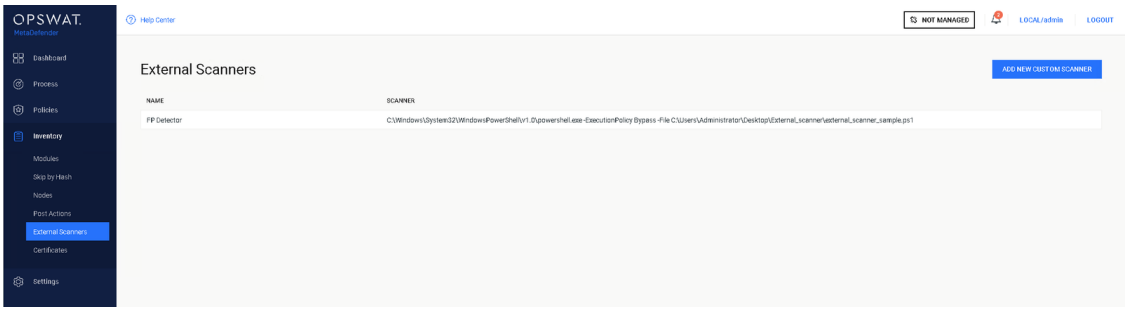
- This feature allows you to define an **External Scanner** which is then invoked through a command line executable or script.
- This executable/script will be called for each scanned file, after all the other engines have scanned - but before the final verdict is decided.
- Additional documentation for this feature can be found here: External Scanners and Post Actions.

The script in the sample below is a Powershell script. For this script to work properly, we need to call the Powershell executable in the **External Scanners** screen of MetaDefender Core, as follows:

V5:



V4:



You will need to specify the location from where Powershell is running in your system, followed by:

- ExecutionPolicy Bypass
- -File TheNameAndPathOfYourScriptFile.ps1

We created a sample Powershell script that attempts to flag suspicious files as **False Positive**:

- The script checks the scan results of the current file and, if the file is flagged as infected by only one engine, the file's hash is subsequently sent to MetaDefender Cloud.

MetaDefender Cloud's results are then analyzed:

- In case the file is flagged as Infected in MetaDefender Cloud by ONLY the same one engine that flagged the file in MetaDefender Core, OR if the file is found to be clean by MetaDefender Cloud, the file will be copied to a **$false_positive** folder for later investigation, and the verdict will be **Suspicious** (2) and threat_ found will be **Suspected False Positive**.
- If the file is flagged by any other engine on MetaDefender Cloud then, verdict will be **Infected** (1) and threat_found will be **Infected - Probably**
- If the file is not flagged by any local engine, the script will return the verdict
- *No Threat Detected**(0).

Input:

1. It is your responsibility to create and populate the system context variable *{{false_positive}}* with a valid folder name before running the script
2. It is your responsibility to create and populate the system context variable *{{apikey}}* with your valid MetaDefender Cloud license key.

3. The script accepts the currently scanned file location as its last command-line argument, and stores it in the variable **$current_file_path.**
4. The script expects to find the scan results JSON on STDIN. it is read into the variable **$scan_results.**

Output:

1. The script will add its verdict (based on results from MetaDefender Cloud) to the result **JSON** and write it to the **STDOUT.**
2. if only the same engine (or no engine at all) flag the file as malicious the script will copy the file to the folder **$false_positive** for later investigation
3. The script has 6 possible return values:

- "0" - Success
- "1" - Input JSON Parse error
- "2" - Copy error
- "3" - file path of the currently scanned file is invalid
- "4" - the destination path of "false positive" is invalid.
- "5" - call to MetaDefender hash lookup failed
- "6" - hash not found on MetaDefender Cloud

Pro Tip:

The script itself can be found and downloaded from the following link:

external_scanner_sample.ps1

Support:

If you have any issues navigating MetaDefender Core's **External Scanners** feature, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How to generate a Core's scan result summary in JSON format

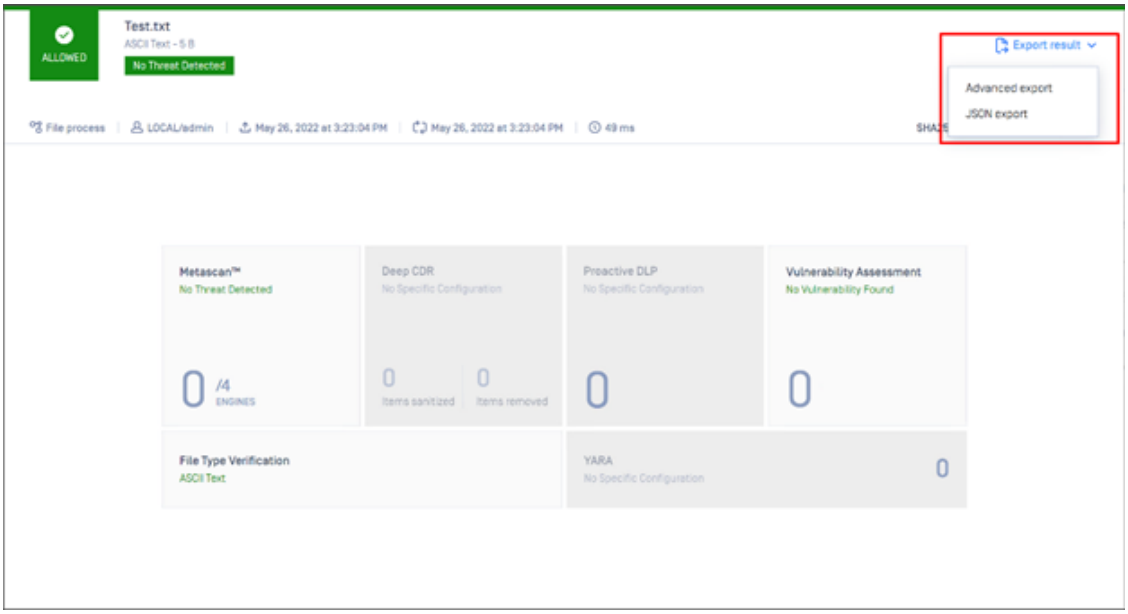
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases.

The following instructions will guide you through the process of saving your MetaDefender Core scan results to a JSON file.

Option 1 (V5)

MetaDefender Core V5 releases include an option to export a JSON result directly from the **Console**. Select **JSON Export** as illustrated in the screenshot below:



Option 2 (V4/V5)

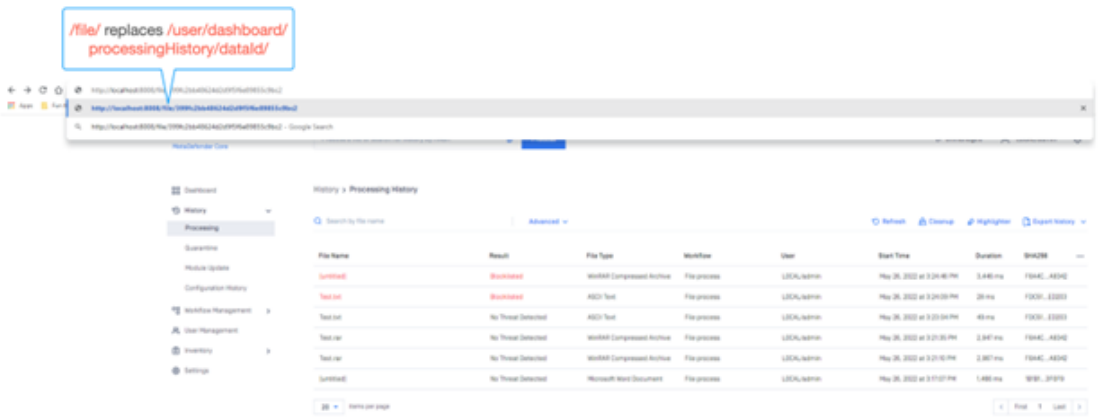
1. On the MetaDefender Core Scan Result Page, modify the URL as follows:

http://localhost:8008/*#/user/dashboard/processingHistory/dataId/*399fc2bb48624d2d9f5f6e89855c9bc2

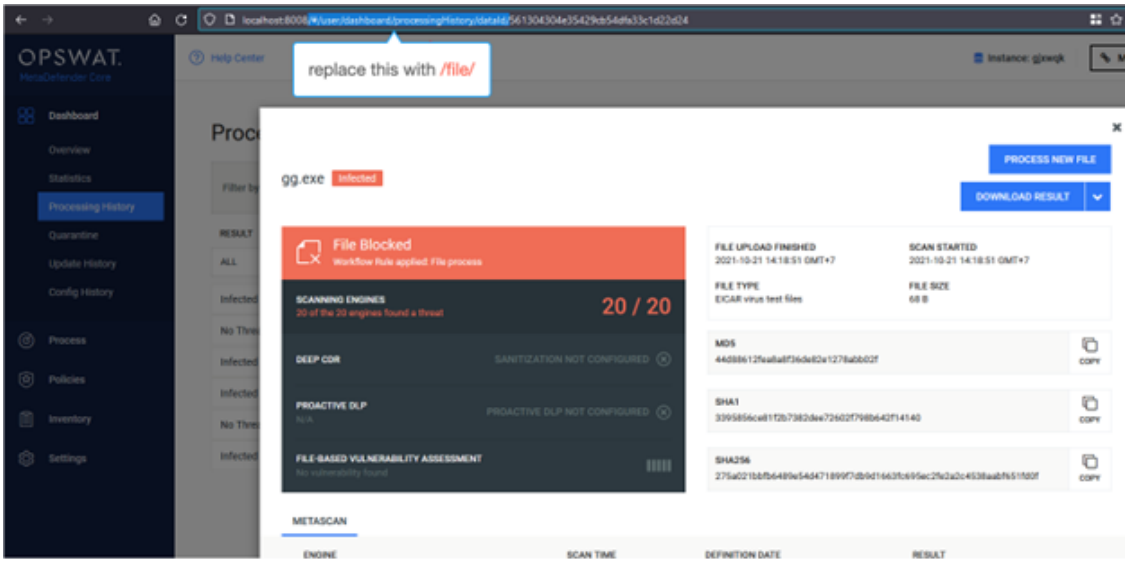
=> http://localhost:8008/file/399fc2bb48624d2d9f5f6e89855c9bc2

Or, as illustrated in the relevant screenshot below:

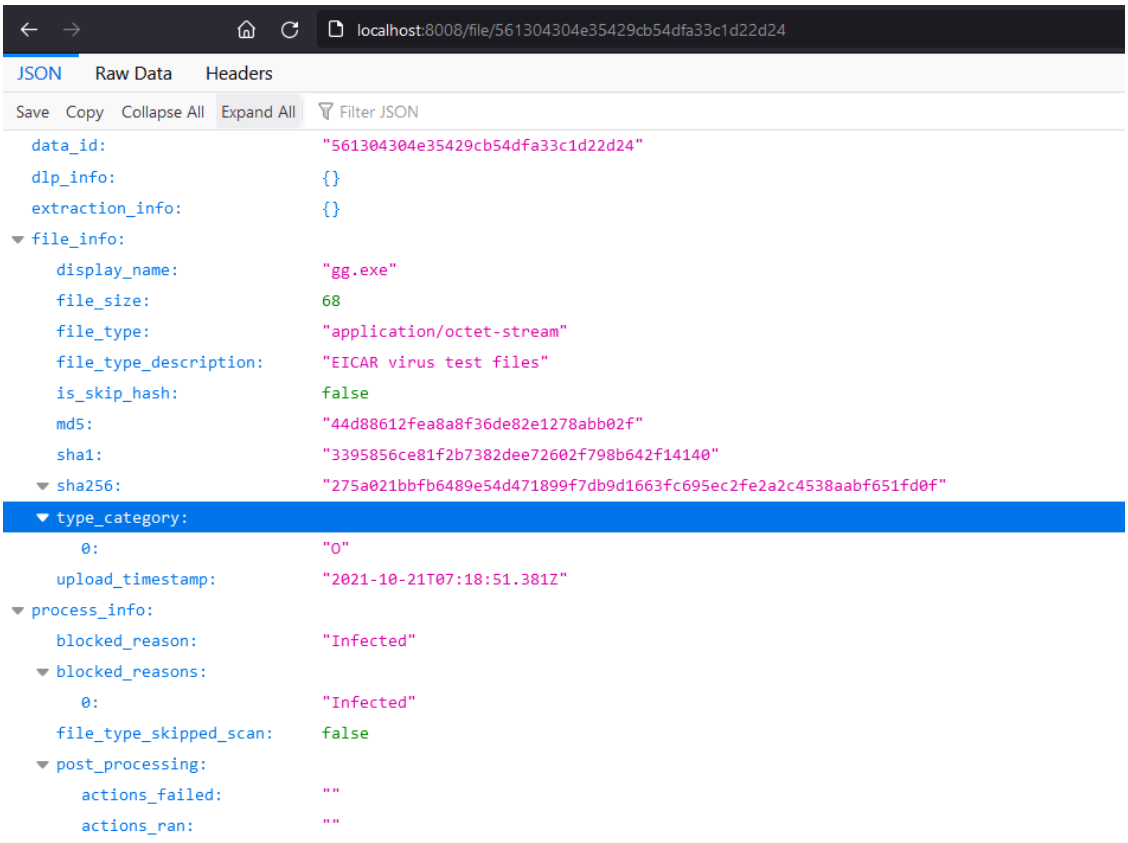
V5:



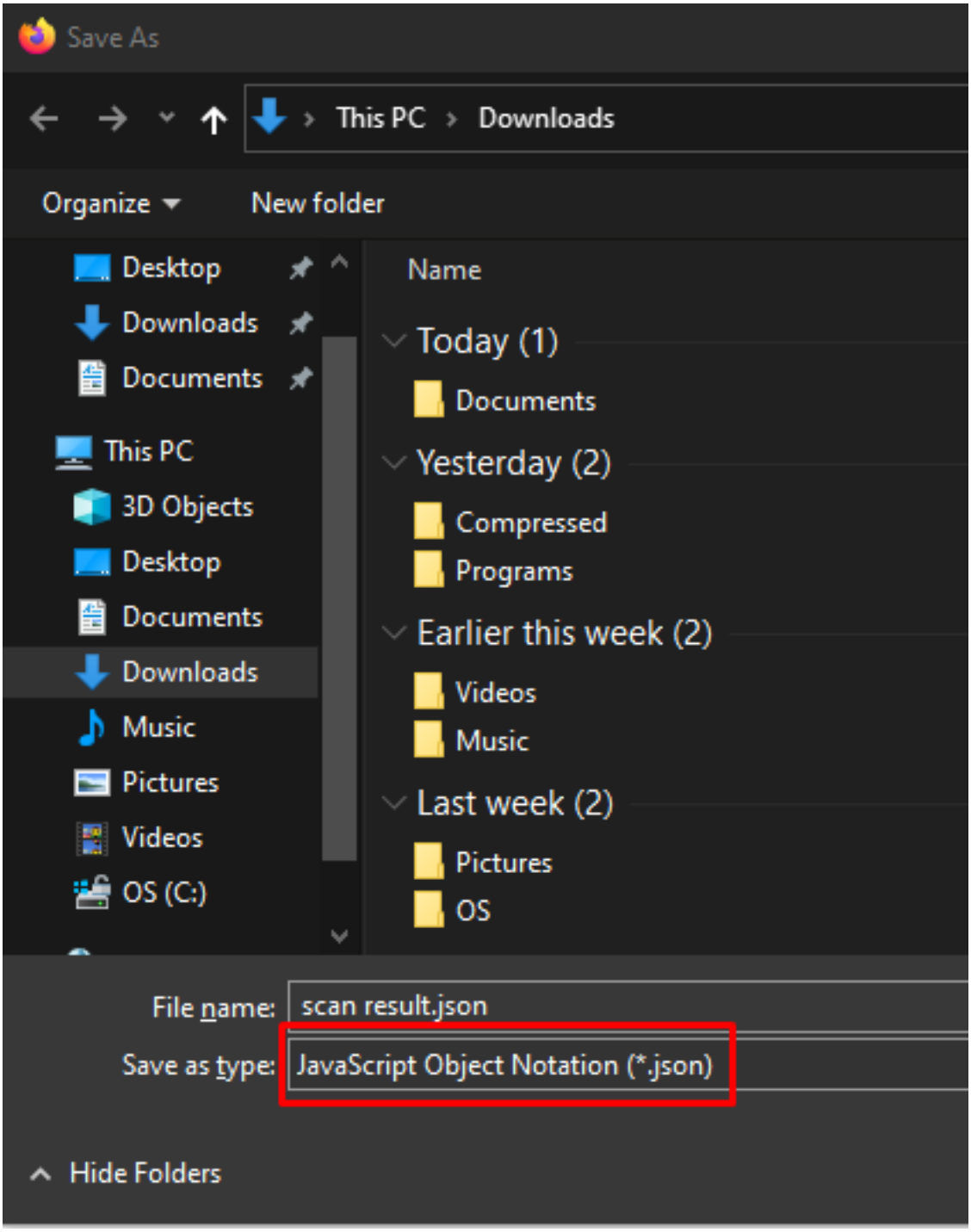
V4:



2. Hit **Enter**, and your screen will update as follows:



3. Finally, hit **Ctrl+S** to save this scan result into a JSON file, as follows:



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How and why are my image files altered during processing with Deep CDR and Proactive DLP?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Pro-Tip:

MetaDefender Core's Deep CDR removes potential embedded threats by assuming all files to be malicious, then sanitizing and rebuilding files to ensure full and safe usability.

The Deep Content Disarm and Reconstruction process will ensure files are free from known and unknown threats, complex and sandbox aware threats, and threats equipped with malware evasion technology.

MetaDefender Core's Proactive DLP prevents potential data breaches and regulatory compliance violations by blocking sensitive and confidential data that may be included in files and emails.

The Data Loss Prevention process will ensure that all files, such as Microsoft Office and PDF docs, are free from sensitive details such as credit card and social security numbers.

To process image files, the Deep CDR engine engages the following process:

1. The image file is decompressed in order to access and process the raw data.
2. The engine then reconstructs a safe-to-use file by processing the pixels from the image.
3. During content disarming and reconstruction, potential steganography (data concealment) is treated by adding noisy data to 'break' hidden elements, while unnecessary data is discarded.
4. In the final stage of reconstruction, the file is compressed with the same image encoding as the original file.

To process image files, the Proactive DLP engine engages the following process:

1. The image file is decompressed in order to access and process the raw data.
2. The decompressed file is systematically scanned for sensitive and possibly confidential information.
3. This information, if any, is redacted or removed.
4. In the final stage of reconstruction, the file is compressed with the same image encoding as the original file.

Please Note:

Although the encoding (at Step 4) is the same, the image compression method may differ from the original method used.

As a result, the size of the graphics file will be altered - but the image quality will be largely retained.

MetaDefender Core guarantees a minimum of 99% image quality retention following processing with either Deep CDR or Proactive DLP.

Please Note:

If you encounter cases that file size changes compare to the original file size, these are the common reasons

1. The document contains images, and as mentioned earlier, the image file size can be changed, leading to the whole document file change
2. Some markup formats may get increased in size due to format decoration changing, e.g., spaces, tabs, or newlines added

Support:

If you believe your **Image Files Are Being Compromised During Deep CDR or Proactive DLP Processing**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Does MetaDefender Core v5 offer real-time antivirus protection on the system where it is installed?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases installed on Windows or Linux systems.

No.

Although MetaDefender Core uses a number of antivirus engines that are typically found in real-time anti-malware products, it does not offer **real-time** protection for the system on which it is installed.

MetaDefender Core is designed to scan only those files that are submitted to it on demand.

Pro Tip:

If real-time protection is required, OPSWAT recommends installing an additional, real-time anti-malware protection product on the MetaDefender Core server.

However, when installing a real-time protection agent on the MetaDefender Core server, be sure to exclude both MetaDefender Core's **Installation** directory and its **Temporary** directory (used for scanning) from the protection agent's coverage.

Please Note:

Be advised that, should you choose to leverage a pre-installed anti-malware engine for real-time protection that also happens to be included as a scan engine in your MetaDefender Core V4 or V5 release, there are similar exception rules that will need to be applied.

Read This Article learn more.

Support:

Feel free to contact OPSWAT Support for expert guidance when installing additional **Protection Agents** or **Programming Exceptions**. Log a support case or chat with our support engineer to get in touch.

Does MetaDefender Core v5 Detect the NotPetya Ransomware?

Check Your Version:

This article applies to all MetaDefender Core releases subsequent to V4.21.2, including all V5 releases, deployed on Windows or Linux systems.

The voracious NotPetya ransomware attacks were first detected in Ukraine in 2017 and quickly spread across Europe and the world, continuing on their devastating course for approximately two years.

Please Note:

OPSWAT was swift to incorporate defenses against this threat into the latest MetaDefender Core technology and, since the group responsible for the NotPetya attacks are still at large today, we (and our vendors) remain vigilant and dedicated to safeguarding organizations against NotPetya and similar threats.

At the heart of the solution, the base MetaDefender Core multi-scanning engine uses up to 33 anti-malware engines to scan files for threats. Our detection rate is dependent on the number of enabled engines, with a higher number of engines increasing malware detection rates overall.

Currently, most of the engines included in our MetaDefender Core base packages acknowledge the NotPetya ransomware threat. Below is a package breakdown with relevant information provided by each of the engine vendors.

Please Note:

1. Lower packages of MetaDefender Core are subsets of higher packages. So, higher packages include the same engines as lower packages - plus more.
2. For a comprehensive list of engines per package and OS, expand the **Metascan** tab on the following webpage: MetaDefender Enterprise Licensing
3. Some of our vendors may already be detecting the NotPetya threat but do not have any official post about it. These vendors are not listed below but will be included as more information becomes available.
4. Specific engine detection is based on the most up-to-date engine definitions. Some latency may occur due to update frequency, update methods, or network speeds.

Windows

MetaDefender Core 8

Avira:

Petya strikes back - Avira Blog

ESET:

“Petya” Ransomware: What we know now | | ESET

Bitdefender:

Massive GoldenEye Ransomware Campaign Slams worldwide users

MBR-Encrypting Ransomware Petya Gets Bitdefender Vaccine

Petya Ransomware Goes Low Level

Quick Heal:

Petya ransomware is affecting users globally, here are things you can do

VirITeXplorer:

Petya or notPetya ? questo è il dilemma.... nuovo ransomware o vecchia conoscenza...

MetaDefender Core 12

CYREN:

Petya Ransomware Rises by Using Similar Exploit As WannaCry - Cyren

MetaDefender Core 16

Emsisoft:

Petya ransomware variant attacks computers worldwide

Zillya!:

Кибертака Украина: В антивирусной компании рассказали, кто может стоять за хакерской атакой Petya.A и чем это грозит — Украина

VirusBlokAda:

New Ransomworm Follows WannaCry Exploits

MetaDefender Core 20

McAfee:

How to protect against Petya, Ransom-WannaCry, and other ransomware | McAfee Support

How to Protect Against Petya Ransomware in a McAfee Environment | McAfee Blog

Sophos:

Petya variants behind global ransomware outbreak – Sophos News

Troj/Petya-AQ - Viruses and Spyware - Advanced Network Threat Protection | ATP from Targeted Malware Attacks and Persistent Threats | sophos.com - Threat Center

Linux

MetaDefender Core 5

Bitdefender:

Massive GoldenEye Ransomware Campaign Slams worldwide users

Petya Ransomware Goes Low Level

ESET: "Petya" Ransomware: What we know now | | ESET

MetaDefender Core 10

Avira:

Petya strikes back - Avira Blog

CYREN:

Petya Ransomware Rises by Using Similar Exploit As WannaCry - Cyren

Quick Heal:

Petya ransomware is affecting users globally, here are things you can do

---

Support:

For more information on virus-specific protections, or to report a suspected vulnerability or threat, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Are there any limitations to the MetaDefender Core scan engines?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

Yes.

Certain custom engines have limitations in terms of OS compatibility, maximum file size allowed for scanning, and supported file types.

The latest list of limitations is outlined below.

File Size and OS Limitation

Engine Name	*File Size Limitation*	*OS Limitations*

Engine Name	*File Size Limitation*	*OS Limitations*
Antiy	900 MB	
Emsisoft	4 GB	
QuickHeal	2 GB	
RocketCyber Linux		Does not function on Redhat and CentOS versions earlier than 8.x.
Aurora		Does not function on Redhat and CentOS versions earlier than 8.x.
Nano AV		Does not function on Amazon Linux 2023
SentinelOne	4 GB	
ClamAV	5 MB (DeepScan disabled) Max scan size (DeepScan enabled)	

Minimum System Resource Requirements

Engine Name	*Minimum System Resource Requirements* —	*OS*
ClamAV	2GB + memory	Windows and Linux
IKARUS	1GB shared memory in docker	Linux

> Warning:
> ClamAV do not work on Redhat 8,9 if FIPS is enabled

> **Please Note:**
> For IKARUS, `--shm-size=1gb` should be added when starting the docker.
>
> If you're using K8s, the section below should be specified in the yaml file

Please Note:

AhnLab's engine is built several times a day. The main daily build is done at 6 PM Korea Standard Time (KST) the day before its version date. Because of this, the DB update may sometimes show the next day's date — this is normal and follows the vendor's process

Code · yaml

```yaml
spec:
  volumes:
  - name: dshm
    emptyDir:
      medium: Memory
  containers:
  - volumeMounts:
      - mountPath: /dev/shm
        name: dshm
```

Supported File type for AI engines engines can be founded here

Additional Installed Services

Engine Name	***Installed Applications***
NETGATE	Spy Emergency (an icon will display on desktop)

Engine Name	**Scan result**
Bitdefender	Password-protected file's result will be returned "Not Scanned" for this engine.

Please Note:

This list is accurate for the latest packages of custom engines.

Support:

If you need help navigating MetaDefender scan engine limitations, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Can MetaDefender Core block renamed files (such as

renamed .exe files)?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

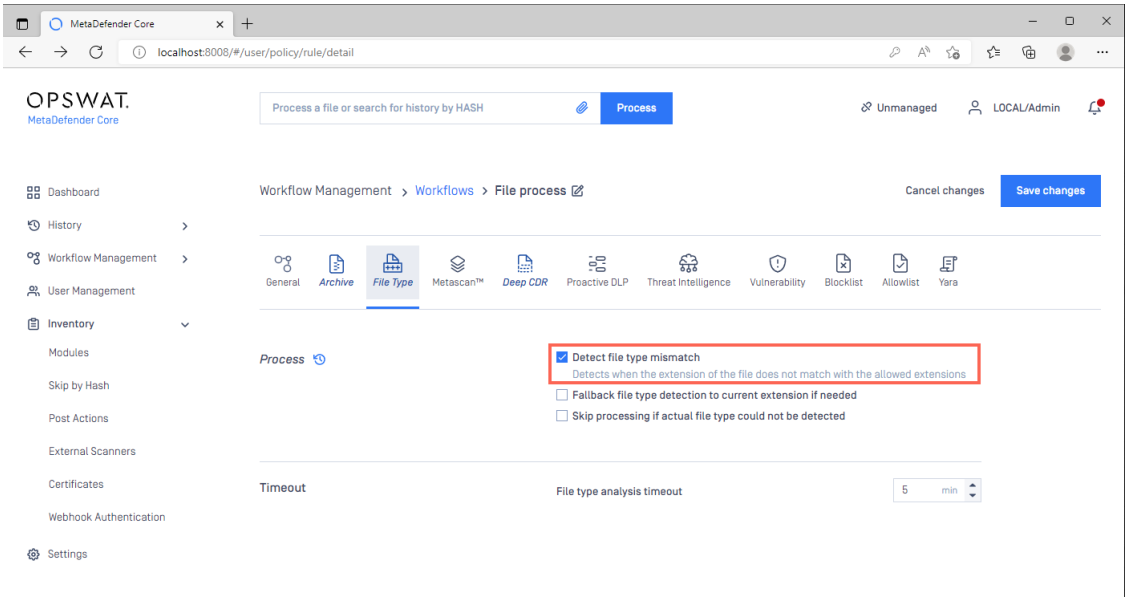
Yes.

Thanks to MetaDefender Core's built-in **File Type Verification Engine**, files that have manually or otherwise had their extensions altered (e.g. a *.exe* file disguised as a *.txt*) can be processed according to their actual format.

Pro Tip:

Often, altered file types can be an indicator of malicious intent, so it is highly recommended that users consider enabling this feature as part of their daily file security workflow.

To configure this feature, please go to your **MetaDefender Core Management Console>workflow Management>Workflows>{Your Relevant Workflow}>File Type** to enable detection as illustrated in the screenshot below.



Core has a first option to detect and block on file type mismatch based on true file type detection.

Detect file type mismatch:

- When enabled, this detects discrepancies between the file extension and the actual file format.
- These files will be blocked and Post Actions can be configured to further process these files as needed.
- Post actions are highly configurable, so specific remediation steps (in the form of policies/rules) can be customized per target/actual file type.

If customer decides to process mismatched file then enabling the third option will allow CDR engine to sanitize files

based on file extension when needed.

Fallback file type detection to current extension if needed:

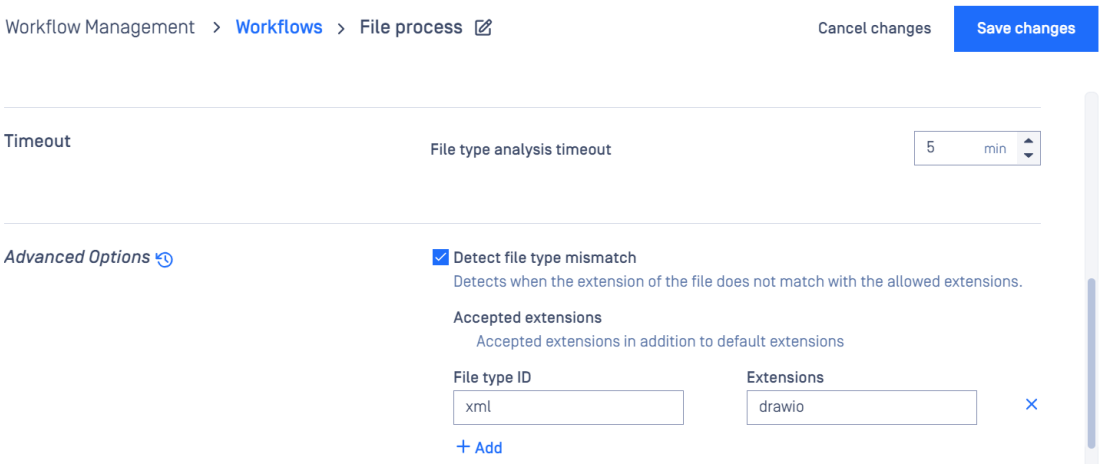
- When enabled, the File Type Verification Engine will work in conjunction with the Deep CDR module to detect and sanitize files according to their true file type.
- This means Deep CDR will only attempt to sanitize according to the given extension in the event that 'true type' sanitization fails.

If the true file type could not be detected for whatever reasons (relevant engine/module encounters issues while processing etc.) then we have a second option to skip and block the processing for such files.

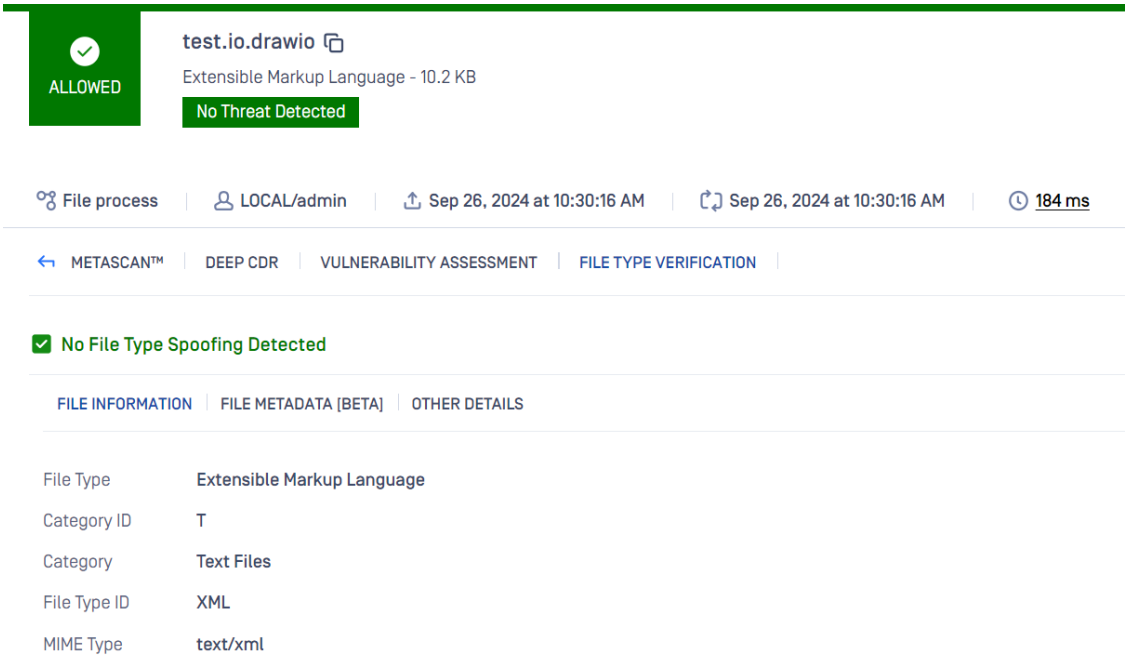
Skip processing if actual file type could not be detected:

- When enabled, this will produce a **Not Scanned** result in cases where the File Type Verification Engine is unable to detect the true file type.
- This should only occur when the engine is crashed/disabled/timed-out/canceled.

Customize mismatch detection settings:



If a file is detected with a specific file type ID but can have a different extension, the user can configure it as shown above to allow possible extensions for that file type. The file type ID can be seen on the scan results page.



Support:

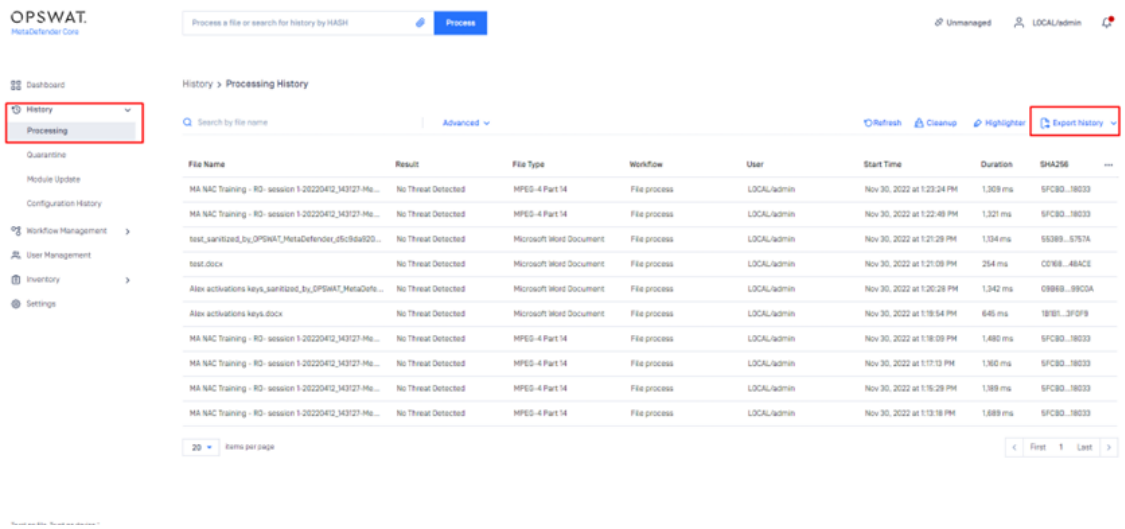
If you have followed the instructions above but are **Having Difficulty Configuring the Filetype Mismatch Feature**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How can I export the processing history from the Core console?

Check Your Version:

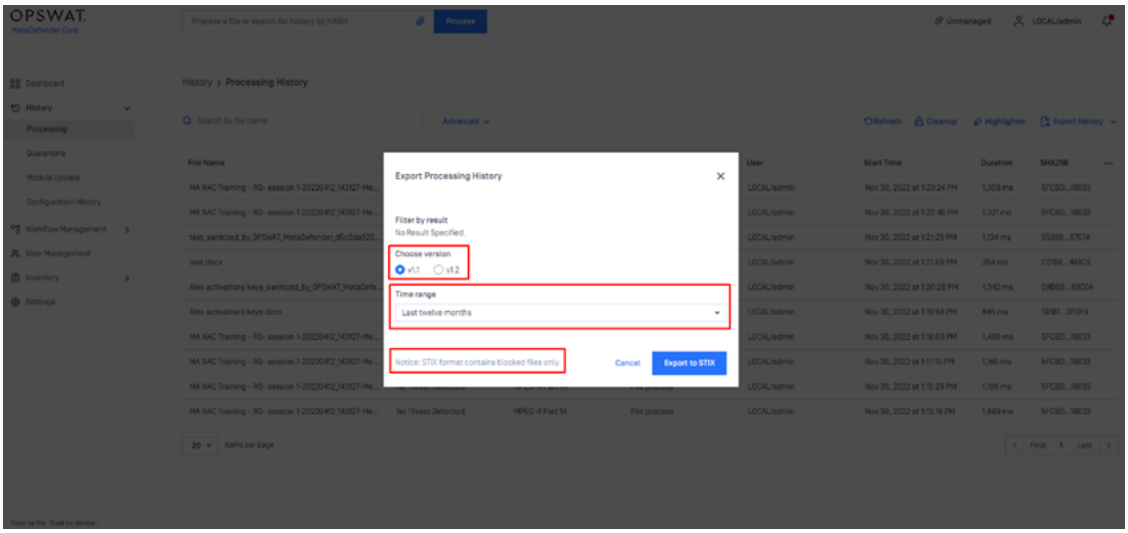
This article applies to all MetaDefender V5 releases deployed on Windows and Linux systems.

To export the processing history from any of the MetaDefender Core deployments users will have to navigate to the Core console\History\Processing tab and select the “export history” tab.



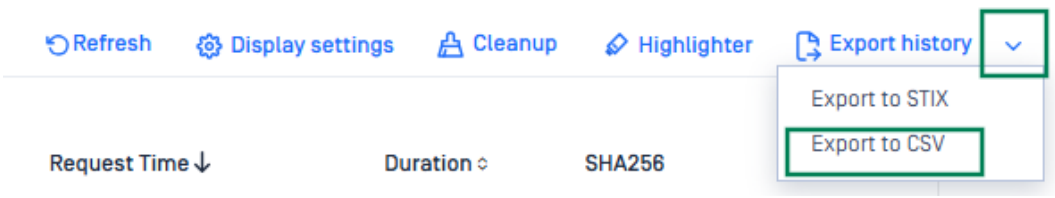
Please take note: STIX format contains blocked files only. A few options are available for the users prior to exporting such as:

- Version
- Time Range



How to export to a CSV file?

1. Click the drop-down arrow and select **Export to CSV**.



Support:

If **Further Assistance** is required, please follow the instructions on How to Create Support Package With Bundle Tools?, then proceed to log a support case or chat with our support engineer.

How to Utilize YARA Rules with MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows and Linux systems.

Overview

The YARA engine in MetaDefender Core scans files against custom rules that you provide, in addition to the curated rule set delivered with the engine. A common point of confusion is that placing a `.yara` file on the server is **not** enough on its own — a custom rule will only match scanned files when **all** of the following are true:

- The rule is reachable through a configured **YARA source**
- A **YARA package** has been generated after the source was added or changed
- The **YARA engine is enabled** in the workflow that scans the file
- The **"Yara Rule Matched" verdict is not treated as Allowed** in that workflow (this is the default behavior and is the most common reason a correctly-written rule appears to do nothing)

This article walks through the full activation flow, plus verification, common gotchas, and example rules to get started with.

Prerequisites

- Administrator access to the MetaDefender Core Management Console (default URL is the server's IP or hostname on the nginx port, `8008`)
- A syntactically valid `.yara` (or `.yar`) rule file. Validate rule syntax with the `yara` command-line tool before adding it as a source if you are unsure
- For local-directory sources: write access to a folder on the MetaDefender Core server
- For network sources: an HTTP or HTTPS URL that returns a `.zip` archive of one or more rule files

Activation Steps

Step 1 — Stage the rule file on the server

Create a dedicated folder on the MetaDefender Core server (for example, `C:\YARA\custom\` on Windows or `/opt/yara/custom/` on Linux) and place the `.yara` rule file inside it. The folder may contain multiple rule files; all of them will be loaded.

Important: When you add the source in the next step, you point MetaDefender Core at the **folder**, not at the individual

rule file. Pointing at a file directly will not work.

Step 2 — Add the folder as a YARA source

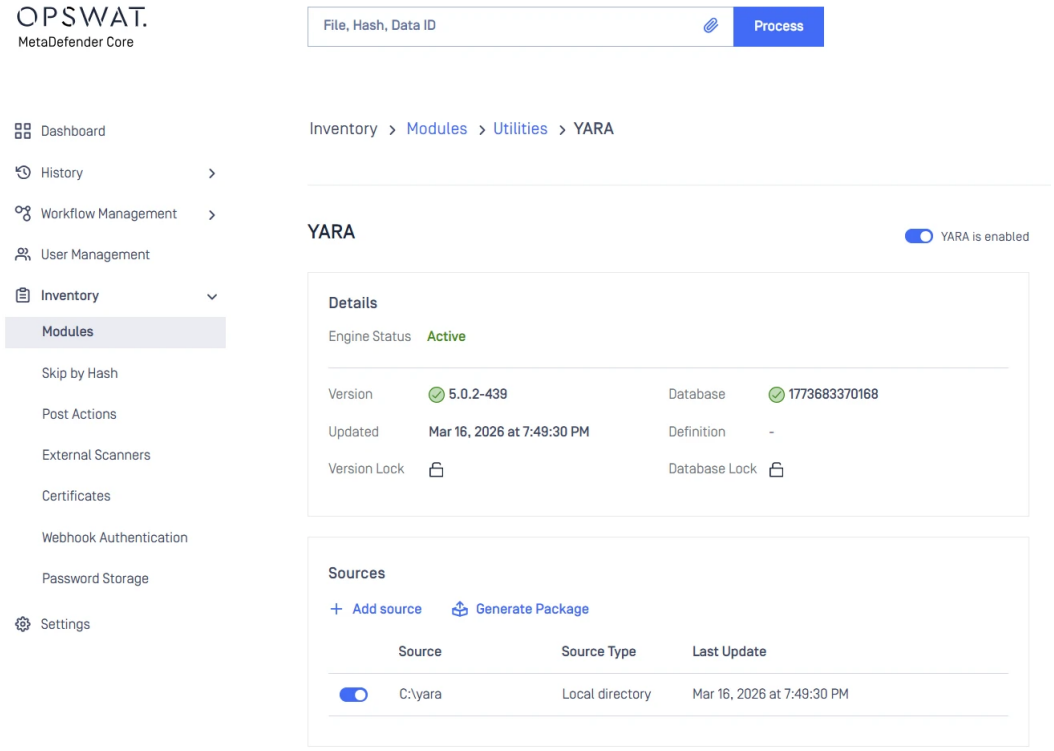
In the Management Console, navigate to:

Inventory Modules Utilities YARA

Click **Add source**. Two source types are supported:

- **Local directory** — the absolute path to the folder containing your rule files on the MetaDefender Core server. MetaDefender Core creates an internal copy of the directory contents for the engine to use
- **Network (HTTP/HTTPS)** — a URL pointing to a `.zip` archive of rule files. The archive is downloaded and its contents are used

Save the source. You can enable or disable each source individually; only enabled sources are included when the package is generated.



Step 3 — Generate the YARA package

Still on the **Inventory Modules Utilities YARA** page, click **Generate package**. This compiles all enabled sources into the active rule package that the engine uses at scan time. The generation start time is displayed next to the button, and the status can be polled.

Note: Adding, modifying, or removing a source does not take effect until you generate a new package. Every rule change requires a regeneration.

Step 4 — Enable YARA in the workflow

The YARA engine must be enabled in the workflow that will scan the file. In the Management Console, navigate to:

Workflow Management → Workflows → [select your workflow] → YARA tab

Confirm the YARA engine is enabled for that workflow, then save.

Step 5 — Make matched files actually blockable (the common gotcha)

By default, MetaDefender Core treats "**Yara Rule Matched**" as an **Allowed** verdict at the workflow level. That means even if your rule fires correctly, the file is still allowed through and the case looks like "YARA isn't working" from the client side.

To make matches actionable:

1. Open the workflow template (**Workflow Management → Templates**) or the workflow rule, and go to the **General** tab
2. Locate "**Override scan results classified as allowed**"
3. **Uncheck** the "**Yara Rule Matched**" entry so it is no longer classified as allowed
4. Save the template/rule

After this change, files matching a YARA rule will be processed as blocked (subject to your other workflow settings).

Verification

To confirm the activation worked end to end:

1. Take a sample file that you know matches your rule (or craft a small test file whose contents match the rule's `strings` and `condition`)
2. Submit it through the workflow you configured in Step 4 — either via the Management Console scan page or via the REST API
3. Open the scan result and expand the YARA section. A successful activation shows the rule name under the YARA engine's results
4. If Step 5 was applied, the overall verdict for the file should be **Blocked** rather than **No Threat Detected**

If the rule does not match, the most common causes are: the package was not regenerated after the source was added (Step 3), the source path pointed at the file instead of the folder (Step 2), or the file did not actually traverse the workflow you configured (Step 4).

Example YARA Rules

The following examples can be used to test the activation flow or as starting points for custom rules. Save any of these into a `.yara` file inside the source folder configured in Step 1.

Simple string match

Code

```
rule HelloWorld
{
    strings:
        $my_text_string = "Hello, World"
    condition:
```

```
        $my_text_string
}
```

Case-insensitive string match

Code

```
rule CaseInsensitiveStringDetect
{
    strings:
        $a = /malware/i
    condition:
        $a
}
```

Binary pattern match

Code

```
rule BinaryPatternDetect
{
    strings:
        $a = { DE AD BE EF }
    condition:
        $a
}
```

Multiple strings match

Code

```
rule MultipleStringsDetect
{
    strings:
        $a = "string 1"
        $b = "string 2"
    condition:
        $a and $b
}
```

Regular expression match

Code

```
rule RegexpDetect
{
    strings:
        $a = /[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+/
    condition:
```

```
        $a
}
```

Managing YARA Sources via REST API

For automation or for environments managed through OPSWAT Central Management, the same operations are available through the MetaDefender Core REST API:

- `GET /yara/sources` — Retrieve all configured YARA sources
- `PUT /yara/sources` — Replace the YARA source configuration (overwrites existing entries)
- `POST /yara/package/generate` — Trigger a new package generation
- `GET /yara/package/status` — Poll the status of the current package generation

Info

If you need further assistance, don't hesitate to create a support case or chat with our support engineer.

What is the Design of MetaDefender Core and Its Protective Measures?

MetaDefender Core functions to ensure the complete safety of the files you receive, preventing any potential activation or triggering upon reaching your machine.

During the scanning process, files sent to MetaDefender Core are temporarily stored in a designated folder. These files are renamed randomly, and their extensions are removed to eliminate any risk of activation.

MetaDefender Core scans the files as raw data, safeguarding against any potential activation during the scan. This method allows for comprehensive analysis without engaging any potentially harmful functions the files may contain.

Once the scanning process is complete, the files are immediately deleted. The duration of the scan can vary, typically ranging from a few milliseconds to several minutes, depending on the size and type of the files.

This mechanism ensures the highest level of safety and reliability, as no files can be triggered or activated during their time on your system.

What does the items sanitized/removed count in Deep CDR sanitization report mean?

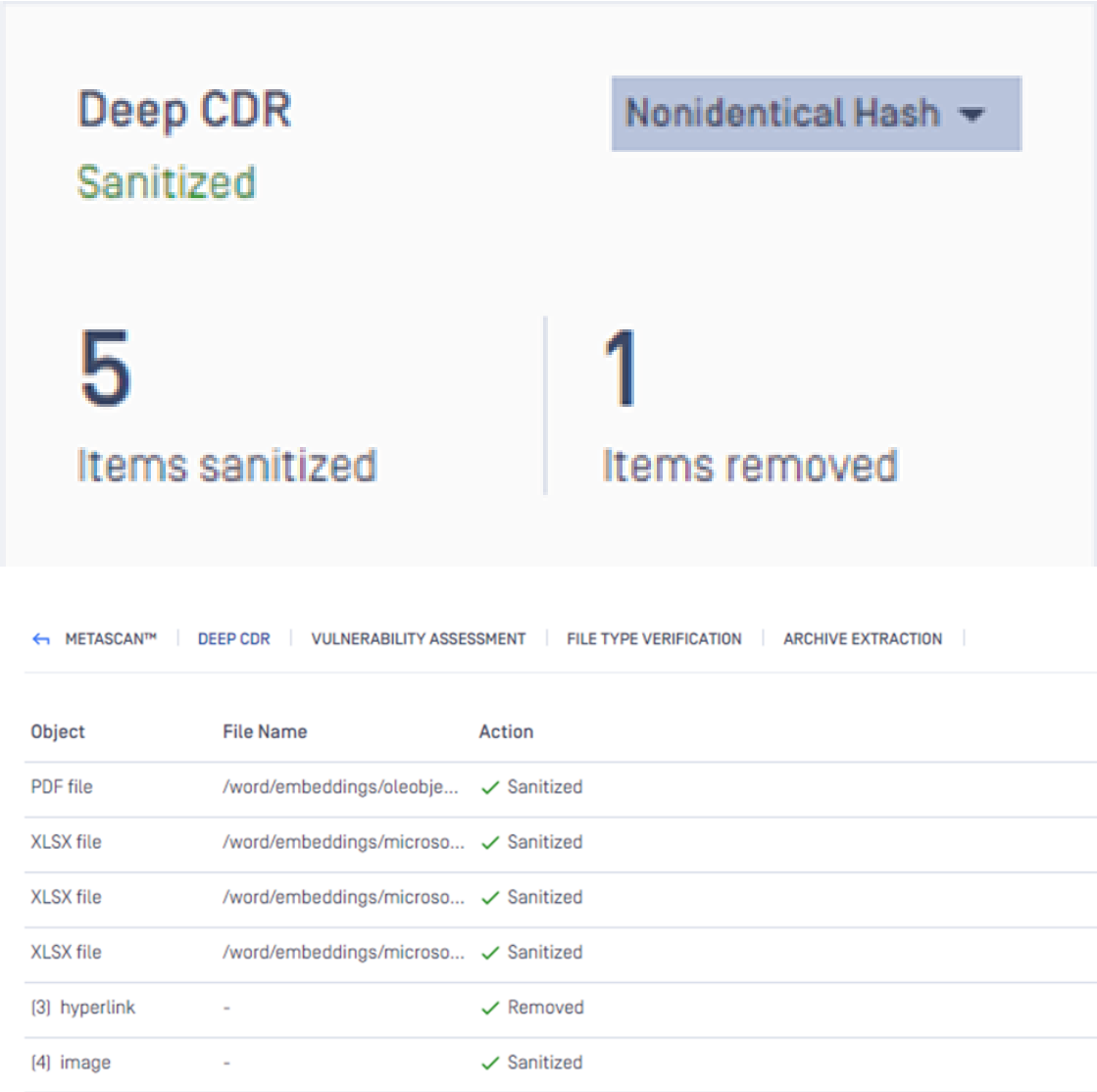
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

After sanitizing a document using Deep CDR, in the processing result afterwards, the number of sanitized/removed

items are counted and listed.

Take the following example: Deep CDR sanitization result for a DOCX document containing in-line images, hyperlinks, and embedded PDF, XLSX files.



The objects are counted as a group when they have the same result or meaning. In the above example, when four images are sanitized successfully, or when 3 hyperlinks are removed, they are grouped together.

For the embedded objects, if Deep CDR's Recursive Level config is turned on (and bigger than 0) and these objects are sanitized, then each file usually has a different result (i.e., a DOCX file, an XLSX file, or a different sanitization detail). Therefore, the embedded objects are separated to show the specific detail of each recursive sanitization.

Support:

For further assistance regarding MetaDefender Core Deep CDR, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why image sanitization doesn't report any object in details?

The most common way attackers hide malicious code in images is through a technique called steganography, where they secretly embed code within the pixels. Later, they use a special method to extract and run the hidden code. Since it's almost impossible to spot this hidden code without knowing exactly how and where it's concealed, Deep CDR doesn't try to detect it. Instead, Deep CDR focuses on disrupting the image's pixel data so the attacker's method for retrieving the hidden code no longer works. In this process, Deep CDR doesn't remove or sanitize any object in the image.

For more image attack vectors and how Deep CDR can help, please refer to this blog post: How Emerging Image-Based Malware Attacks Threaten Enterprise Defenses

Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

What are the requirements to process XDW/XBD/XCT files with CDR?

Bellow you can find the entire list of requirements:

- Version DocuWorks 9 or newer on the system to use this feature.
  - Deep CDR Supported File Types
  - DocuWorks
  - DocuWorks 9.1
- The trial versions will not work, a paid version is necessary.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

How do we utilize API uploads so the correct file type and sanitization occurs every time?

Check Your Version:

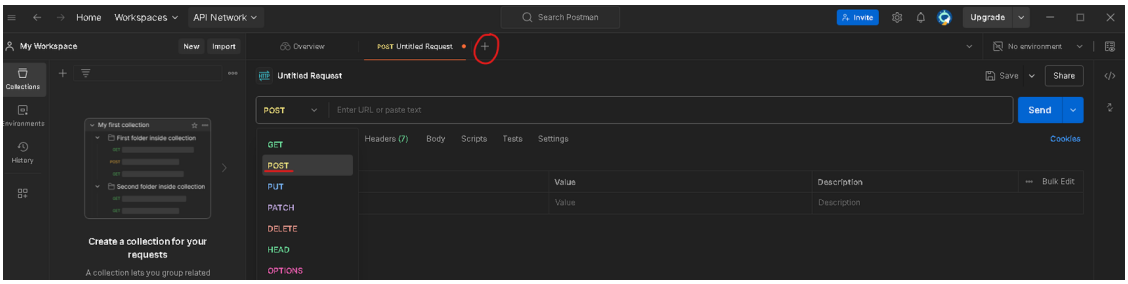
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Use the ‘binary’ upload method when sending files to Core for scanning via API. If you add files to the body of a ‘Post’ API call using another method such as ‘form-data’, the content of the file that our Core server receives gets

modified. This can result in our system incorrectly identifying the file type and being unable to fully sanitize the files. The 'binary' method is the only supported method for attaching files to the body of a 'Post' API call.

Example using Postman:

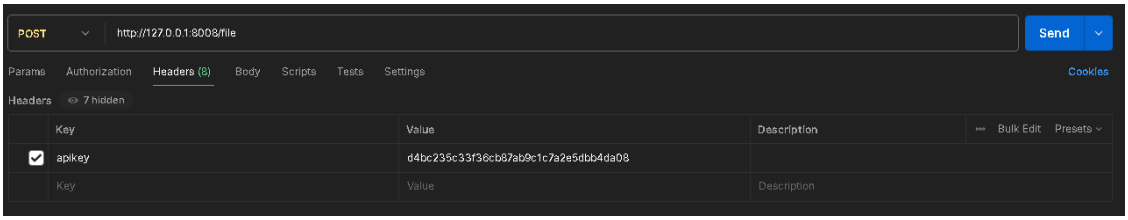
1. Press the + Button at the top of the Postman tool to begin a new API request.
2. Select 'Post' as the call type from the drop down menu



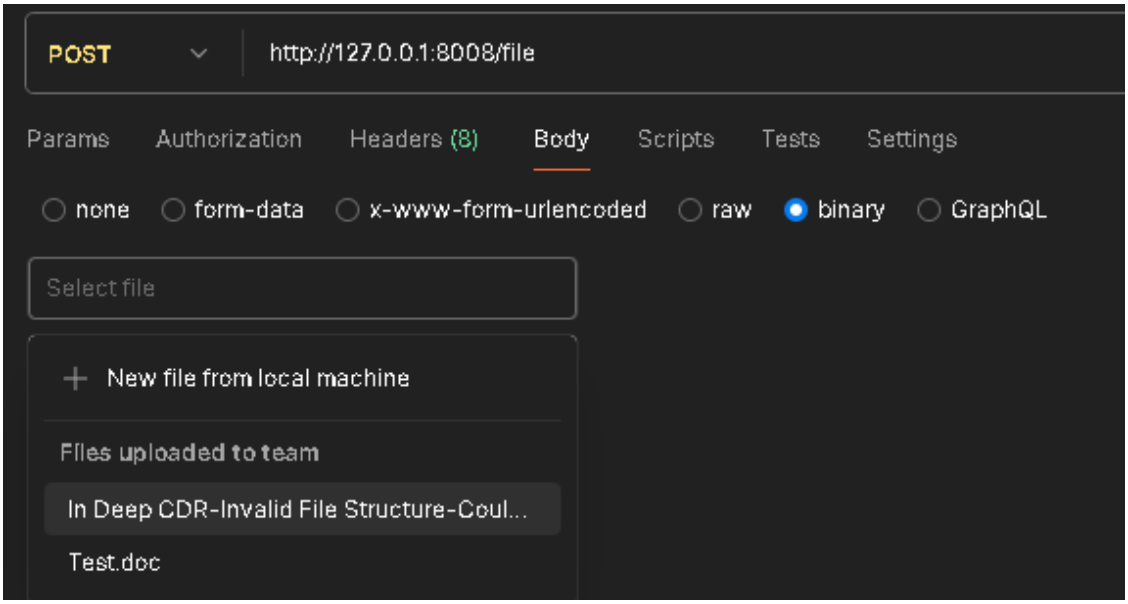
3. Enter the full URL for your MetaDefender Core server's API interface.



4. Add any optional headers outlined in POST - Analyze File (Asynchronous mode). Below, I will use the 'apikey' header



5. Change to the body of the API call and select 'binary' as your upload type.
6. Select your file by pressing the + button and browsing to the file you would like scanned



7. Press Send.

If the scan was successful, you should receive a server response of '200' along with the data_id associated with the file scan.



Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

What are the differences between AVs in OPSWAT and the other services?

When comparing antivirus (AV) scanning capabilities, it's essential to understand how configurations and engine versions impact detection performance and reliability. OPSWAT products are specifically designed to protect critical infrastructure, focusing on stability, privacy, and minimizing false positives. This approach differs from other services, which may prioritize higher detection rates at the cost of increased false positives or reduced data privacy. Below, we explore the key differences in configurations and engine versions between OPSWAT and other services.

Configurations

Some antivirus engines have a cloud scanning feature that is disabled by default in OPSWAT products. Enabling this feature can significantly increase detection capabilities, but it requires sending data to the vendor's servers for processing. This trade-off may have privacy and security implications.

- **Engines with Cloud Scan Features**. OPSWAT disables cloud scanning to prioritize data privacy and local scanning,

while other services may enable it to enhance detection.

Engine	Configuration
Lionic	Enable Cloud Scan
Sophos	Enable Cloud Scan
AhnLab	Enable Cloud Scan
Xvirus	Enable Cloud Scan

- **Other Configurations**

Additional engine configurations can also improve detection rates but may require adjustments depending on the specific use case.

Engine	Configuration
ClamAV	Enable PUP/PUA Detection
ESET	Enable Archive Extraction
Ikarus	Enable Archive Extraction
Huorong	Enable Archive Extraction
Crowstrike ML	Change Threshold Detection to lower level

Engine Versions

OPSWAT typically employs official release versions with selected features tailored for protecting critical infrastructure. This approach minimizes errors and ensures stability. In contrast, other services may use earlier or beta versions to maximize detection rates. While these versions might provide higher detection, they are more prone to false positives and instability.

Examples of Differences in Engine Behavior:

1. **Antiy, CrowdStrike Falcon and QuickHeal**

- OPSWAT: Cloud scanning is not used.
- Other services: May enable cloud scanning to improve detection.

2. **McAfee**

- OPSWAT: Artemis detection is excluded. Reference: Some of my virus detections are called 'Artemis' | McAfee Support.
- Other services: May include Artemis detection, increasing detection sensitivity but potentially introducing more false positives.

3. **Webroot SMD**

- OPSWAT: Uses the SMD version for scanning.
- Other services: May rely on File Reputation, which could provide broader detection at the cost of stability or accuracy.

By focusing on stability, data privacy, and critical infrastructure protection, OPSWAT's approach differs significantly from other services that may prioritize maximum detection at the expense of reliability or privacy.

Support:

If you have any further questions regarding the differences between AVs in OPSWAT and the other services, please proceed to create a support case or chat with our support engineer.

Does OPSWAT Use Security Checks on Our Code, Libraries, Credentials, etc?

Yes, we implement comprehensive security checks throughout our development lifecycle to ensure the safety and integrity of our products.

Please refer to the following security checks that we conduct:

1. **Static Application Security Testing (SAST)**

- **SonarQube**: Integrated into the development process to analyze code quality and security. All issues are resolved before release.

2. **Dynamic Application Security Testing (DAST)**

- **Tenable**: Conducts weekly scans on live applications to identify vulnerabilities. All findings are addressed prior to release.

3. **Software Composition Analysis (SCA)**

- **Blackduck**: Scans dependencies daily to detect known vulnerabilities. Any issues are resolved before deployment.

4. **Secret Scanning**

- **Gitleaks**: Monitors each code commit for exposed credentials or sensitive information.

We also leverage open-source tools to enhance our security posture:

- **Trivy**: Scans containers for vulnerabilities and misconfigurations.
- **OWASP ZAP**: Conducts dynamic security testing for web applications.

Support:

If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

How to access and analyze detailed scan results for an infected file?

Check Your Version:

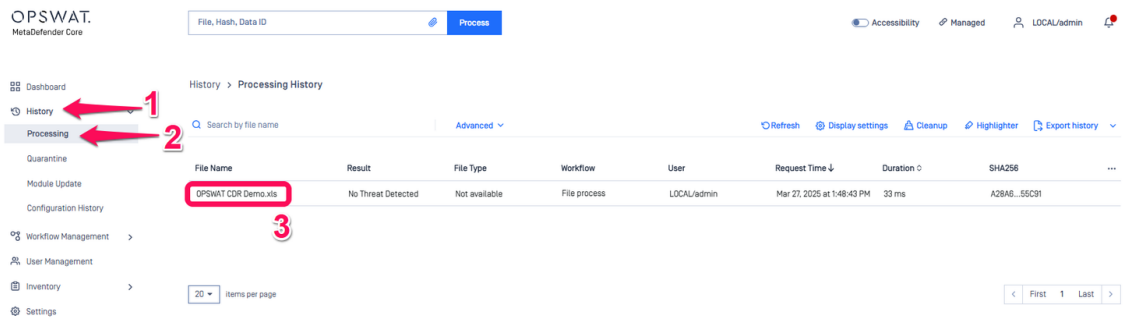
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

OPSWAT's MetaDefender Core provides in-depth scan results through both the management console and API, offering crucial insights into detected threats. Each scan result includes:

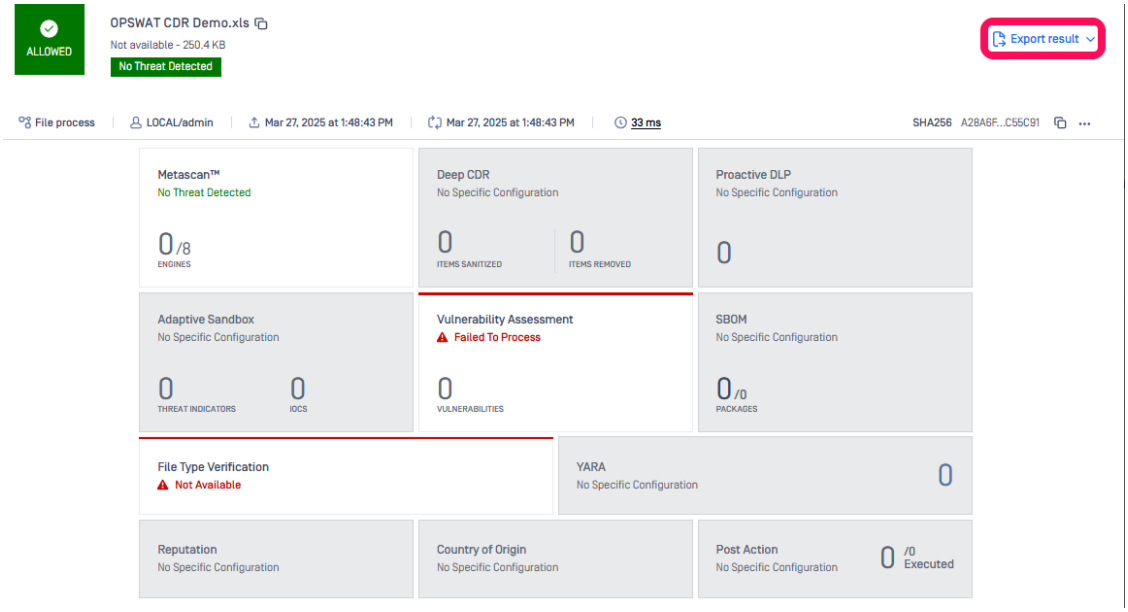
- **Scanning Engine**: The antivirus or security engine that flagged the file.
- **Threat Name**: The identifier assigned to the detected threat.
- **Metadata**: Additional details such as file hash, scan timestamp.

Using the Management Console UI:

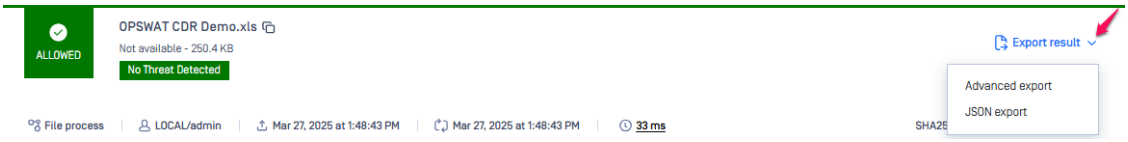
- Click the **History** tab on the left, select **Processing** from the drop-down list, and then click the file to view the scan result.



- Navigate to the top right corner and click on the **copy button** to retrieve the file hash. You can choose to export the file hash in SHA1, SHA256, or MD5.
- You can download the result in various formats. To save it as a PDF, click the **Export Result** button.



Or click the arrow button to export the file in your preferred format.



Using API Calls:

Refer to the API documentation at MetaDefender Core File Analysis API. Scans are done asynchronously, and each scan request is tracked by a **data ID**. You need to initiate file scans and retrieve the results using two separate API calls.

- **Upload a file to scan**:
  - Use the `POST /file` API endpoint to upload the file.
  - The response will include a `data_id`, which is a unique identifier for the submission. Example response: `{ "data_id": "61dffeaa728844adbf49eb090e4ece0e" }`
- **Fetch the analysis result**:
  - Use the `GET /file/{data_id}` API endpoint with the `data_id` obtained in the previous step.
  - Continue polling this endpoint until the `progress_percentage` in the response reaches 100%. The scan completion can be traced using the `scan_results.progress_percentage` value from the response. Example request:

Code · bash

```
curl --request GET \ --url 'http://localhost:8008/file/{data_id}' \ --header 'apikey: {apikey}'
```

- **Retrieve the scan results**:

- Once the scan is complete, you can retrieve the results using the `data_id` or the file's hash (e.g., MD5, SHA1, SHA256).

Scan results can be retrieved in JSON format, enabling further automation and integration with security workflows.

While MetaDefender Core provides high-level detection details, for a more comprehensive analysis, you can cross-reference the reported threat name with the corresponding antivirus vendor's threat intelligence database. These databases contain detailed descriptions, attack vectors, and mitigation strategies for known malware.

By leveraging both MetaDefender's scan results and vendor-specific threat intelligence, you can gain a deeper understanding of the nature of the threat and take appropriate remediation actions.

Support:

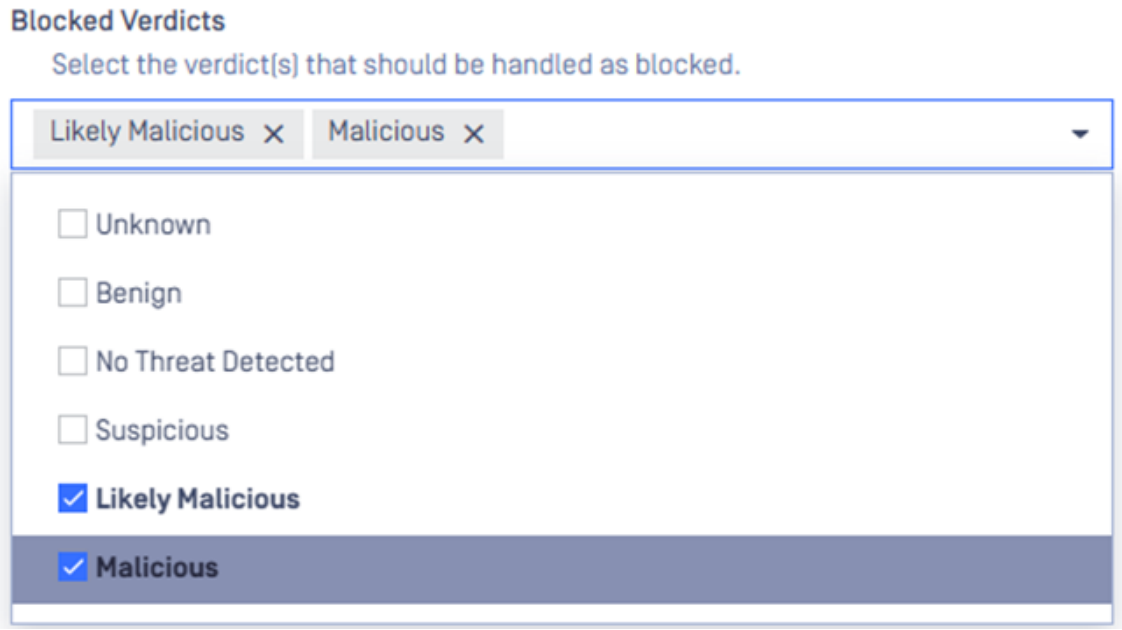
If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why MetaDefender Core blocks Sandbox Verdicts that are not marked to be blocked?

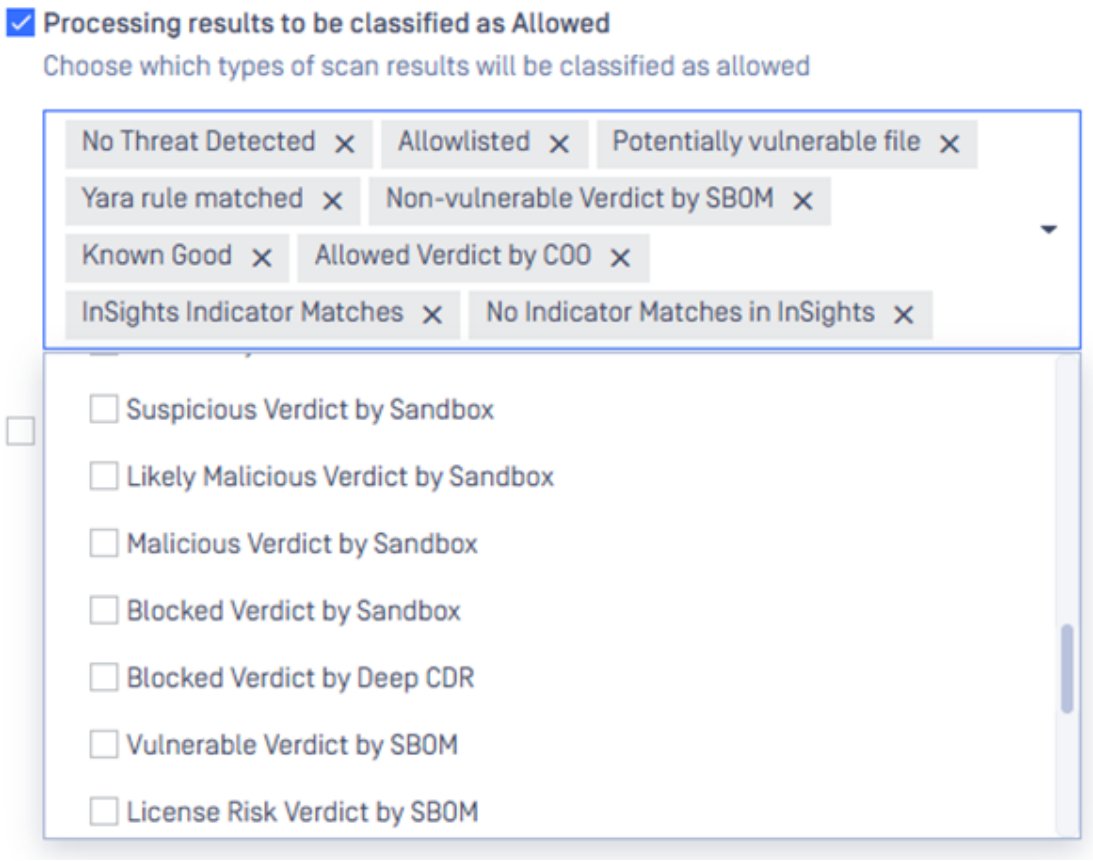
Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

By default, Sandbox Module for MetaDefender Core allows customizing the verdicts for which a scan should be marked as blocked:



However, it is very important to note that the following option under **Workflow** > **General** Tab will directly override the verdicts to be classified as Allowed, once activated:



What does that mean?

For instance, if Suspicious verdict is not marked to be blocked by Sandbox under the Engine Tab, but the **Processing results to be classified as Allowed** option is active without the **Suspicious Verdict by Sandbox** in the list of options, any Suspicious Verdict raised by Sandbox will directly be blocked, regardless of the individual engine blocking classifications.

*This is a security-first enhancement prioritized over the user experience with the main purpose of preventing accidental misconfigurations.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to scan multiple files at once with MetaDefender Core

By default, using MetaDefender Core on its own will allow scanning only one file at a time. We do offer the option to manually submit files for scanning, but this does not support folders or multiple files at once.

To achieve the later (i.e. scanning an external drive, for example), there are multiple ways:

1. Using MetaDefender Kiosk in integration with MetaDefender Core

2. Using MetaDefender Endpoint in integration with MetaDefender Core

3. Custom scripting: A custom-made application that browses through a directory, automatically sending files one-

by-one for scanning to MetaDefender Core

Support:

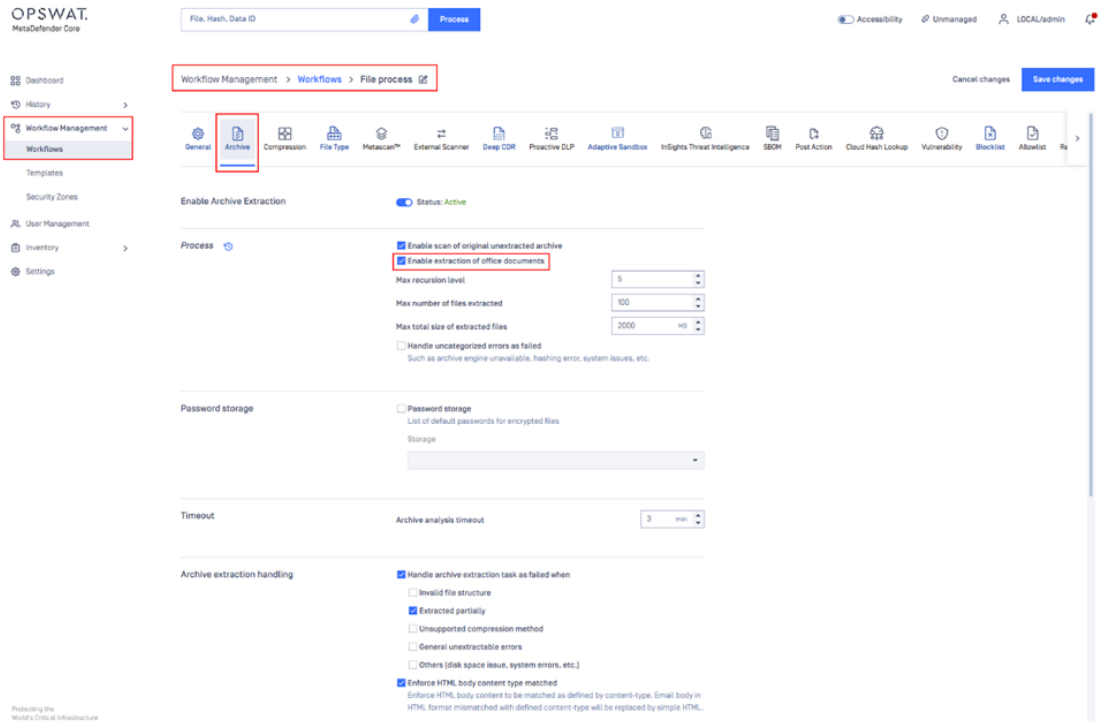
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Is MetaDefender Core able to extract, detect and block malicious embedded macros?

In short, yes! MetaDefender Core can extract and block files if malicious macros are embedded.

Macro objects are typically stored in Office Documents files such as **xlsm, docm, ppt** and so on, kindly make sure that the following option is enabled under your workflow, so that Core can extract Office Documents contents:

Workflow Management > Workflows > [any workflow] > Archive Tab > **Enable extraction of office documents**



Using the MetaScan Engines, it can detect the infected files and directly block them. For a more proactive approach, the DeepCDR Engine (What is CDR?) is able to deliver a clean version of the file by sanitizing/removing embedded data according to the configuration in place as per each file type, reconstructing the base file.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Does OPSWAT provide a file dataset that can be used to

evaluate MetaDefender Core's capabilities?

We do not offer pre-prepared test files for evaluating **MetaDefender Core's** capabilities. However, you may obtain safe, non-malicious test files from reputable third-party sources that specialize in antivirus and network performance testing. These files can be found through trusted file repositories available online.

If you are looking to test the malware detection capabilities, we recommend using the **EICAR test file**, which is a widely recognized standard for antivirus testing.

You can download the **EICAR test file** from the official **EICAR website**: Download Anti Malware Testfile - EICAR

⚠ **Disclaimer:** The use of third-party test files is **at your own discretion**. Please ensure compliance with your organization's security policies before downloading and using test files.

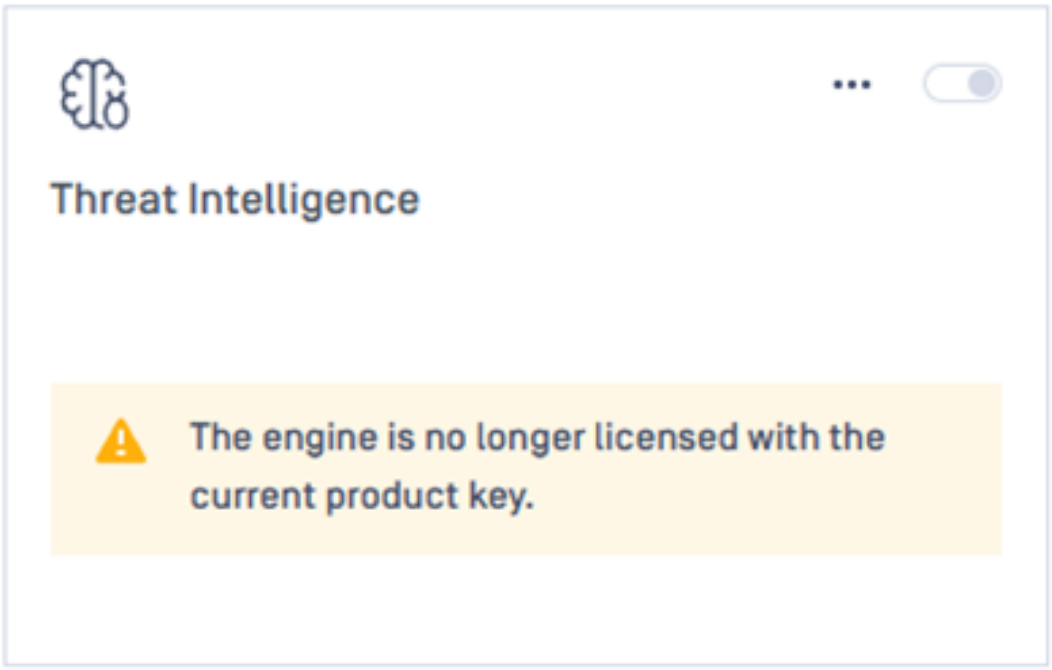
Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

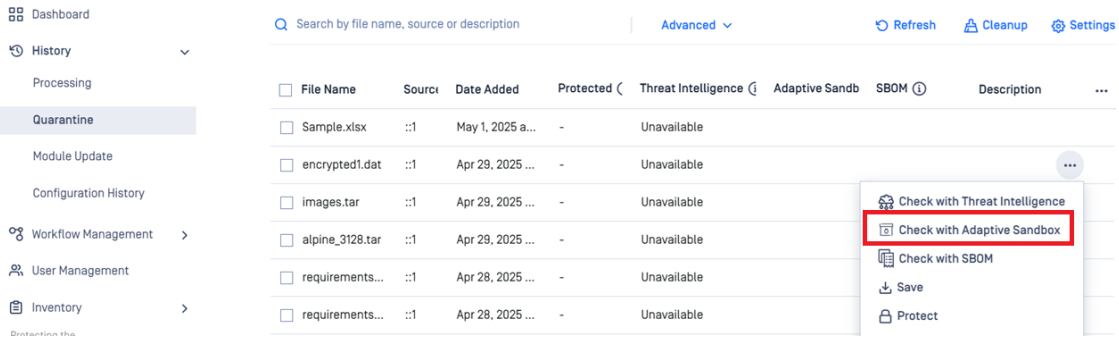
How would discontinuing Threat Intelligence affect customers?

- **Status**: The legacy Threat Intelligence engine in MetaDefender Core has already been deprecated, and its successor is now in development.
- **Legacy functionality**: The former engine functioned as a free utility for on-demand analysis of quarantined files, helping teams validate suspected threats after containment.
- **What's changing**: The new engine embeds advanced machine-learning threat pattern correlation directly in MetaDefender Core and works in sync with Sandbox to deliver zero-day detection; the discontinued quarantine-analysis workflow will not return.

Upon renewal, you may see this engine become "Unlicensed" when you activate a new license key. This simply happens because the engine is already deployed, but the new activation key doesn't include it anymore. You can simply ignore this status or just delete the engine package.



Functionality wise, you can still use directly from the quarantine section by leveraging our Adaptive Sandbox integration.



For details on how to deploy locally or how to configure a remote sandbox, see here: Sandbox

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

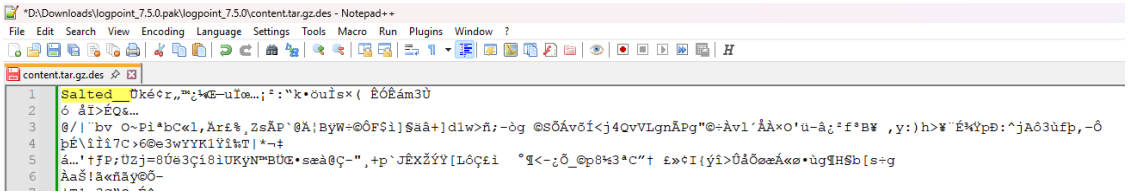
How to add custom detection for a file type?

Info:

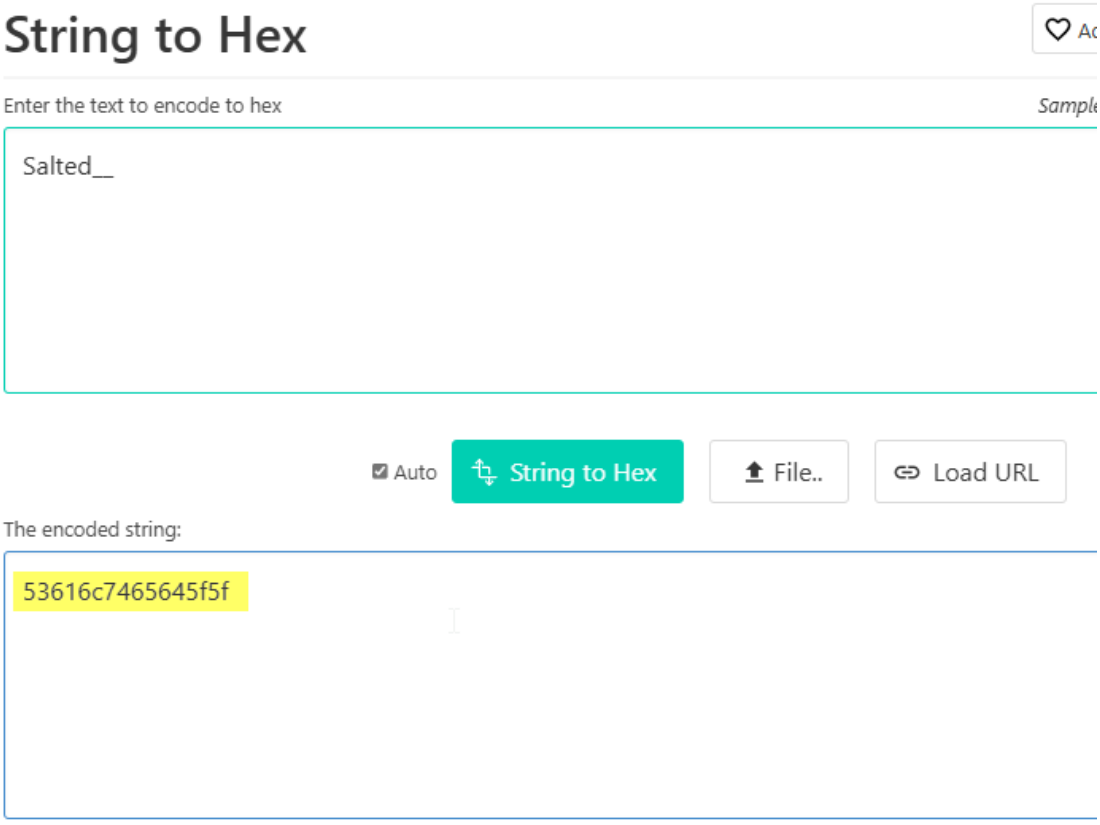
The custom detection is only supported from File Type version 7.4 and later

Here are the step-by-step instructions to add a custom detection rule:

- **Identify a signature pattern of the file type** you wish to add custom detection for. Take a .des file as an example. It is a file encrypted using OpenSSL's `enc` command with a salted password. Its well-known signature pattern is that the file content usually starts with `Salted__`. OpenSSL prepends this 8-byte header to indicate that the file was encrypted with a salted key derived from a password.



- Convert the signature pattern to a hexadecimal byte sequence
- You can use an online converter such as https



This hexadecimal byte sequence will be used to define the custom rule

- Open a Text editor and create a new text file
- Copy the following content into the new file

```
<CustomRule ver="1.0">
    <Info>
        <description>openssl enc'd data with salted password</description>
        <id>DES</id>
        <mime>application/octet-stream</mime>
        <group>OPENSSL_ENC</group>
        <extension>des, dat</extension>
        <score>0.9</score>
    </Info>
    <FrontBlock>
        <Pattern>
            <Bytes>53616C7465645F5F</Bytes>
            <Pos>0</Pos>
        </Pattern>
    </FrontBlock>
</CustomRule>
```

In this example, this rule is designed to detect files encrypted using **OpenSSL's `enc` command with a salted password**. Here's a detailed breakdown of what each important part of the rule does:

 `<Info>`**Section**

This section provides **metadata** about the custom rule:

`<description>openssl enc'd data with salted password</description>`

- Describes the rule's intent: identify files encrypted with `openssl enc` using a password and salt.

<id>DES</id>

- Internal File Type ID of the rule. It's labeled `DES`, but this ID is **arbitrary**—it could refer to the common usage of DES encryption in OpenSSL (e.g., `openssl enc -des -salt`), though other ciphers can also use the same salt header.

<mime>application/octet-stream</mime>

- Sets the **MIME type** for detected files. `application/octet-stream` is a generic binary file type.

`<group>OPENSSL_ENC</group>`

- Places this rule in the **OPENSSL_ENC** group, useful for organizing and classifying rules logically.

`<extension>des, dat<extension>`

- Assigns file **extensions** to this rule. Files with `.des` or `.dat` extensions that match the pattern will be recognized as matching this rule.

<Score>0.9</score>

- The score for this detection.

Important Notes:

The current use case is to turn an unknown (DATA) or not surely (non-DATA with score < 1.0) (detected by native rules of the engine) file type into a user-custom one with higher score.

There can be cases in which a file matches both a custom and a built-in rule. In order to prioritize the detection resulted from the custom rule, the custom rule should be defined with a high confidence score, e.g., 1.1.

<FrontBlock>

This section defines the **byte pattern** that identifies the file format: <FrontBlock>

`<Pattern>`

Specifies a signature pattern.

`<Bytes>53616C7465645F5F</Bytes>`

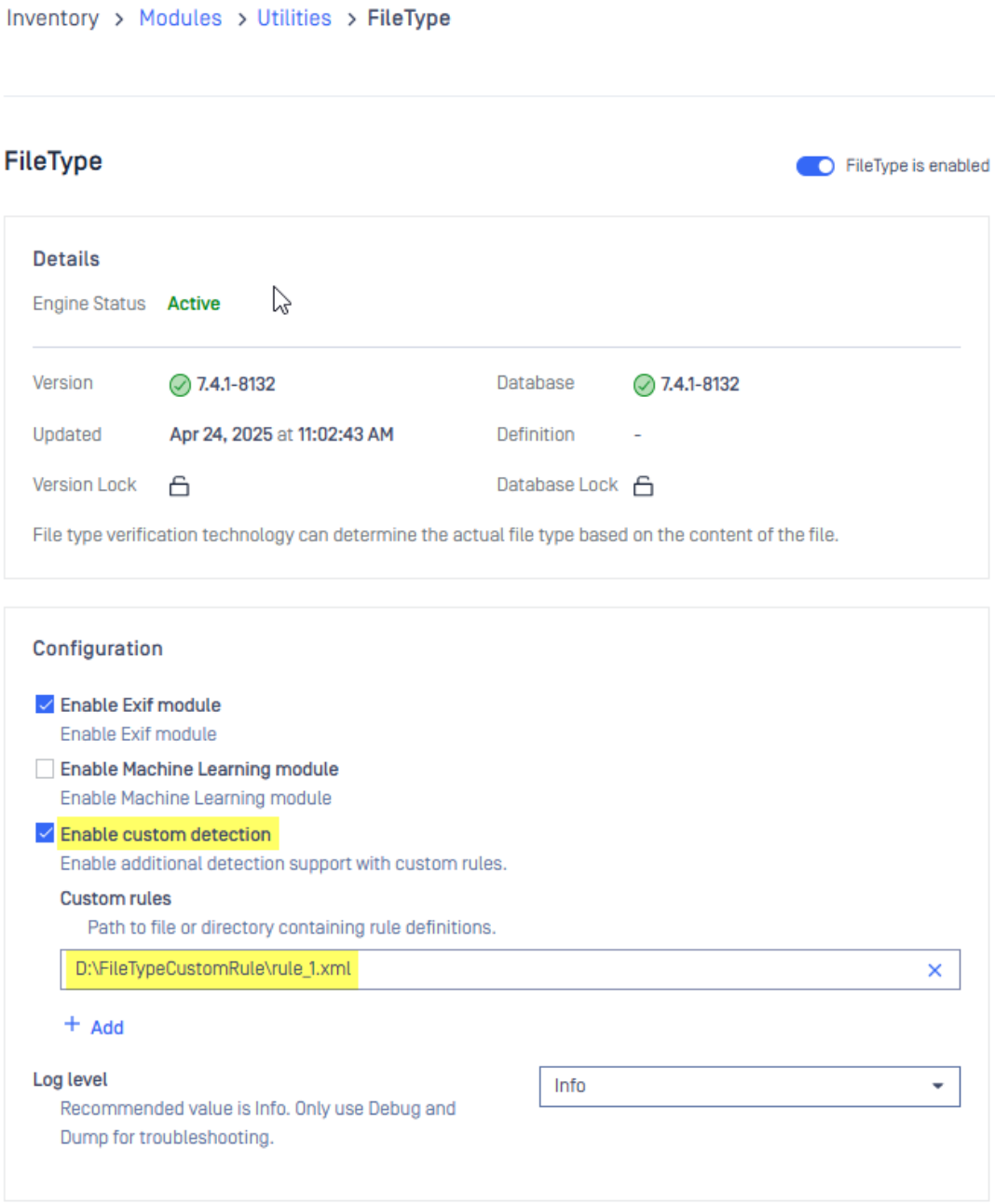
- This is a **hexadecimal byte sequence** in the step #2

`<Pos>0</Pos>`

- Specifies the **offset** in the file where the signature must appear. In this case, the signature must be **at the beginning** of the file (byte position 0).

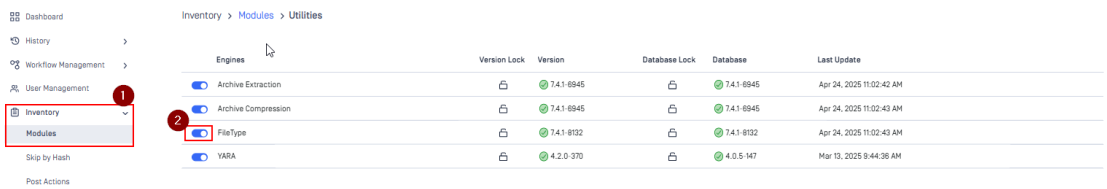
For details about the XML format with fields description, check Rule definitions

- Save the file with .xml extension
- Add the customer rule at MD Core UI > Inventory > Modules > Utilities > FileType engine > “Enable custom detection” > Input the path to the XML rule file

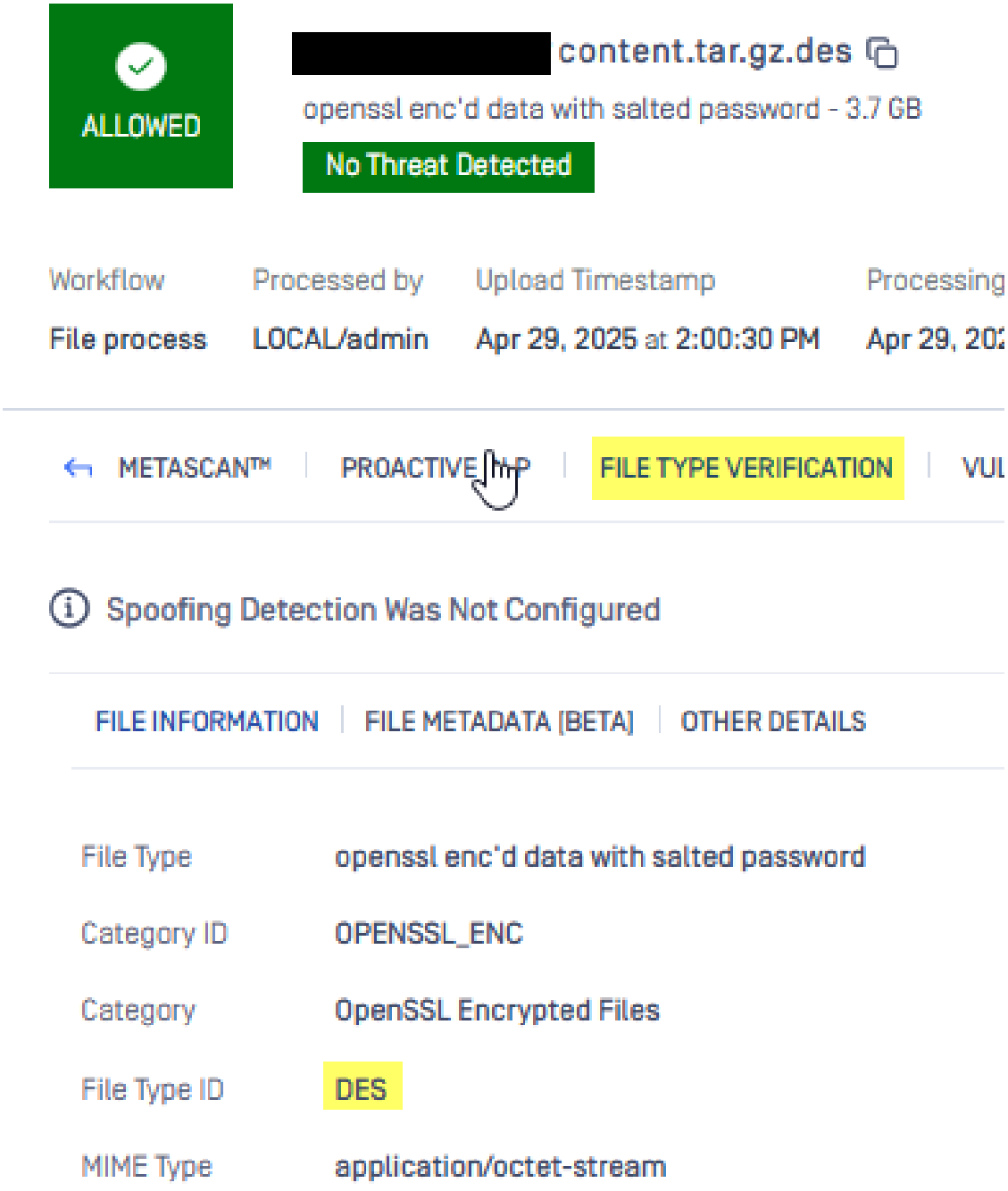


Check Enable custom detection to see how to enable custom detection

- Disable/Enable the FileType engine for the change to take effect.



- Scan the file with MD Core, you will see something like this in the File Type Verification section



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What can I do if MetaDefender Core shows a high number of files that failed to scan?

Check Your Version:

This article applies to all MetaDefender V5 releases deployed on Windows or Linux systems.

Context

This can be due to several different factors, some of which can be figured out and remediated on a DIY basis, and several of which can only be discovered by OPSWAT Support after you open a Support case. Instructions below.

Mostly it comes down to, one or more of Core's engines are offline or updating during the time the scan is performed.

Set the failure threshold

Under normal circumstances, all active engines scan every file.

However, there are some situations in which scanning might be impacted and a "failed to scan" result is observed. In such situations, the behavior may be caused by a number of factors:

- Some engines failed to scan, and exceeded the scan failure threshold settings in place on the Core side.
- Core itself has a fatal problem e.g. Core service crashed.
- Due to configurations:
  - Failed the scan when sanitization timeout or failed.
  - Failed the scan when archive extraction failed.

Please Note:

MetaDefender V5 includes an option that modifies the behavior of the product when a specific engine fails to scan - such as in the case mentioned above.

Here, MetaDefender Core can be configured to fail the scan if any of the engines report problems. In other words, it can toss out all partially complete scans.

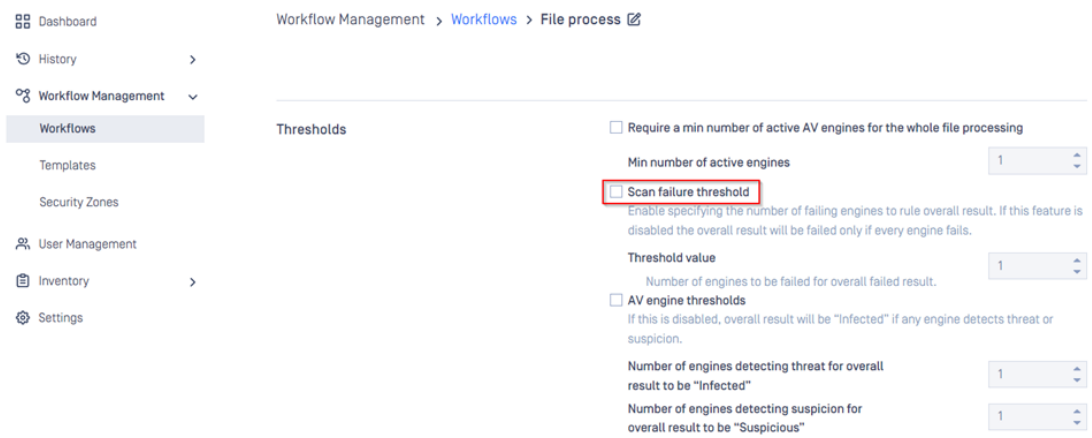
Failing to disable this option can give the illusion that MetaDefender Core is failing to scan large amounts of files.

MetaDefender Core V5 Releases

To disable this option on V5, follow these steps:

1. In your web browser, navigate to the MetaDefender Core **Management Console** at http://localhost:8008/ (you may need to log in under admin privilege)
2. On the left menu, navigate to **Workflow Management** **Workflow** to list all available rules
3. Click on a workflow you are using for the current scan to modify the setting
4. Click on the **Metascan** tab

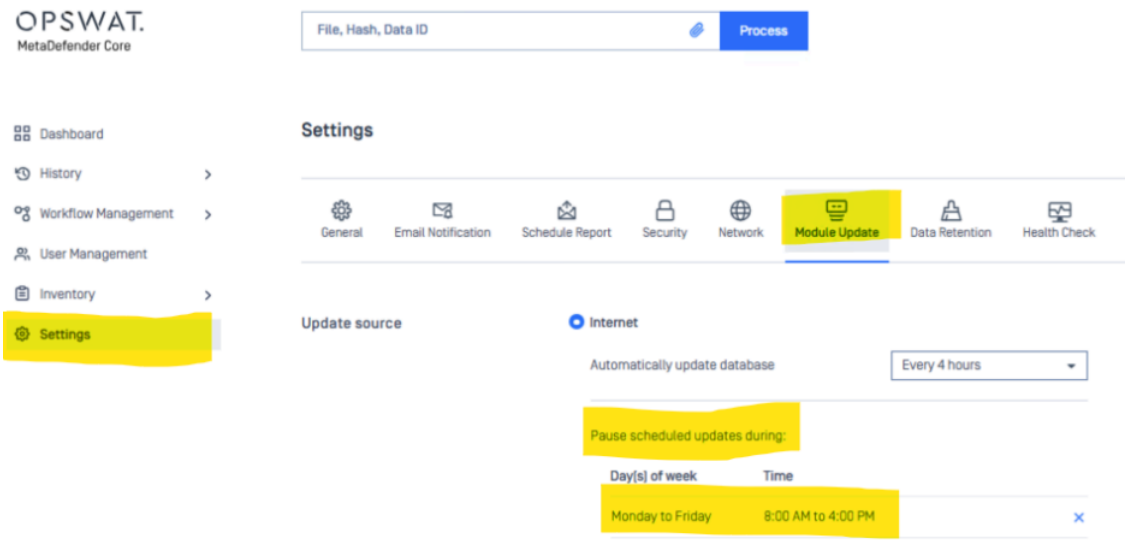
Uncheck the option **Scan Failure Threshold** shown below.



If that didn't fix it

With separate update windows for anywhere from 4 to 20 AV engines, and separate update windows for each of the utility engines, this could result in periods throughout the day where multiple engines are sporadically unavailable.

To minimize the impact of these updates, you can ensure that all engines are set to update **outside of peak hours**. This can be done from the Settings menu > Module Updates. Just pick one or more blackout windows to exclude the times when you expect the most scan volume, as shown below.



Dependencies

If the errors persist, it's possible Core may be missing a dependency. Typically, this will be the latest C++ redistributable.

Please install the latest C++ package for your platform from the Microsoft website and then restart the MetaDefender Core service.

Once Core is online again, please monitor whether the Failed to Scan errors persist.

If they do, it's time to open a Support case.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why did the required password pop up when scanning an unencrypted file?

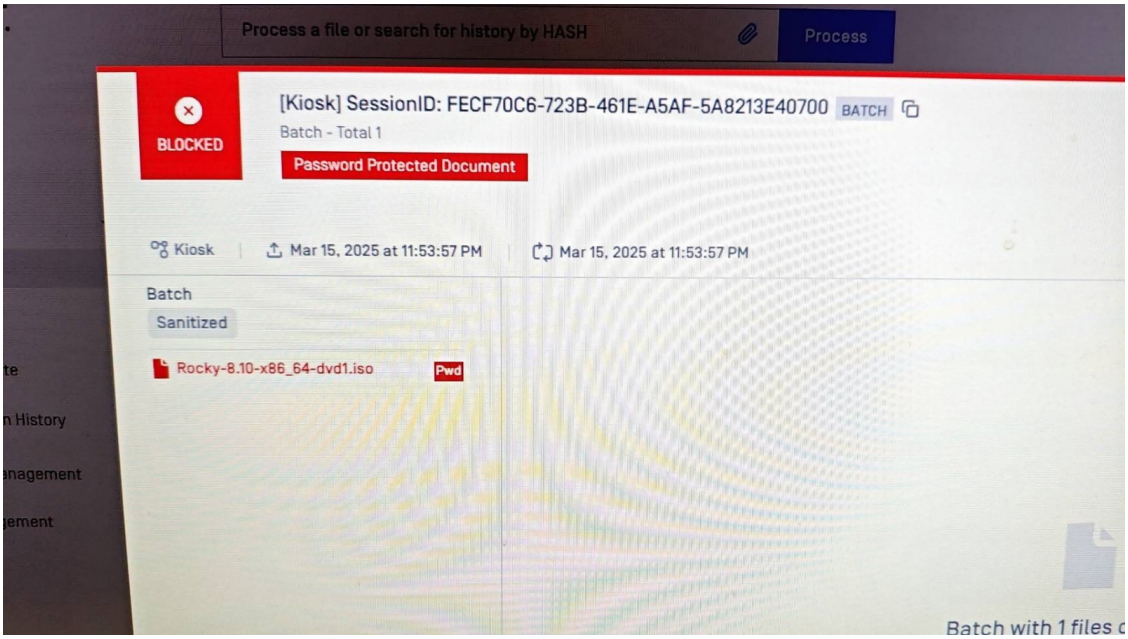
This article explains why MetaDefender Core or MetaDefender Kiosk may unexpectedly prompt for a password when scanning files that are not encrypted.

Issue:

During the scanning of a file of **Rocky-8.10-x86_64-dvd1.iso**

Which is the iso file from Download - Rocky Linux

As the file is being scanned, a password popup appears on Kiosk screen appearing that the password is needed for this file `multitoc.asc`. in order to continue scanning the file. This is happening due to the Rocky Linux preventing some users from viewing the contents of the file. (Image below)

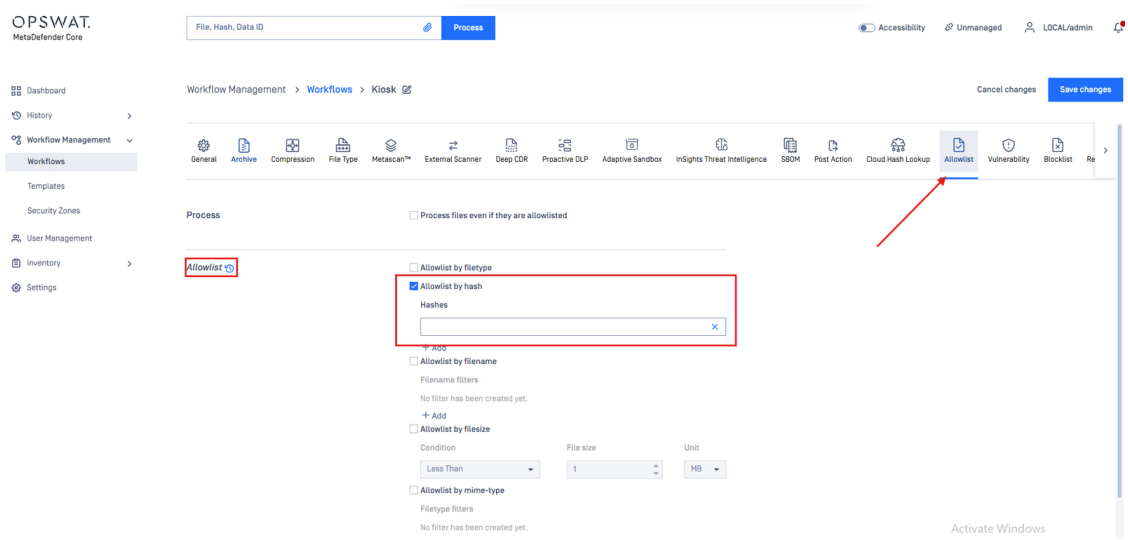




Current Workaround:

As `multitoc.asc` is an encrypted file included by Rocky Linux, and cannot be decrypted without a valid password, the recommended workaround is to **add the hash of this file to the Allowlist**. Doing so will instruct MetaDefender Core to **bypass scanning this specific file**, allowing the rest of the ISO to be scanned without interruption.

> **Warning:**
>
> Before proceeding with this step, please ensure that you are **fully confident that the file is from a trusted source** and does not pose any security risks. **This action should be taken at your discretion.**



Additional Information:

Due to content protection policies by the Rocky Linux distributor, certain files within their ISO may not be accessible for scanning or extraction. Since MetaDefender Core requires access to file contents for scanning, it cannot process

encrypted files like `multitoc.asc` without a password.

Unfortunately, in this scenario, **there is no alternative method to scan the encrypted file** unless valid credentials are provided.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to Test the Syslog Server functionality from MetaDefender Core?

Windows

Execute this in PowerShell.

UDP Syslog

Code · powershell

```
$syslogServer = "<syslog_server_ip>"
$syslogPort = 514
$message = "<34>Test Syslog message"

$udpClient = New-Object System.Net.Sockets.UdpClient
$udpClient.Connect($syslogServer, $syslogPort)
$encodedMessage = [System.Text.Encoding]::ASCII.GetBytes($message)
$udpClient.Send($encodedMessage, $encodedMessage.Length)
$udpClient.Close()
Write-Output "Syslog message sent."
```

TCP Syslog

Code · powershell

```
$syslogServer = "<syslog_server_ip>"
$syslogPort = 601
$message = "<34>Test Syslog message`n"

$tcpClient = New-Object System.Net.Sockets.TcpClient($syslogServer, $syslogPort)
$stream = $tcpClient.GetStream()
$writer = New-Object System.IO.StreamWriter($stream)
$writer.WriteLine($message)
$writer.Flush()
$writer.Close()
$stream.Close()
$tcpClient.Close()
```

```
Write-Output "Syslog message sent."
```

Linux

Using the Linux command below.

UDP Syslog

Code · bash

```
echo "<34>Test Syslog message" | nc -u -w1 <syslog_server_ip> 514
```

TCP Syslog

Code · bash

```
echo "<34>Test Syslog message" | nc <syslog_server_ip> 601
```

Info

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

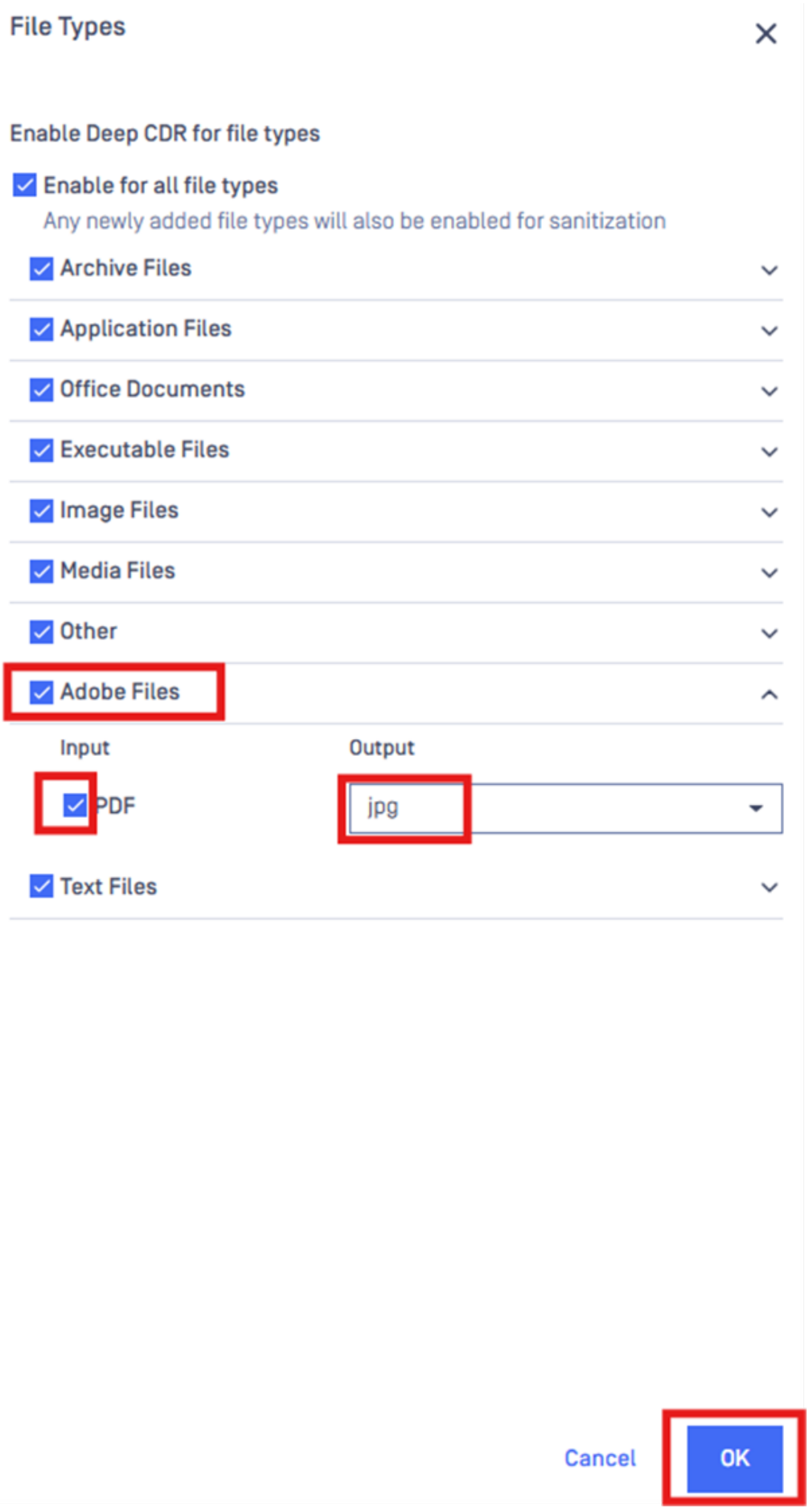
When and how to use “PDF to Image Configuration” Deep CDR feature?

Introduction:

This knowledge base article provides detailed instructions on how and when to use the “PDF to Image Configuration” feature which is part of Deep CDR.

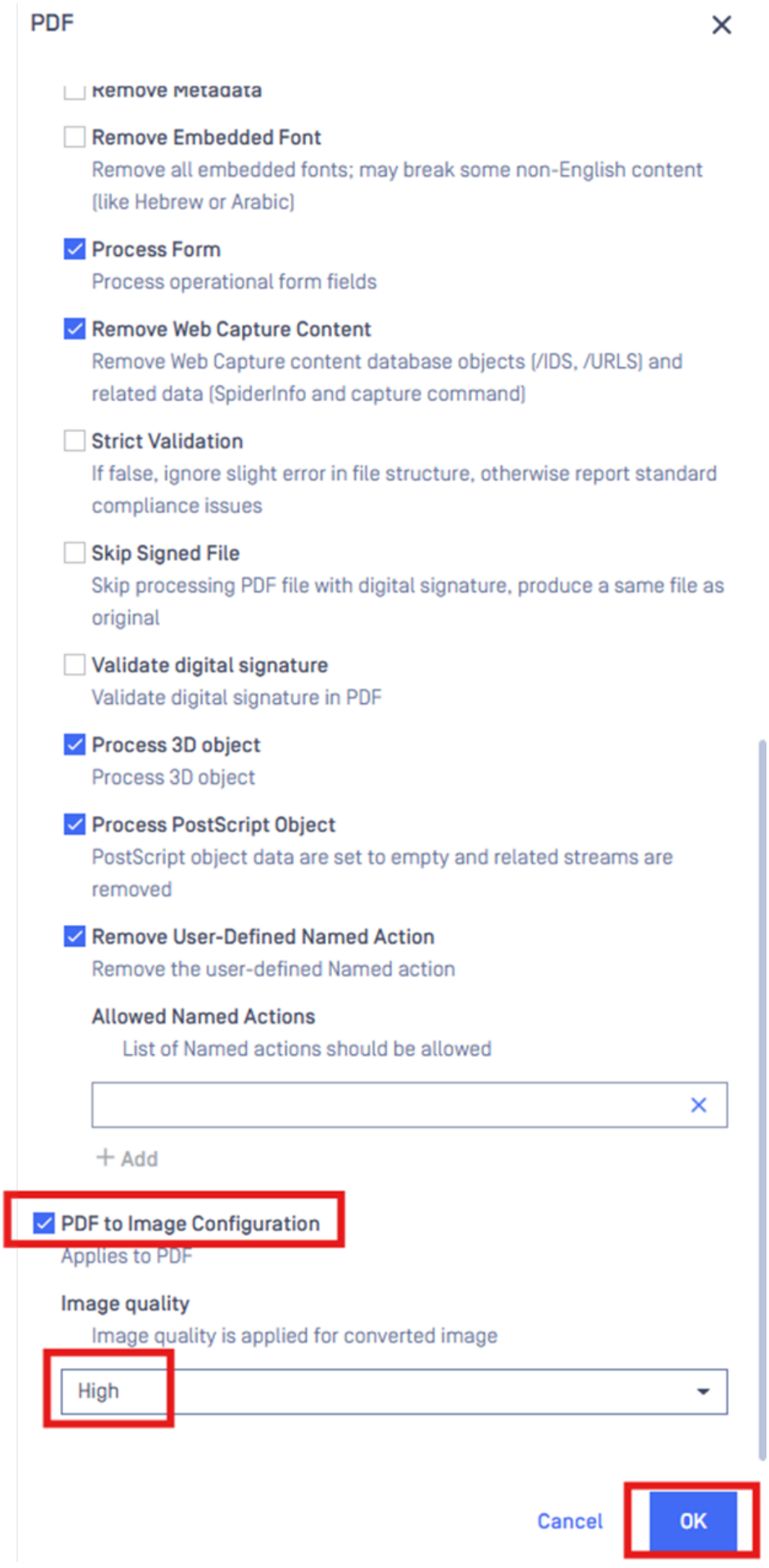
Details:

“PDF to Image Configuration” feature is taking effect only when you use PDF to image(ex.: jpg etc.) conversion for PDF files.


File Types
Enable Deep CDR for file types
Enable for all file types
Any newly added file types will also be enabled for sanitization
Archive Files
Application Files
Office Documents
Executable Files
Image Files
Media Files
Other
Adobe Files
Input
Output
PDF
jpg
Text Files
Cancel
OK

This can be configured using the following path: in MetaDefender Core when Deep CDR engine is licensed using the following path: Under “Workflow Management” - click on“Workflows” - Choose the workflow in scope - click on tab “Deep CDR” - Enable Deep CDR if it’s disabled - scroll down to the section “Enable for filetypes” - click on “Edit” - expand “Adobe Files” - enable “PDF” - under section “Output” select “jpg” - click “OK” - click “Save changes”

The feature “PDF to Image Configuration” can be found in MetaDefender Core when Deep CDR engine is licensed using the following path: Under “Workflow Management” - click on“Workflows” - Choose the workflow in scope - click on tab “Deep CDR” - Enable Deep CDR if it’s disabled - scroll to the bottom of the page - click “Advanced Configuration” - scroll down to the section “File type handling” - expand “Adobe” - click on “PDF” - in the new window scroll down as per screenshot below -click “OK” - click “Save changes”


PDF
Remove Metadata
Remove Embedded Font
Remove all embedded fonts; may break some non-English content (like Hebrew or Arabic)
Process Form
Process operational form fields
Remove Web Capture Content
Remove Web Capture content database objects (/IDS, /URLS) and related data (SpiderInfo and capture command)
Strict Validation
If false, ignore slight error in file structure, otherwise report standard compliance issues
Skip Signed File
Skip processing PDF file with digital signature, produce a same file as original
Validate digital signature
Validate digital signature in PDF
Process 3D object
Process 3D object
Process PostScript Object
PostScript object data are set to empty and related streams are removed
Remove User-Defined Named Action
Remove the user-defined Named action
Allowed Named Actions
List of Named actions should be allowed
+ Add
PDF to Image Configuration
Applies to PDF
Image quality
Image quality is applied for converted image
High
Cancel
OK

Below is an example of such implementation:

- Using the sample file:



Conversion: PDF to JPG.

- **LOW setting:** file size 340KB, dimension is 1759x2490 pixels.



- **MEDIUM setting**: file size 916KB, dimension is 3497x4950 pixels.



- **HIGH setting**: file size 1.55MB, dimension is 4960x7020 pixels.



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I send specific log entries to syslog integration?

Check Your Version:

This article applies to all MetaDefender Core version 5.0 and newer releases deployed on Windows and Linux

systems.

Introduction:

This knowledge base article provides a way to choose specific MSGID's and their urgency level so that based on the Syslog configured level, to filter them against all other log entries.

Details:

There are situations where the syslog integration needs to be configured to filter the log entries shared and to leverage only specific log entries that the customer is interested in. This allows the customer to avoid sending unwanted logs to the syslog integration.

For example, if we want to only view the errors in the syslog integration but at the same time we are interested in the scan results to see if there are any infections or issues with scanning, we can configure the syslog integration using the error log level but how do we add log entries that have information log level?

By configuring the override function, we can configure specific msgid's log level from information to error.

[INFO] 2025.04.28 05:58:30.407: (core.workflow) Processing finished ... [msgid: 82]

After the change is implemented:

[ERROR] 2025.04.28 06:00:02.032: (core.workflow) Processing finished ... [msgid: 82]

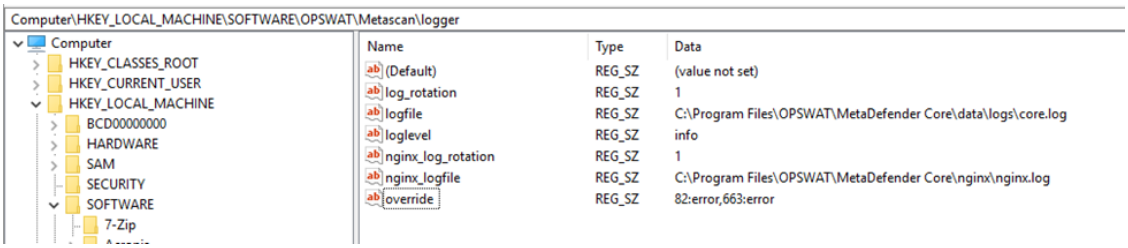
To make the above change please follow the instructions below for each OS:

Windows:

Please open the regedit and navigate to:

HKEY_LOCAL___MACHINE\SOFTWARE\OPSWAT\Metascan\logger

Create a new string value (REG_SZ) named "override" and enter all the msgid's followed by the new log level and then add a comma to add a new value. Please see example below:



Linux:

Please edit the configuration file /etc/ometascan/ometascan.conf in the logger section as follows: ** Please navigate to the [logger] section and add the "override" entry and enter all the msgid's followed by the new log level and then add a comma to add a new value. Please see example below:

```
root@alex-191:/home/alex# cat /etc/ometascan/ometascan.conf
[global]
dbmode=1
restaddress=*
restport=8008
[logger]
log_rotation=1
logfile=/var/log/ometascan/ometascan.log
loglevel=info
nginx_log_rotation=1
nginx_logfile=/var/log/ometascan/nginx-ometascan.log
override=82:error,663:error
```

The override option changes the log level for both syslog integration and the log information.

After the changes are made, please restart the services to implement the changes.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I send the MetaDefender Core logs in Windows event log?

Check Your Version:

This article applies to all MetaDefender Core version 5.0 and newer releases deployed on Windows systems.

Introduction:

This knowledge base article provides a way to replicate the MetaDefender Core logs in the Windows event logs. These are helpful for easy log capture when there are already existing syslog integrations with the Windows event log.

Details:

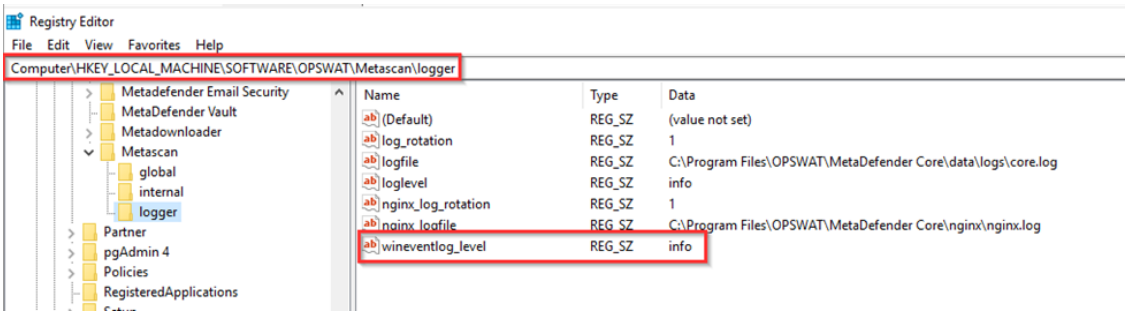
There are situations where the syslog integration is already configured with the Windows event log and it's easier to get the MetaDefender Core logs from the Windows event log than to create a new integration with the MetaDefender Core.

In such cases we do have the option to replicate the MetaDefender Core logs in the Windows event log by performing the below steps.

Please open the regedit and navigate to:

HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger

Create a new string value (REG_SZ) named _"_wineventlog_level" and enter one of the following values: debug, info, warning, error.



After the changes are made, please restart the services to implement the changes.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Does installing an antivirus impact a containerized deployment?

Both Docker and Kubernetes deployments of MetaDefender Core use storage internal to their containers for the temporary files generated during the scanning process. For this reason, installing any form of antivirus or real-time protection solution on the host operating system will **not** have any impact on the operation of MetaDefender Core.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why does the DeepCDR engine delete the content of the HTML file?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

When scanning HTML files, the Deep CDR engine reconstructs the content to remove potentially malicious or hidden elements that could pose a security risk. One common reason for content being removed is the presence of specific CSS attributes or styles used to hide information from the visible page is the font-size:0px.

```
test-mail

------=_Part_316646736_828665848.1743118257422
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

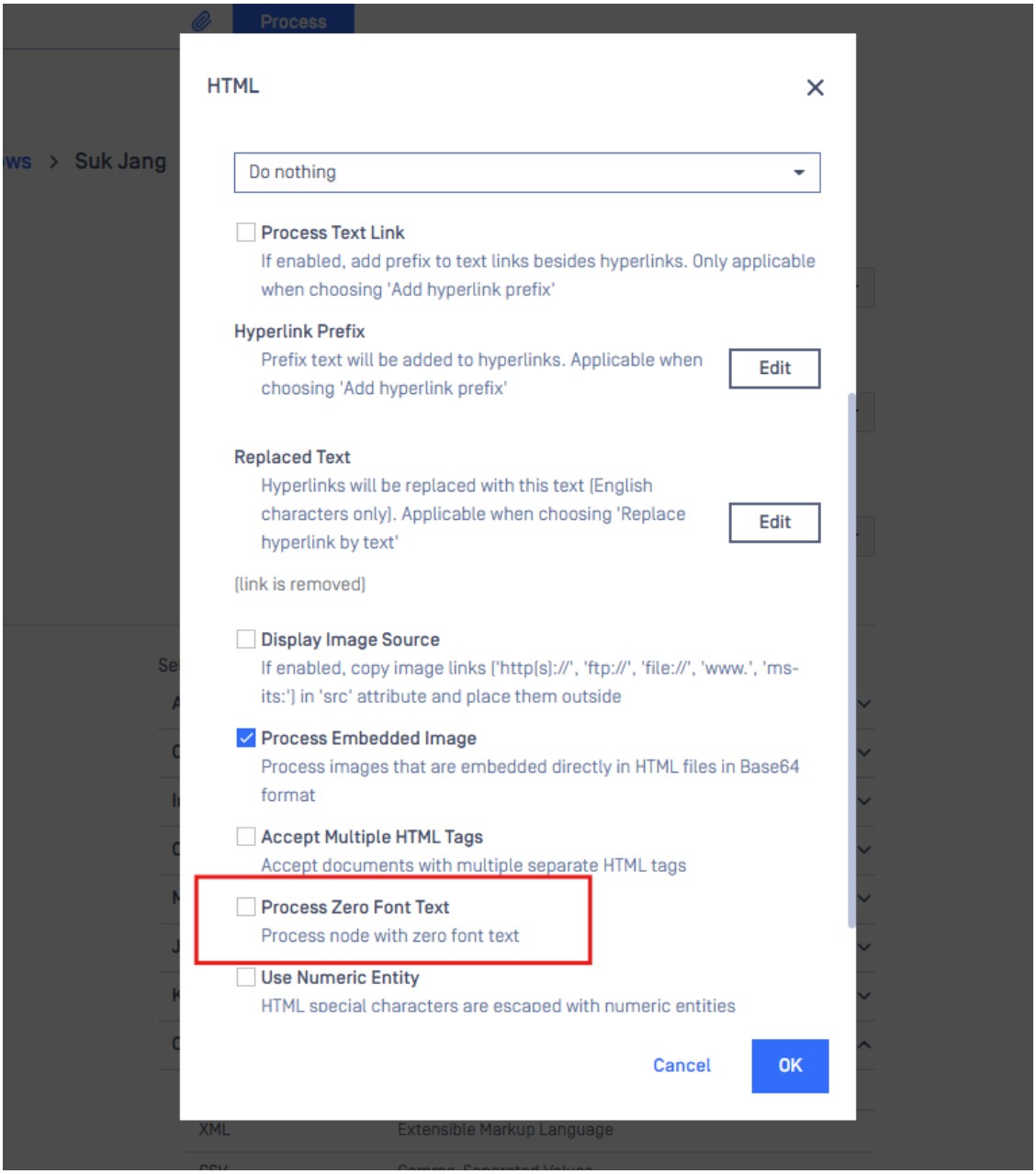
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
</head>
<body>
<div id=3D"emailPreHeader" style=3D"mso-hide:all; visibility:hidden; opacity:0; color:transparent; mso-line-height-rule:exactly;
line-height:0; font-size:0px; overflow:hidden; border-width:0; display:none !important;">
This is a testing file for scanning proccess
</div>
</body>
</html>
------=_Part_316646736_828665848.1743118257422--
```



As seen from the image all the content in the got removed.

Since Deep CDR recognizes these CSS attributes as methods used to conceal content, the information inside `<div>` elements that use specifically (font-size:0px) is removed during the sanitization process.

For a current workaround, we will need to disable the “Process Zero Font Text” to make sure that the Deep CDR does not recognize those elements as zero font text.



Support:

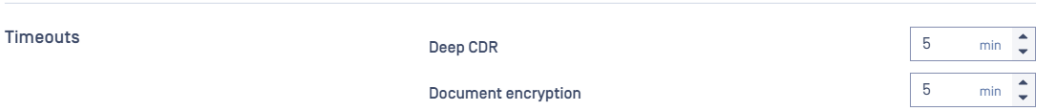
If you have followed all of the steps above but are still unable to resolve the issue. Please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What is the Document Encryption Timeout setting and how does it work?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

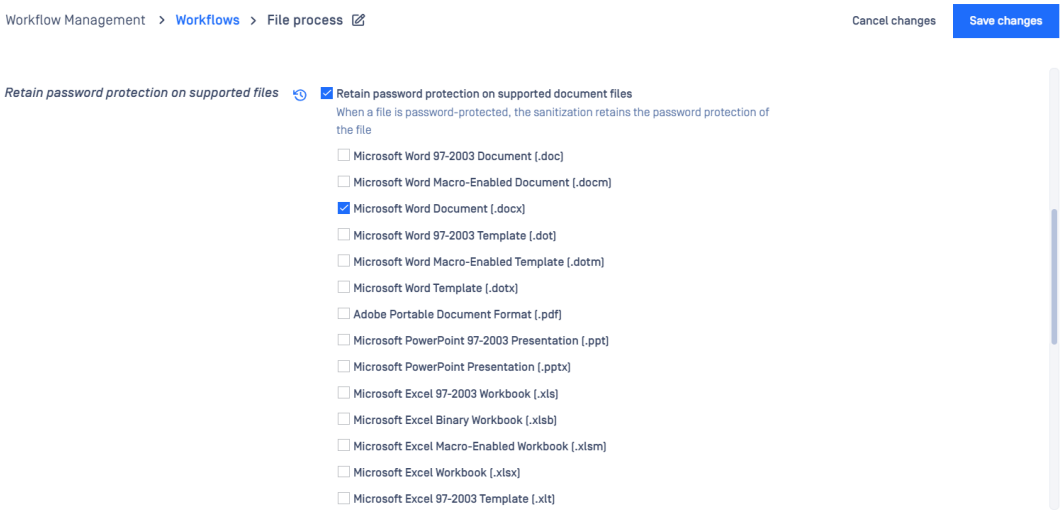
The **Document Encryption Timeout** is a configurable setting located under: **Workflows > DeepCDR > Timeouts > Document encryption**



This setting specifies the maximum duration MetaDefender Core allows for re-encrypting a sanitized document during the Deep CDR process.

When is this setting used?

The Document Encryption Timeout is only applicable when the option **"Retain password protection on supported document files"** is enabled.



When MetaDefender Core detects an encrypted document or archive (e.g., password-protected PDF, Word, or Excel file), the system performs the following steps:

1. Attempts to decrypt the file using a list of preconfigured passwords.
2. Once successfully decrypted, the Deep CDR engine sanitizes the file to remove potential threats.
3. The sanitized content is then recompressed and re-encrypted using the original password.
4. If this re-encryption process exceeds the configured timeout value, MetaDefender Core will return a timeout error.

Why you may need to modify this setting

By default, the timeout is set to 5 minutes. In some environments, especially when processing large or complex encrypted files, the default timeout value may not be sufficient to complete the re-encryption process. If you notice timeout errors during the sanitization of encrypted documents, you may need to increase the Document Encryption Timeout to ensure these files can be processed successfully without interruption.

Adjusting this setting can help balance security, performance, and user experience depending on your organization's needs.

Support:

If **Further Assistance** is required, please proceed to **create a support case or chat with our support engineer**.

How do I manually clean up the MD Core engine's local folder?

Check Your Version:

This article applies to all MetaDefender Core versions.

Warning:

We recommend taking these steps when MetaDefender Core is out of service, since deleting the engine folder will stop the current processing.

Internet access is required. In an air-gap environment, please prepare an offline engine package before applying this.

In some cases, MetaDefender Core keeps the engine folders after updating to a newer version, which can cause engine update issues and increase the disk space MetaDefender Core uses over time. Below is how to manually delete the engine folders.

1. Stop MetaDefender Core services
2. Browse and delete folders inside these installation locations (by default):

- C:\Progam Files\OPSWAT\MetaDefender Core\data\engines
- C:\Progam Files\OPSWAT\MetaDefender Core\data\resources
- C:\Progam Files\OPSWAT\MetaDefender Core\data\updates\av
- C:\Progam Files\OPSWAT\MetaDefender Core\data\updates\db\

3. Delete the file "C:\Program Files\OPSWAT\MetaDefender Core\data\updates\metadescriptor"
4. Start the MetaDefender Core services again and manually click the button “Update All” to update the engine

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with our support engineer**.

How can I handle a high CPU load on my Core server

Overview This article outlines steps and best practices to troubleshoot and resolve high CPU load issues on your OPSWAT Core server. A high CPU load can lead to decreased file processing performance and can affect the overall

system throughput. By following these guidelines, you can optimize your server's performance and ensure efficient scanning operations.

To effectively troubleshoot and resolve high CPU load issues on your OPSWAT MetaDefender Core server, please follow the steps and best practices outlined below. High CPU load can lead to decreased file processing performance and affect overall system throughput. By adhering to these guidelines, you can optimize your server's performance and ensure efficient scanning operations.

Symptoms

- Noticeable slowdown in file scanning and processing
- Increased average processing time per object
- Elevated CPU utilization consistently exceeding recommended thresholds

Potential Causes

- **Configuration Settings**: Overly aggressive parallel scan counts or a high queue size can stress the CPU.
- **Engine Issues**: Outdated or misconfigured scanning engines (e.g., FileType engine issues) may consume excessive CPU resources.
- **System Resources**: Inadequate hardware (e.g., insufficient CPU cores or suboptimal system settings) can lead to high CPU usage.
- **System Tuning**: Power plans, BIOS settings, or operating system configurations that are not optimized for performance can also contribute to high CPU load.

Troubleshooting Steps

1. **Review System Requirements**

Before proceeding, ensure that your server meets the Recommended System Configuration for MetaDefender Core. Inadequate hardware specifications can lead to high CPU usage and degraded performance.

2. **Monitor System Resources**
    1. Use system monitoring tools to assess CPU, memory, and disk I/O usage.
    2. Identify if CPU usage spikes coincide with specific scanning activities.

3. **Review MetaDefender Core Configuration**
    1. Verify that the engine parallel count and maximum queue size settings match your workload. Adjust these settings if necessary.
    2. Check for any misconfigurations in the MetaDefender Core settings that might be triggering unnecessary load.

4. **Update Scanning Engines**
    1. Ensure that all scanning engines, especially the FileType engine, are updated to the latest versions to mitigate known issues that may cause high CPU usage.

5. **System and Hardware Optimization**
    1. Confirm that your server hardware meets the recommended specifications. For instance, using a high-

performance power plan and ensuring BIOS settings are optimized (e.g., enabling Turbo Boost on CPUs).

2. Consider increasing CPU cores if the current hardware is consistently overburdened.

6. **Review Logs and System Health**

1. Examine MetaDefender Core logs to detect any error messages or alerts related to CPU load.
2. Use the built-in health check features to determine if the high CPU load is affecting other system components.

7. **Restart the MetaDefender Core Service**

1. After applying configuration changes, restart the service to ensure the changes take effect.

Recommendations

- **Adjust Configuration Settings**: Tweak engine parallel counts and queue sizes based on your typical workload to avoid overloading the CPU.
- **Regular Updates**: Maintain the latest version of all scanning engines and MetaDefender Core software to benefit from performance improvements and bug fixes.
- **Optimize Hardware**: If performance issues persist, evaluate whether a hardware upgrade (such as adding more CPU cores or faster memory) is required.
- **Follow Best Practices**: Refer to the Performance Optimization and Fine-Tuning Guide for in-depth recommendations on system tuning and optimal configurations.

Additional Resources

- **Performance Optimization and Fine-Tuning Guide**: Learn how to maximize overall performance through system and application tuning. Read more
- **How to Increase Parallel Count and Max Queue Size**: Detailed steps for adjusting these settings in your MetaDefender Core configuration. Read more
- **Troubleshooting MetaDefender Core**: A comprehensive guide to diagnosing and resolving common issues. Read more

By following these guidelines and recommendations, you can effectively manage high CPU loads on your OPSWAT MetaDefender Core server and ensure smooth, efficient operation. If issues persist, consider reaching out to OPSWAT Support for further assistance.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Does MetaDefender Core generate a file download link after all scan sessions?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

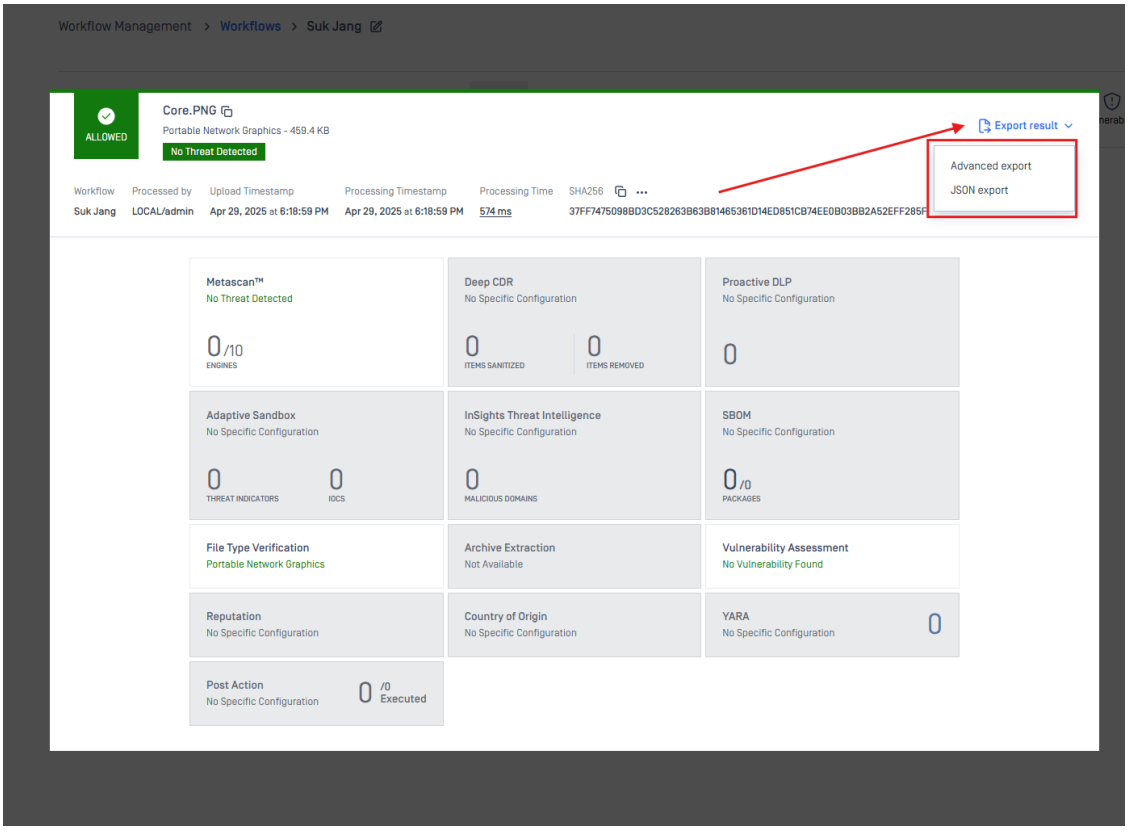
Workflow without DeepCDR, Proactive DLP

When a file is scanned through MetaDefender Core without Deep CDR or Proactive DLP enabled, the scan session will not generate any download link for the scanned file.

Once the scan is completed, MetaDefender Core provides two options for retrieving scan results:

- **Download JSON export**
- **Download the PDF scan report**

In this configuration, the original file remains unchanged, as no sanitization or DLP policy is applied.



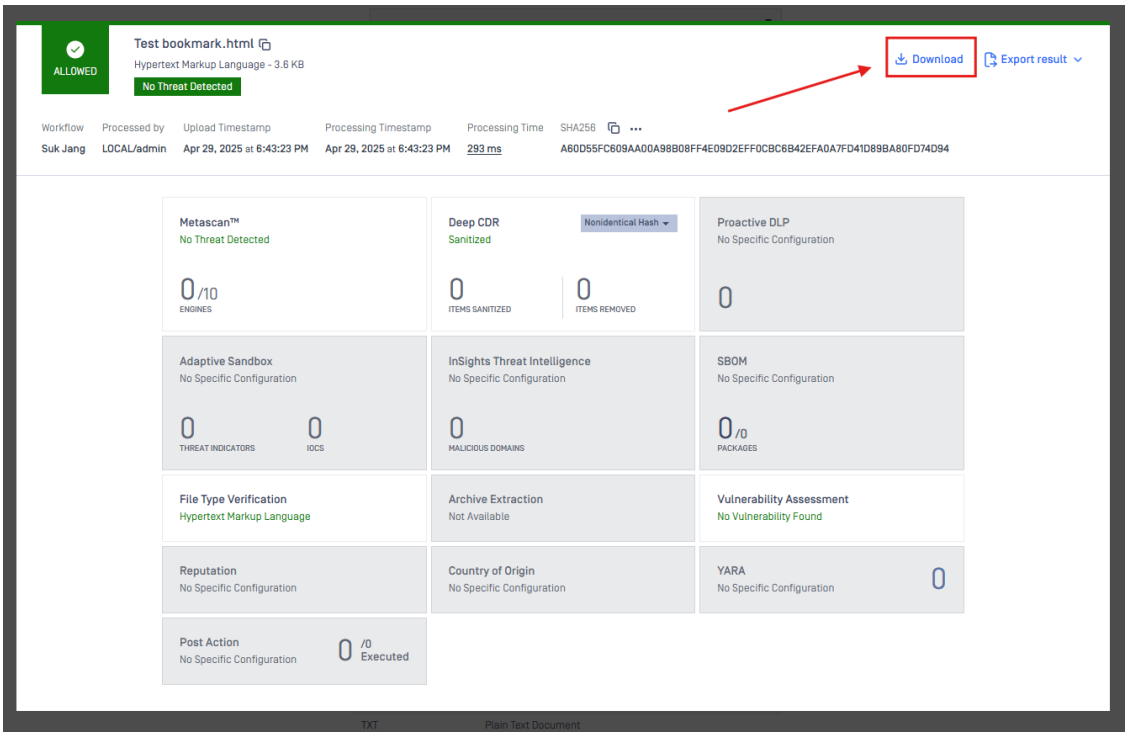
Workflow with DeepCDR Or DLP

When Deep CDR is enabled in the workflow:

- The original file is scanned and **sanitized**, meaning any potentially harmful components (such as embedded scripts, macros, or obfuscated code) are **removed or replaced**.
- Deep CDR reconstructs the file to retain usability while eliminating threats, which may result in **structural or visual changes** to the file.

Note:

The output is a sanitized version of the file. The original file is not retained after scanning.

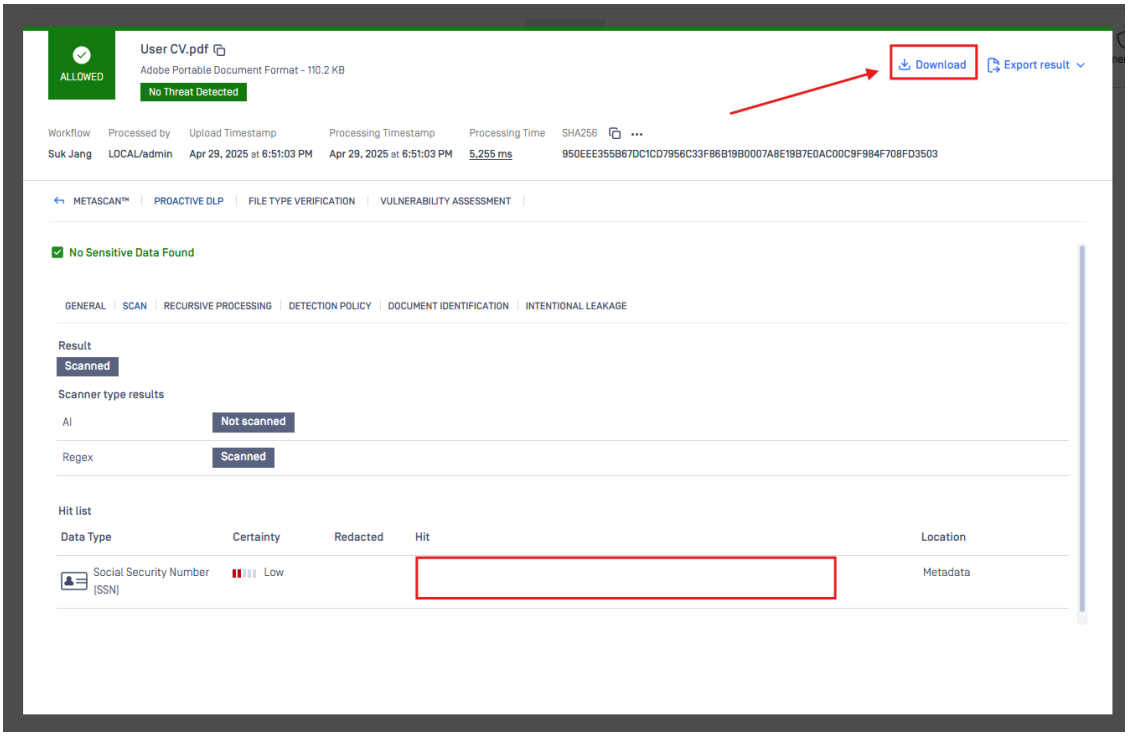


When DLP (Data Loss Prevention) is enabled in the workflow:

- Files are scanned for sensitive content such as personally identifiable information (PII), credit card numbers, or custom-defined patterns.
- The file may be blocked, quarantined, or redacted if a match is found based on your configured policy.
- Unlike Deep CDR, DLP does not reconstruct the file. Instead, it focuses on adding a watermark and controlling the distribution of confidential data.

Note:

Proactive DLP will add a watermark to the areas containing sensitive data of the user.



Support:

If you have followed all of the steps above but are still unable to resolve the issue. Please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Can Workflow Settings Be Updated Globally Across Multiple Workflows?

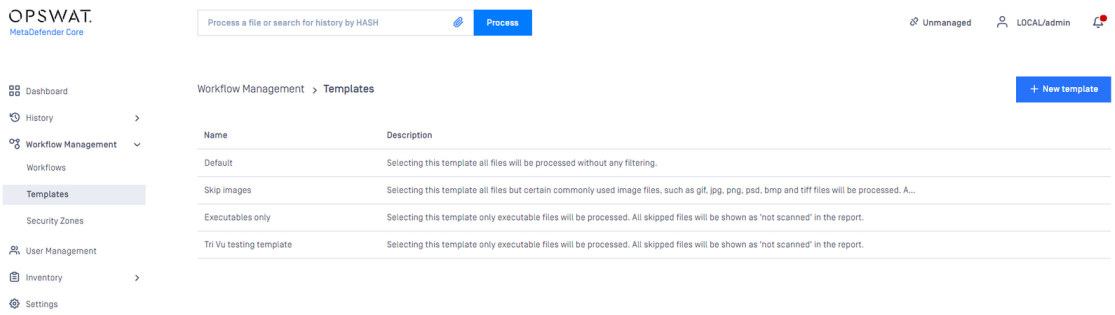
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

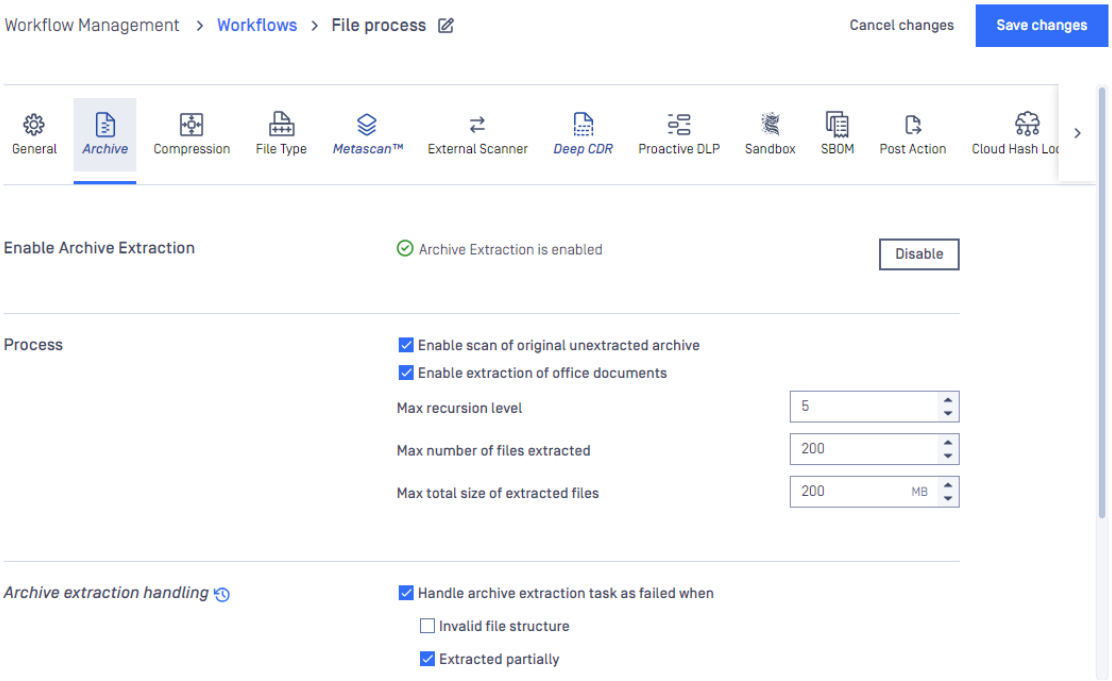
Description

Customers managing a large number of workflows often encounter scenarios where they need to **update a specific setting across all or selected workflows**. For example, changing the **file size upload limit** from 30 MB to 300 MB in all workflows, without altering any other configuration parameters.

The Workflow templates page is found under **Policy > Workflow templates** after successful login.



When you click on a workflow template, a window appears displaying various tabs that show different properties related to the template.



Click "Save Changes" to confirm and apply your updates.

For more information regarding different kind of properties of the workflow, please refer to this article: Workflow Configuration Template

Current Capabilities

✅ *Workflow Templates* __

- Workflow templates are designed to standardize configurations.
- Changes made to a template are propagated to all workflows based on that template.
- This is useful when workflows are intended to share an identical configuration.

⚠️ *Limitations of Workflow Templates* __

- Templates apply a **full configuration**, not selective updates.
- If workflows have unique settings, applying a template will **override all existing configurations**, not just the intended setting.
- Templates **cannot apply partial changes** such as updating only one setting (e.g., file size limit) across all

workflows without affecting other parameters.

Is Partial or Global Update Possible?

Currently, there is **no built-in feature** to:

- Selectively update one or more parameters across all workflows.
- Apply changes to a group of workflows without affecting unrelated settings.
- Track such partial changes centrally across all workflows.

Workflows must be **updated individually** if they do not share a common template.

Recommendation for Large-Scale Workflow Management

- For new deployments, consider grouping workflows under a **template-based structure** to streamline future updates.
- If workflows are highly customized, updating them will need to be done individually or via custom automation using available APIs (if supported).
- For large-scale environments with repetitive update needs, feature enhancement discussions may be warranted.

Feature Enhancement Possibility

This use case highlights a potential need for:

- A **centralized management tool** that allows applying a specific configuration change across multiple workflows.
- A method to track such changes in the **configuration history** for auditing purposes.

While this is not currently available, feedback is being collected to help prioritize such capabilities in future product iterations.

Summary

Question	Answer
Can I update a single setting across all workflows?	Not directly—this requires individual updates unless workflows share a template.
Will changing a template update workflows?	Yes, but it will overwrite the full configuration, not just one setting.
s there a way to track changes globally?	Changes are tracked per workflow; no centralized history exists for global changes.
Is a more flexible update method planned?	This is a known request and under consideration for future development.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Does MetaDefender Core support Elliptic Curve (EC) Private Key and Certificate for enabling HTTPS?

Yes, MetaDefender Core supports Elliptic Curve private key and certificate for enabling HTTPS.

Example: PEM-encoded EC private key format

Code · html

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIFXkB9C7UeznRzZKj...
-----END EC PRIVATE KEY-----
```

Supported Curves

Elliptic Curve Cryptography (ECC) offers many different curve sets. You can list all supported curves using the following command:

Code · html

```
openssl ecparam -list_curves
```

However, most modern browsers only support **P-256** and **P-384**.

Therefore, we recommend using P-256 (also known as `prime256v1`) or P-384 for HTTPS with MetaDefender Core.

How to Verify Your EC Private Key

To inspect your EC private key, use the following command:

Code · html

```
openssl ec -in yourkey.key -text -noout
```

Explanation:

- `-in ec_private.key`: Path to your EC private key file (PEM format).
- `-text`: Prints a human-readable summary of the key (including public and private components).
- `-noout`: Suppresses output of the encoded key.

Example Output:

Code · html

```
read EC key
Private-Key: (256 bit)
priv:
    73:a1:34:...
pub:
    04:51:88:...
ASN1 OID: prime256v1
NIST CURVE: P-256
```

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to manage false positives in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core version 5.0 and newer releases deployed on Windows and Linux systems.

How to Manage False Positives in MetaDefender Core

False positives—clean files incorrectly flagged as threats—can disrupt operational workflows and cause unnecessary concern. MetaDefender Core offers several methods to handle false positives, depending on the scenario and your internal security policies. Use the options below in the recommended order:

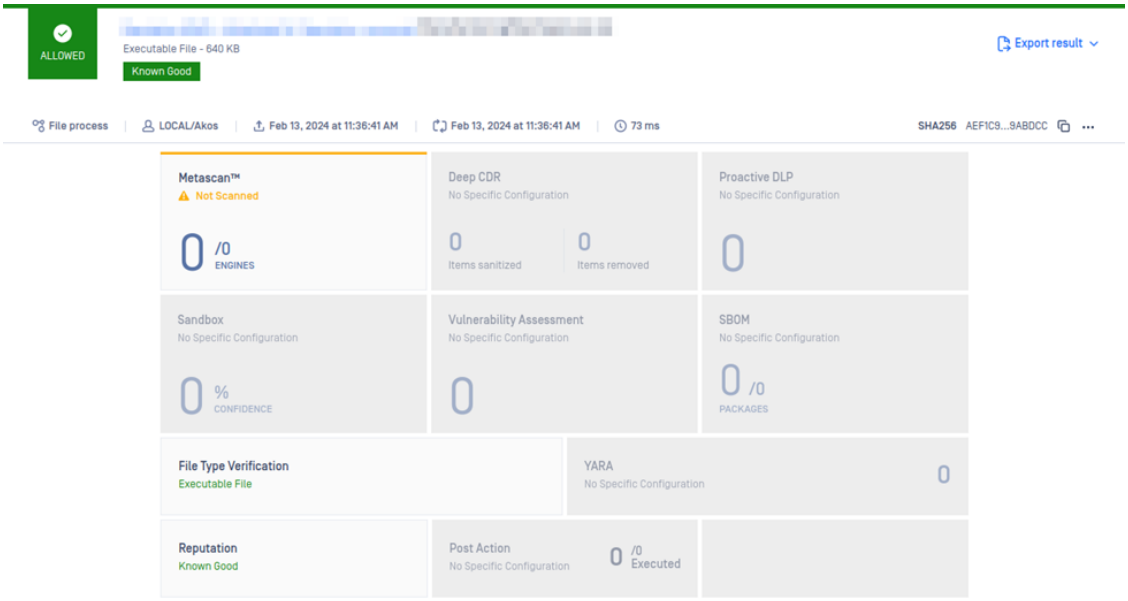
1. Verify That the File Is a False Positive, Not an Actual Threat

Before taking any further action, it is essential to confirm that the file is indeed a false positive.

- Review the detection details provided by each engine.
- Test the file in a controlled environment to validate its behavior.
- If you suspect a false positive, you can submit the file to OPSWAT for further analysis:
  - Log into My OPSWAT.
  - In the left-hand side menu, select “Support,” and then choose “Report False Detection” from the submenu.
  - Provide the file you would like OSPWAT's malware analyst team to investigate. Note that the maximum size for file uploads is restricted to 2GB.
  - In the Submission Type section, specify if you suspect the file is a false positive (i.e., a clean file incorrectly flagged) or a false negative (i.e., a malicious file not detected). Note: You should not be overly concerned about categorizing your submission as a false positive or false negative. OPSWAT utilizes a thorough validation process for analyzing submitted files.

See official guidance: How do I get a False Detection (False Positive or False Negative) corrected?

- Once the file has been confirmed as a false positive, the Reputation engine will be updated and will override the AV engines detection with a Known good verdict.



- Also, the AV vendors will be notified of the false positives and in time these will no longer be flagged.

2. Use the Sandbox Engine for Behavioral Analysis

To gain deeper insight into the file's behavior, enable scanning with the **Sandbox** engine.

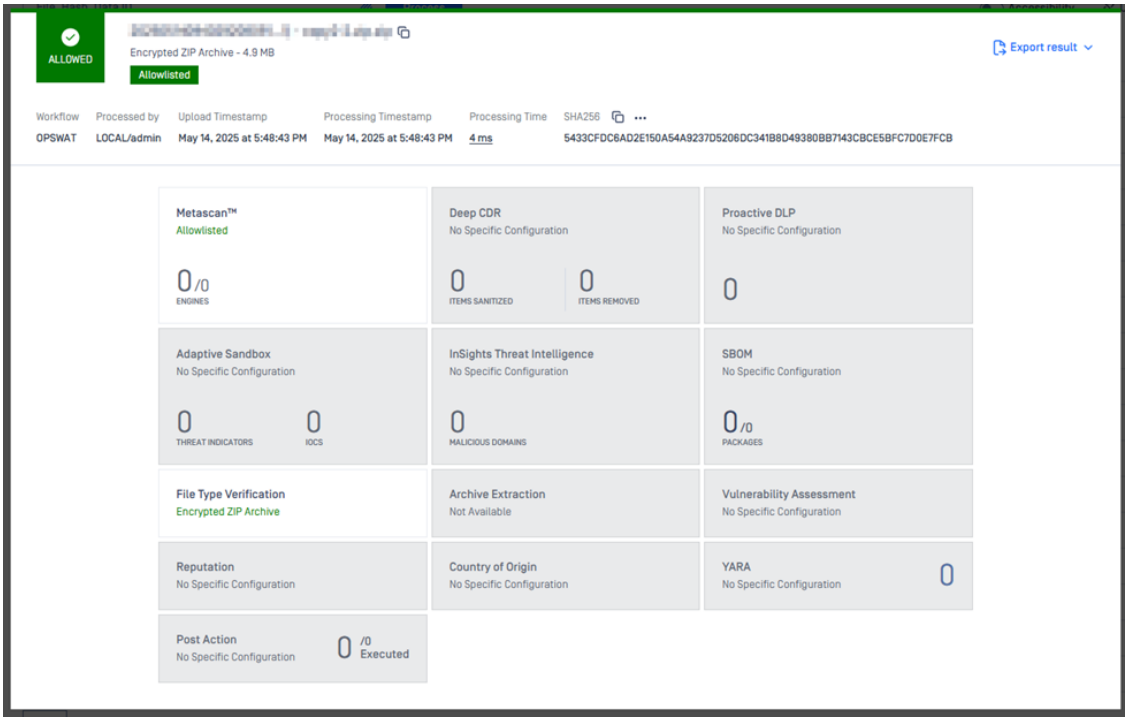
- The sandbox analyzes files in a controlled virtual environment to detect malicious behavior beyond traditional AV signatures.
- This additional layer helps validate whether a detection is accurate or a false positive.

Using sandbox analysis is especially helpful for suspicious scripts, executables, or documents where static detection yields inconclusive results.

3. Use the Allow List for Known Trusted Files (With Caution)

If you are certain the file is safe (e.g., it is a known internal script or tool), you can add its hash to the **Allow List** to prevent future detections:

- Go to **Workflow Management -> Workflows -> your workflow -> Allowlist -> Allowlist by HASH**
- Add the file's SHA256 hash
- Save the configuration

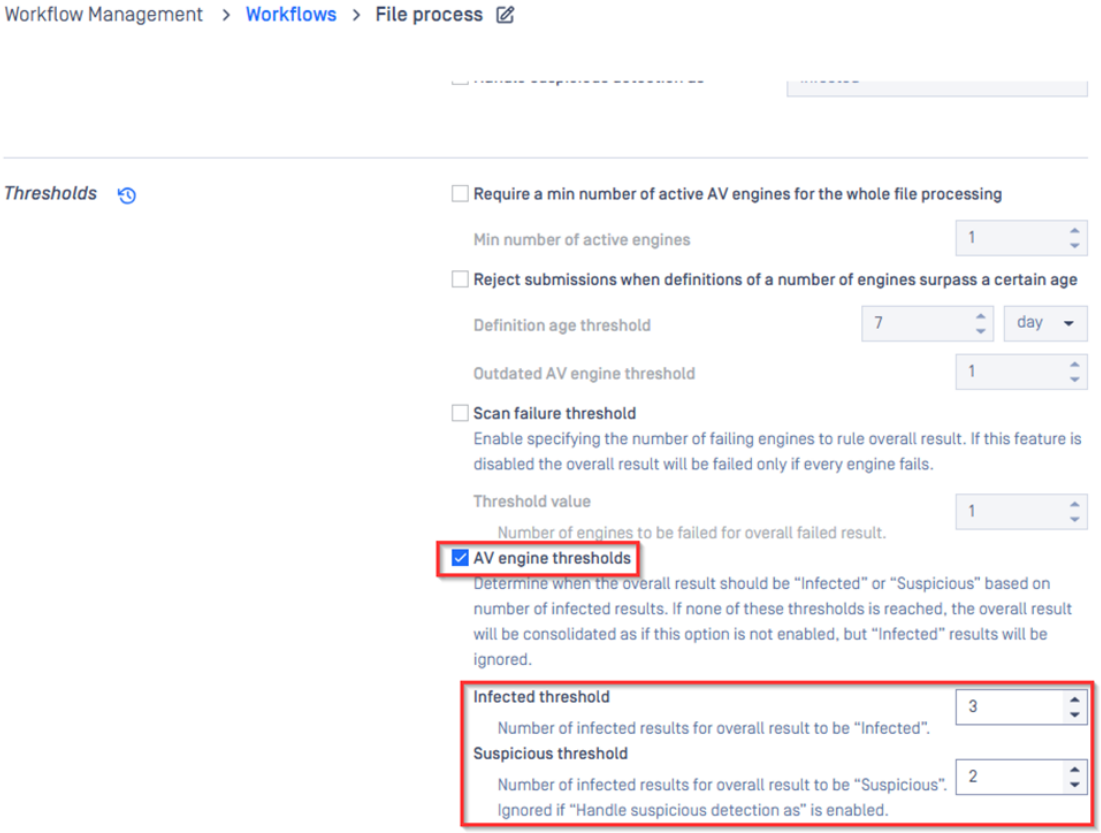


⚠ **Warning**: Adding a file to the Allow List means it will bypass all scanning. This creates a potential security risk. Always ensure that internal security policies are followed, and use this option **only when absolutely necessary**.

4. Adjust Engine Scoring Thresholds in Metascan Workflow (Advanced Use)

MetaDefender Core allows you to configure thresholds for how many engines must flag a file before it is classified as **infected** or **suspicious**.

- Go to **Workflow Management -> Workflows -> your workflow -> Metascan**
- Adjust the values for AV engines thresholds:
  - **Number of engines to classify as Infected**
  - **Number of engines to classify as Suspicious**
- **Save the configuration**



⚠ **Warning**: Lowering these thresholds increases the risk of allowing actual threats into the environment. Use this option only when:

- You are using a **large number of engines** (e.g., 16, 20, or MAX packages)
- You have **sufficient compensating controls** in place
- You are following **internal security guidelines**

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

How to handle split archive files?

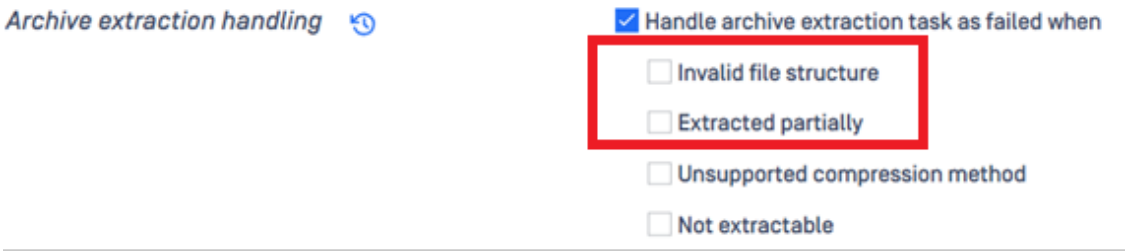
A **split archive file** is a compressed file that has been divided into multiple smaller parts, typically to simplify storage, sharing, or transfer of large data sets. These files usually follow a naming convention such as `.001`, `.002`, and so on. Each part contains a portion of the original data.

Some archive formats store metadata in each individual part, allowing partial extraction. However, others store metadata only in the first part (e.g., `.001`), meaning the remaining parts cannot be extracted independently.

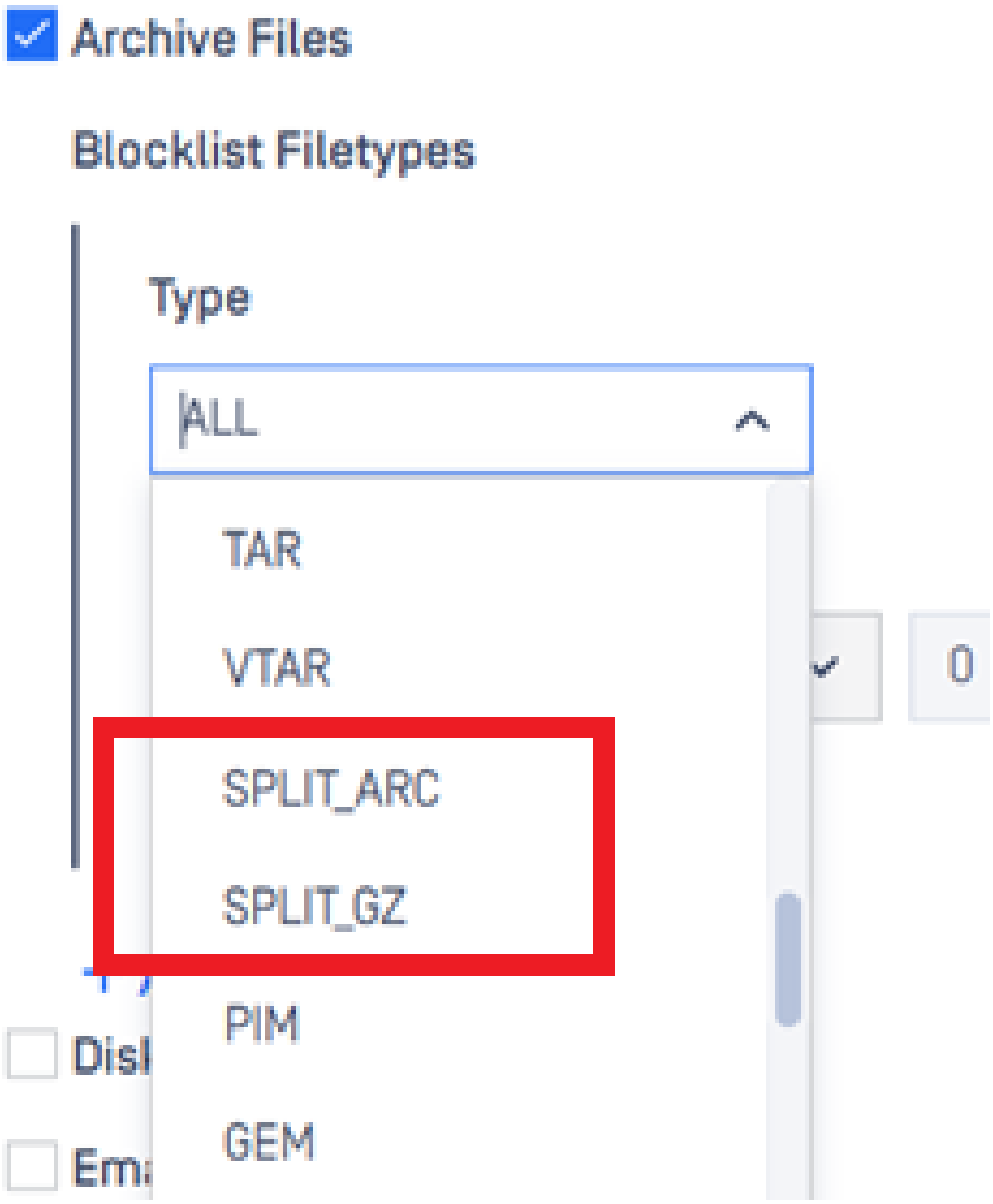
The **Archive Extraction** module attempts to extract as much data as possible and passes the extracted content through a scanning process to ensure maximum security. However, due to the nature of split archives, the extraction process often results in outcomes like **"Partially Extracted"** or **"Corrupted Archive."**

What can customers do with split archive files?

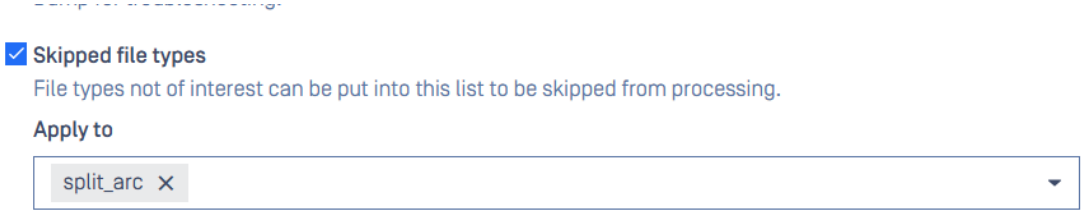
- If customers prefer not to have these files blocked under such conditions, they should avoid enabling specific options in the **Archive Extraction** tab within the workflow configuration.



- The multipart upload API works in some file types, such as zip and 7z, but not with all split archive file types (rar, ...). This API concatenates the split files by appending their data, without properly reconstructing the archive structure. A proper combination of split archives requires dedicated extraction tools such as **7-Zip**, **WinRAR**, or similar. We recommend that all parts of a split archive be combined on the client side before being sent to MetaDefender Core. This ensures that the archive engine can properly extract the contents, and the scan process is applied to the complete file.
- If this approach is not feasible, customers may choose to **block split archives and investigate them manually at a later time**. This can be configured using the **Blocklist** option within a workflow.



- It is not suggested, but if customers want to skip the extraction for the split archive, please use this configuration in the Modules > Utilities> Archive extraction



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to check the Nginx version used in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows, Linux.

Overview:

This article explains how to determine the Nginx version used in your current MetaDefender Core installation.

On Windows:

1. Open Terminal (Command Prompt or PowerShell).
2. Navigate to the MetaDefender Core installation folder, specifically to the nginx subfolder: `cd "C:\Program Files\OPSWAT\MetaDefender Core\nginx"`
3. Run the following command: `nginx.exe -v`

On Linux:

1. Go to the `/usr/lib/ometascan/` directory.
2. Run the following command: `./nginx -v`

Verify the Change:

The command will output the Nginx version. For example: `nginx version: nginx/1.26.2`

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

What are the List of folders that should be removed after MetaDefender Core uninstallation on Windows?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

Please find below the list of folders that should be removed after performing a clean uninstallation of MetaDefender Core on Windows:

Code · html

```
C:\Program Files\OPSWAT\MetaDefender Core
C:\Windows\Temp\ometascan
```

If you have set a custom `tempdirectory` in Registry, please also try to remove that folder as well.

Support:

If **Further Assistance** is required, please proceed to **create a support case or chat with our support engineer**.

How to Sanitize Non-Compliant MDZIP Files?

Check Your Version:

This article applies to DeepCDR version 7.4 and later, which requires MetaDefender Core version 5.14.0 or higher to function properly.

Issue: Some MDZIP files from customers may not strictly follow the MDZIP format, preventing them from being sanitized. R&D is unable to relax the format rules to support these non-compliant files.

Resolution: These files can still be sanitized as regular ZIP files, provided the settings are configured to ensure the following:

- The sanitized files retain their original `.mdzip` extension.
- All internal file names remain unchanged. This is crucial for CAMEO to read the files correctly after sanitization.

Workarounds:

To keep the `.mdzip` extension, consider these options:

- Manually rename the extension to `.mdzip` after sanitization.
- Wait for the new feature enhancement in Core 5.15.0.

To preserve the internal file names after sanitization:

- Enable the setting: `Preserve entry name in sanitized files`. This feature has been available starting from DeepCDR version 7.4 and later, and it needs MetaDefender Core version 5.14.0 or above to work correctly.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I receive MD Core scan results using a static callback URL without specifying the URL in request headers?

Info:

- This article applies to MD Core version 5.5.0 and later
- Please note that the webhook server must be managed and hosted by you.
- From version 5.13.0, certificate loading is enhanced to ensure Webhook communication over TLS.

Overview

MetaDefender Core allows the use of webhooks for post-scan actions. You can pre-configure a static callback URL under the *Post Action* section (via the **Inventory** menu). When enabled, MD Core will send scan results to this static URL using multi-threaded asynchronous workflow processing. This setup is ideal for disaster recovery scenarios where header-based dynamic URLs are not feasible.

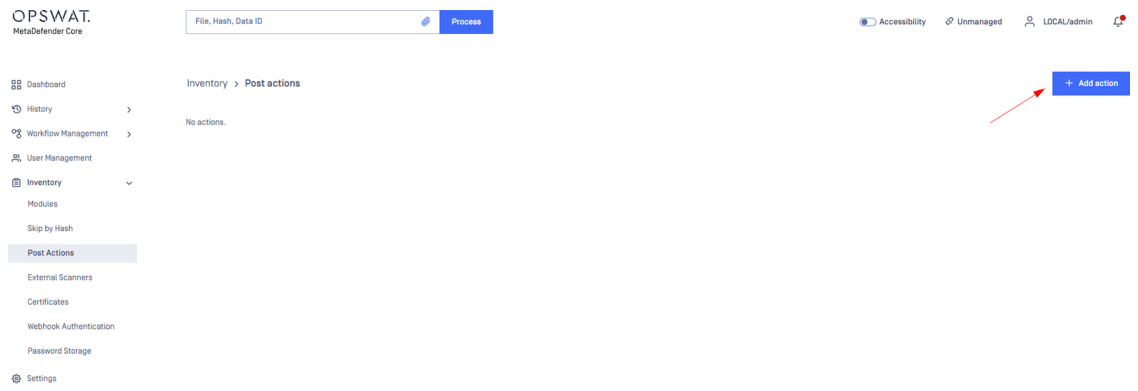
Security Consideration:

By default, webhook callbacks from MetaDefender Core are sent over an unsecured connection. To enhance security and protect against potential man-in-the-middle attacks—which could lead to unauthorized alteration of scan results—MetaDefender Core supports mutual TLS (mTLS) authentication. This mechanism ensures secure communication between the MetaDefender Core server and the client receiving the callback.

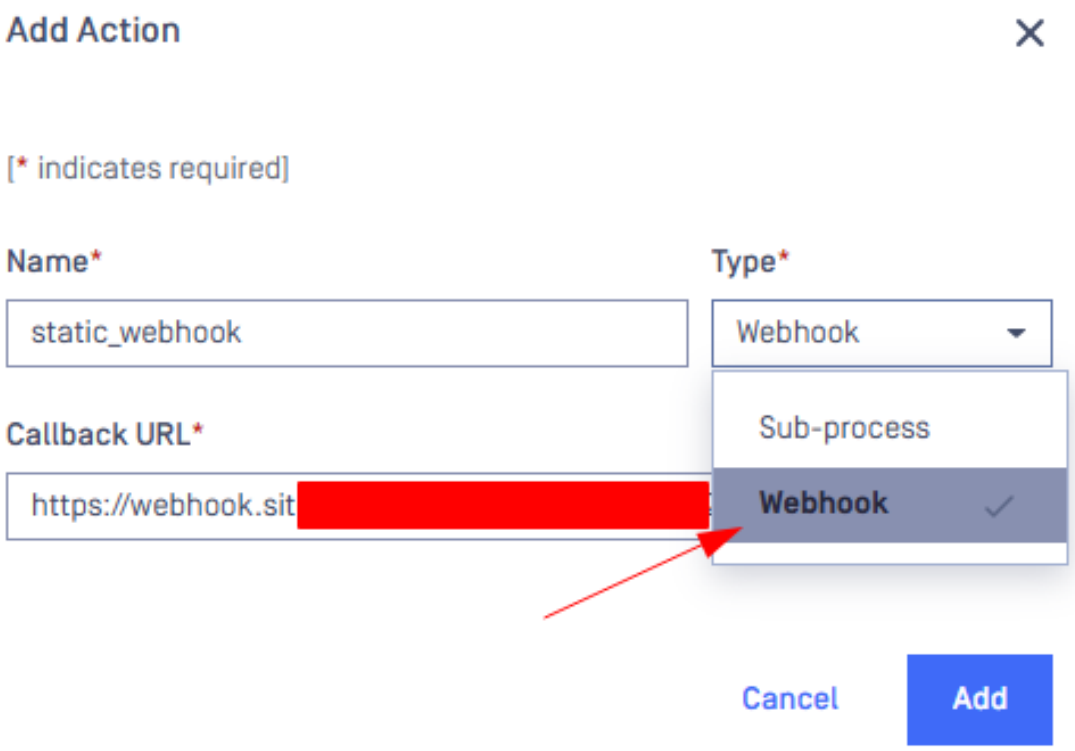
For detailed guidance on configuring webhook authentication, please refer to the official documentation: Webhook Authentication Configuration

Workflow configuration

1. On the left panel, navigate to Inventory > Post Actions > Add action



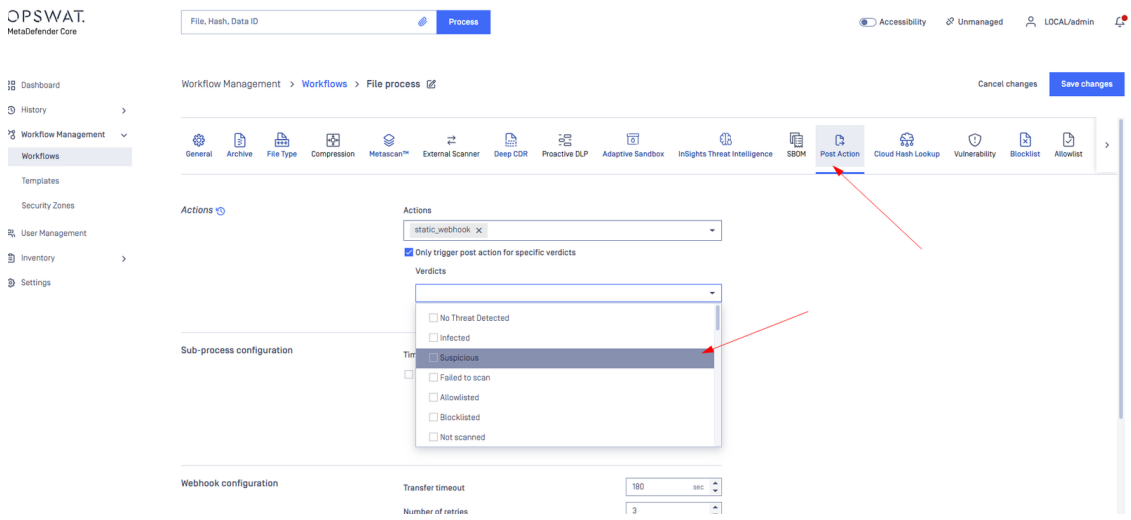
2. Choose Webhook type, then setup the Name and static Callback URL. Press the “Add” button.



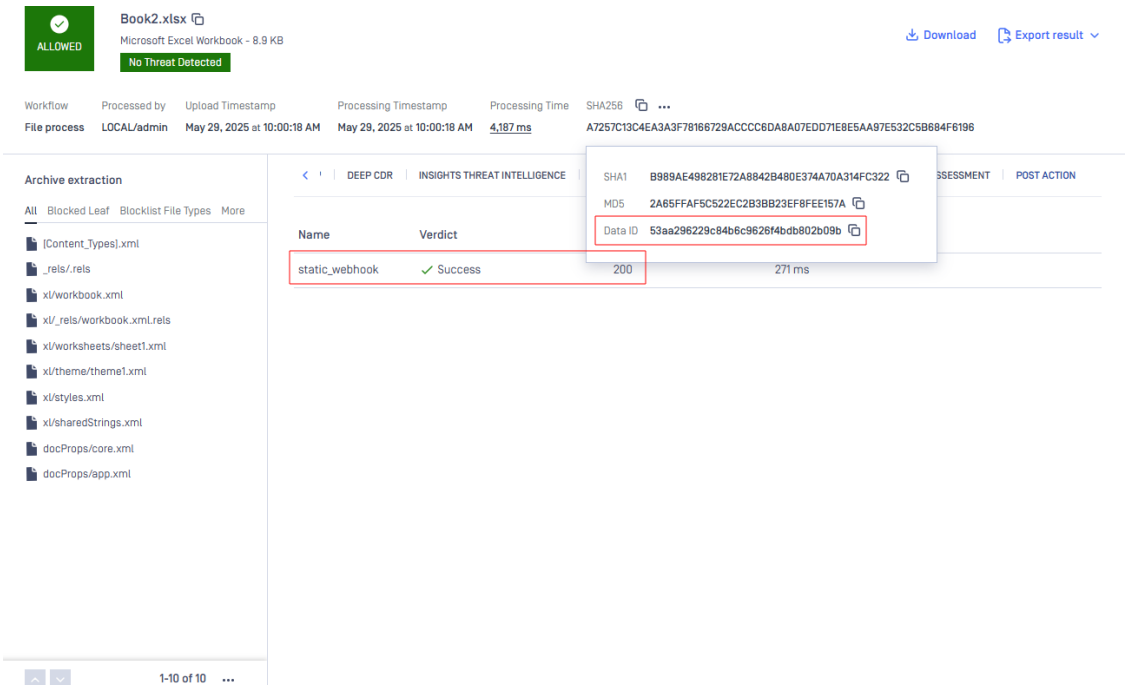
3. You shall receive the successful message afterwards.



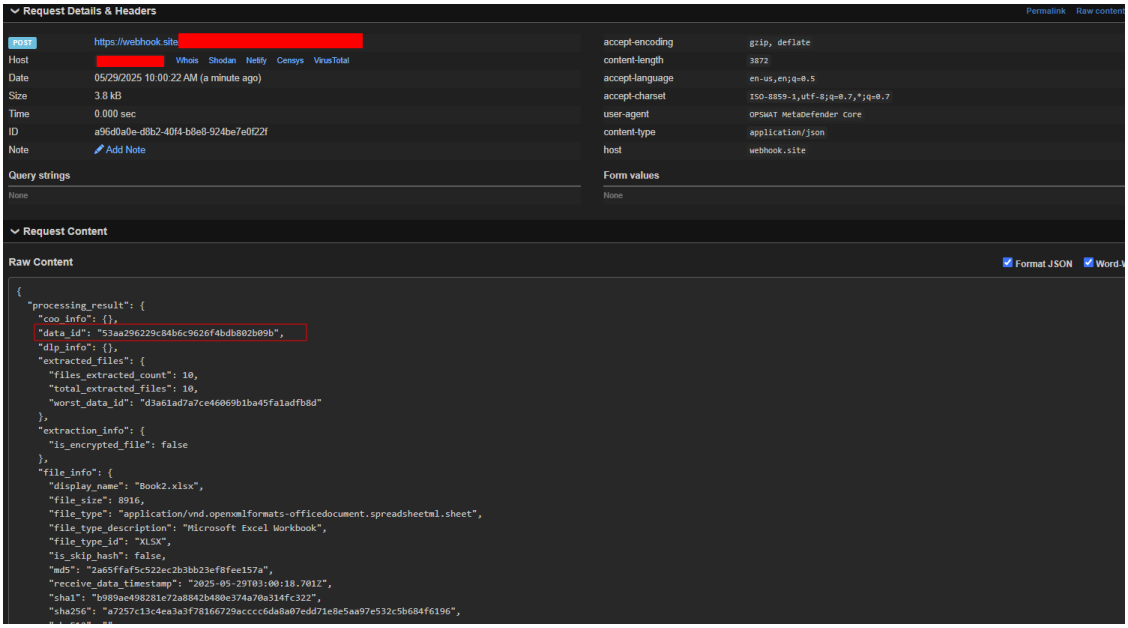
4. After successfully added the action, navigate to the Workflow that you would like to trigger the Post-action, for e.g ***FileProcess.*** Then, choose the Post-Action tab to enable the action on the workflow



5. **Save** the configuration and verify the result



Result from webhook:



Support:

If you require further assistance, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What is the difference between the Archive Extraction Engine V.S. Metascan Antivirus engines extraction?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Description:

The OPSWAT Archive Extraction engine works independently from the Metascan Antivirus engines.

Archive Extraction Engine

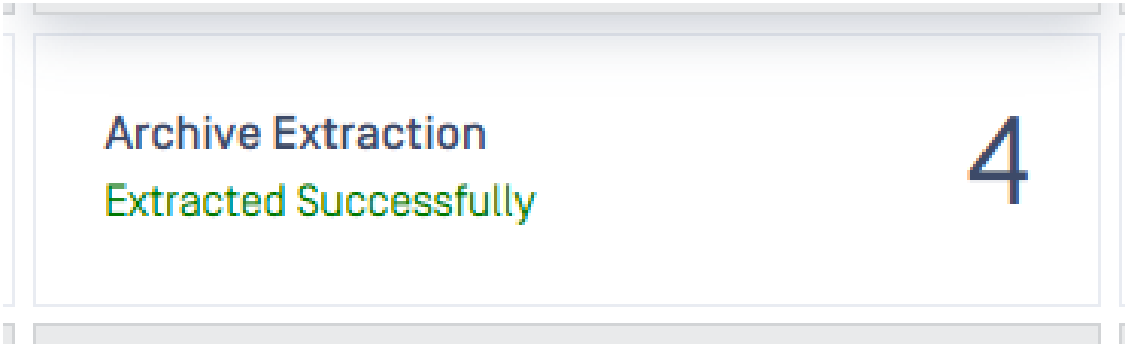
Location:

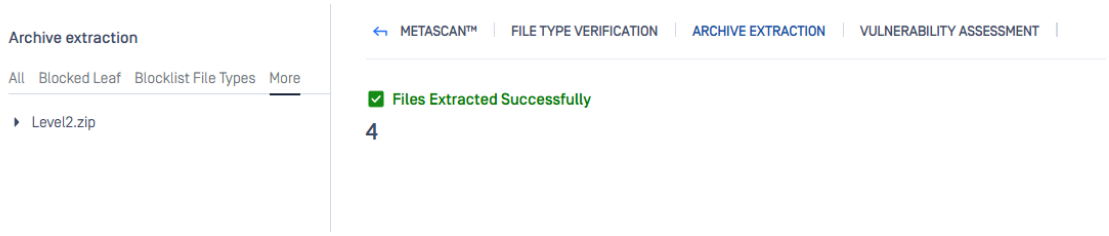
Inventory > Modules > Utilities > Archive Extraction



Functionality:

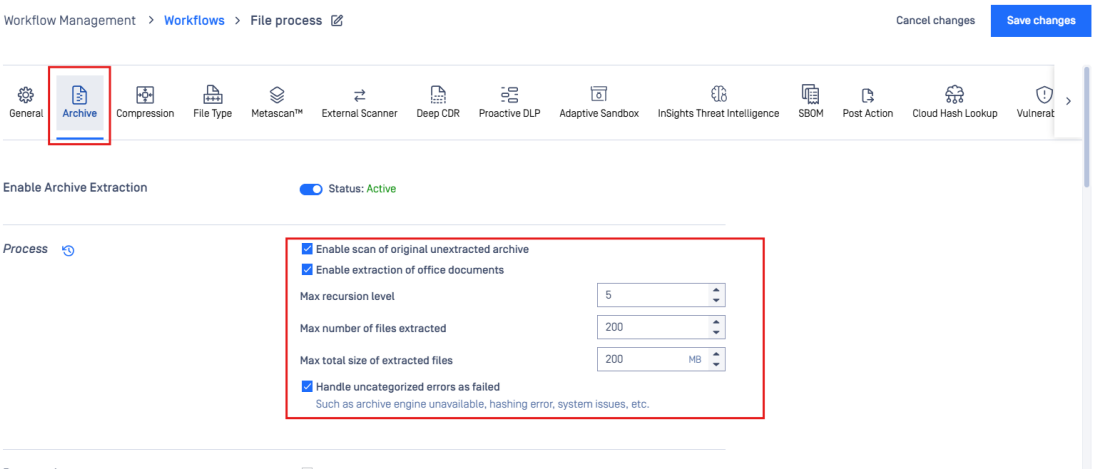
- Supports more file types than individual Metascan Antivirus engine extractors.
- OPSWAT integrates 7-Zip into its Archive Engine to facilitate efficient archive extraction, for more details: Differences between OPSWAT Archive engine and 7-zip
- Generally faster than using individual Metascan Antivirus engine-based extractions.
- Recommended Archive Extraction method.
- Better performance and GUI visibility. Shows the complete recursive path and breakdown for extracted files.





- Configurable extraction limits that can be set globally for each file.

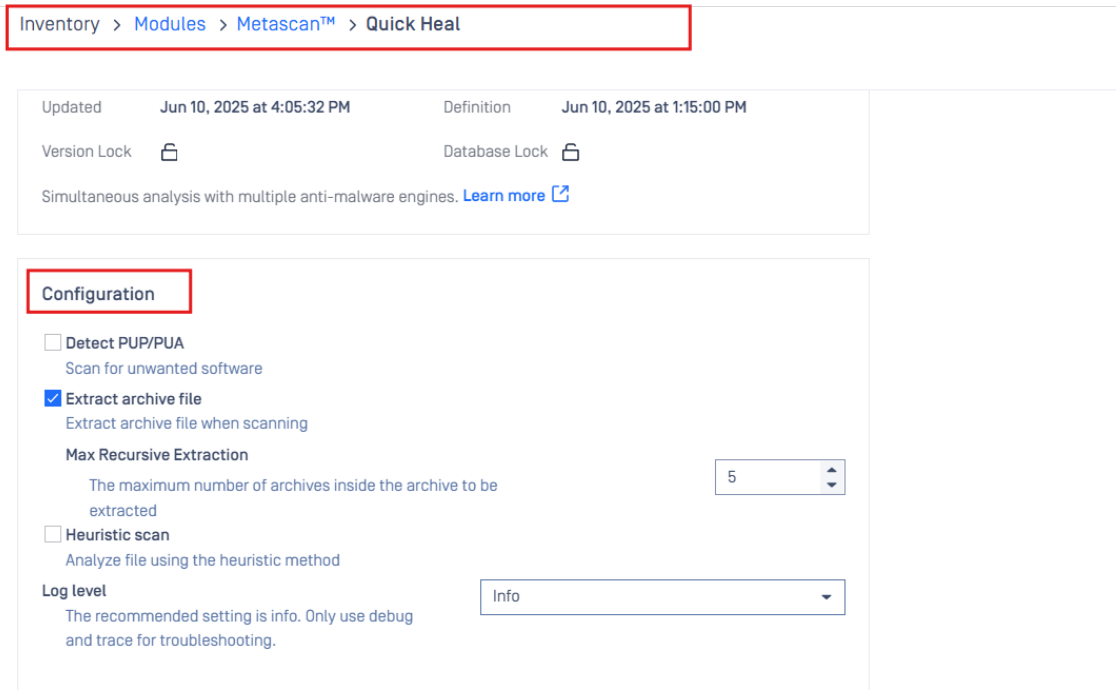
__ Example: __



Metascan Antivirus Engines

Location:

Inventory > Modules > Metascan > click into individual engines

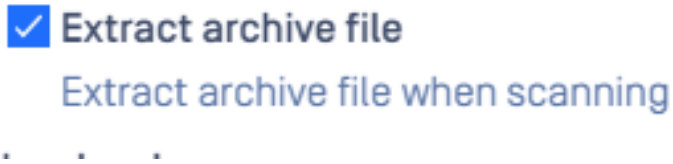


Functionality:

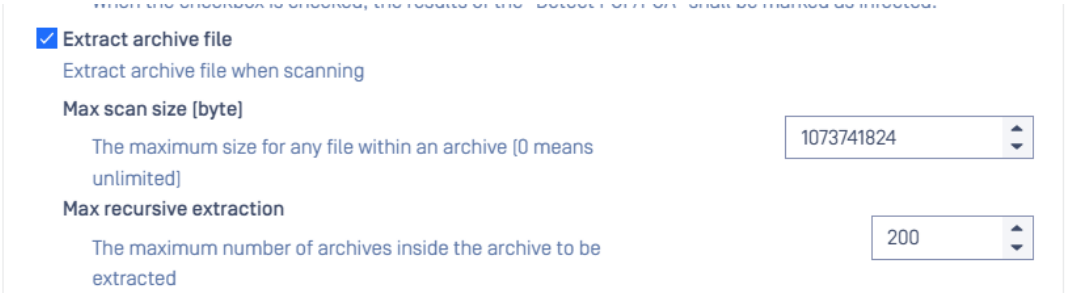
- Antivirus engines can extract files independently, this results in sequential processing, where the same archive may be extracted multiple times by different engines, this may lead to performance degradation.
- Best use if you are unable to enable the Archive Extraction Engine.
- Limited visibility, the GUI currently does not show the extracted files by each engine.
- Archive limits are set by the Antivirus vendor. Some contain configurable limits, whereas some do not have a setting.

Examples:

Ahnlab:



Avira:



Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Why does MetaDefender Core Identifys Only the First Part of a Split Archive as an Archive File?

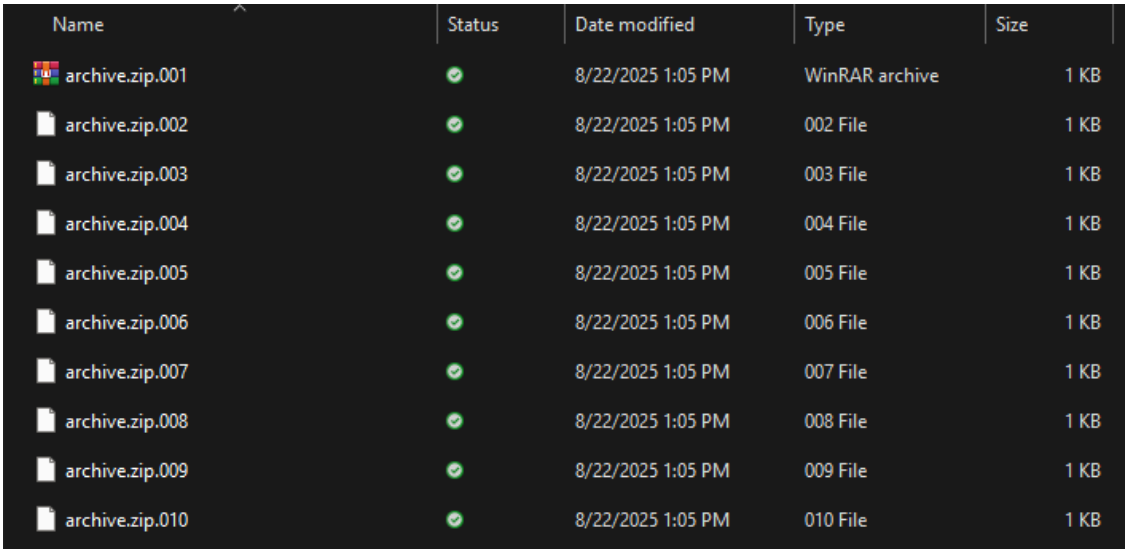
When scanning split archives with MetaDefender Core, you may notice that only the first segment (e.g., **archive.zip.001**) is detected as an archive, while subsequent parts (**.002**, **.003**, etc.) are labeled as **data**. This behavior is expected and stems from how split archives are structured and how File Type Engine interprets them.

What Are Split Archives?

Split archives are large compressed files divided into smaller segments for easier distribution and storage. Instead of a single **archive.zip**, you might see:

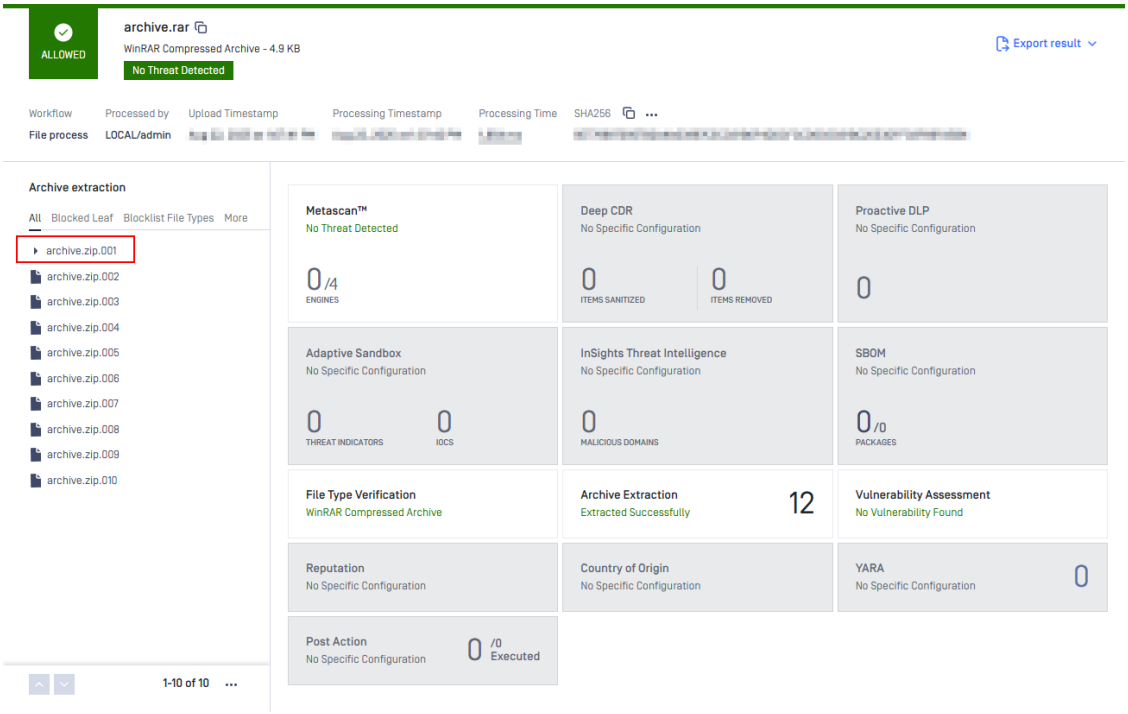
Code · html

```
rchive.zip.001
archive.zip.002
archive.zip.003
...
```



Name	Status	Date modified	Type	Size
archive.zip.001	✓	8/22/2025 1:05 PM	WinRAR archive	1 KB
archive.zip.002	✓	8/22/2025 1:05 PM	002 File	1 KB
archive.zip.003	✓	8/22/2025 1:05 PM	003 File	1 KB
archive.zip.004	✓	8/22/2025 1:05 PM	004 File	1 KB
archive.zip.005	✓	8/22/2025 1:05 PM	005 File	1 KB
archive.zip.006	✓	8/22/2025 1:05 PM	006 File	1 KB
archive.zip.007	✓	8/22/2025 1:05 PM	007 File	1 KB
archive.zip.008	✓	8/22/2025 1:05 PM	008 File	1 KB
archive.zip.009	✓	8/22/2025 1:05 PM	009 File	1 KB
archive.zip.010	✓	8/22/2025 1:05 PM	010 File	1 KB

These parts are **not independent archives**; they must be reassembled before the data can be accessed. Tools like **WinRAR** and **7-Zip** automatically join these segments when you open the first part.



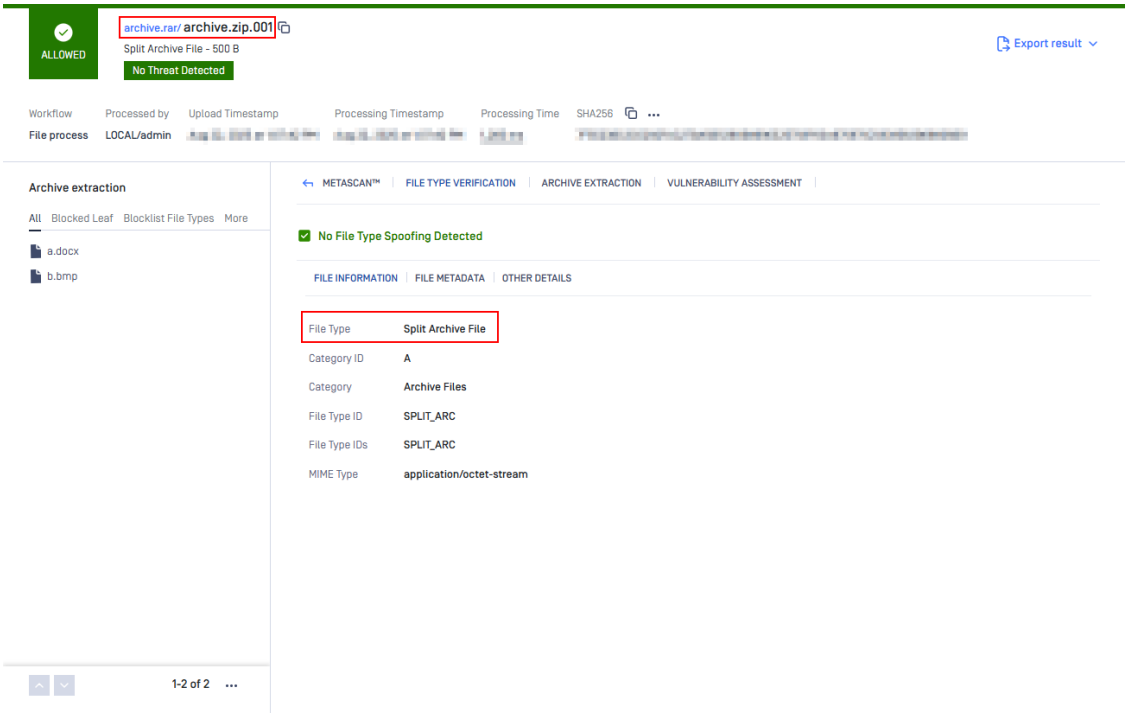
How MetaDefender Core Processes Split Archives

1. First Part Contains the Archive Header

The **.001** file includes:

- The **archive header** (e.g., ZIP or RAR signature).
- Information about the compressed files inside.
- The initial portion of the compressed data.

MetaDefender Core leverages the **File Type Engine** to detect this header and classifies the file as a **Split Archive File**, as shown in the screenshot:



This allows MetaDefender Core to treat **.001** as the entry point for scanning and extraction.

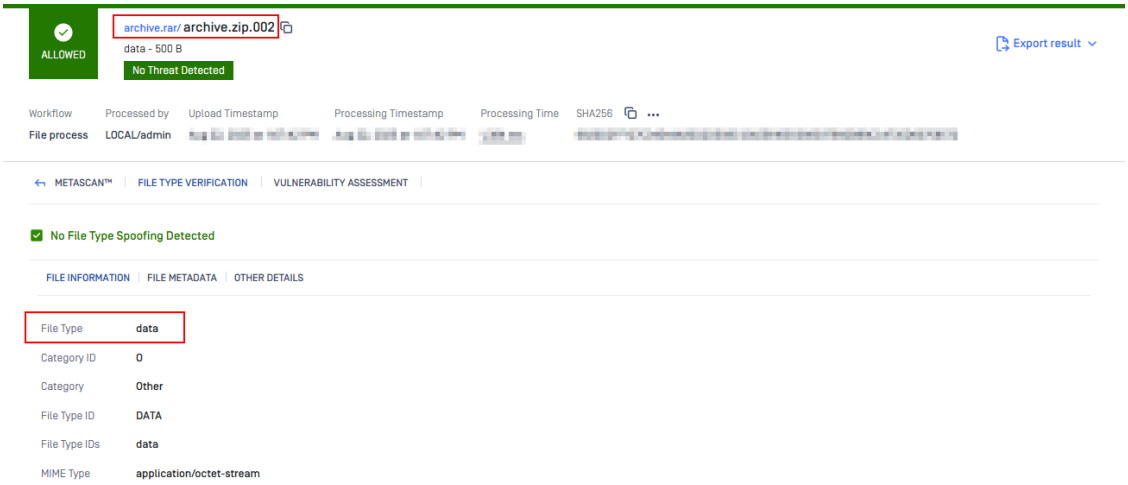
2. Subsequent Parts Contain Only Raw Data

Files like **archive.zip.002**, **archive.zip.003**, etc.:

- Contain **continuation data only**.
- Lack headers or metadata to identify them as archives.
- Cannot be opened or scanned independently.

This is why MetaDefender Core labels them as **data**:

This allows MetaDefender Core to treat **.001** as the entry point for scanning and extraction.



This is standard behavior: these segments are inert on their own and meaningful only when combined with **.001**.

Conclusion

MetaDefender Core's classification of only the first segment of a split archive as an archive file is not an error - it's a reflection of how split archives are designed. The first file (**.001**) contains the critical header and structure needed to rebuild and scan the entire archive. The remaining files are simply data blocks that have no standalone meaning without the first part.

By focusing on **.001**, MetaDefender Core ensures accurate detection, efficient processing, and full threat analysis of multi-part archives. Users should always provide all parts of the archive during scans and remember that **without .001, the rest of the files cannot be properly analyzed or extracted**.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Where can I find sample logs for quarantine errors?

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows and Linux systems.

Overview

When configuring a SIEM solution to parse specific logs, administrators may need to reference the exact error lines associated with quarantine errors. These errors are often difficult to reproduce intentionally. To assist with this,

sample log outputs are provided for common quarantine-related error codes.

Reference: Error Message Description Table – OPSWAT Documentation

Sample Logs by Error ID

Error ID 1182: Error occurred quarantining file

Code · html

```
10.40.50.231 May 27 10:22:44 daemon err 2025-05-27T03:22:44Z LE11-D1R5KLN3 MSCW[15604] Error adding new file to quarantine, file_id='e12ccfae3d58cbf874b122e4bbe8bf96bfd0b8d4874e56b6ed0d8823d3e2e79e', data_id='05c793857bea467e9b206060a9573c85', reason='Force exception path' [msgid: 1182]
```

Error ID 1185: Error reading file in quarantine

Code · html

```
10.40.50.231 May 27 11:07:14 daemon err 2025-05-27T04:07:14Z LE11-D1R5KLN3 MSCW[26524] Can't open quarantined file, fileId='09f414597c14cbfbfc2689a32e135b4dec11882d8d8ec7ebcfd73edd58cff0af', owner='transferquarantinedfile__2171213cb80', path='/non/existent/path/to/force/failure' [msgid: 1185]
```

Error ID 1186: Error writing data to quarantined file

Code · html

```
10.40.50.231 May 27 10:39:03 daemon err 2025-05-27T03:39:03Z LE11-D1R5KLN3 MSCW[21584] Error adding data to quarantined file, fileId='09f414597c14cbfbfc2689a32e135b4dec11882d8d8ec7ebcfd73edd58cff0af', owner='1898921464864', reason='Forced error for testing' [msgid: 1186]
```

Error ID 1188: Could not finalize quarantined file

Code · html

```
10.40.50.231 May 27 10:54:00 daemon err 2025-05-27T03:54:00Z LE11-D1R5KLN3 MSCW[6400] Error finalizing file, fileId='e12ccfae3d58cbf874b122e4bbe8bf96bfd0b8d4874e56b6ed0d8823d3e2e79e', owner='1841321109408' [msgid: 1188]
```

Usage

- Use these samples to test SIEM parsing configurations.
- Map `msgid` values to error IDs for correlation.
- Validate log ingestion, alerting, and reporting workflows without reproducing real error conditions.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

Can Syslog Reliably Detect MetaDefender Core Process Crashes?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Question

Can “warning” or “error” **Syslog** messages be used as a reliable signal that critical MetaDefender Core processes (e.g., `ometascan.exe`, `engineprocess.exe`, `nginx.exe`, `postgres.exe`) have **crashed** or become **unresponsive**? If not, are there *any* crash events that appear in Syslog?

Short answer

No. Do not rely on Syslog severity (warning/error) to detect process crashes or hangs. A hard crash (e.g., OOM, SIGKILL, power loss) often prevents the application from emitting any Syslog line at all. Even when errors are logged, they’re not guaranteed to uniquely indicate a crash of a specific process.

Use **direct process/service monitoring** at the OS level and/or infrastructure health checks.

Why Syslog isn’t reliable for crash detection

- **Crashes are abrupt.** On OOM/SIGKILL or sudden exits, the process cannot log a “we are crashing” message.
- **Severity isn’t specific.** “warning”/“error” messages can reflect transient conditions, timeouts, or engine-specific issues—not a confirmed crash.
- **Log pipeline can be affected.** If logging/forwarding is colocated, a crash or high load can drop/skip messages.

Recommended approach (process-level monitoring)

Processes to monitor

- `ometascan.exe` – main MetaDefender Core process
- `engineprocess.exe` – engine worker processes (ClamAV may appear as `clamd.exe`)
- `nginx.exe` / `nginx` – web front end
- `postgres.exe` / `postgres` – database (when bundled)

Windows

- **Service/Process checks**
  - PowerShell (example

Code · html

```
$names = 'ometascan','engineprocess','nginx','postgres'
$down = $names | ForEach-Object {
  if (-not (Get-Process -Name $_ -ErrorAction SilentlyContinue)) { $_ }
}
if ($down) { Write-Error "Missing processes: $($down -join ', ')" }
```

- - Configure **Service Recovery** (Services Recovery tab) to restart on failure and run a script to alert.
- **Crash signals you *can* forward to Syslog**
  - Windows Event Log *Application* / *System* (e.g., **Application Error** / **Service Control Manager** events).
  - Use an agent (e.g., NxLog/Winlogbeat) to forward these **OS-level** crash/stop events to your SIEM/Syslog.

Linux

- **Process checks**
  - `pgrep -x ometascan || echo "ometascan missing"`
  - `pgrep -x nginx || echo "nginx missing"`
  - `pgrep -x postgres || echo "postgres missing"`
- **systemd watchdog (if applicable)**
  - Use `systemctl is-active <unit>` and configure `Restart=on-failure`.
- **Crash signals you *can* forward to Syslog**
  - `journald` / `rsyslog` can send kernel/app crashes (e.g., `segfault`, OOM killer) to your SIEM.

*Tip: Also monitor **UI/API liveness** (TCP/HTTP) and **queue/backlog** metrics where available. A process might be up but unresponsive.*

What Syslog is good for (and not)

Use Syslog for	Avoid relying on Syslog for
Operational messages, scan outcomes, warnings that require triage	Definitive detection of *process death* or *hangs*
Trend analysis and correlation in SIEM	Immediate auto-remediation triggers

Minimal alerting examples

Windows (scheduled every minute)

Code · html

```
$must Have = 'ometascan','engineprocess','nginx','postgres'
$missing = $mustHave | Where-Object { -not (Get-Process -Name $_ -ErrorAction
SilentlyContinue) }
if ($missing) {
  # send to your alerting endpoint here
  Write-Error "MetaDefender Core critical processes down: $($missing -join ', ')"
}
```

Linux (cron every minute)

Code · html

```
missing=()
for p in ometascan engineprocess nginx postgres; do
  pgrep -x "$p" >/dev/null || missing+=("$p")
done
[ ${#missing[@]} -gt 0 ] && echo "Missing: ${missing[*]}" >&2
```

Conclusion

- **Do not** treat Syslog warning/error messages as a dependable indicator of MetaDefender Core process crashes.
- **Do** implement **OS-level process/service monitoring** and forward **OS crash/service events** (Windows Event Log, journald/rsyslog) to your SIEM for alerting and correlation.

Can MetaDefender Core Scan Embedded Payloads or Inline File Content?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core does **not** natively scan content that is embedded directly within a payload, such as files included inline in an email body or as raw binary/base64 in an API request.

To scan such content, it must first be programmatically extracted and submitted to MetaDefender Core as a standalone file or data stream via its API.

Supported Scanning Methods

There are two two scanning methods that MetaDefender Core supports

1. Remote Scan (HTTP(S) Payload Upload)

- File content is included directly in the HTTP(S) request body.
- Core scans the uploaded data as a standalone file.
- Embedded content must be extracted, saved, and then submitted via the API.

2. Local File Path Scan

- Used when the file is already accessible to Core on the local system or network.
- The file path is passed in the request, and Core reads the file directly.
- This method does not process any content embedded in the request payload.

In short:

Scan Method	File Content in Request	Scans Embedded Payload?	Notes
Remote Scan (Upload)	✅ Yes	✅ Yes (if submitted as file)	Recommended after extracting embedded content
Local File Path Scan	❌ No	❌ No	Only references a file path Core can access

Steps to Configure

To scan embedded or inline content:

1. **Extract the embedded content** Programmatically extract the embedded file from the email body, base64 blob, or structured payload. This may involve parsing and decoding the content before saving it.
2. **Convert it to a file or stream** Save the extracted content as a temporary file or buffer that can be submitted via HTTP.

3. **Upload the file to MetaDefender Core** Use the `POST /file` API endpoint to submit the file for scanning:

Code · html

```
curl -X POST "http://<core_address>:8008/file" \
  -H "apikey: <your_api_key>" \
  -F "file=@/path/to/extracted-file"
```

- The response will contain a `data_id`, which uniquely identifies the scan session:

Code · html

```
{
  "data_id": "61dffeaa728844adbf49eb090e4ece0e"
}
```

4. **Poll for scan results** Use the `GET /file/{data_id}` API to check the scan status:

Code · html

```
curl -X GET "http://<core_address>:8008/file/61dffeaa728844adbf49eb090e4ece0e" \
  -H "apikey: <your_api_key>"
```

- Keep polling until `scan_results.progress_percentage` reaches `100`.

Verify the Change

To confirm the embedded content was scanned successfully:

1. **Check scan completion**

- Wait until `progress_percentage` in the `/file/{data_id}` response reaches `100`.
- Example response field:

Code · html

```
"scan_results": {
  "progress_percentage": 100,
  ...
}
```

2. **Review the scan results**

- Once complete, examine the `scan_results` section for detected threats, actions taken, and engine

breakdowns.

3. **Search by hash (optional)**

- You can also retrieve results later using the file's hash (MD5, SHA1, SHA256) with:

```
Code · html

GET /hash/{hash_value}
```

4. **Use the Core Management Console (optional)**

- Navigate to **History** to view processed files and scan statuses via the GUI.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

How can I export complete archive scan results from MetaDefender Core using the REST API (session_id + extraction APIs)?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows systems.

Why this is needed

When scanning archives, the Core UI export may not include all archive-child details or the full set of extraction issues customers want to review. In those cases, you can authenticate via the REST API, then retrieve either (a) the top 100 extraction errors or (b) the full archive report including all extracted child files.

The examples below use PowerShell, Postman, and Linux curl, but none of these tools is mandatory. You can use any REST client that can send JSON requests and custom HTTP headers.

Prerequisites

- MetaDefender Core base URL (for example `http(s)://<core-host>:<core-port>`)
- A local/LDAP/AD user that can log in via API (username + password)
- The **data_id** of the **parent archive** scan you want to report on (from scan history / prior submission).

Login to in and obtain the session_id

What you get from Login

`POST /login` returns a JSON response containing `session_id` (and `oms-csrf-token`). The `session_id` can be used as the `apikey` header for protected API calls.

PowerShell

1. Set your Core URL and credentials:

```
Code

$CoreUrl  = "http://<core-host>:<core-port>"
$Username = "admin"
$Password = "admin"
```

2. Call **POST /login** and capture `session_id`:

```
Code

$loginBody = @{ user = $Username; password = $Password } | ConvertTo-Json
$loginResp = Invoke-RestMethod -Method Post -Uri "$CoreUrl/login" -ContentType "application/
json" -Body $loginBody
$sessionId = $loginResp.session_id
$sessionId
```

Postman

1. Create (or choose) an environment and add variables:

- `baseUrl` = `http://<core-host>:<core-port>`
- `user` = your username
- `password` = your password

2. Create a request:

- **Method:** `POST`
- **URL:** `{{baseUrl}}/login`
- **Headers:** `Content-Type: application/json`
- **Body (raw / JSON):**

Code

```
{
  "user": "{{user}}",
  "password": "{{password}}"
}
```

3. In **Tests**, store the returned session:

Code

```
pm.environment.set("session_id", pm.response.json().session_id);
```

Linux curl tool

1. Set your Core URL and credentials:

Code

```
CORE_URL="https://<core-host>:<core-port>"
USERNAME="admin"
PASSWORD="admin"
```

2. Call `POST /login`:

Code

```
curl -sS -X POST "$CORE_URL/login" \
  -H "Content-Type: application/json" \
  -d "{\"user\":\"$USERNAME\",\"password\":\"$PASSWORD\"}"
```

3. Copy the returned `session_id` value and store it in a shell variable:

Code

```
SESSION_ID="<SESSION_ID>"
```

Fetch top 100 extraction errors inside an archive

This uses: **GET** `/file/{data_id}/extraction-errors` (described as “Fetch top 100 of extraction errors inside an archive”).

PowerShell

1. Set the target archive `data_id`:

```
Code

$dataId = "<PARENT_ARCHIVE_DATA_ID>"
```

2. Call the endpoint with the `apikey` header:

```
Code

Invoke-RestMethod -Method Get `
  -Uri "$CoreUrl/file/$dataId/extraction-errors" `
  -Headers @{ apikey = $sessionId }
```

Postman

1. Add an environment variable:

- `data_id` = `<PARENT_ARCHIVE_DATA_ID>`

2. Create a request:

- **Method:** `GET`
- **URL:** `{{baseUrl}}/file/{{data_id}}/extraction-errors`
- **Headers:** `apikey: {{session_id}}`

3. Send the response returns the extraction error list (top 100 per the endpoint description).

Linux curl tool

1. Set the target archive `data_id`:

```
Code

DATA_ID="<PARENT_ARCHIVE_DATA_ID>"
```

2. Call the endpoint:

```
Code

curl -sS -X GET "$CORE_URL/file/$DATA_ID/extraction-errors" \
```

```
  -H "apikey: $SESSION_ID"
```

Fetch the full archive report including all extracted child files

This uses: **GET** `/archive/{data_id}` (described as “Fetch analysis reports containing all files in archive”).

PowerShell

1. Set the target archive `data_id`:

Code

```
$dataId = "<PARENT_ARCHIVE_DATA_ID>"
```

2. Call the endpoint with the `apikey` header:

Code

```
Invoke-RestMethod -Method Get `
  -Uri "$CoreUrl/archive/$dataId" `
  -Headers @{ apikey = $sessionId }
```

Postman

- **Method:** `GET`
- **URL:** `{{baseUrl}}/archive/{{data_id}}`
- **Headers:** `apikey: {{session_id}}`

Send the response returns the archive-level report including all files extracted/analyzed under that archive scan.

Linux curl tool

1. Set the target archive `data_id`:

Code

```
DATA_ID="<PARENT_ARCHIVE_DATA_ID>"
```

2. Call the endpoint with the `apikey` header:

Code

```
curl -sS -X GET "$CORE_URL/archive/$DATA_ID" \
  -H "apikey: $SESSION_ID"
```

Notes and common pitfalls

- If you get an HTTP error code **403 (Forbidden) / 401 (Unauthorized)**, re-check that you are passing `apikey: <session_id>` (the `session_id` from `/login` is used as `apikey`).
- If you get an HTTP error code **404 (Not found)**, confirm the `data_id` is correct and still retained by your Core retention policy (older scan artifacts may have been purged).

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Can MetaDefender Core Process Microsoft Purview Information Protection Encrypted Files?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Some organizations use **Microsoft Purview Information Protection (MPIP)** to classify and encrypt sensitive data. A common question is whether **MetaDefender Core (MD Core)** can scan and process these encrypted and classified files directly.

This article clarifies MetaDefender Core's current capabilities and provides guidance on handling such files effectively.

Issue

- Customers want to process files that are **encrypted and classified** by Microsoft Purview Information Protection using MetaDefender Core engines.
- There is uncertainty about whether MetaDefender Core natively supports these encrypted formats.
- No official documentation exists on direct integration or support for Microsoft Purview encrypted files.

Resolution

MetaDefender Core cannot process encrypted files from Microsoft Purview Information Protection. To ensure successful scanning:

- Process files with MetaDefender Core **before** they are encrypted and classified by Microsoft Purview Information Protection.

Programmatic Handling

If the goal is to detect **classification labels** (not the encrypted content itself), this can be achieved through **regex-based detection in Proactive DLP** on the file's metadata. Refer to the following documentation for implementation guidance: Leverage metadata info added by data classification systems

> **Note:**
> This method only enables label detection — it does not decrypt or scan the encrypted file content.

Additionally, encrypted files protected by Microsoft Purview are currently detected by MetaDefender Core with the following **file type identifiers**:

- `MSO-ENCRYPTED-MSIP`
- `PDF-ENCRYPTED-MSIP`

Customers can configure policies in MetaDefender Core to **allow or block** these file types based on their organization's security requirements.

Additional Notes

- MetaDefender Core supports a wide range of file formats, but encrypted content is treated as opaque data unless decrypted before scanning.
- Ensure your data flow enforces **"scan before encrypt"** as a best practice.

> **Support:**
> If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Why does MetaDefender Core not quarantine the same malicious file twice?

> **Check Your Version:**
> This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Behavior:

When scanning a known malicious file on MetaDefender Core, and quarantine blocked file is enabled, the first instance is quarantined successfully.

However, when scanning the same known blocked file again, the second instance is not quarantined because the original instance is already present in the MetaDefender Core quarantine.

Explanation:

Before adding a file to the quarantine, a hash lookup is performed to avoid adding duplicate files. Thus, if a file with the same hash already exists in the quarantine, it won't be added again.

Note:

This only works if **hash calculation** is enabled in the workflow. If hash calculation is disabled, the same duplicate file will be added to quarantine again, since the hash is unknown and cannot be compared.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

When Is the “Database Maintenance Required” Notification Triggered, and How Does the ometascan-vacuum-db Tool Improve Performance?

Check Your Version:

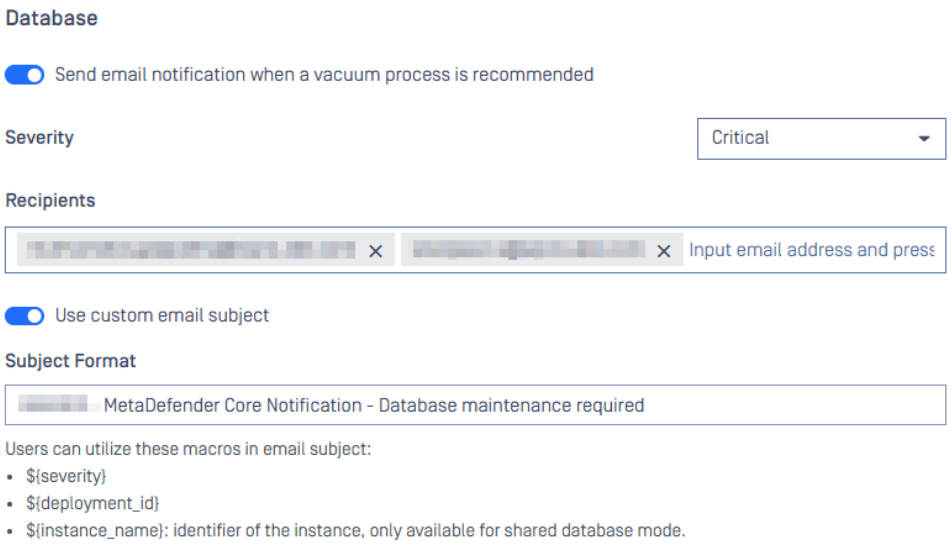
This article applies to **MetaDefender Core version 5.12.0** and above

Administrators can configure Email Notification settings to receive alerts about detected threats and incidents, providing real-time updates on potential security risks.

The **Database Monitor** keeps users informed about the current database status and notifies them when a vacuum process is required to maintain optimal performance.

For additional information about configuring email notifications, refer to:

- **Email Notification**



When Is the "Database Maintenance Required" Notification Triggered?

This notification is sent when the database accumulates a significant amount of obsolete data that could degrade performance or increase disk usage. Specifically, the trigger is based on a fixed threshold of **200,000 dead tuples**.

Dead tuple monitoring process:

- The system checks for dead tuples every hour.
- If the number of dead tuples exceeds the threshold, a "Database Maintenance Required" email notification is sent.
- After the notification is sent, the next check will occur after 24 hours.

Example: Each time a file scan result is deleted from the Scan History, the number of dead tuples increases. When these deleted entries reach 200,000 and the scheduled check interval has passed, the system automatically triggers a notification.

What Happens When Running the ometascan-vacuum-db Tool, and How Does It Improve Performance?

Running the `ometascan-vacuum-db` tool removes obsolete row versions (dead tuples), reclaims unused storage space, and updates table statistics. This helps the database query planner operate more efficiently and ensures smoother performance.

Key benefits include:

- Reduced database size on disk
- Faster query execution for reporting, UI access, and scan record retrieval
- Improved overall responsiveness of MetaDefender Core

Conclusion

Regular database maintenance is essential to ensuring consistent system performance and reliability. The **"Database Maintenance Required"** notification provides an early warning that helps prevent potential slowdowns caused by data fragmentation. By running the `ometascan-vacuum-db` tool promptly after receiving the notification, administrators can optimize storage usage, maintain faster query speeds, and enhance the overall efficiency of **MetaDefender Core**.

Keeping your database clean and well-maintained not only improves responsiveness but also contributes to a more stable and scalable environment.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to Allow Encrypted Archives in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Summary

By default, MetaDefender Core (MD Core) blocks encrypted archive files since their contents cannot be scanned. If it is necessary to allow encrypted archives to pass through (for example, when integrating with MetaDefender Email Gateway Security (EGS)), the workflow can be configured to classify these files as **"Allowed."**

This article explains how to configure the workflow so that encrypted archives are not blocked and can be forwarded even when their contents cannot be scanned.

Symptoms

Encrypted archive files sent via email are automatically blocked by MetaDefender Core or MetaDefender Email Gateway Security (EGS).

Cause

Encrypted archives cannot be scanned because their contents are protected by a password. By default, such files are blocked to prevent potential threats from bypassing security checks.

How to Allow Encrypted Archives

Follow the steps below to configure MetaDefender Core to allow encrypted archives:

1. **Access the Configuration Interface** Log in to the MetaDefender Core management console.
2. **Select the Workflow** Choose the workflow used by MetaDefender Email Gateway Security (EGS) or the relevant integration.

3. **Open the Process Settings** Navigate to **General Process**.
4. **Enable Allowed Processing Results** Enable the option **"Processing results to be classified as Allowed."**
5. **Select Encrypted Archive** Under this section, select **"Encrypted archive."**
6. **Save the Configuration** Click **Save** to apply the changes.
7. **Verify the Configuration** Confirm that the new setting is saved successfully.

Expected Result

Encrypted archives that cannot be scanned are classified as **Allowed** and forwarded according to the configured workflow.

Warning

Please note that allowing **Encrypted Archive** files may introduce potential security risks. Attackers often use encrypted archives to bypass content inspection and deliver malicious files.

We strongly recommend that you **carefully evaluate this option** before enabling it, ideally through a **cross-review process** involving multiple teams (e.g., security, compliance, and IT operations).

For example, if a system problem arises, an administrator might quickly allow encrypted archives to restore functionality — which is exactly the scenario attackers aim to exploit to gain access and compromise user machines.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

What are the pros and cons of using MetaDefender Core's archive extraction engine vs. AV engine archive handling?

Check Your Version:

This article applies to all MetaDefender Core versions

MetaDefender Core Archive Extraction Engine

Pros:

- Provides detailed scan results for each file inside the archive in the scan report.
- Delivers more effective threat detection by leveraging multiple AV engines and additional modules (e.g., CDR, DLP) that can detect threats beyond the scope of traditional AV engines.
- Required if your MetaDefender Core deployment includes **CDR** or **DLP** and you want to apply these technologies to files within archives. (You can disregard this if your license does not include CDR or DLP.)

- Might be faster for large archives being scanned by many AV engines. However, it will depend on many other factors when it comes to performance.

Cons:

- May introduce noticeable processing overhead, especially when scanning small archives or using only a few AV engines. This can result in additional seconds of total processing time.

AV Engine Archive Handling

Pros:

- Can be faster for small archives, particularly when only a few AV engines (e.g., two) are enabled.

Cons:

- May become slower when scanning large archives with many AV engines, since each engine must extract and scan the same archive independently.
- Not all AV engines support archive handling, and OPSWAT does not maintain a list of which ones do.
- The range of supported archive formats may be narrower than that of MetaDefender Core's archive extraction engine.
- Scan reports do not include per-file details for the contents of archives.

Recommendation

- **Use MetaDefender Core archive extraction** if you need detailed reporting, use CDR or DLP, or require the broadest file type coverage and most comprehensive threat detection.
- **Use AV engine archive handling** if your deployment is performance-sensitive, uses only a small number of AV engines, and does not rely on CDR or DLP.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Will adding a new Ethernet Interface to the MetaDefender Core machine impact its functionality?

Version support:

This applies to all versions before **MetaDefender Core 5.17.0**

v5.17.0 introduces support for teamed NIC (Network Interface Card) configurations, preserving Deployment ID and maintaining stable license activation when network adapters are changed or switched within the team.

Adding an additional **Ethernet interface** to a system running MetaDefender Core does not negatively affect the product's operational capabilities. When properly configured, MetaDefender Core can operate seamlessly across multiple network interfaces.

MetaDefender Core supports serving both its **REST API** and **Web UI** on multiple IP addresses. This behavior is governed by the `restaddress` registry configuration parameter. Administrators may choose to:

- Allow MetaDefender Core to listen on all available IP addresses, or
- Restrict MetaDefender Core to a specific IP address if required.

For detailed configuration instructions, refer to the MetaDefender Core Configuration documentation.

MetaDefender Configuration

While adding a secondary Ethernet interface is safe, administrators must use caution when modifying the **primary NIC**—specifically its **MAC address**. The MAC address of the initially configured primary network interface is tied to the system's **Deployment ID**.

Changing the primary NIC's MAC address will cause MetaDefender Core to update its Deployment ID. This triggers a license validation event.

Does the Deployment ID change at any time?

If a Deployment ID change occurs, MetaDefender Core automatically recognizes the modification and prompts for license reactivation. Activation based on the updated Deployment ID ensures continuous service availability.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Does MetaDefender Core support HTTP2 Protocol?

Description

This article clarifies whether MetaDefender Core supports the HTTP/2 protocol for client-server communications. It is intended for customers and integrators who are evaluating network, performance, or compatibility requirements when deploying MetaDefender Core behind load balancers, reverse proxies, or modern web servers.

Response

No, MetaDefender Core does not support the HTTP/2 protocol.

MetaDefender Core currently supports HTTP/1.0 and HTTP/1.1 only for all API and web-based communications. Native HTTP/2 connections are not supported by the MetaDefender Core application itself.

Technical Details

- MetaDefender Core's embedded web services and APIs are designed to operate over HTTP/1.x.

- Features specific to HTTP/2—such as multiplexed streams, header compression (HPACK), and binary framing—are not implemented at the application level.
- When deployed behind a reverse proxy or load balancer (e.g., NGINX, HAProxy, F5), HTTP/2 may be used externally (client → proxy), but traffic must be downgraded to HTTP/1.1 between the proxy and MetaDefender Core.

Deployment Implications

- This limitation does not affect standard API functionality, performance, or security in supported configurations.
- If HTTP/2 is required for client-facing traffic, it is recommended to terminate HTTP/2 at a reverse proxy and forward requests to MetaDefender Core using HTTP/1.1.
- This approach is fully compatible with common enterprise architectures and does not require changes to MetaDefender Core.

Summary

- HTTP/2 supported: ✗ No
- HTTP versions supported: ✓ HTTP/1.0, HTTP/1.1
- Recommended workaround: Use HTTP/2 termination at a reverse proxy, forward to MetaDefender Core over HTTP/1.1

Support:

If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

How to track Settings changes in MetaDefender Core

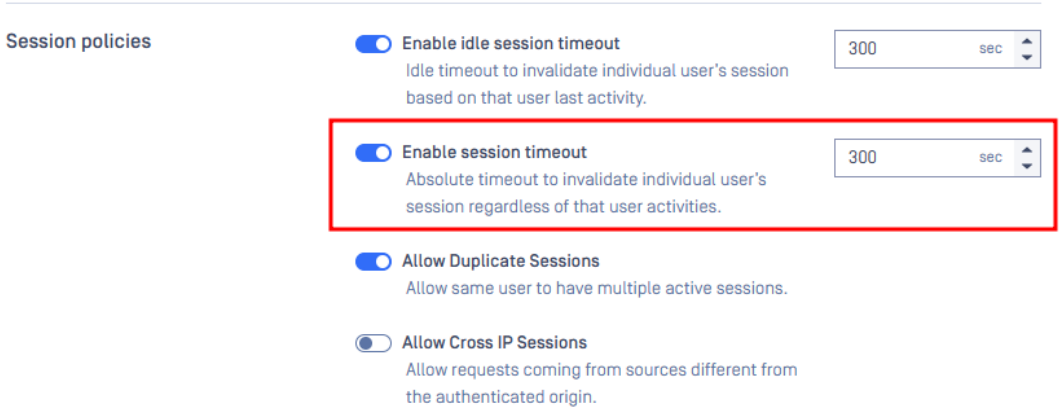
Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

This article explains how to track settings changes in MetaDefender Core.

For example: enabling **Session timeout** in Settings > Security:



Steps to Configure

- Such changes can be tracked on the UI > History > Configuration history:



- Admins can also setup a remote syslog server, and this change will show up as:

Code

```
Value changed, key='/admin/config/session', parameter='absoluteSessionTimeout', old='0',
new='300000', user='LOCAL/admin'
```

Verify the Change

- After setting up syslog server, make some configuration changes and observe if the syslog server records these changes.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How Does OPSWAT Determine Which Files Are Acceptable for Content Disarm and Reconstruction (CDR)

Overview

Content Disarm and Reconstruction (CDR) is a proactive cybersecurity technology offered by OPSWAT that sanitizes files by removing potentially malicious elements while preserving usability. A critical part of this process is file acceptability determination — the mechanism that decides which files can be safely processed by CDR.

This article explains how OPSWAT determines whether a file is acceptable for CDR processing.

File Type Risk Assessment Criteria

✅ Is it a productivity file?

This can be debatable. While the file type may be intended for machine consumption rather than human reading, it could still be considered a “productive file” depending on the specific customer use case.

✅ Is there a public specification for this file type?

Yes — there appear to be public specifications or references available, which is a good indicator for potential support.

✅ Is this file type a proprietary format?

No — we couldn’t find any indication that this format is proprietary, which makes analysis and processing more feasible.

✅ Are there known attacks associated with this file type?

No known historical attacks have been found related to this format, which makes it less likely to be supported. However, if the customer has specific threat intelligence or incident data, we encourage them to share it with us for a more accurate assessment.

⚠️ Are there potential risk objects in this file type?

This is one of the key factors in determining whether CDR support is feasible. If a file format contains only static text and numerical data, without any interactive or executable components that typically introduce risk, CDR may offer limited benefit and might not be necessary.

Does MetaDefender Core support scanning of .parquet.gpg files?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems

This is a two-part inquiry, since the file extension represents a combination of formats:

.parguet extension refers to the Apache Parquet file format. .gpg indicates that the file has been encrypted using the OpenPGP standard.

When a Parquet file is encrypted with OpenPGP, the gpg extension is appended to the existing extension, resulting in a combined extension file.

This article applies to all version of MetaDefender Core

As of File-Type Engine version 7.7.0 MetaDefender Core supports both: • The Parquet file format • The OpenPGP-encrypted file format This means that files are supported beginning with the release of File Type Engine 7.7.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Using the metadata Header to Track API File Submissions in MetaDefender Core

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

MetaDefender Core supports a `metadata` HTTP header on the `POST /file` API endpoint. This header can be used to attach custom, client-defined metadata to a file submission.

This is useful when you need to identify which internal record, transaction, user session, case, ticket, or unique file submission a scan belongs to.

The `metadata` header accepts a JSON object. The field names are defined by the client, so there is no fixed list of required metadata keys.

Use case

In some environments, the file name alone is not enough to reliably identify where a scan result came from.

For example:

- A single source system may submit many files.
- Multiple files may have the same name.
- The same file may be submitted more than once.
- The same file hash may appear in multiple submissions.
- A sanitized file must be correlated back to the original scan request.
- A scan result must be mapped back to an external ticket, case, transaction, or database record.

To avoid ambiguity, clients can submit a unique identifier, such as a UID, in the `metadata` header and store the returned MetaDefender Core `data_id` for later lookups.

How it works

When a file is submitted through `POST /file`, the client can include a `metadata` header containing a JSON object.

Example metadata payload:

Code

```
{
  "uid": "abc123",
  "source_system": "system-A",
  "client_ref": "ref-001"
}
```

The `metadata` header can be used for:

- Additional parameters for predefined post actions and external scanners, as part of STDIN input.
- Customized macro variables for watermarking text, used by the Proactive DLP engine feature.
- Additional context or verbose information for each file submission, appended into the JSON scan result.

It is strongly recommended to URL encode the `metadata` header before sending it to MetaDefender Core to help prevent unexpected issues related to encoding errors or unsafe characters.

Note for post-action and external scanner integrations

When metadata is passed to post-action scripts or external scanners through STDIN, characters that could be used for command injection, such as spaces, parentheses, angle brackets, ampersands, pipes, and similar characters, may be HTML-entity-escaped before being passed to the script or scanner.

If your post-action script or external scanner reads metadata from STDIN, make sure it accounts for this escaping behavior.

Example file submission

Code

```
curl --request POST \
  --url 'http://<core-host>:8008/file' \
  --header 'Content-Type: application/octet-stream' \
  --header 'apikey: <your_apikey>' \
  --header 'filename: test.txt' \
  --header 'metadata: {"uid":"abc123","source_system":"system-A","client_ref":"ref-001"}' \
  --data-binary '@/path/to/test.txt'
```

Example response:

Code

```
{
  "data_id": "2381c1b26a2e4de1bdb4d64883f5b91a",
  "status": "In Progress"
```

```
}
```

The returned `data_id` should be stored by the client application. It is the primary MetaDefender Core identifier used to retrieve the scan result and, if applicable, the sanitized file.

Recommended correlation flow

1. Generate a unique identifier on the client side, such as a UID.
2. Submit the file to MetaDefender Core using `POST /file`.
3. Include the UID in the `metadata` header.
4. Capture the `data_id` returned by MetaDefender Core.
5. Store the mapping between the client UID and the MetaDefender Core `data_id`.
6. Retrieve the scan result using `GET /file/{data_id}`.
7. Retrieve the sanitized file, if available, using `GET /file/converted/{data_id}`.

Example client-side mapping:

Code

```
{
  "uid": "abc123",
  "data_id": "2381c1b26a2e4de1bdb4d64883f5b91a",
  "source_system": "system-A",
  "submitted_file_name": "test.txt"
}
```

Verify metadata in the scan result

After submitting the file, retrieve the result by `data_id`:

Code

```
curl --request GET \
  --url 'http://<core-host>:8008/file/<data_id>' \
  --header 'apikey: <your_apikey>'
```

The scan result should include the submitted metadata in the JSON response.

Example:

Code

```
{
  "data_id": "2381c1b26a2e4de1bdb4d64883f5b91a",
  "metadata": {
```

```
    "uid": "abc123",
    "source_system": "system-A",
    "client_ref": "ref-001"
  },
  "process_info": {
    "...": "..."
  },
  "scan_results": {
    "...": "..."
  }
}
```

Retrieve the sanitized file

If the selected workflow performs sanitization and a sanitized output is available, retrieve it using the same `data_id`:

Code

```
curl --request GET \
  --url 'http://<core-host>:8008/file/converted/<data_id>' \
  --header 'apikey: <your_apikey>' \
  --output sanitized_file
```

This means the same `data_id` can be used to correlate:

- The original file submission
- The scan result
- The submitted metadata
- The sanitized output, if one was produced

PowerShell test example

The following example submits `C:\test.txt` with metadata, captures the returned `data_id`, and verifies that the metadata is present in the scan result.

Code

```
$coreUrl = "http://<core-host>:8008"
$apikey = "<your_apikey>"
$filePath = "C:\test.txt"

$metadata = @{
    uid = "abc123"
    source_system = $env:COMPUTERNAME
    client_ref = "ref-001"
} | ConvertTo-Json -Compress
```

```
$headers = @{
    "apikey" = $apikey
    "filename" = [System.IO.Path]::GetFileName($filePath)
    "metadata" = $metadata
}

$submitResponse = Invoke-RestMethod `
    -Method Post `
    -Uri "$coreUrl/file" `
    -Headers $headers `
    -ContentType "application/octet-stream" `
    -InFile $filePath

$dataId = $submitResponse.data_id
Write-Host "Data ID: $dataId"

$result = Invoke-RestMethod `
    -Method Get `
    -Uri "$coreUrl/file/$dataId" `
    -Headers @{ "apikey" = $apikey }

$result.metadata | ConvertTo-Json -Depth 10
```

Expected metadata output:

Code

```
{
  "uid": "abc123",
  "source_system": "HOSTNAME",
  "client_ref": "ref-001"
}
```

Linux curl test example

The following example submits `/tmp/test.txt` to MetaDefender Core with a custom UID in the `metadata` header, captures the returned `data_id`, and then retrieves the scan result to verify that the metadata is present.

Code

```
CORE_URL="http://<core-host>:8008"
APIKEY="<your_apikey>"
FILE_PATH="/tmp/test.txt"

METADATA='{"uid":"abc123","source_system":"linux-host","client_ref":"ref-001"}'

SUBMIT_RESPONSE=$(curl -sS --request POST \
  --url "$CORE_URL/file" \
  --header "Content-Type: application/octet-stream" \
```

```
  --header "apikey: $APIKEY" \
  --header "filename: $(basename "$FILE_PATH")" \
  --header "metadata: $METADATA" \
  --data-binary "@$FILE_PATH")

echo "$SUBMIT_RESPONSE"
```

Extract the `data_id` from the response using standard shell tools:

Code

```
DATA_ID=$(echo "$SUBMIT_RESPONSE" | sed -n 's/.*"data_id"[[:space:]]*:[[:space:]]*"\([^"]*\)".*/
\1/p')

echo "Data ID: $DATA_ID"
```

Retrieve the scan result and verify that the metadata is present:

Code

```
curl -sS --request GET \
  --url "$CORE_URL/file/$DATA_ID" \
  --header "apikey: $APIKEY"
```

In the returned JSON, check for the `metadata` section. It should contain the values submitted with the file, for example:

Code

```
"metadata": {
  "uid": "abc123",
  "source_system": "linux-host",
  "client_ref": "ref-001"
}
```

Important notes

- The `metadata` header applies to `POST /file`.
- The `metadata` header is only honored on `POST /file`. If submitted to `POST /process`, the header is ignored and no metadata will be returned in the result.
- The value must be a valid JSON object.
- It is strongly recommended to URL encode the `metadata` header before sending it to MetaDefender Core.
- Use simple metadata field names, without any spaces.
- Metadata is only returned when querying scan results by `data_id`. Queries by file hash do not return the `metadata` field.

- Store the UID-to-`data_id` mapping on the client side for reliable lookups.
- Use `data_id` for retrieving the scan result and sanitized output.
- Sanitized files are only available if the selected workflow performs sanitization and the processed file is retained according to the configured retention policy.
- For tracking the originating system, deployment, host, or product, use the separate `client-identity` header instead.

Troubleshooting

Metadata does not appear in the result

Confirm that the scan result is being queried by `data_id`:

```
Code

GET /file/{data_id}
```

Metadata is not returned in hash-based result queries.

API returns HTTP 400

Confirm that:

- The `metadata` header contains a valid JSON object.
- The metadata value is correctly escaped or URL encoded.
- Metadata field names do not contain spaces.

Sanitized file cannot be retrieved

Confirm that:

- The selected workflow performed sanitization.
- A sanitized output was produced.
- The sanitized file has not been removed by retention cleanup.
- The correct `data_id` is being used.

Post-action script receives escaped characters

If a post-action script or external scanner reads metadata from STDIN, some characters may be HTML-entity-escaped before being passed to the script. Update the script logic to handle these escaped values if needed.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

How is the Main Result Determined in MetaDefender Core Scanning?

Check Your Version:

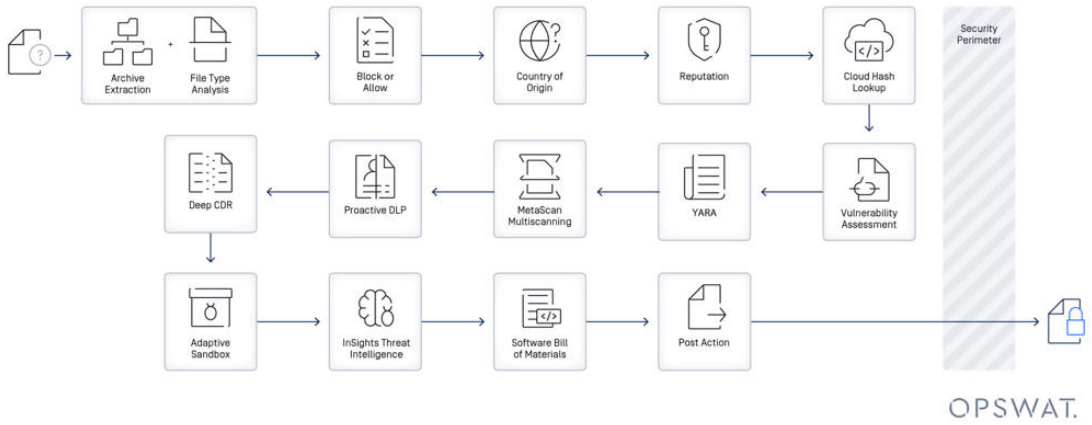
This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

MetaDefender Core scanning involves multiple sequential checks to determine the security status of a file.

Each step in the process contributes to the final decision, but the **main result** displayed is based on the earliest decisive action in the scanning flow.

Scanning Flow

The scanning process follows this order:



1. **Archive Extraction**
2. **File Type Analysis**
3. **Blocklist / Allowlist Check**
4. **Country of Origin**
5. **Reputation**
6. **Cloud Hash Lookup**
7. **Vulnerability Assessment**
8. **YARA Rules**
9. **Metascan (+ External Scan)**
10. **Proactive DLP**
11. **Deep CDR (Content Disarm & Reconstruction)**

12. **Adaptive Sandbox**
13. **Software Bill of Materials**
14. **Post Action**

Main Result Logic

If a file is blocked for multiple reasons during the scanning process, the **main result** will reflect the **highest-priority decisive reason** based on the scanning flow.

For example:

- If a file is:
  - **Known Bad** (due to **Reputation**),
  - flagged as **Potentially Vulnerable** (by **Vulnerability Assessment**),
  - and **infected** (detected by **Metascan**),

  The **main result** will display **Known Bad**, because **Reputation** is evaluated earlier in the flow and takes precedence.
- Other block reasons (e.g., Vulnerability Assessment, Metascan infection) will **not appear in the main result** but can be viewed by clicking the **three dots (…)** in the scan report for detailed information.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

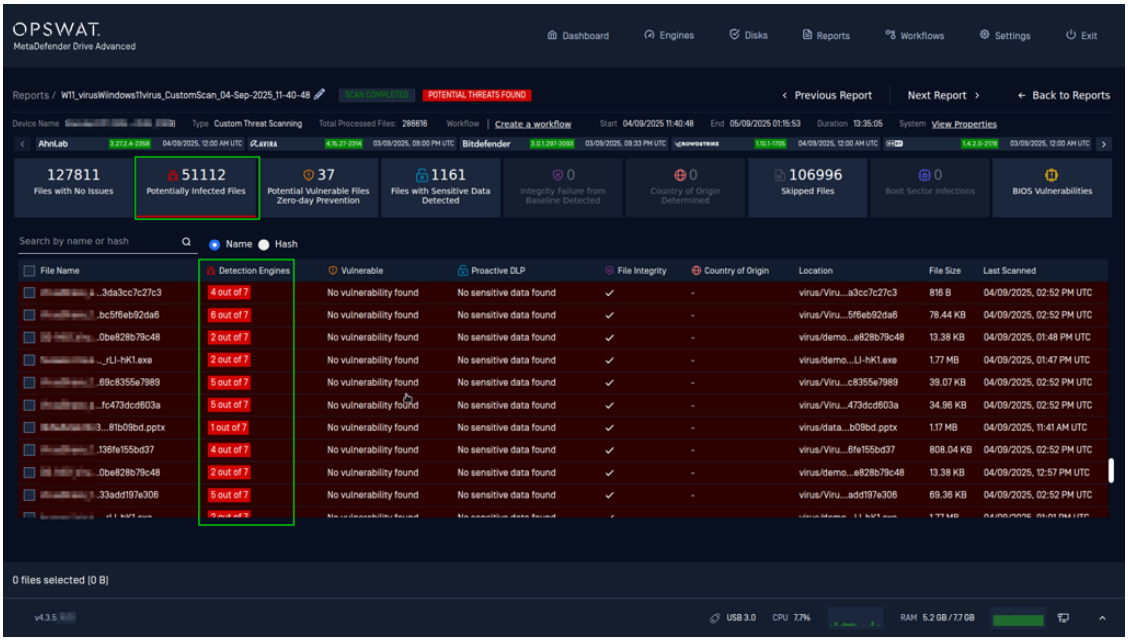
What do “Potentially Infected Files” mean in MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core v5 releases deployed on Windows systems.

Summary

A file shown under **“Potentially Infected Files”** is **not automatically confirmed as infected by MetaDefender Core itself**. It means **one or more integrated AV engines returned a malicious or suspicious verdict for that file**, and Core is **surfacing the engines’ results** so you can take a risk-based decision.



The example above is showing a MetaDefender Drive interface with some files that have been scanned by MetaDefender Core that have been flagged as Potentially Infected Files.

What triggers “Potentially Infected”

MetaDefender Core performs **multi-scanning** and aggregates vendor engine outputs. If at least one engine reports a threat (or flags it as suspicious), the file can appear as **potentially infected** in the results view/reporting.

Related scan result meanings (engine verdict level):

- **Infected**: one or more threats found by the Metascan AV engines.
- **Suspicious**: classified as a potential threat without a specific identification.

Recommended handling (best practice)

Because the finding originates from AV vendors’ detections, the safe operational recommendation is:

1. **Block / quarantine the file** (or prevent onward delivery) until it is confirmed clean.
2. **Review which engine(s) flagged it** and the **threat name** reported.
3. If the file is expected to be legitimate, treat it as a **suspected false positive** and **submit it for expert analysis** (see below).

This approach reduces the chance of allowing a true positive through while the detection is still under investigation.

How to identify exactly what was flagged (step-by-step)

Option A — Management Console (UI)

1. Go to **History** → **Processing** (Processing History).
2. Open the scan entry for the file (or the parent archive, then check contained files).
3. In the scan details, locate the **engine detection list** to see:
   - engine name
   - verdict
   - threat name / signature (where provided)

Option B — API / exported report

- Export the scan result (or retrieve via API) and review the per-engine results in the scan details.

How to confirm / correct a false positive

If you believe the file is clean but was flagged:

1. Log into **My OPSWAT**.
2. Go to **Support** → **Report False Detection**.
3. Submit the **file (or hash, depending on portal options)** for investigation by OPSWAT's analyst team.

After confirmation/correction, re-scan once the relevant engines' detections are updated (vendor-side or via analysis feedback loop).

Optional: reduce noise (with caution)

If your workflow classifies files as "Infected" based on thresholds (e.g., X engines must detect), you can tune those thresholds in the workflow settings. **Be cautious**: lowering thresholds can increase security risk.

If **Further Assistance** is required, please proceed to log a support case or chat with one of our support engineers.

Does OPSWAT MetaDefender Core Support Scanning TXF Files?

Check your version:

This article applies to MetaDefender Core and is applicable to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

Yes, TXF (Tax Exchange Format) files are supported by OPSWAT MetaDefender Core.

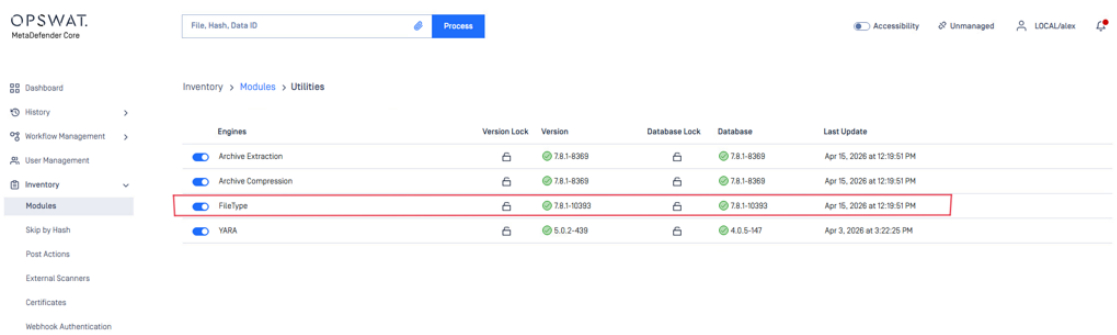
Support for scanning TXF files was introduced starting with **File Type Module version 7.8.0**. If your system is running an earlier version of the File Type Module, TXF files may not be properly recognized or scanned.

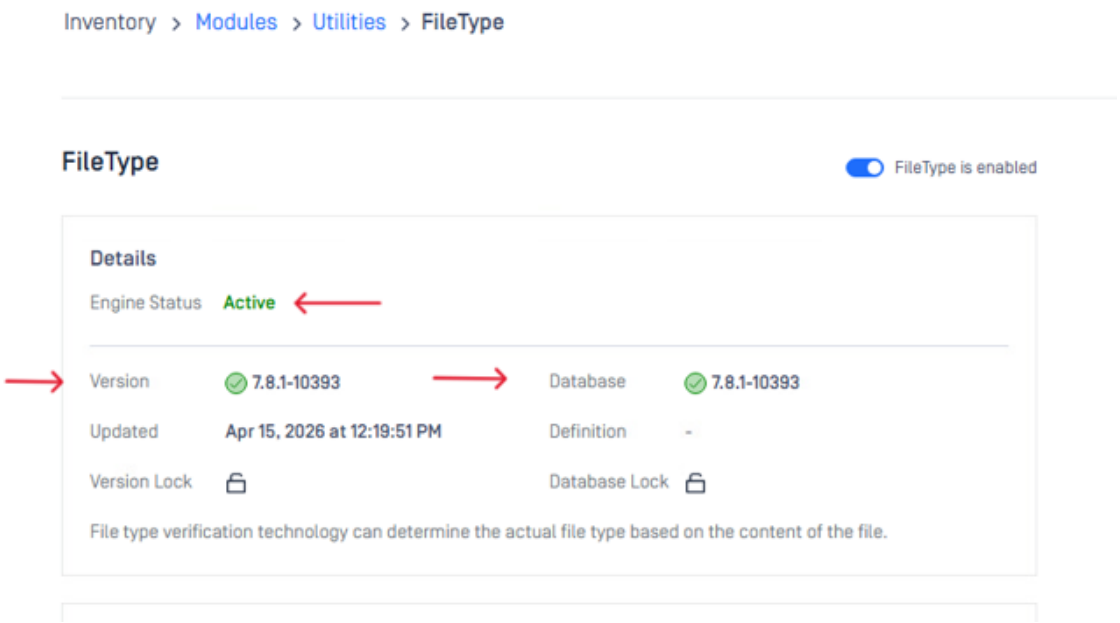
Resolution / Steps

To ensure TXF files are supported:

1. Verify your current File Type Module version in MetaDefender Core.
2. Update your modules to the latest available version.
3. Confirm that the File Type Module is at **version 7.8.0 or later**.

In order to check the File Type Module status and version, in your MetaDefender Core web UI, navigate to: Inventory > Modules > Utilities > File Type. Make sure the engine is enabled - **Engine status: Active**, and there is a **green checkmark next to Version and Database.**





Keeping your modules up to date is essential to ensure full file type coverage and optimal detection capabilities.

Additional notes:

No additional configuration is required once the appropriate module version is installed.

Regular module updates are recommended to maintain support for newly added file formats and improvements.

Support:

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

Why Are Office Documents Counted as Multiple Files During Archive Extraction?

Check Your Version: This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue:

Some customers observe that Microsoft Office documents (such as DOCX or XLSX) may be treated as multiple files by security inspection engines, even when the document does not contain any visible attachments or embedded files. This behavior can lead to unexpected blocking or processing limits being reached.

Explanation:

Modern Microsoft Office files use the Office Open XML format. These file types are ZIP-based containers, not single flat files. Internally, an Office document consists of multiple components, including:

- XML files that define document content and structure
- Metadata and relationship files
- Style definitions, headers, and document properties
- Optional embedded objects (even if not apparent to the user)

During archive or content extraction, the security engine processes these internal components individually rather than treating the document as one opaque file. As a result, each internal element is counted separately.

Why Office Documents Cannot Be Treated as a Single File:

From a security perspective, Office Open XML documents cannot be handled as a single “black-box” file. Analyzing only the outer container would prevent inspection of internal components and could allow threats such as:

- Embedded malicious payloads
- Malicious macros
- Exploit-based XML content

- ZIP bombs disguised as Office documents

To effectively detect and prevent these threats, the security engine must unpack and analyze each internal component of the document.

Is it possible to count the whole Office document as a single file?

No. Office Open XML documents are intentionally decomposed during inspection so that their internal structure can be fully analyzed. Treating them as a single file would reduce visibility and weaken security protection.

Support: If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

How to View SSL/TLS Certificate Details from the Command Line (Windows and Linux)

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Issue:

When troubleshooting an SSL/TLS certificate issue, you may need to confirm that the certificate contains the required information, such as the Subject Alternative Name (SAN), expiration date, or issuer. You can view these details from the command line on both Windows and Linux without installing additional certificate management tools.

Solution:

Use the commands below to print the contents of a certificate file. Both PEM-encoded (`.pem`, `.crt`) and DER-encoded (`.cer`, `.der`) certificates are supported.

Windows:

Windows includes the built-in `certutil` tool. Open **Command Prompt** or **PowerShell** and run one of the following.

Offline environment - Print the certificate details only:

Code

```
certutil -dump C:\path\to\cert.cer
```

(Optional) For Internet connected environment - Print the certificate details and validate the full certificate chain, including CRL/OCSP revocation checks:

Code

```
certutil -verify -urlfetch C:\path\to\cert.cer
```

Linux:

OpenSSL must be installed before you can run the commands in this section. Most distributions include it by default; if not, install it with your package manager.

Ubuntu / Debian / Linux Mint

```
Code

sudo apt install openssl -y
```

RHEL / CentOS / Fedora / Rocky Linux

```
Code

sudo dnf install openssl -y
```

Once OpenSSL is installed, run the command that matches your certificate format.

PEM-encoded certificate:

```
Code

openssl x509 -in cert.pem -noout -text
```

DER-encoded (.cer) certificate:

```
Code

openssl x509 -in cert.cer -inform DER -noout -text
```

> ***Tip:*** *If you are not sure which format the file is in, open it in a text editor. A PEM file begins with `-----BEGIN CERTIFICATE-----`; a DER file appears as unreadable binary.*

Verifying the Output

In the output, check the following fields to confirm the certificate is correct:

Field	What to check
Issuer	The Certificate Authority (or your internal CA) that signed the certificate. For a self-signed certificate, Issuer and Subject are identical.
Subject	The Common Name (CN) of the server the certificate was issued to.

Field	What to check
Not Before / Not After	The validity period. Confirm the current date falls within this range.
Subject Alternative Name (SAN)	All hostnames and IP addresses the certificate is valid for. The hostname clients use to connect must appear here.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

When should I apply the new license?

Check Your Version:

This article applies to all OPSWAT Metadefender products with a License key on Windows and Linux systems.

Overview

You may activate your new renewal license (activation key) **before the current license expires**, once provided by OPSWAT.

In many cases, you will be provided the renewal license key **several weeks prior to the expiration of the existing license**. Activating the new license key in advance helps to ensure **continuous operation and prevents any potential service disruption**.

Applying the new license

Please find the license documentation for the product at https://www.opswat.com/docs

For online activation, please hit activate license and paste the key.

For offline activation, please follow the steps provided on the screen.

When to Use “Deactivate”

In most cases, you do not need to deactivate the existing license before applying the new one.

However, if the system’s network configuration has changed such as the MAC address, there will be a new deployment ID, you may need to select Deactivate before activating the new license.

If you encounter an activation error:

1. Click Deactivate.
2. Repeat the activation process using the new license key.
3. If you continue to encounter an error, please contact OPSWAT support.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

How to check whether AV engines are running the latest update from the OPSWAT update server?

Check Your Version: This article applies to all MetaDefender Core versions. The Health Check outdated threshold mentioned in the Recommendation section requires MetaDefender Core 5.21.0 or later.

Overview

This article explains how to confirm that the antivirus (AV) engines on MetaDefender Core are running the latest engine and definition (database) packages published on the OPSWAT update server. Use it when you need to verify update status beyond the engine status shown in the web console, for example after a failed update or when building monitoring and alerting.

The check has two parts:

1. Read what is installed on MetaDefender Core using the Engines Status API.
2. Compare it against the latest packages published by OPSWAT in the MetaDescriptor file.

Prerequisites

- A MetaDefender Core API key (or session ID) with permission to call `GET /stat/engines`.
- Network access to `update.dl.opswat.com` from the machine where you download the MetaDescriptor file. In offline environments, download the file on an internet-connected machine.

Steps

Step 1: Check the installed packages with GET /stat/engines

1. Call `GET /stat/engines` on MetaDefender Core (see the Engines Status API documentation). For example:

Code · bash

```
curl -H "apikey: <YOUR_API_KEY>" http://<CORE_HOST>:8008/stat/engines
```

2. For each engine, note these fields:

- `def_time` — the timestamp of the definition (database) package currently installed.
- `download_time` — the time MetaDefender Core downloaded the **engine** package. This is not the download time of the definition package, which is updated more often than the engine itself.
- `eng_ver` — the version of the engine package currently installed.

3. Compare `def_time` with your expected update frequency. If it is older than expected, continue to Step 2 to confirm whether a newer package is available.

Note: When an update check completes successfully but no newer package is available, neither `def_time` nor `download_time` changes. Both fields describe the package currently installed, not the result of the most recent update check. On their own, they cannot confirm that the installed packages are the latest published by OPSWAT.

Step 2: Compare against the latest packages published by OPSWAT

1. Download the MetaDescriptor file from `http://update.dl.opswat.com/metadescriptor`.
2. Find the entry for the engine using its `engine_id`. Example (values are for illustration only):

Code

```
engine_id: bitdefender_1_linux
engine_type: av
engine_package_version: 3.0.1.399-2537
engine_url: package/bitdefender_1_linux/bitdefender_1_linux-engine-1784748762-1784748769.yml
database_package_version: 7.101529
database_url: package/bitdefender_1_linux/bitdefender_1_linux-database-1789740974-1789741035.yml
```

3. Identify the latest packages:

- `engine_package_version` — the latest engine package.
- `database_package_version` — the latest definition (database) package.

4. To find when a package was released on the OPSWAT update server, read the first Unix timestamp in its URL. In the example above, `1789740974` in `database_url` corresponds to 2026-09-18 14:16:14 UTC.
5. Compare the published values with the values installed on MetaDefender Core (from Step 1).

Interpret the results

Result	Meaning
`def_time` is the same as or later than the release time of the latest database package	The engine is running the latest definitions.
`def_time` is earlier than the release time of the latest database package	A definition (signature) update is pending.
Installed engine version differs from `engine_package_version`	An engine update is pending.

If an update is pending for longer than your update frequency, check the update source and frequency settings, network or proxy access to the OPSWAT update server, and whether the engine is pinned to a specific version.

Recommendation

For monitoring and alerting, combine all three signals:

- **Health Check outdated threshold** (Health Check API, version 5.21.0 or later) as a coarse alarm for outdated engines.
- `def_time` from Engines Status (`GET /stat/engines`) to track the age of installed definitions.
- **MetaDescriptor comparison** to confirm the installed packages match the latest versions published by OPSWAT.

Support: If you need further assistance, please log a support case or chat with our support engineers.

How to Use a RAMDISK for the tempdirectory?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

In order to improve the file scan speed, a custom *tempdirectory* can be set for Metadefender Core.

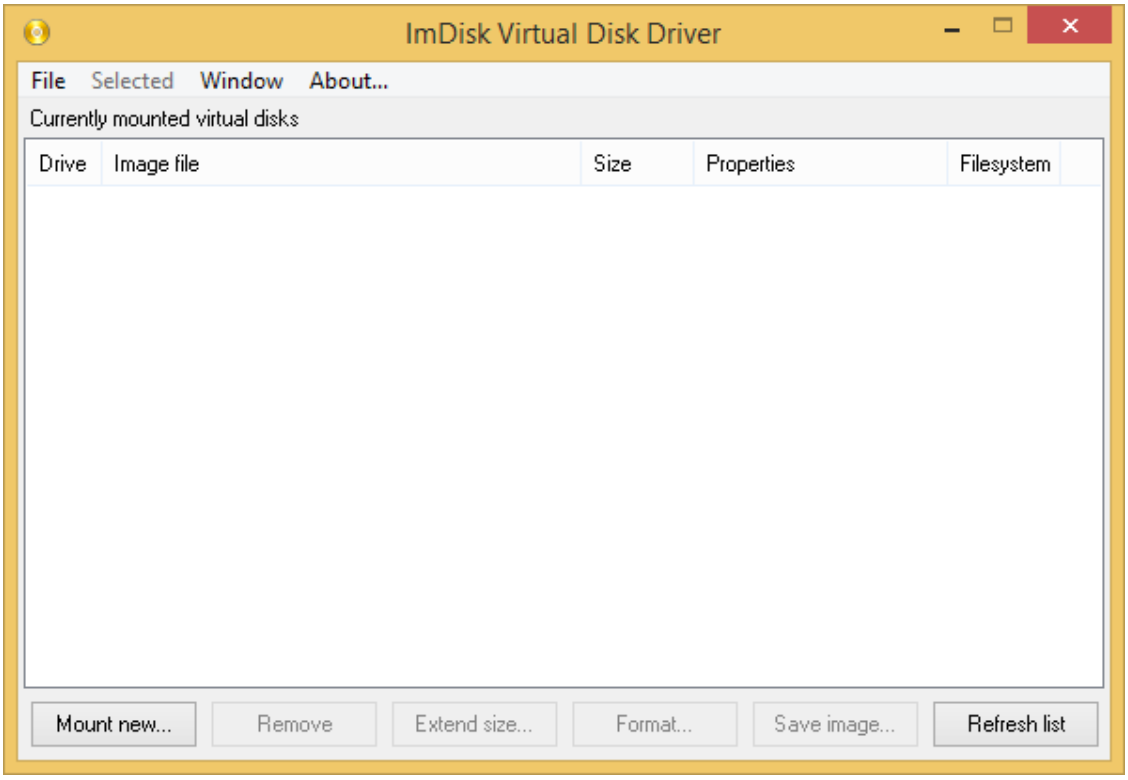
Instructions for Windows

Step 1:

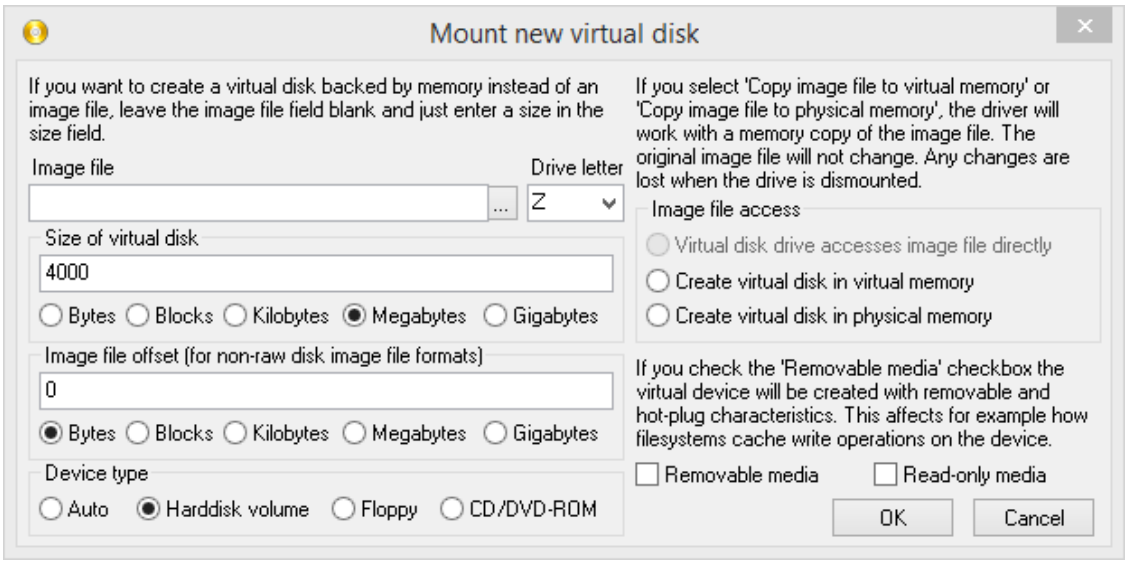
Create a RAMDISK on your system.

We recommend the following tool for this: Tools and utilities for Windows

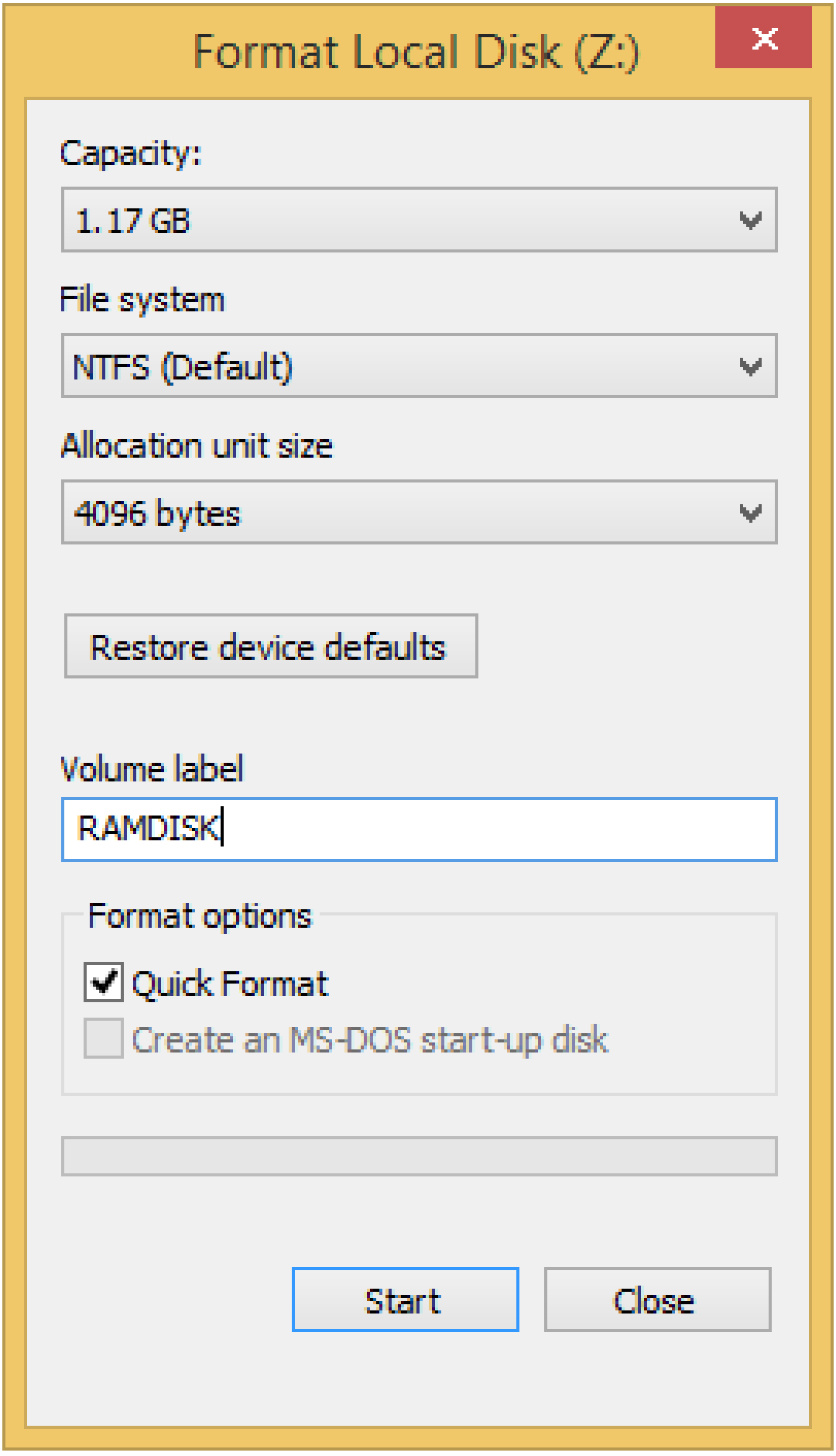
After installation, use the **ImDisk Virtual Disk Driver** application, to create a new RAMDISK.



Mount new virtual disk:



After disk creation, windows will ask you to format the new disk.

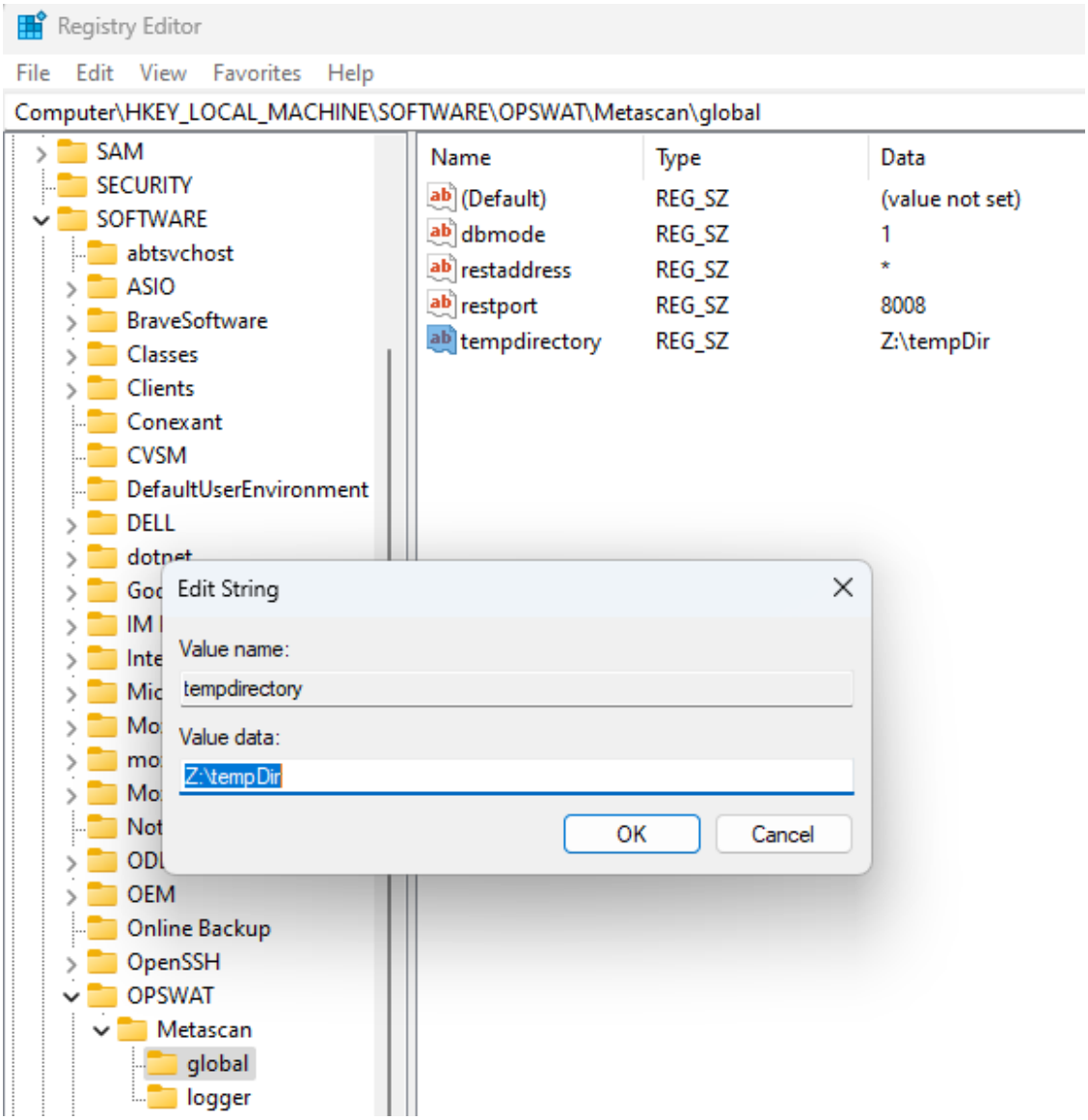

Format Local Disk (Z:)
Capacity:
1.17 GB
File system
NTFS (Default)
Allocation unit size
4096 bytes
Restore device defaults
Volume label
RAMDISK
Format options
Quick Format
Create an MS-DOS start-up disk
Start
Close

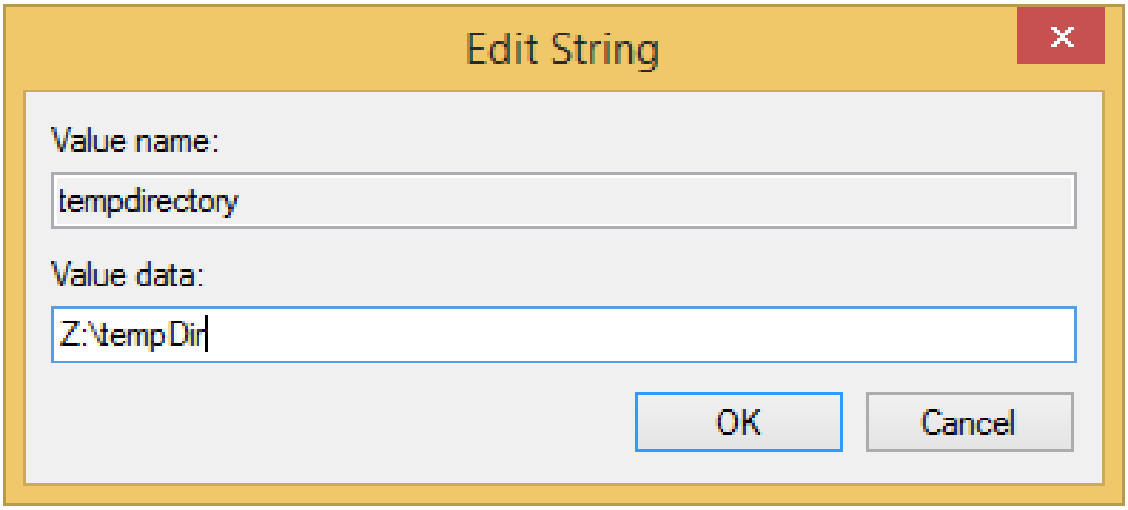
Step 2:

Create, or edit the following registry entry:

> **Info**
>
> HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global\tempdirectory

where *tempdirectory* should be a string value with a desired location, for example: *Z:\tempDir*





The recommended minimum size for the *tempdirectory* is at least 10GB, and also:

- For non archive files: the average file size * 40
- For archive files: the average archive size * 200

Step 3:

The changes to take effect restart the **ometascan** service.

After the service restart, your custom directory will be used for temporary file storage during file scan.

Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

Why does ClamAV seem to be slower than other engines?

Check Your Version:

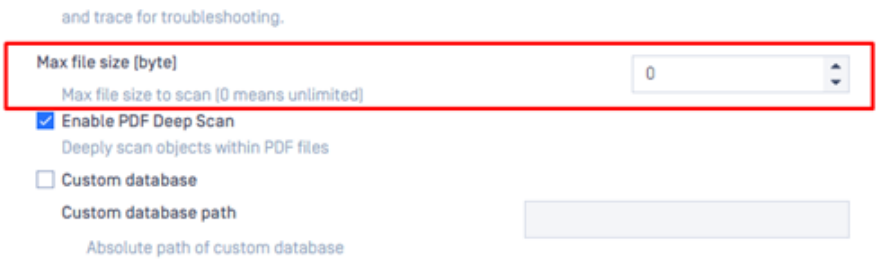
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Due to certain engine-specific design factors, ClamAV is known to be one of the slower performing engines included in the MetaDefender Core system.

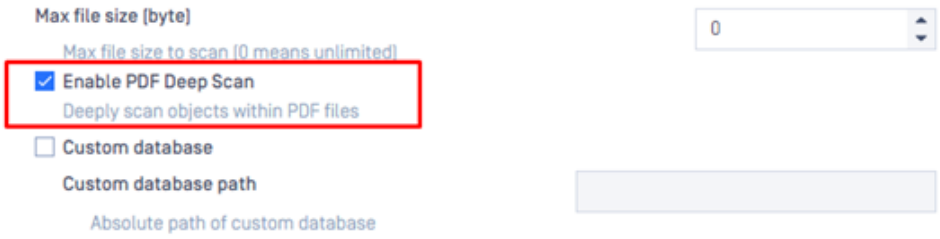
However, ClamAV's slow scan speed is easily counterbalanced by its high detection rate, quality, and stability.

As a workaround for the slow scan time, users can change some settings on the actual engine as described below:

- Limiting the file size that the engine should process by changing "**Max File Size**"



- Disabling **"PDF Deep Scan"** also makes the PDF scan faster but the trade-off is less detection rate.



Pro Tip:

Should you still wish to remediate slow performance, OPSWAT suggests you set the **Engine Timeout** on MetaDefender Core for **Engine Level.**

ClamAV will then scan as usual, but for files that take longer to scan, ClamAV will stop its scanning process.

Support:

If you are having issues or require further assistance regarding **Clam AV**s performance on MetaDefender Core, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How does MetaDefender Core determine the runtime nginx.conf location on Windows?

Info:

Check Your Version:

This article applies to **all MetaDefender Core releases deployed on Windows**.

Issue:

After upgrading to MetaDefender Core 5.17.1 or later (see **Windows Temporary Folder Path Changes for SYSTEM**

Account for details regarding the Local System account), or configuring the MetaDefender Core service to run under a different Windows account, users may be unable to locate the runtime `nginx.conf` file.

Historically, the runtime configuration was commonly found under:

```
C:\Windows\Temp\ometascan\nginx.conf
```

However, depending on the MetaDefender Core version and the account configured to run the service, the runtime configuration may be generated in a different location.

Main cause:

The runtime `nginx.conf` is generated automatically during MetaDefender Core service startup.

The location is determined by the temporary directory resolved for the account configured in the **Log On As** setting of the **OPSWAT MetaDefender Core** service.

For services running as **Local System**:

MetaDefender Core Version	Runtime Location
Earlier than 5.17.1	`C:\Windows\Temp\ometascan\`
5.17.1 and later	`C:\Windows\SystemTemp\ometascan\`

For services running under **any Windows user account**, the runtime configuration is generated under the temporary directory resolved for that account.

For example:

```
C:\Users\<username>\AppData\Local\Temp\ometascan\
```

The exact location depends on the value of the account's **TEMP** environment variable.

Resolution:

Step 1. Verify the account running the MetaDefender Core service

Open:

```
services.msc
```

Locate:

```
Code

OPSWAT MetaDefender Core
```

Open **Properties** → **Log On** and identify the account configured to run the service.

Step 2. Locate the runtime configuration

If the service runs as **Local System**:

- MetaDefender Core earlier than **5.17.1**

```
Code

C:\Windows\Temp\ometascan\
```

- MetaDefender Core **5.17.1 or later**

```
Code

C:\Windows\SystemTemp\ometascan\
```

If the service runs under a **Windows user account**, navigate to that account's temporary directory.

Typical example:

```
Code

%TEMP%\ometascan\
```

or

```
Code

C:\Users\<username>\AppData\Local\Temp\ometascan\
```

The runtime directory contains files such as:

- `nginx.conf`
- `ometascan-nginx.pid`
- NGINX temporary directories

Step 3. If the directory is missing

Restart the **OPSWAT MetaDefender Core** service.

The runtime configuration is regenerated automatically during service startup.

Notes:

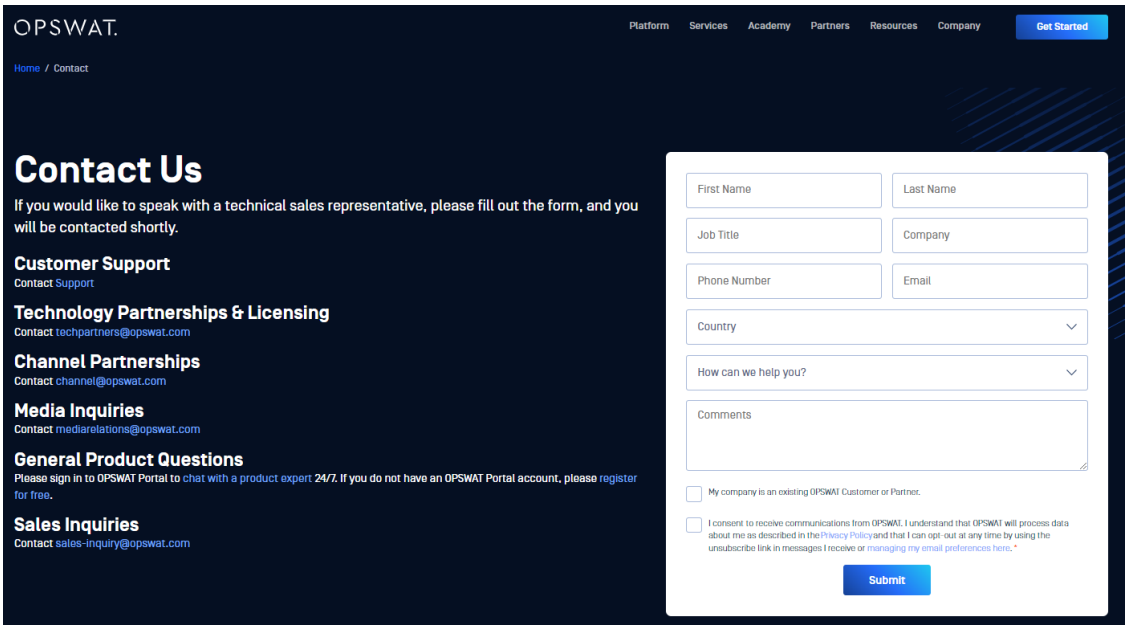
- The generated `nginx.conf` is the runtime configuration used by the embedded NGINX instance.
- The runtime `nginx.conf` should **not** be modified directly.
- If custom NGINX configuration is required, follow the supported method for loading additional NGINX configuration files.

If Further Assistance is required, please proceed to log a support case or chat with our support engineer.

How do I obtain a Trial License for MetaDefender Core?

If you're interested in obtaining a trial license for any of our products, including MetaDefender Core, please follow the steps below:

1. Navigate to the OPSWAT **Contact** page via This Link
2. Fill out the form with all of the required information.
3. Include additional details, as well as your reason for obtaining the trial license, in the **Comments** field.
4. Upon completion, click **Submit,** as highlighted in the screenshot below.
5. You should be contacted by the OPSWAT CX team shortly with a trial key for the requested product.

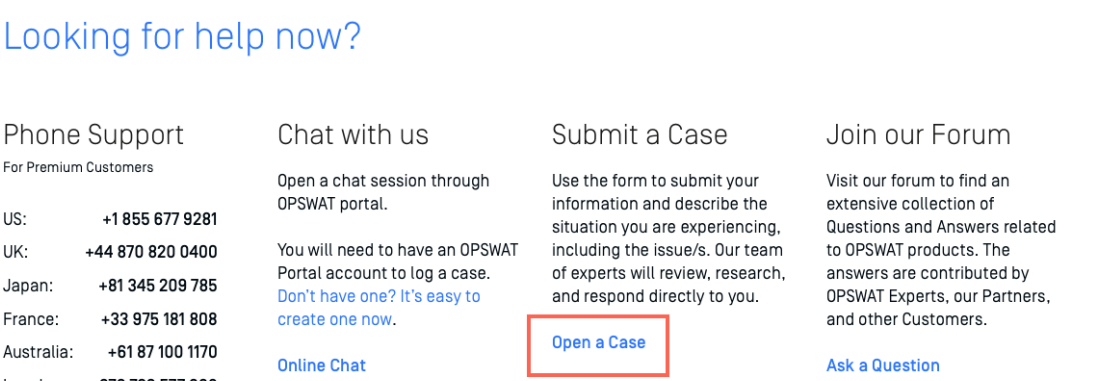


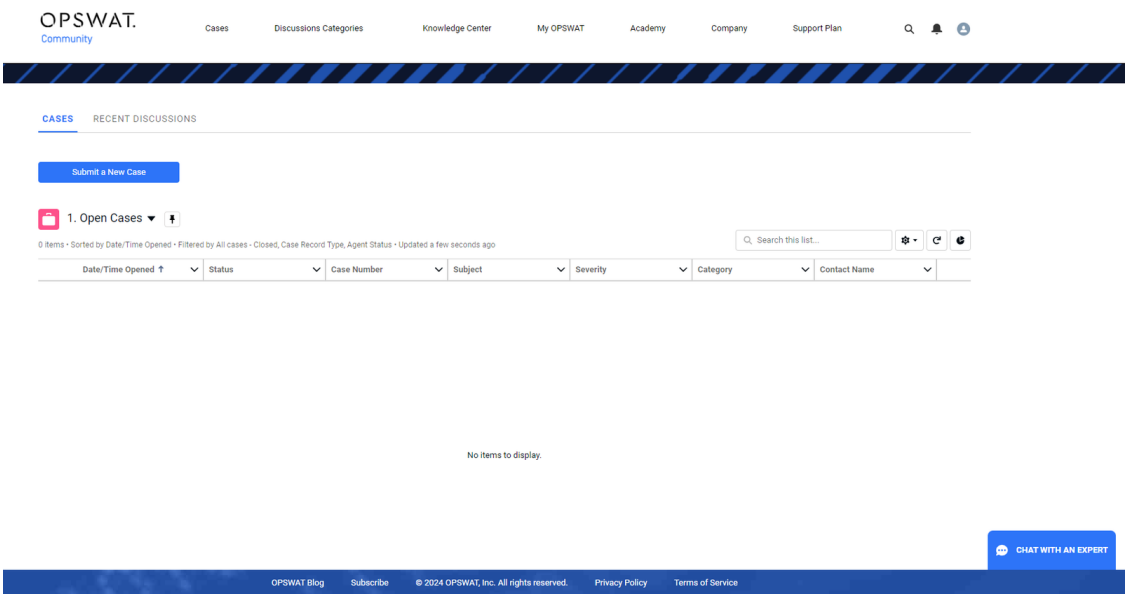
Support:

Should you receive no contact from OPSWAT within 48 hours of submitting the contact form, please follow the instructions below to request your trial via a support ticket:

1. Click This Link to visit our **Support** page.
2. Click the **Open a Case** option, as highlighted in the first screenshot below.
3. When redirected to the OPSWAT Portal, register your free OPSWAT account to open a case.
4. Log into the OPSWAT Portal.
5. Click **New**, as highlighted in the second screenshot below.
6. Choose your product and fill out all the necessary fields included in the case form.
7. Include your original details and comments, for attention: the OPSWAT CX team.

Finally, click **Save** on your support ticket and await your reply from OPSWAT.





How long is the support lifecycle for a specific version of MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems. Version 3 and prior releases are no longer under support.

OPSWAT provides support on each release of MetaDefender Core for 18 months after the publication of the **following** release of the product. So, once a new release is published, you have 18 more months of support on the previous release.

Keep in mind, however, that bug fixes and enhancements are applied only to the next release of a product, not to the current release or historical releases - even when those releases are still under support.

In some cases, hot-fixes can be provided for the current release of the product, and then incorporated as regular fixes in the next release.

Pro Tip:

OPSWAT highly recommends that users regularly upgrade to the latest version of MetaDefender Core and not wait until the end of a release's support cycle.

Please take note that running an unsupported product version may result in engine failure at any time.

Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

To upgrade, follow these steps:

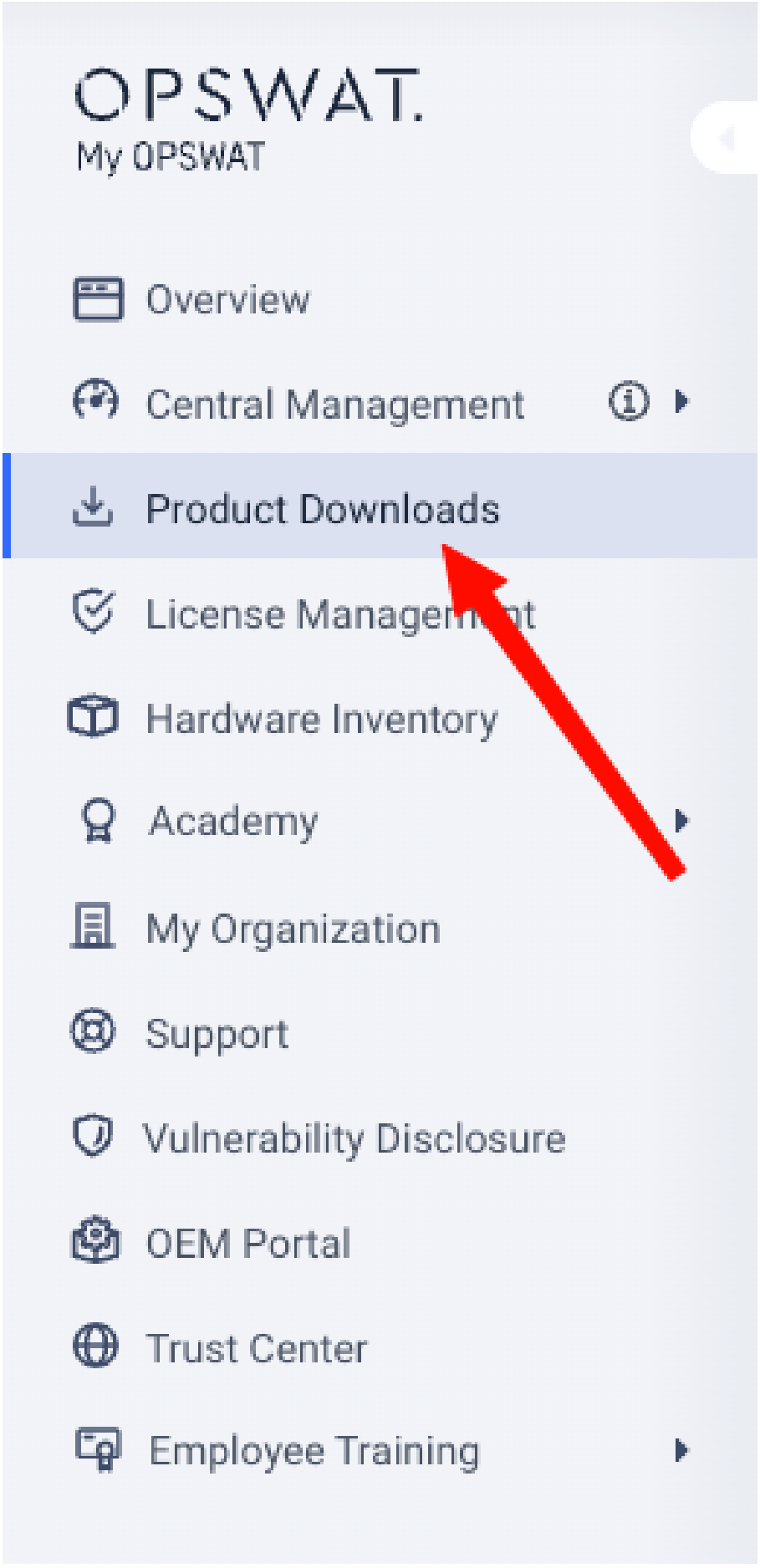
1. Browse Release notes for details on the latest MetaDefender Core version release.
2. Visit My OPSWAT to download a new version under the relevant package license.

If you are upgrading from a V4 or V5 release, visit Product Upgrading and follow the links.

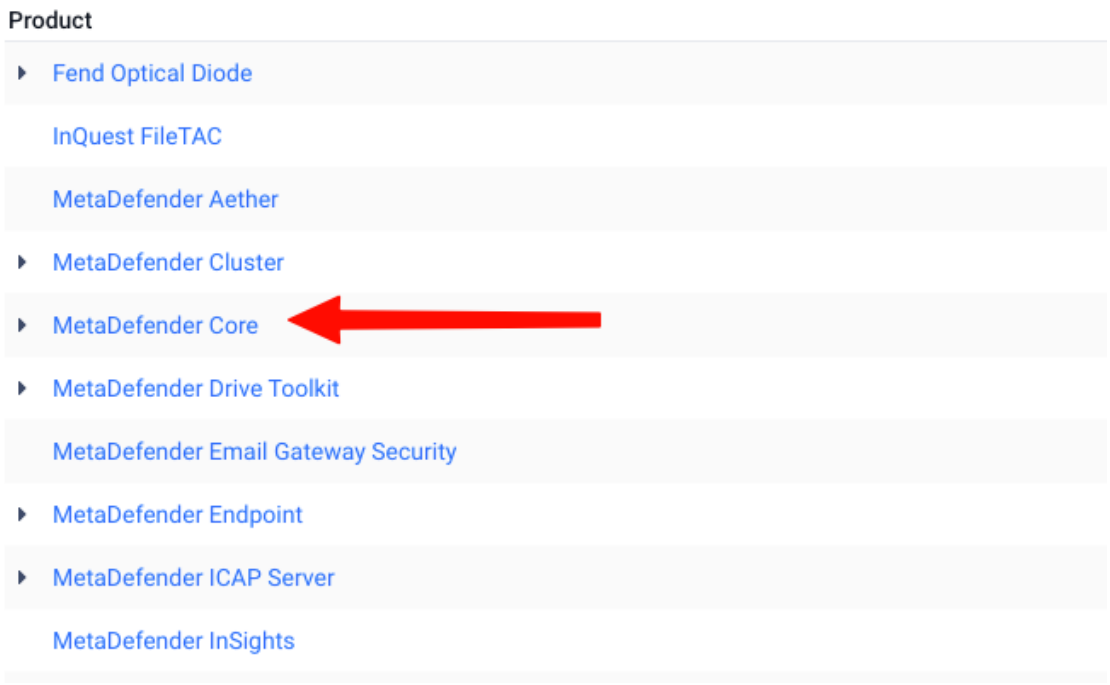
How can I check the support lifecycle for a specific MetaDefender Core version?

To check the release and end-of-life information for a specific MetaDefender Core version:

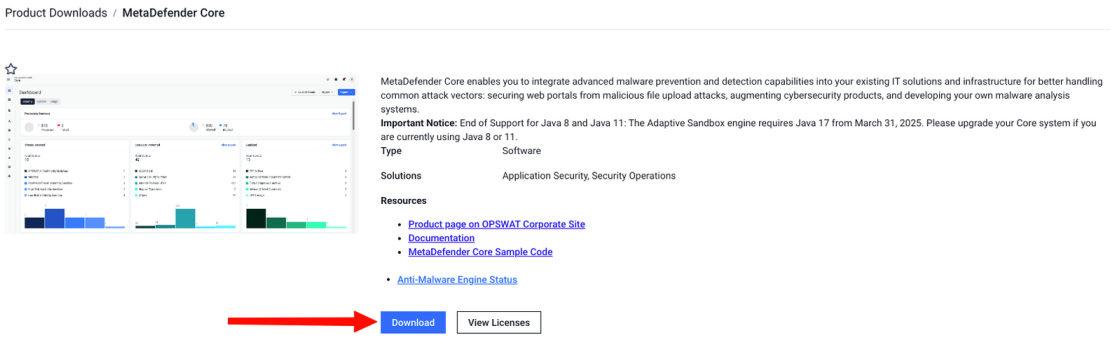
1. Sign in to My OPSWAT.
2. Go to **Product Downloads**.


OPSWAT.
My OPSWAT
Overview
Central Management
Product Downloads
License Management
Hardware Inventory
Academy
My Organization
Support
Vulnerability Disclosure
OEM Portal
Trust Center
Employee Training

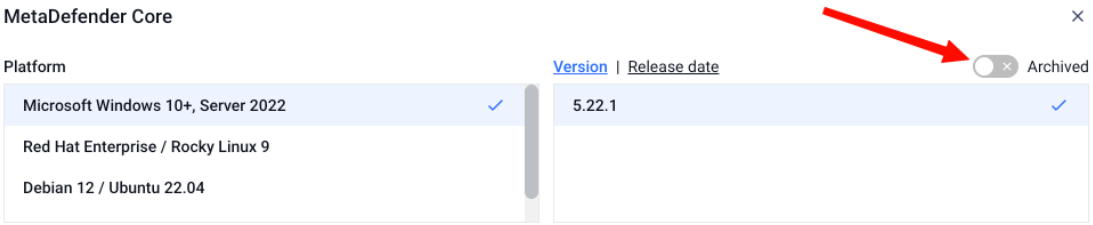
3. Select **MetaDefender Core**.



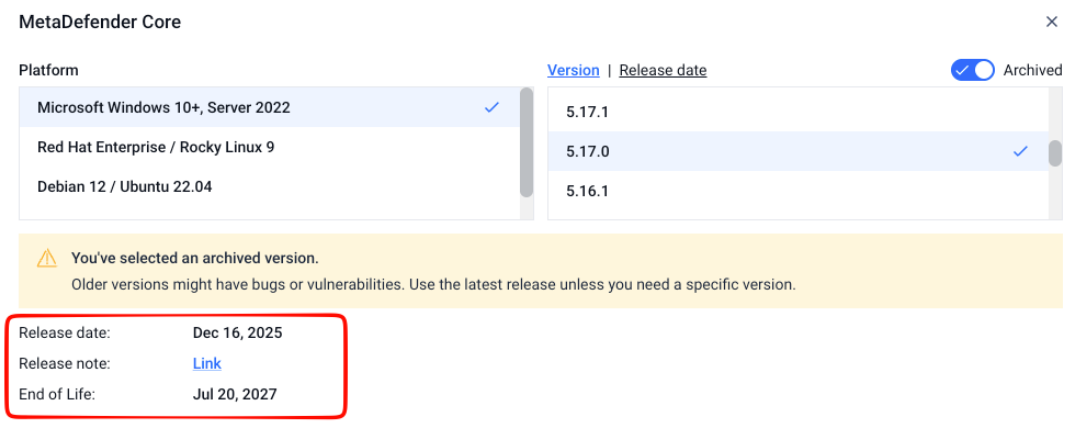
4. Click **Download**.



5. Enable **Archived** to display previous MetaDefender Core releases.



6. Select the version you want to check and review the version information displayed on the page, including its release and end-of-life information.



Support:

For any issues or queries regarding supported versions of our **MetaDefender Core**, please follow these instructions on How To Create a Support Package, before creating a support case or chatting with our support engineer.

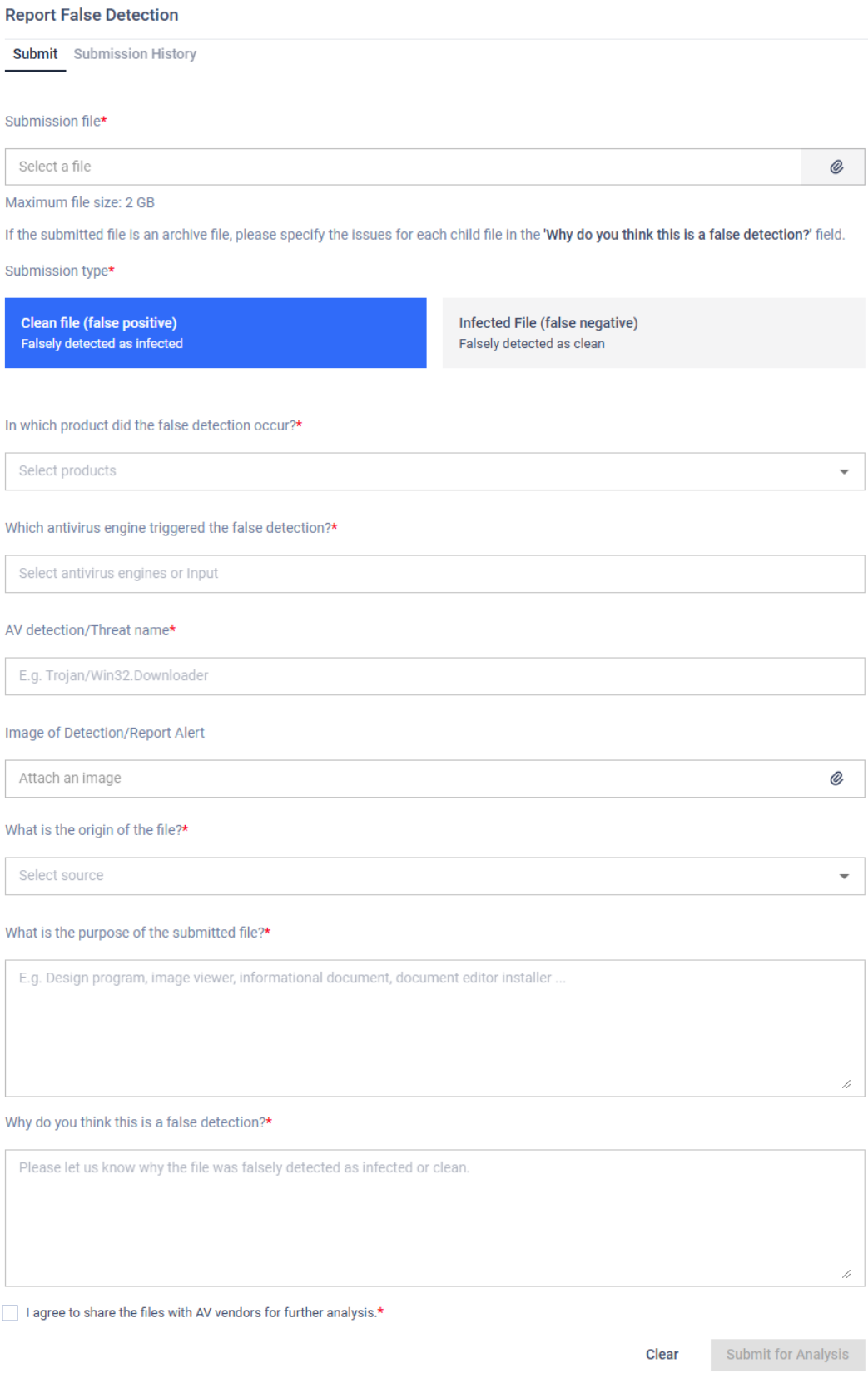
How do I get a False Detection (False Positive or False Negative) corrected?

If you are using an OPSWAT's multi-scanning feature and believe the one or more of the associated Anti-malware engines is incorrectly labeling (i.e. falsely detecting) whether a file is malicious, you can have that file analyzed by OPSWAT's expert analyst team simply by logging into My OPSWAT (my.opswat.com) and submitting a report.

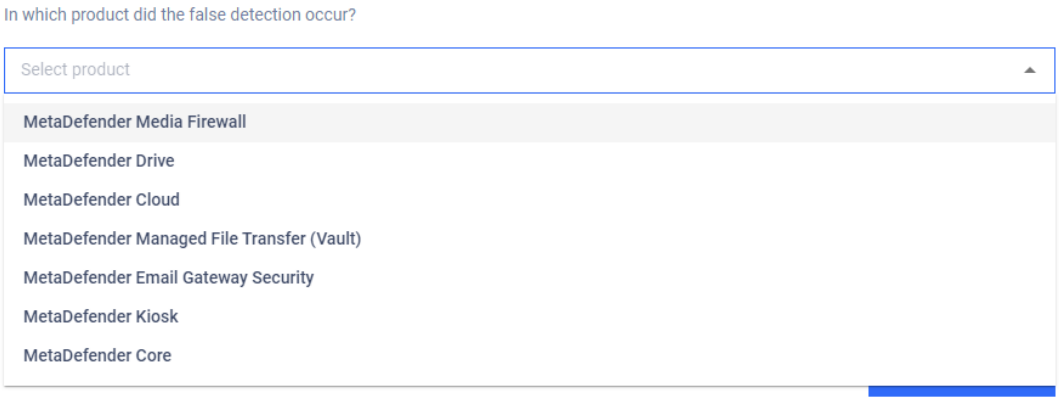
OSPWAT's malware analyst team commits to conducting a prompt triage within seven 7 business days for enterprise users and with dedicated efforts for community users. They will then recommend appropriate actions and ensure updates to the OPSWAT Reputation Engine if necessary. Additionally, if a potential false positive is identified, the analyst team will promptly inform the relevant antivirus vendor.

Follow these steps to submit the report of the file that you think has been misidentified:

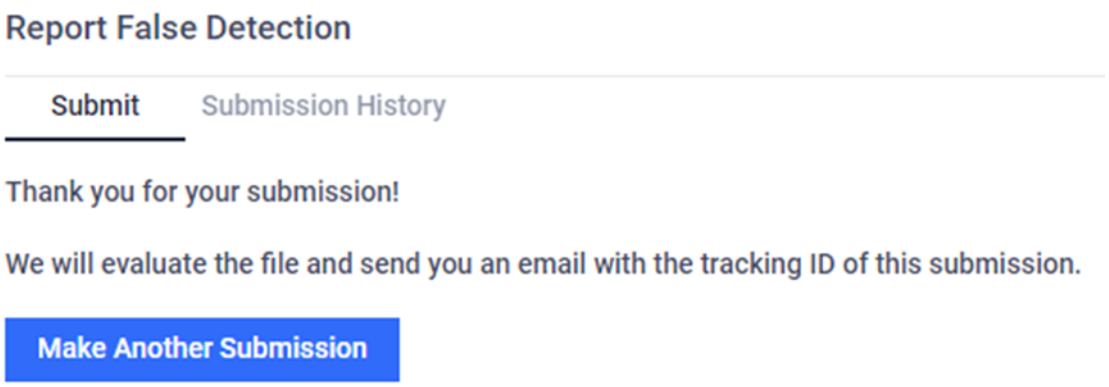
- Log into My OPSWAT.
- In the left-hand side menu, select "Support," and then choose "Report False Detection" from the submenu.
- Provide the file you would like OSPWAT's malware analyst team to investigate. Note that the maximum size for file uploads is restricted to 2GB.
- In the Submission Type section, specify if you suspect the file is a false positive (i.e., a clean file incorrectly flagged) or a false negative (i.e., a malicious file not detected). Note: You should not be overly concerned about categorizing your submission as a false positive or false negative. OPSWAT utilizes a thorough validation process for analyzing submitted files.



- Choose the OPSWAT product you were using that provided the suspected incorrect results. You can select multiple products.



- We recommend that you provide a detailed description explaining why you think the detection is wrong, as this information provides context that better helps us analyze the file.
- Finally, click on the 'Submit for Analysis' button. A notification will pop up to confirm your submission. You will also have the option to make another submission.



You can view your complete submission history under the 'Submission History' tab. This includes all relevant details such as File Name, Hash types (MD5/SHA1/SHA256), submission category, submission time, file size, and status of your submission.

User submissions from enterprise accounts are typically analyzed and processed within 7 business days. Community user submissions are handled with best efforts, aiming for timely processing. Upon completion of the analysis, you will receive notification regarding the validity of your submission.

What is Included in the MetaDefender Core Support Package?

When troubleshooting issues with MetaDefender Core, OPSWAT support may request a support package. Here's a support article, "How to Create Support Package With Web Console," that explains how to generate a support package. You can choose either an Advanced or Lightweight support package depending on the level of information you would like to collect for analysis.

1. Config Folder

The `config` folder contains configuration details of the MetaDefender Core server. These configurations define various settings and operational parameters. More details on the MetaDefender Core configuration can be found

here.

2. Crashdump Folder

The `crashdump` folder includes memory dumps generated when MetaDefender Core crashes. These dumps provide insights into system failures and help in debugging crash-related issues.

3. Database Folder

The `database` folder consists of:

- Logs from the PostgreSQL server used by MetaDefender Core.
- A PostgreSQL database dump, which captures the state of the database for further analysis.

4. Enginelogs Folder

The `enginelogs` folder contains logs for all the engines integrated into MetaDefender Core. These logs are useful for identifying engine-specific issues or performance-related concerns.

5. Log Folder

The `log` folder contains:

- The primary log files of MetaDefender Core.
- Logs related to the Nginx engine, which may be useful for identifying network or proxy-related issues.
- Starting from MetaDefender Core **v5.17.0**, you can also export **Windows Event Logs** from the Log section. These logs are useful when troubleshooting application crashes.

6. Process Folder

The `process` folder includes details about the active MetaDefender Core processes. This information helps in diagnosing issues related to system performance and resource utilization.

7. files.info

The `files.info` file lists the contents of the MetaDefender Core installation directory. This file helps in verifying the integrity of installed files and identifying missing or modified components.

8. hw.info

The `hw.info` file provides hardware details of the system running MetaDefender Core. This information is useful for ensuring that the server meets the recommended hardware specifications.

9. network.info

The `network.info` file contains network-related details of the MetaDefender Core server. This information is

essential for diagnosing connectivity issues.

10. os.info

The `os.info` file includes details about the operating system on which MetaDefender Core is running. This data helps in identifying OS-related compatibility or configuration issues.

Conclusion

The MetaDefender Core support package is a crucial resource for diagnosing and troubleshooting issues. When providing this package to OPSWAT support, ensure that all relevant folders and files are included to facilitate efficient issue resolution.

> **Support:**
> If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What dependencies does MetaDefender's Deep CDR for Linux have?

> **Check Your Version:**
> This article applies to all MetaDefender Core releases deployed on Linux systems.

From Deep CDR engine version 8.0.2, MetaDefender's **Deep CDR** for Linux will no longer require libgdiplus to be preinstalled.

Starting with version 6.0 (released in December 2021), MetaDefender's **Deep CDR** for Linux will require that **libgdiplus** be pre-installed**.**

Please follow the relevant instructions below to install libgdiplus on your Linux system:

Rocky Linux 9.x

- Download the prebuilt libgdiplus6.0.4.tar.gz .
  - File size: 1,857,019 bytes
  - MD5: 759a6a3641a1bc83363336f860cd7f25
  - SHA256: 222ec848c62bdc886ebed980f87d4e56ff3341c5e9af1c242a2e9706b674a4b5
- Run these commands to install, be aware of the "Copy and extract libgdiplus6.0.4.tar.gz ..." step.

Code · bash

```
### Install runtime dependency
```

```
dnf install -y 'dnf-command(config-manager)'
dnf config-manager --set-enabled crb
dnf makecache
dnf install -y tar giflib-devel libjpeg-devel giflib libjpeg libpng cairo libtiff libexif glib2
### Copy and extract libgdiplus6.0.4.tar.gz file to your installed folder (this folder can be
read by metascan) (for example :/var/lib/libgdiplus6)
mkdir /var/lib/libgdiplus6
tar -xvf libgdiplus6.0.4.tar.gz -C /var/lib/libgdiplus6
echo "/var/lib/libgdiplus6/6.0.4/lib" | sudo tee /etc/ld.so.conf.d/libgdiplus.conf
ldconfig
### Double check libgdiplus was installed successfully
ldconfig -p | grep libgdiplus
### Restart Core to make change applied
systemctl stop ometascan
systemctl start ometascan
```

Rocky Linux 10.x

- Download the prebuilt libgdiplus6.0.4_rl10.tar.gz.
  - File size: 2,541,804 bytes
  - MD5: 4228c639a307fbdb8f735b29ada1ba42
  - SHA256: 8c340a8e95daaaa968766768e462f3b5e87a32c700c17a357b6a3fd4dce310b9
- Run these commands to install, be aware of the “Copy and extract libgdiplus6.0.4_rl10.tar.gz ...” step.

Terminal · bash

```
### Copy and extract libgdiplus6.0.4_rl10.tar.gz file to your installed folder (this folder can
be read by metascan) (for example :/var/lib/libgdiplus6)
mkdir /var/lib/libgdiplus6
tar -xvf libgdiplus6.0.4_rl10.tar.gz -C /var/lib/libgdiplus6
echo "/var/lib/libgdiplus6/6.0.4/lib" | sudo tee /etc/ld.so.conf.d/libgdiplus.conf
ldconfig
### Double check libgdiplus was installed successfully
ldconfig -p | grep libgdiplus
### Restart Core to make change applied
systemctl stop ometascan
systemctl start ometascan
```

Ubuntu 22.04

Code · bash

```
sudo apt install gnupg ca-certificates
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys
3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF
```

```
echo "deb https://download.mono-project.com/repo/ubuntu stable-focal main" | sudo tee /etc/apt/
sources.list.d/mono-official-stable.list
sudo apt update
sudo apt install libgdiplus
```

Ubuntu 24.04

Code · bash

```
sudo apt install wget
sudo wget http://ftp.de.debian.org/debian/pool/main/libw/libwebp/
libwebp6_0.6.1-2.1+deb11u2_amd64.deb
sudo wget http://ftp.de.debian.org/debian/pool/main/libj/libjpeg-turbo/libjpeg62-
turbo_2.1.5-4_amd64.deb
sudo wget http://ftp.de.debian.org/debian/pool/main/t/tiff/libtiff5_4.2.0-1+deb11u5_amd64.deb
sudo wget http://ftp.de.debian.org/debian/pool/main/libg/libgdiplus/
libgdiplus_6.0.4+dfsg-2_amd64.deb
sudo dpkg -i libwebp6_0.6.1-2.1+deb11u2_amd64.deb || sudo apt-get install -f
sudo dpkg -i libjpeg62-turbo_2.1.5-4_amd64.deb || sudo apt-get install -f
sudo dpkg -i libtiff5_4.2.0-1+deb11u5_amd64.deb || sudo apt-get install -f
sudo dpkg -i libgdiplus_6.0.4+dfsg-2_amd64.deb  || sudo apt-get install -f
```

Red Hat 8

Code · bash

```
rpmkeys --import "http://keyserver.ubuntu.com/pks/lookup?
op=get&search=0x3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF"
su -c 'curl https://download.mono-project.com/repo/centos8-stable.repo | tee /etc/yum.repos.d/
mono-centos8-stable.repo'
yum install libgdiplus
```

Red Hat 9

Code · bash

```
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
sudo dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo dnf -y install libgdiplus
```

Debian 12

Code · bash

```
sudo apt install wget
sudo wget http://ftp.de.debian.org/debian/pool/main/libw/libwebp/
```

```
libwebp6_0.6.1-2.1+deb11u2_amd64.deb
sudo wget http://ftp.de.debian.org/debian/pool/main/t/tiff/libtiff5_4.2.0-1+deb11u5_amd64.deb
sudo wget http://ftp.de.debian.org/debian/pool/main/libg/libgdiplus/
libgdiplus_6.0.4+dfsg-2_amd64.deb
sudo dpkg -i libwebp6_0.6.1-2.1+deb11u2_amd64.deb || sudo apt-get install -f
sudo dpkg -i libtiff5_4.2.0-1+deb11u5_amd64.deb || sudo apt-get install -f
sudo dpkg -i libgdiplus_6.0.4+dfsg-2_amd64.deb  || sudo apt-get install -f
```

Amazon Linux 2023

Code · bash

```
sudo curl https://download.mono-project.com/repo/centos8-stable.repo | sudo tee /etc/
yum.repos.d/mono-centos8-stable.repo
sudo dnf install libgdiplus
```

Support:

If you have followed the steps above but are still having trouble installing **Libgdiplus** or running MetaDefender Core's **Deep CDR** on a Linux system, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What links, target-services or target host-IPs need to be allowed for Metadefender Core to function accurately?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems.

If you have installed, or if you wish to install or use, the MetaDefender Core system in a restricted environment, you will have to allow access to the following hosts' to ensure MetaDefender Core's accurate functioning:

- https://activation.dl.opswat.com - this is for product activation/licensing.
- https://update.dl.opswat.com - this is for fetching engine/database updates.

Attention:

Since August 2023, we've migrated to AWS Cloudfront as our CDN for Update Servers. Depending on your location, you'll be directed to specific Edge Locations, each with its own unique IP ranges. Ensure you've allowed the correct Cloudfront IP ranges by referring to: https://ip-ranges.amazonaws.com/ip-ranges.json

Pro Tip:

Even though the OPSWAT update servers host updates for all of the available engines we support, the custom engines might occasionally try to connect to their own cloud for updates.

This can be disabled under **Firewall**, after which the engines will only be updated via OPSWAT.

Configure MetaDefender Core's proxy configuration settings if applicable for update server and activation server URLs.

Please Note:

IP address based allowlisting might fail after some time, since OPSWAT employs a **CDN (Content Delivery Network)** to achieve faster updates globally, and the IP addresses of edge servers might change over time.

Info:

If the internal policy requires certificate validation, the firewall must allow access to the CRL and OCSP URLs of the OPSWAT Activation and Engine Update servers' certificates, as well as their Intermediate CA (ICA) certificate. To extract the URLs that need whitelisting, please follow How to retrieve the CRL and OCSP URLs for the Server and ICA Certificates?

Support:

If you have any difficulty **Installing** and **Setting Up** MetaDefender Core in a restricted environment, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I set up exclusions on my anti-malware software to prevent disruption of the MetaDefender Core scanning process, when my corporate policy does not allow it?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Anti-malware engines included in MetaDefender Core do not install real-time protection agents.

If you already have an anti-malware product installed on your system which also happens to be one of the anti-malware engines in your version of MetaDefender Core, it will interfere with the MetaDefender Core scanning process.

Pro Tip:

For this reason, OPSWAT recommends that you disable the real-time protection of your anti-malware product. If your corporate policy does not allow you to disable your real-time anti-virus product, you will need to add some exception rules.

To implement your exception rule, you must exclude the following from the relevant anti-malware product's real-time protection:

- the **OPSWAT Installation** folder which by default also includes the folder where MetaDefender Core is creating its temporary files
- the **ometascan**, **engineprocess**, **engineprocess32** and **nginx** processes (note that some engines will need to run on different process instances (e.g. **ClamAv**) that are managed by the **engineprocess** parent process).

If you do not add this exception or if you do not disable real-time protection, results returned by MetaDefender Core scanning will not be consistent, and the return value of the scans would be one of the following:

- **Clean**: If your existing anti-malware product was able to clean the threat
- **Failed** (or other errors): If your existing anti-malware product removed the file before MetaDefender Core could scan it.

Please Note:

If you are using Symantec Endpoint Protection as your local AV, please adjust the settings as instructed in This KB Article.

Pro Tip:

Should you require a real time protection agent to be installed in addition to MetaDefender Core, please ensure that it is not a product featured among the MetaDefender Core's included scanning engines.

Support:

If you need help learning how to add an exception rule (to exclude a given folder from an AV products scan area) please tell us what product you are using and we may be able to help you - but be sure to include the product version. To contact us, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I deploy MetaDefender Core to an offline Windows environment?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed in offline Windows environments.

Pro Tip:

MetaDefender Core supports deployment in either online or offline environments.

There are four steps to getting the product up and running in an offline environment:

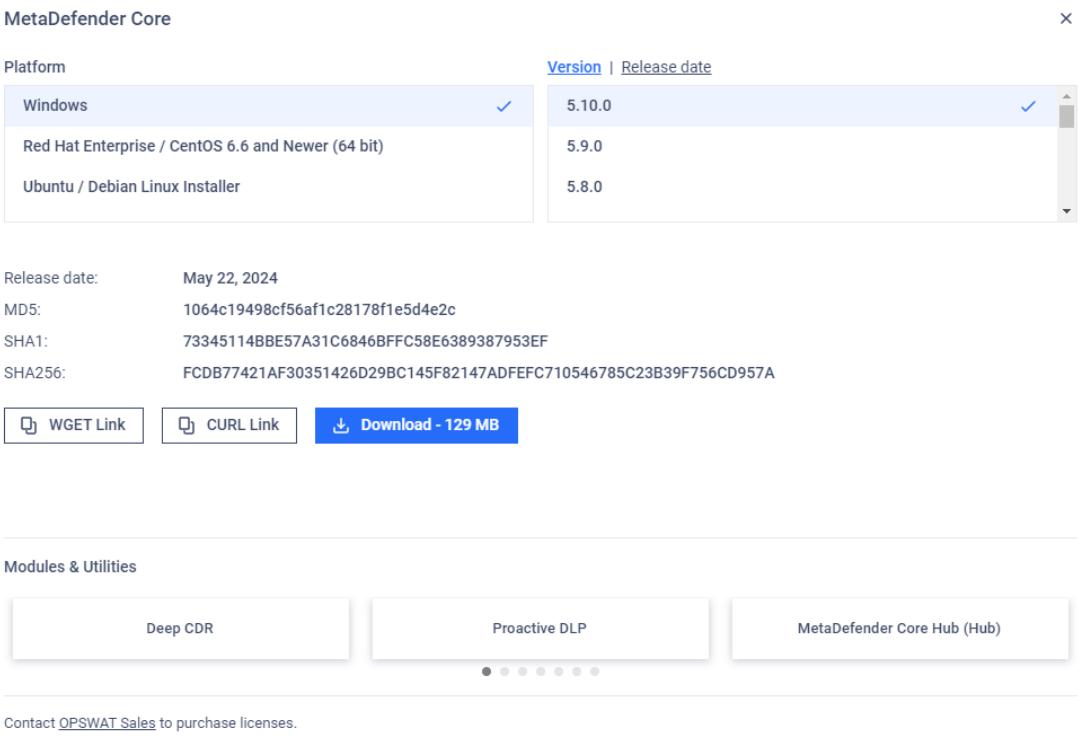
1. **Install MetaDefender Core on the offline server.**
2. **Activate your license.**
3. **Install and configure the MetaDefender Update Downloader utility.**
4. **Apply the offline updates that are downloaded by the update utility to the offline MetaDefender Core server.**

A. Installing MetaDefender Core

Please Note:

The MetaDefender Core installation packages can be downloaded from My OPSWAT.

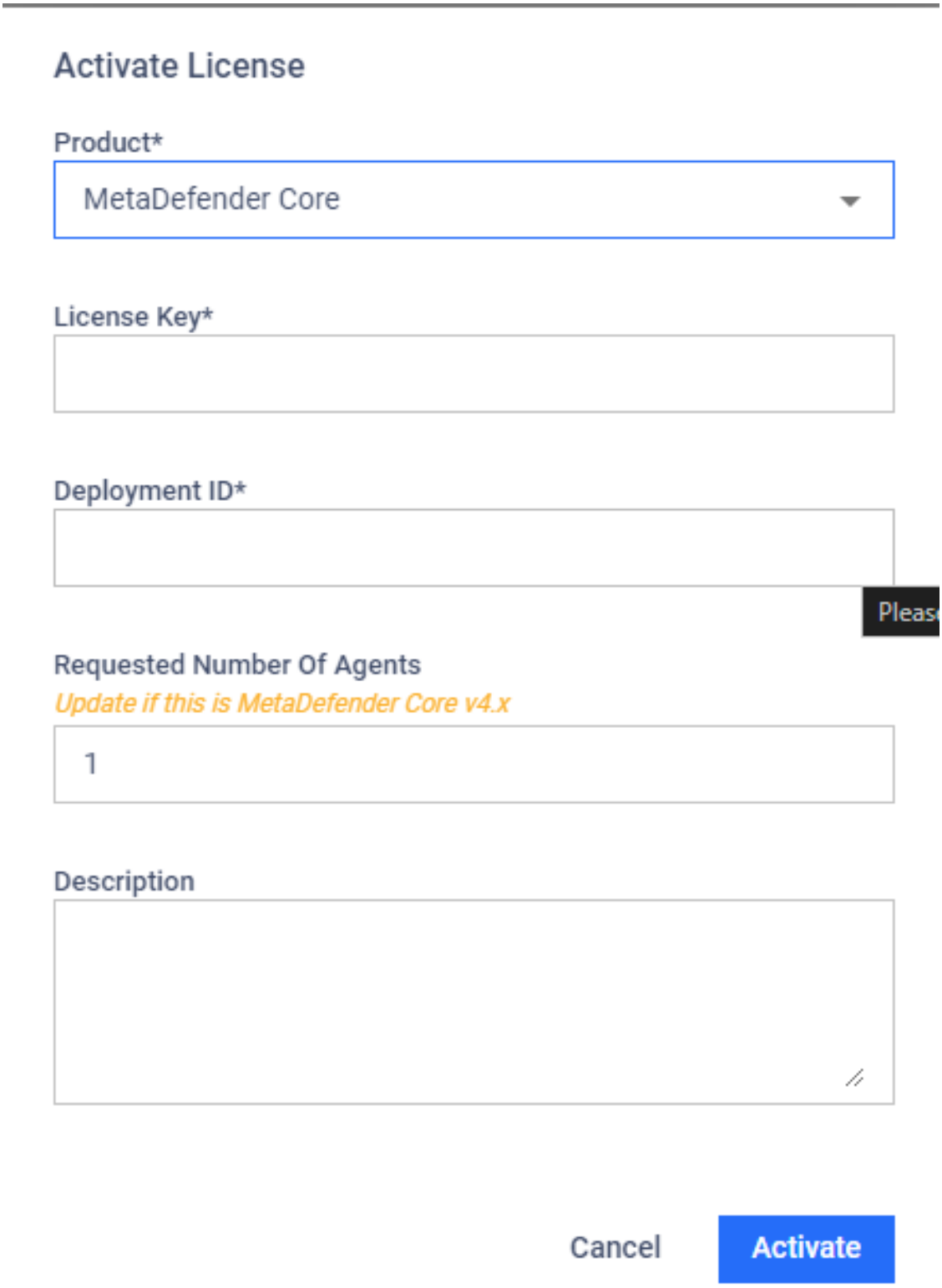
1. After logging into the portal, go to the Downloads page to select the MetaDefender Core package you wish to download.
2. The supported platforms include Windows 7 and later releases.
3. Download the appropriate installer and run it on the Windows system:



B. Activating your license

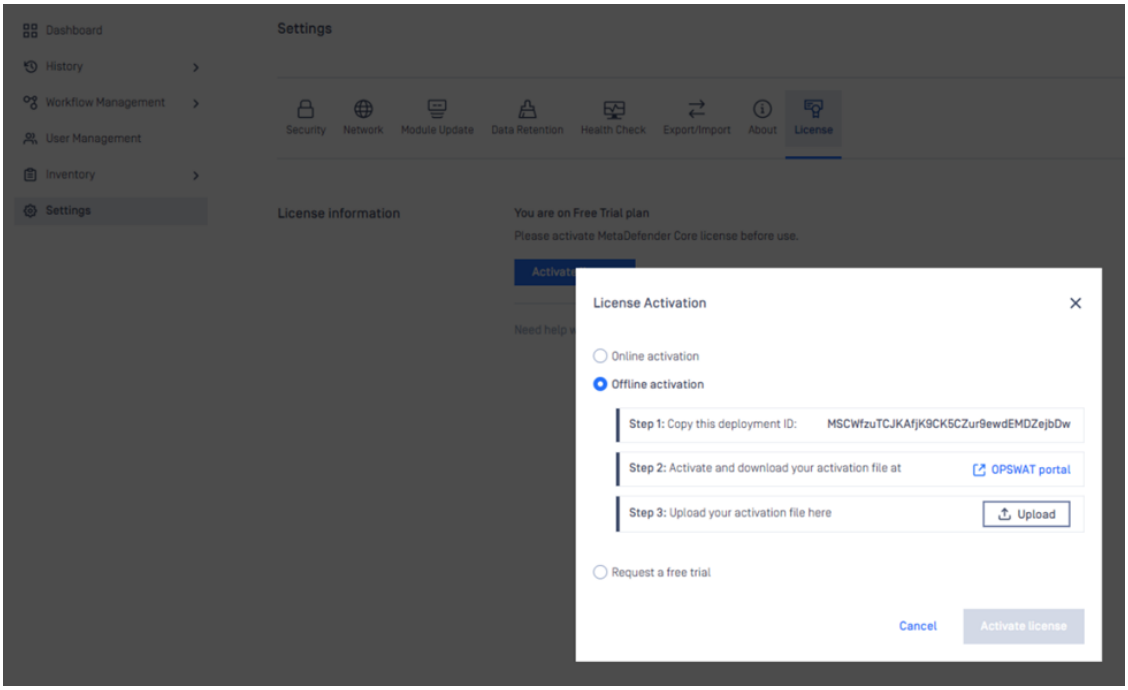
After MetaDefender Core is installed, you will need to activate your MetaDefender Core installation:

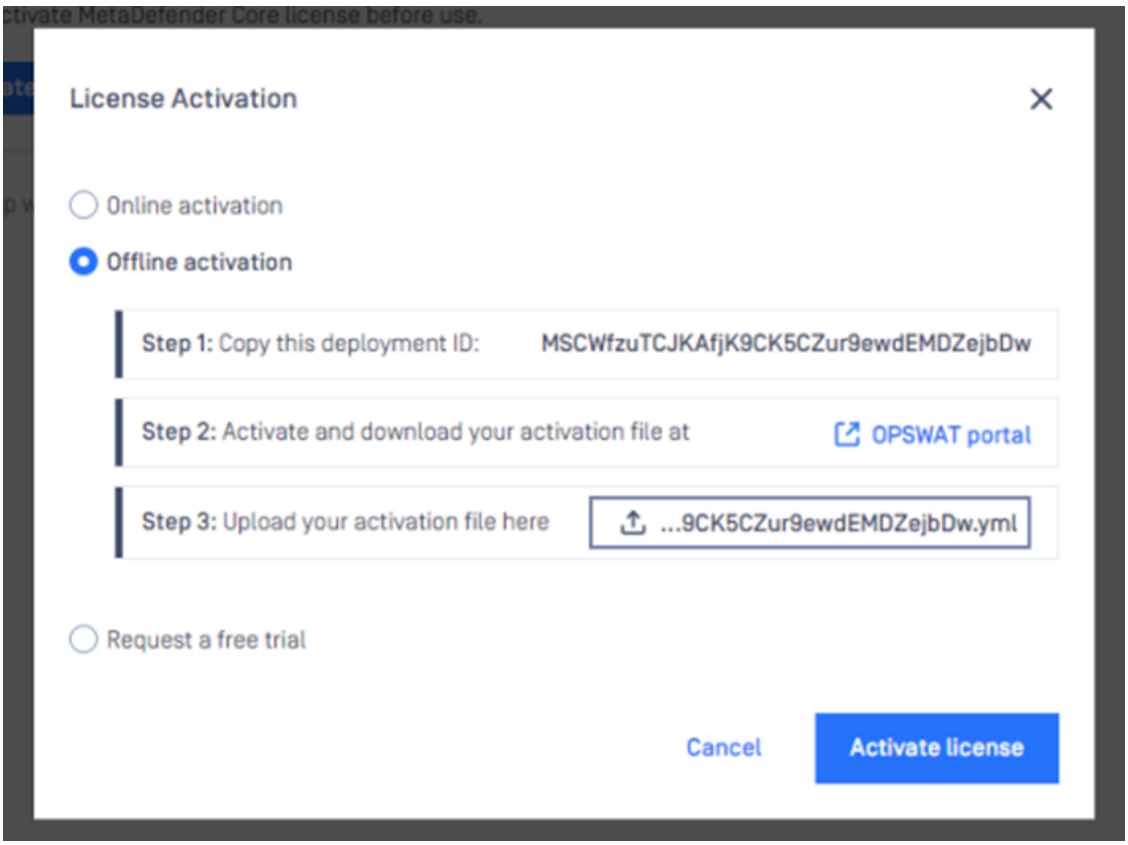
1. Log in to your MetaDefender Core **Management Console** at http://localhost:8008/. Complete the required steps of the Basic Configuration Wizard.
2. Go to **Settings > License > Activate > Offline**.
3. Here, you can follow the steps to:
    1. Copy your **Deployment ID** to the clipboard, then open a new browser tab.
    2. Log in to the My OPSWAT and navigate to the **License Management** page and click on **Activate License** button on the top right.
    3. Fill in the required details to retrieve your **Activation File**.



d. Return to the Management Console to upload your **Activation File.**

e. Finally, **Activate** your license, as shown in the screenshots below.



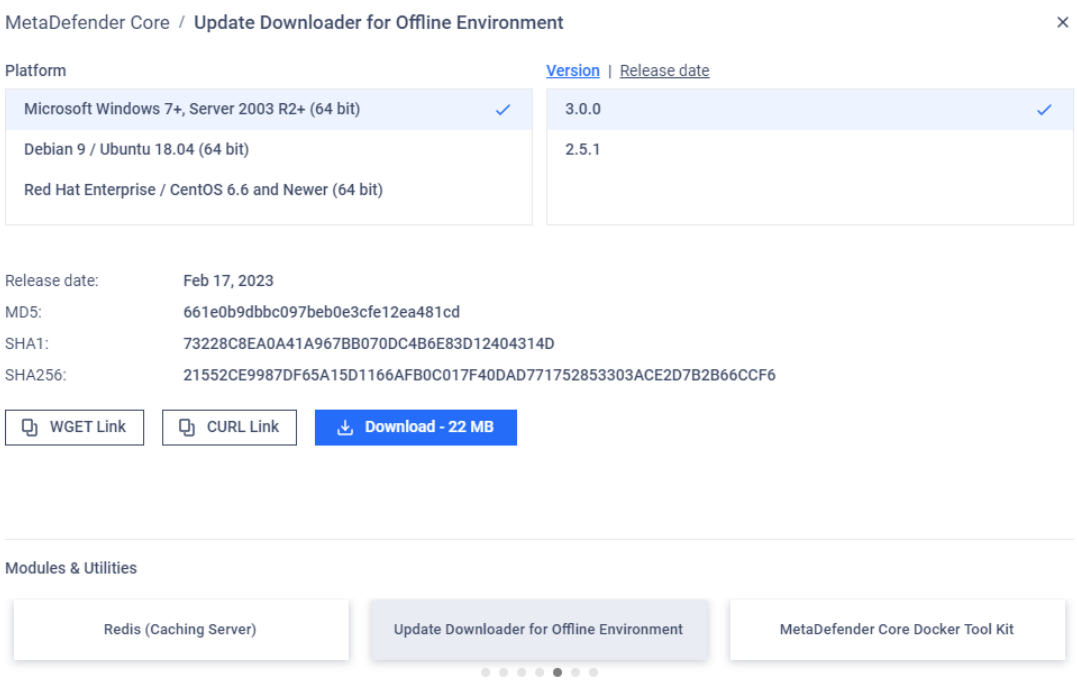


C. Installing the MetaDefender Update Downloader utility

Please Note:

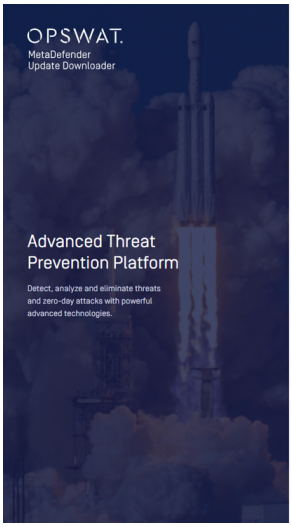
If your MetaDefender Core server is deployed offline, you will need to install the **Update Downloader** utility to download the anti-malware definition updates to be applied to the server.

You can download the **Downloader** utility from My OPSWAT under **Modules & Utilities**.



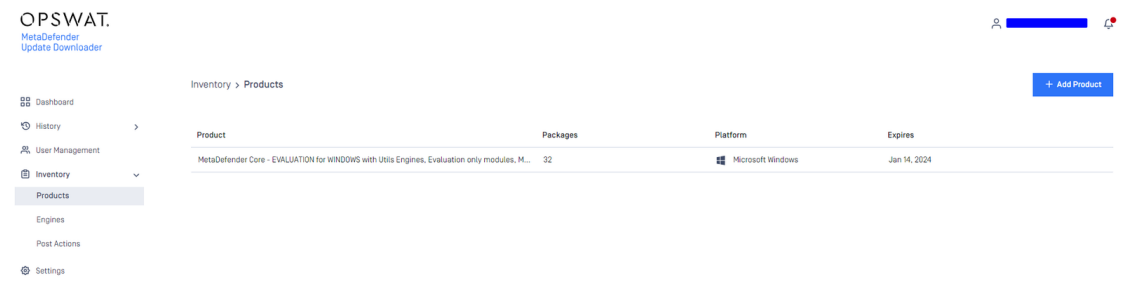
Once you have installed the Update Downloader, apply your license key to activate the product:

1. Log into the **Update Downloader Management Console at http://<server>:8028/**.

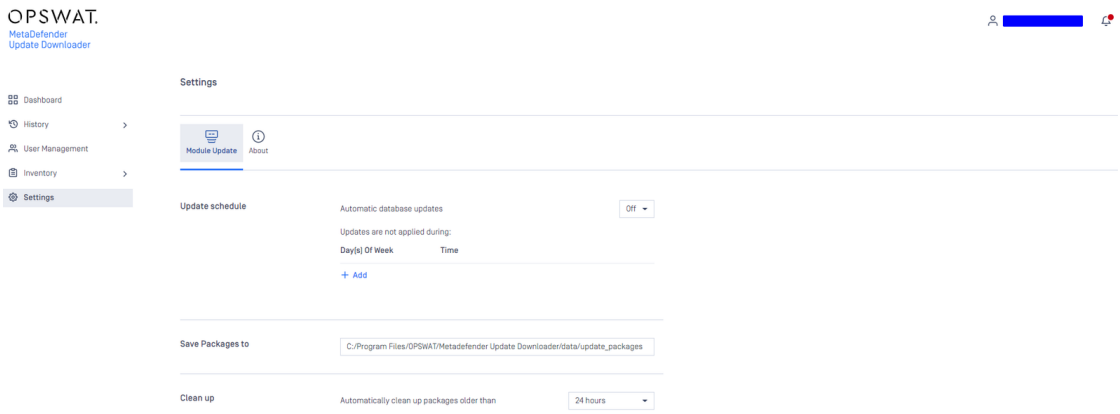




2. Add your product on the **Inventory** page, **Product** tab.



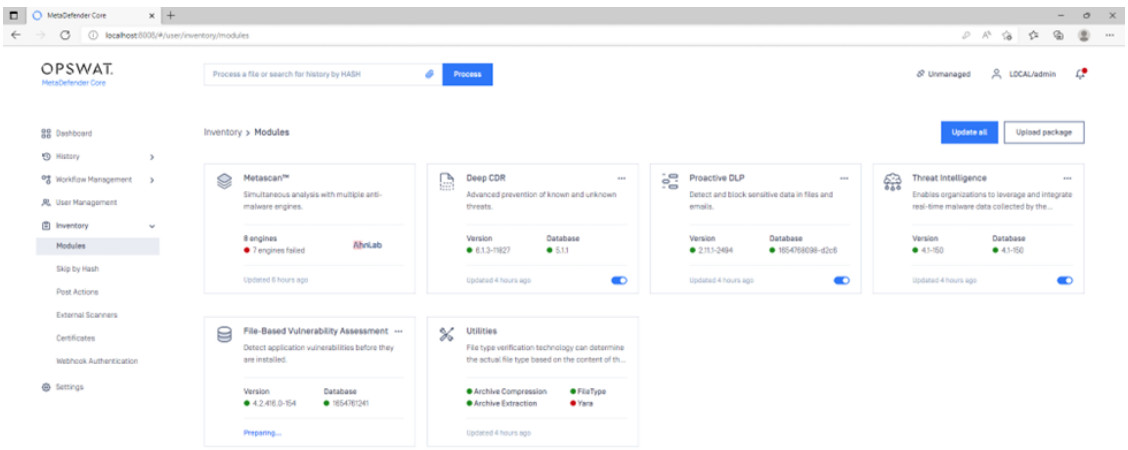
3. Update the configuration for update download and package generation on the **Settings** page of the **Management Console**.



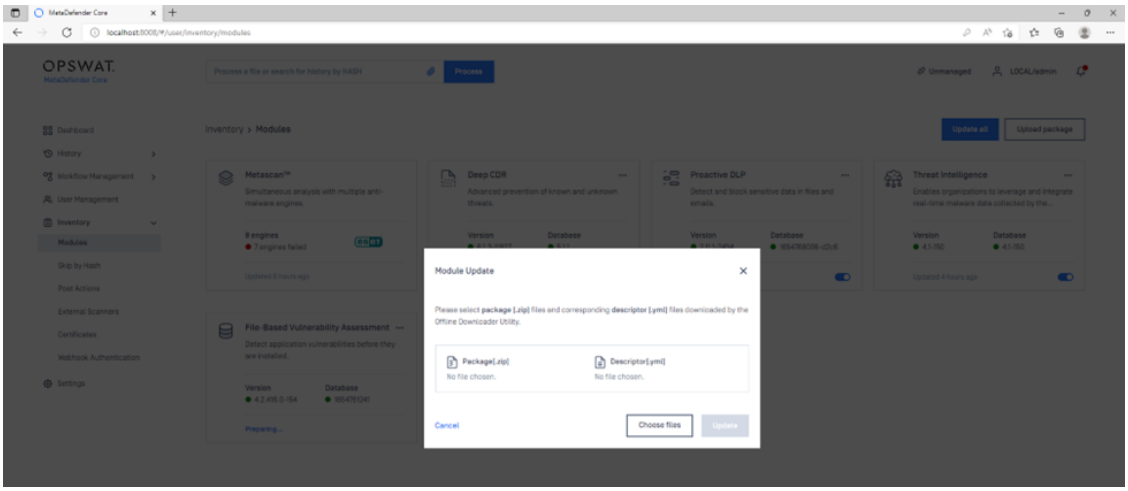
D. Applying offline updates

Once the update packages have been downloaded, you can apply them to the offline MetaDefender Core server through the MetaDefender Core Management Console:

1. Copy all of the update packages from the directory where the **Update Downloader** saves the updates, to the offline MetaDefender Core system.
2. Upload the updated packages through the MetaDefender Core **Management Console**.



3. To save the update packages, select the engine update packages from the directory where the **Update Downloader** is configured.



Support:

If you have any difficulty setting up MetaDefender Core in an offline Windows environment, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I deploy MetaDefender Core to an offline Linux environment?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed in offline Linux environments.

Pro Tip:

MetaDefender Core supports deployment in either online or offline environments.

There are four steps to getting the product up and running in an offline environment:

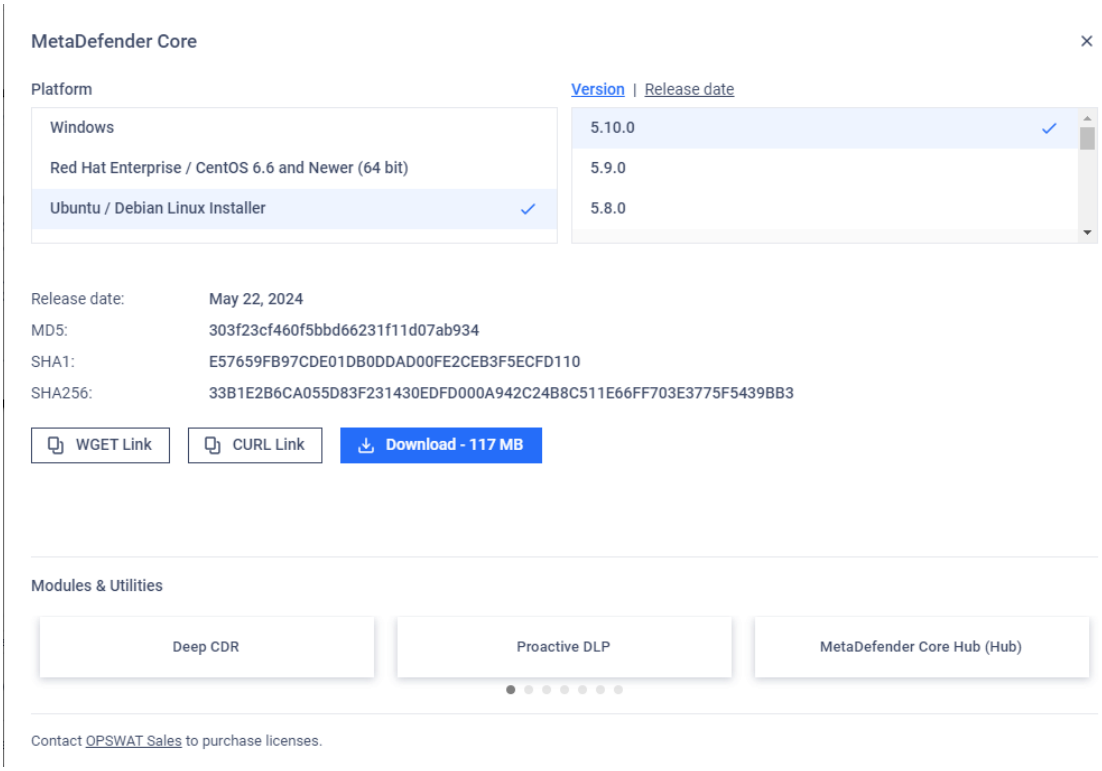
1. **Install MetaDefender Core on the offline server.**
2. **Activate your license.**
3. **Install and configure the MetaDefender Update Downloader utility.**
4. **Apply the offline updates that are downloaded by the update utility to the offline MetaDefender Core server.**

A. Installing MetaDefender Core

Please Note:

The MetaDefender Core installation packages can be downloaded from My OPSWAT.

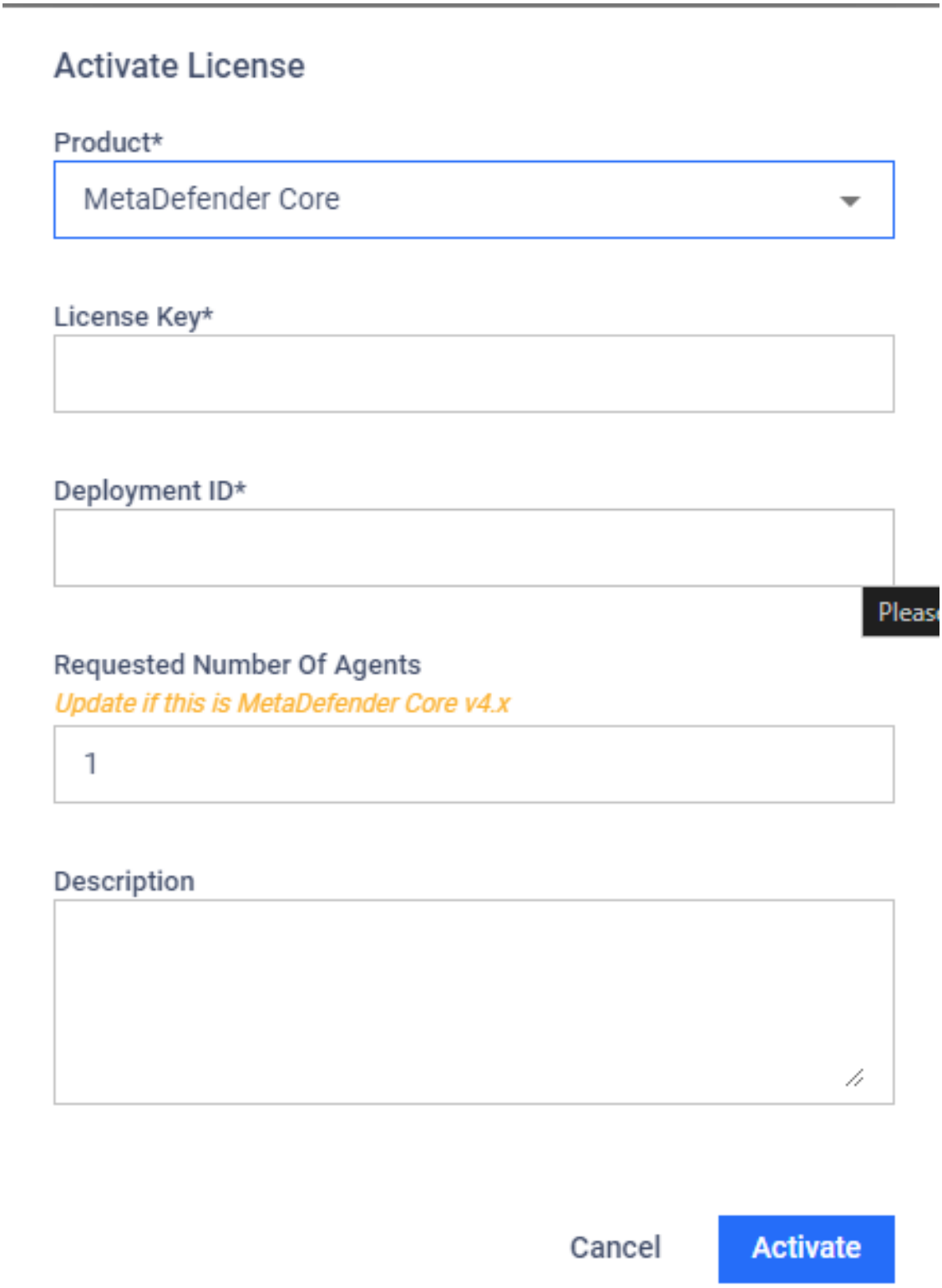
1. After logging into the portal, go to the Downloads page to select the MetaDefender Core package you wish to download.
2. The supported platforms include Red Hat Enterprise, CentOS, Debian, and Ubuntu.
3. Download the appropriate installer for your distribution:



B. Activating your license

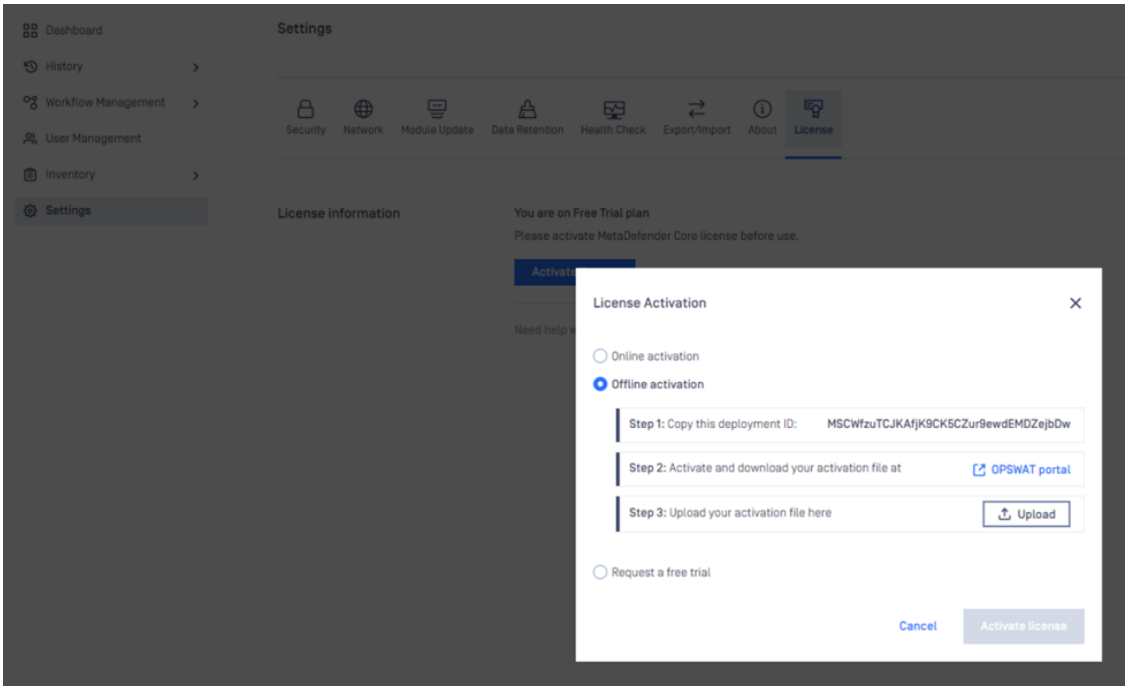
After MetaDefender Core is installed, you will need to activate your MetaDefender Core installation:

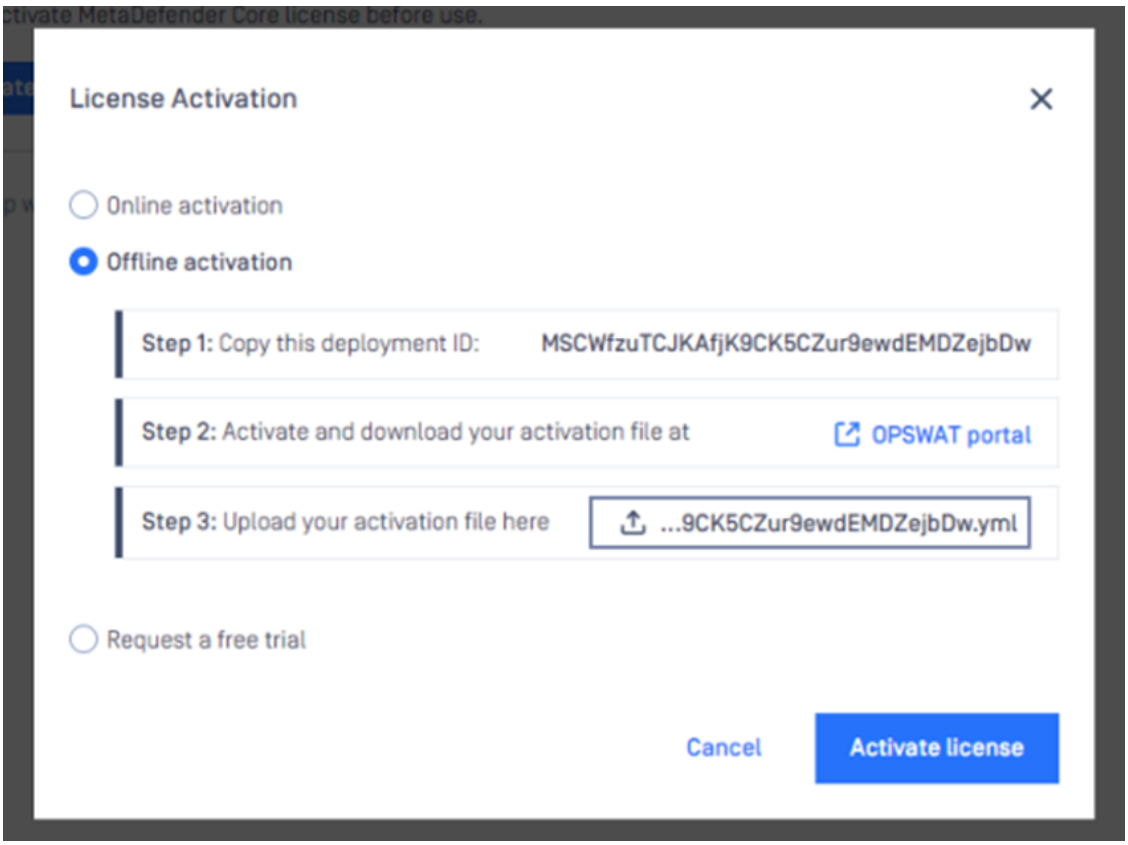
1. Log in to your MetaDefender Core **Management Console** at http://localhost:8008/. Complete the required steps of the Basic Configuration Wizard.
2. Go to **Settings > License > Activate > Offline**.
3. Here, you can follow the steps to:
    1. Copy your **Deployment ID** to the clipboard, then open a new browser tab.
    2. Log in to the My OPSWAT and navigate to the **License Management** page and click on **Activate License** button on the top right.
    3. Fill in the required details to retrieve your **Activation File**.



d. Return to the Management Console to upload your **Activation File.**

e. Finally, **Activate** your license, as shown in the screenshots below.



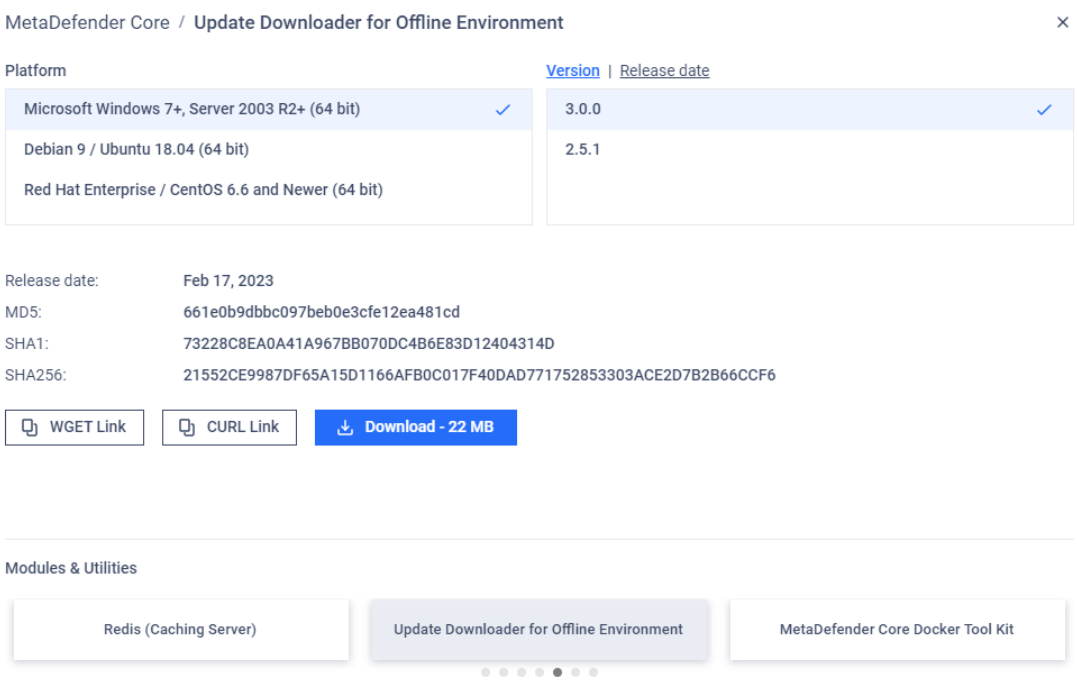


C. Installing the MetaDefender Update Downloader utility

Please Note:

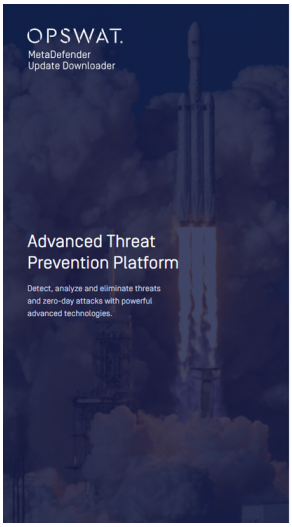
If your MetaDefender Core server is deployed offline, you will need to install the **Update Downloader** utility to download the anti-malware definition updates to be applied to the server.

You can download the **Downloader** utility from My OPSWAT under **Modules & Utilities**.



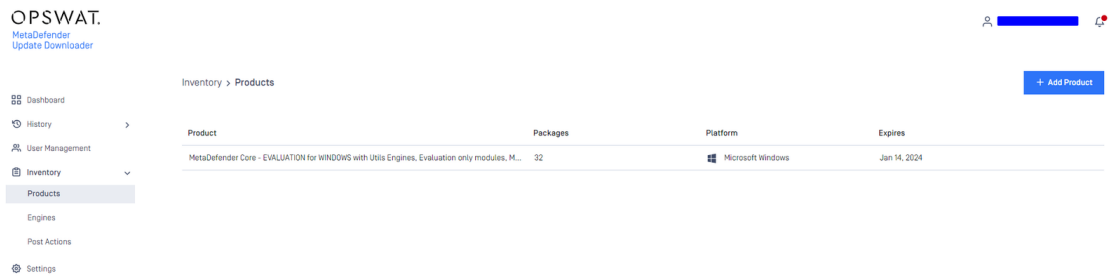
Once you have installed the Update Downloader, apply your license key to activate the product:

1. Log into the **Update Downloader Management Console at http://<server>:8028/**.

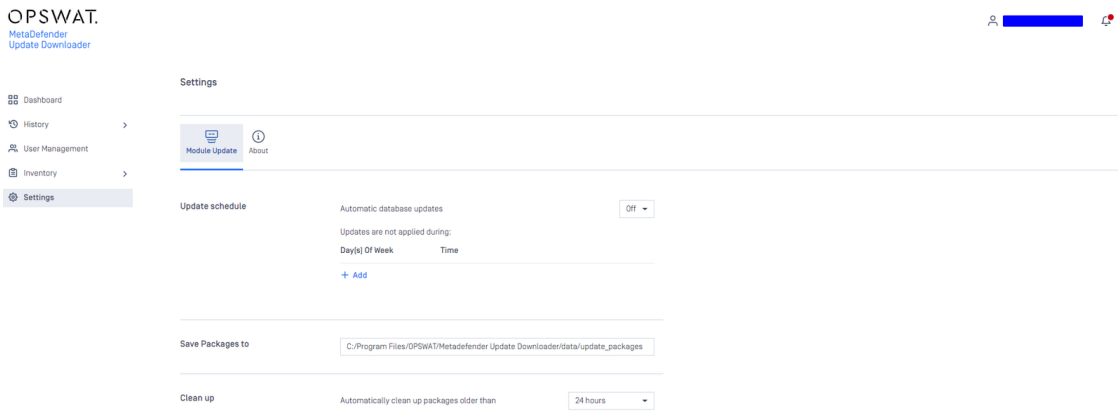




2. Add your product on the **Inventory** page, **Product** tab.



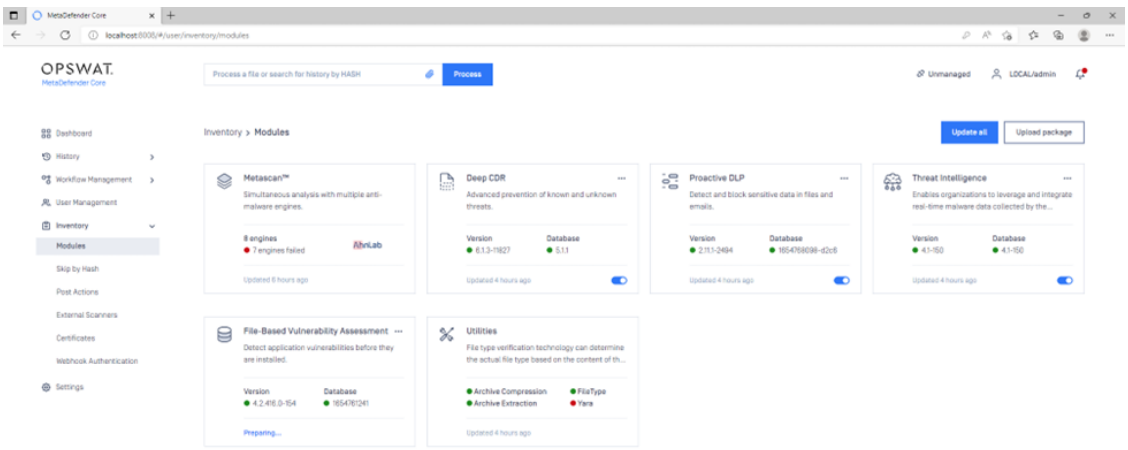
3. Update the configuration for update download and package generation on the **Settings** page of the **Management Console.**



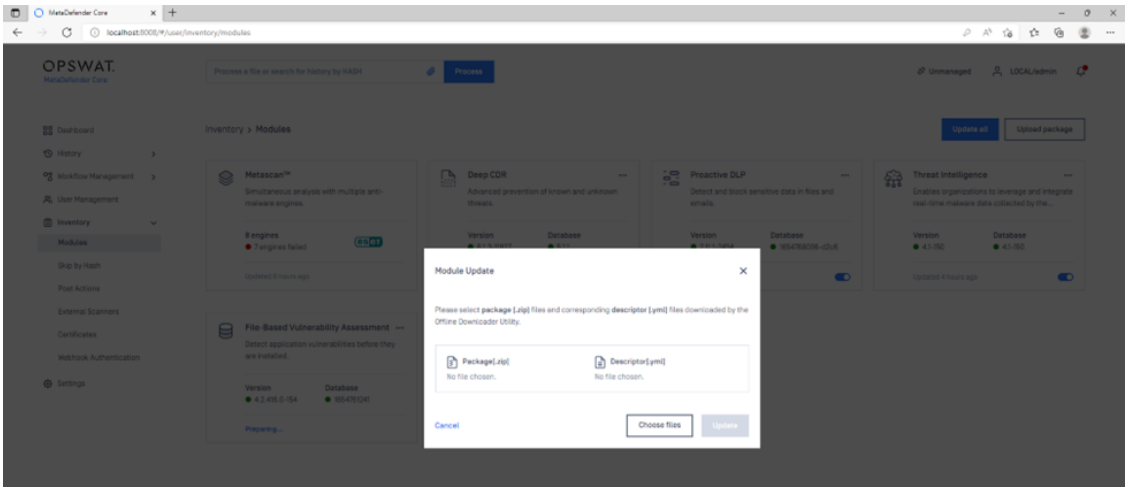
D. Applying offline updates

Once the update packages have been downloaded, you can apply them to the offline MetaDefender Core server through the MetaDefender Core Management Console:

1. Copy all of the update packages from the directory where the **Update Downloader** saves the updates, to the offline MetaDefender Core system.
2. Upload the updated packages through the MetaDefender Core **Management Console**.



3. To save the update packages, select the engine update packages from the directory where the **Update Downloader** is configured.



Support:

If you have any difficulty setting up MetaDefender Core in an offline Linux environment, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Does the Deployment ID change at any time?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

New in v5.17.0:

Starting from **MetaDefender Core v5.17.0**, the product now supports **teamed NIC (Network Interface Card) configurations** without causing Deployment ID changes. This enhancement ensures that license activation remains stable even when network adapters within the team are changed or switched. With this improvement:

- Deployment ID will remain consistent across NIC team reconfigurations.
- Service interruptions due to MAC address changes in teamed NIC setups are eliminated.

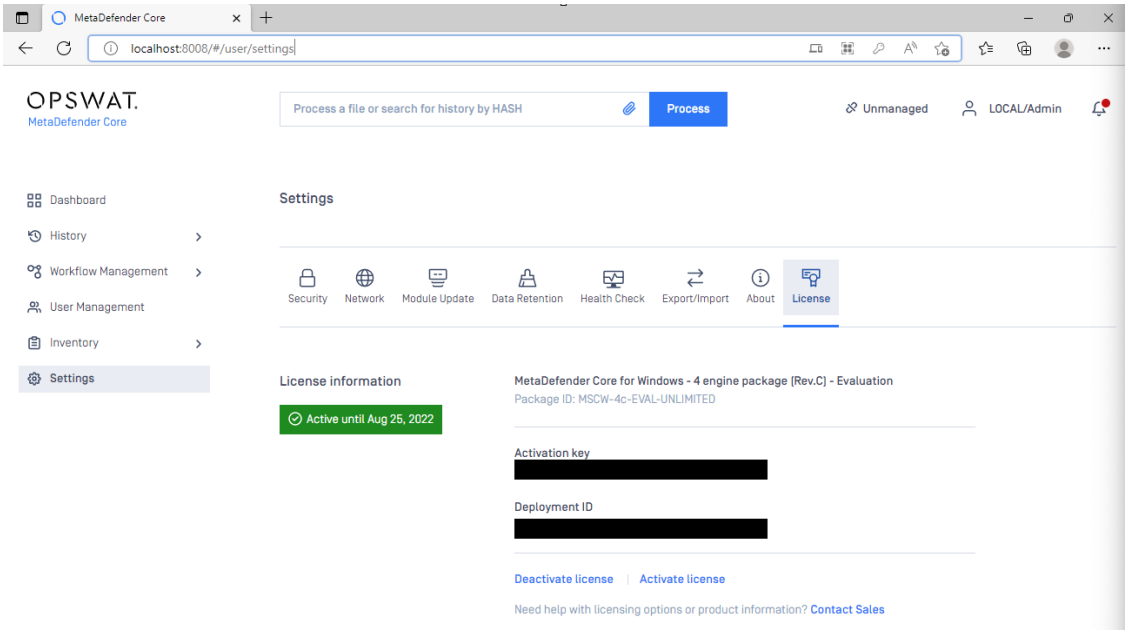
For more information, please refer to Release notes - MetaDefender Core

For users running versions prior to v5.17.0, please refer to the following instructions:

License holders should be aware that changes or updates to the hardware or network components of the system on which MetaDefender Core is installed could result in changes to the Deployment ID.

Pro Tip:

To check whether your Deployment ID has changed, go to your **MetaDefender Core Management Console>Settings>License** tab, where your package ID, activation key, deployment ID and license expiry date are all listed.



Error

In some cases, if MetaDefender Core is deployed on a server using **teamed NIC**, the deployment ID may change due to changing physical (MAC) address of the server's network adapter.

Cause: NIC teaming, also known as Load Balancing/Failover (LBFO) or NIC bonding, allows you to combine multiple physical network adapters into a single logical network card

When using teamed NIC, because the primary team member (primary NIC) might change in a non-deterministic manner (e.g. at each boot, NIC disable/enable action, or other reconfiguration activities), the primary team member might change, and the MAC address of the team might vary.

- Since MetaDefender Core's deployment ID is based partly on the MAC address of the network adapter, varying MAC address also leads to deployment ID changes and deactivation of license - causing service interruptions.

Example of a teamed NIC:

Code · bash

```
Ethernet adapter Team:
Connection-specific DNS Suffix  . :
Description . . . . . . . . . . . : Microsoft Network Adapter Multiplexor Driver
Physical Address. . . . . . . . . : XX-XX-XX-XX-XX-XX
```

- `Microsoft Network Adapter Multiplexor Driver` is a kernel mode driver to team up with multiple network

adapters to provide one unified device to utilize the entire network bandwidth and load balancing among underlying adapters.

Solution: To avoid MetaDefender Core service interruption due to changing deployment ID, please perform the necessary steps to ensure the teamed NIC's physical address remain the same after reboot.

- One way to achieve this is by configuring **a static MAC address** for the teamed NIC.

Please Note:

Operating system upgrades, license upgrades, license renewals and other factors unrelated to system hardware and network components should not affect your Deployment ID.

However, in the unlikely event that you notice an unexplained change to your Deployment ID, please contact support as instructed below.

Support:

If you require further assistance with your **Licensing or Deployment ID**, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

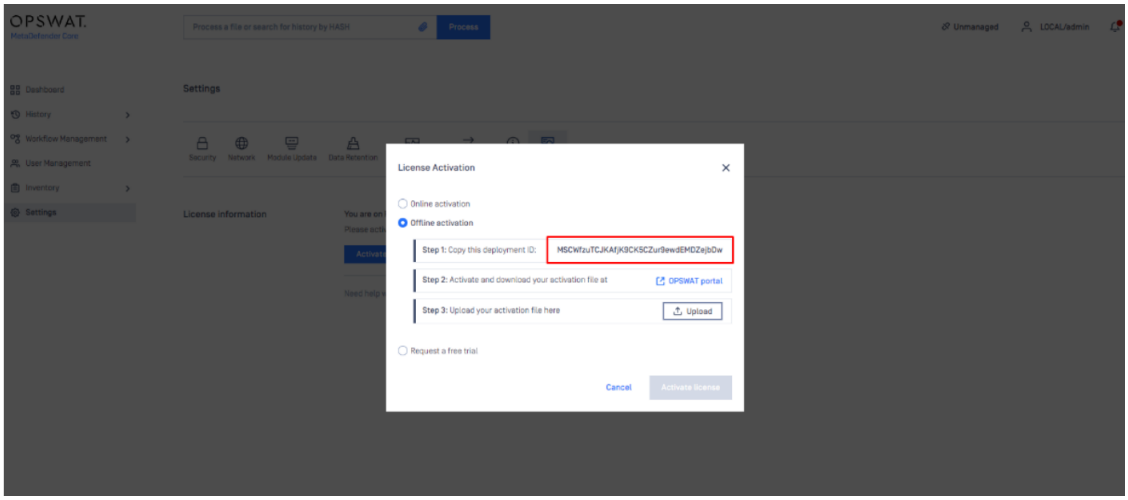
How do I activate my offline deployment of MetaDefender Core?

Check Your Version:

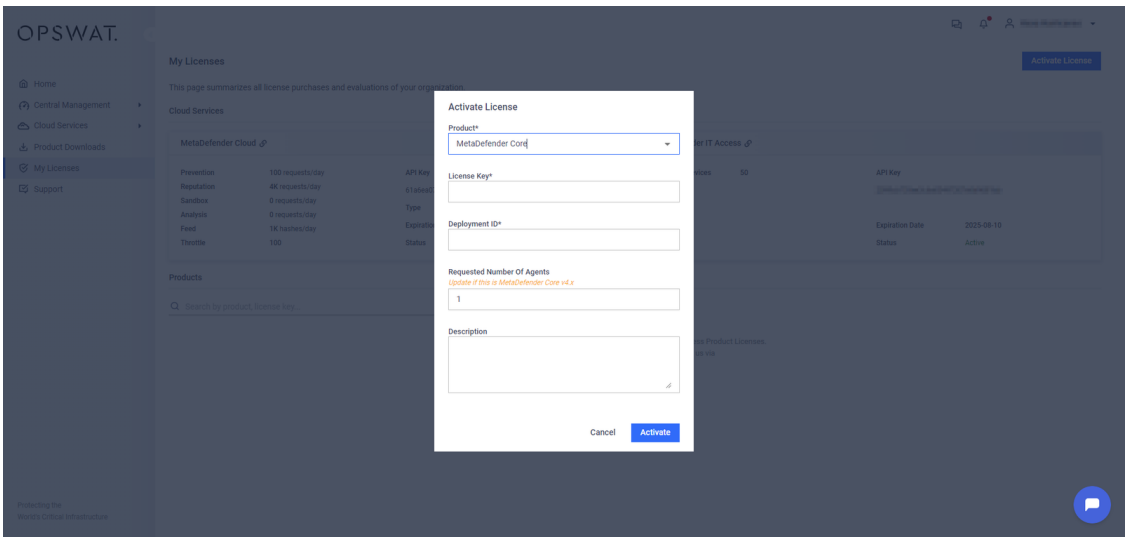
This article applies to all MetaDefender Core V5 releases deployed on offline Windows or Linux systems.

The steps for activating MetaDefender Core in an offline environment are as follows:

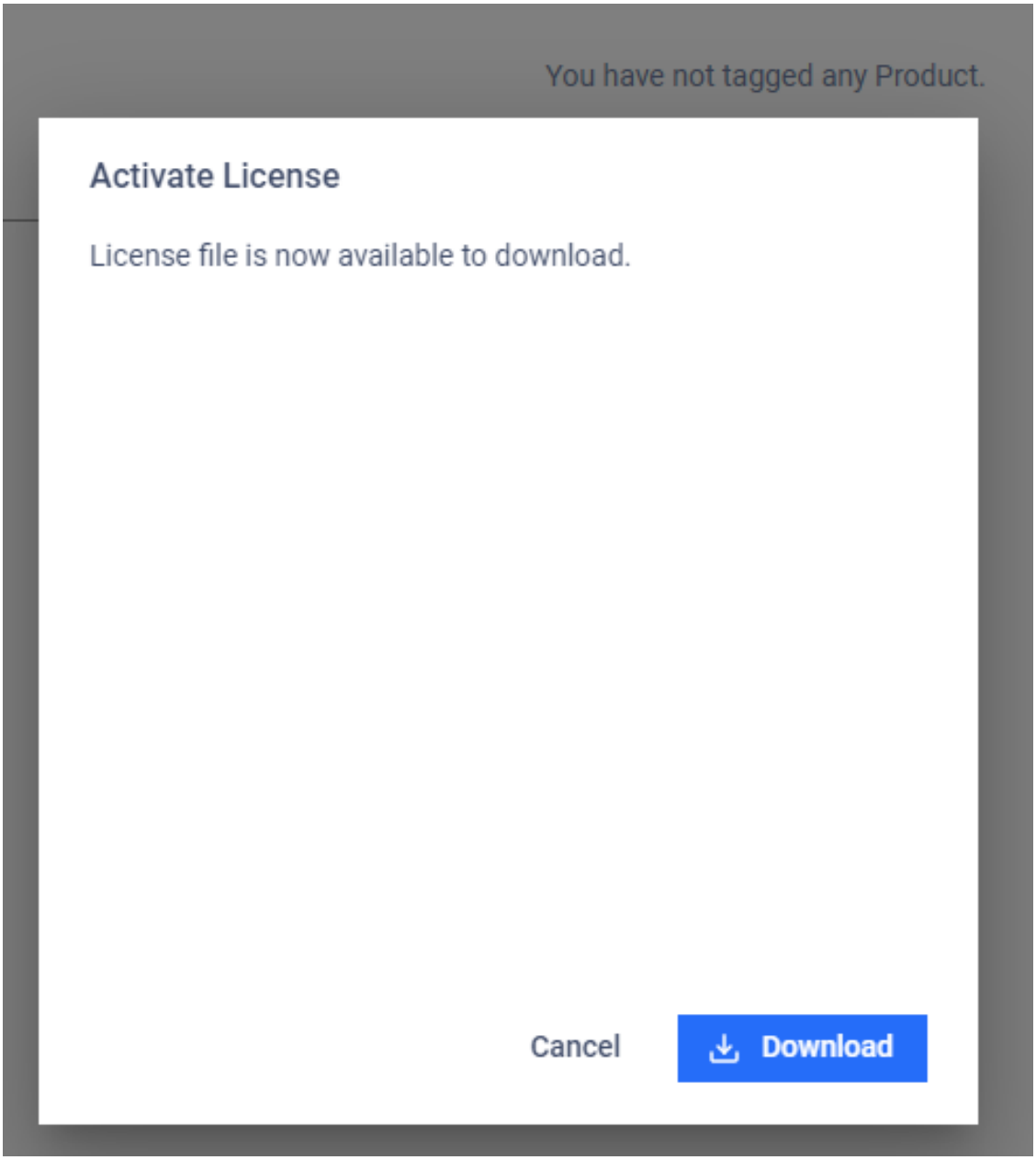
1. Go to your **MetaDefender Core Management Console>Settings>License** tab.
2. Select the **Offline Activation** option.
3. Copy your unique **Deployment ID**, as highlighted in the screenshot below.



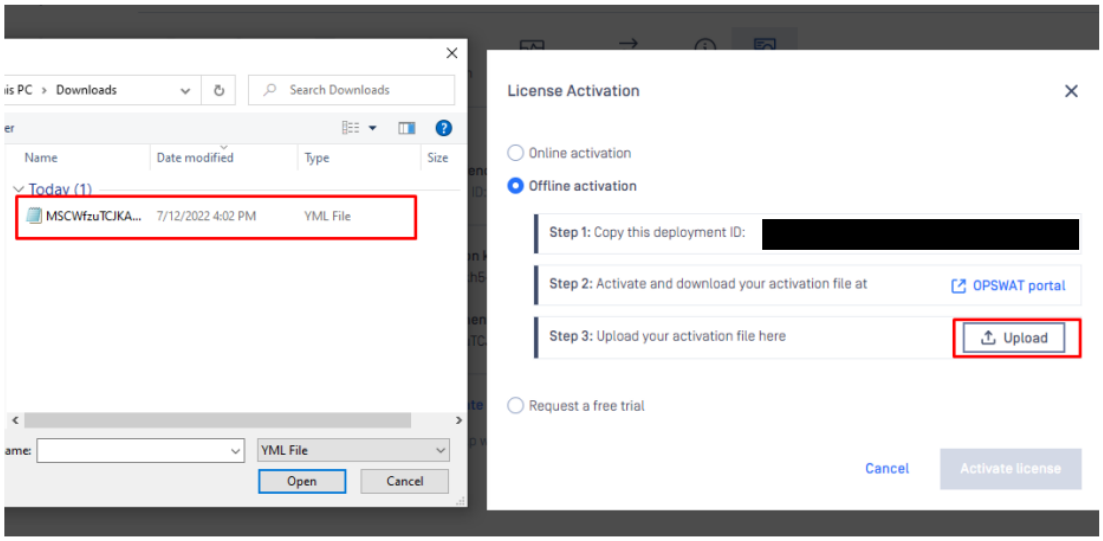
4. Navigate to MyOPSWAT here and click Activate License on the left side panel.
5. Fill out all necessary information, including your **Activation Key**, **Deployment ID** and selection of the **Package** you require.



6. Click **Activate**, then download it to a secure location.



7. Return to the **MetaDefender Core Management Console>License Activation** tab to upload your activation file (.yml file).
8. Finally, click **Activate License** to activate your offline deployment of MetaDefender Core.



Please Note:

Once you have completed the steps above, your offline core deployment will be activated.

Support:

If you have any difficulty **Activating MetaDefender Core In an Offline Windows or Linux Environment**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I activate/deactivate a MetaDefender Core Deployment via API calls?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Please Note:

In order to activate/deactivate your MetaDefender Core deployment via API calls, you will need to:

1. Obtain your valid **License/Activation Key**.
2. Obtain your **Deployment ID**.
3. Use these details to apply the relevant **API Calls** listed below.

To obtain the above mentioned details via API, please Read This.

Activate your deployment via API

Apply the following API call, as illustrated in the screenshot below:

Code · bash

```
curl -X GET "https://activation.dl.opswat.com/activation?key=<activation-key>&deployment=<deployment unique-ID>&quantity=<quantity>"
```



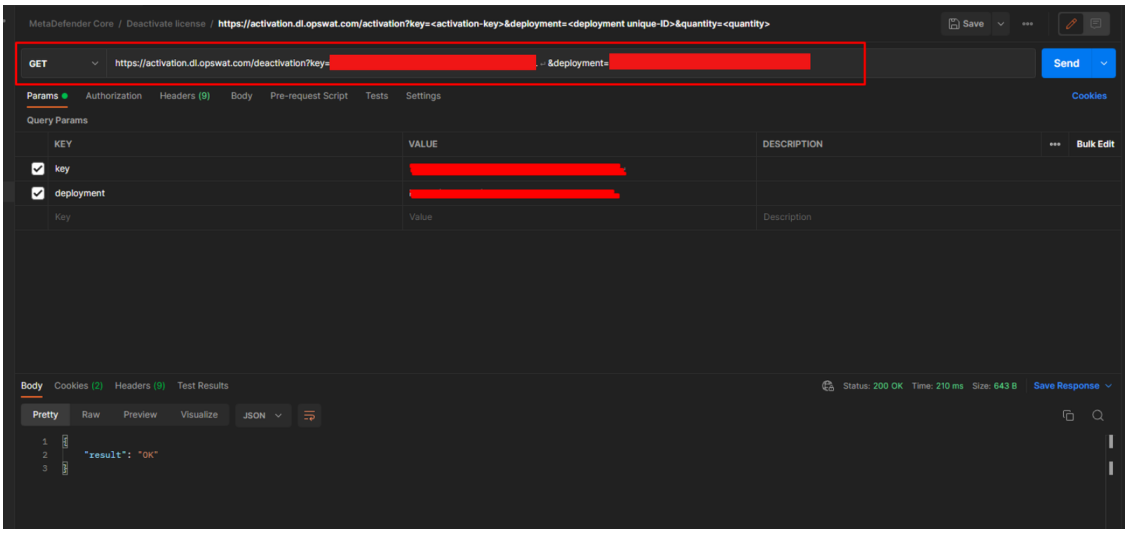
After running the API call described above users will need to use the “save response file” option and use the .yml file to activate the Core deployment.

Deactivate your deployment via API

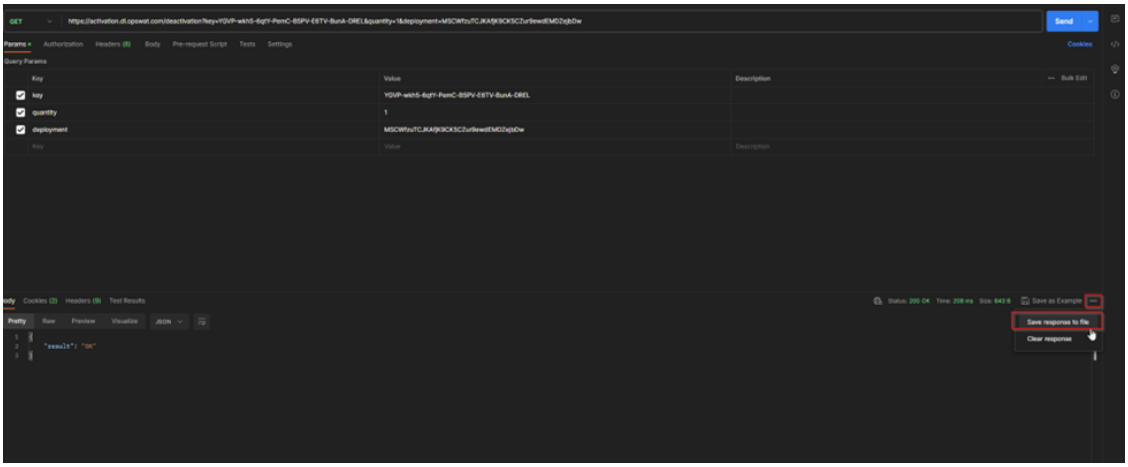
Apply the following API call, as illustrated in the screenshot below:

Code · bash

```
curl -X GET "https://activation.dl.opswat.com/deactivation?key=<activation-key>&deployment=<deployment unique-ID>"
```



After running the API call described above the mentioned deployment will be removed from the activation key.



Support:

If you have any difficulty **Activating or Deactivating a MetaDefender Core Deployment Via API Calls**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I update to a new license key on a Metadefender Core server?

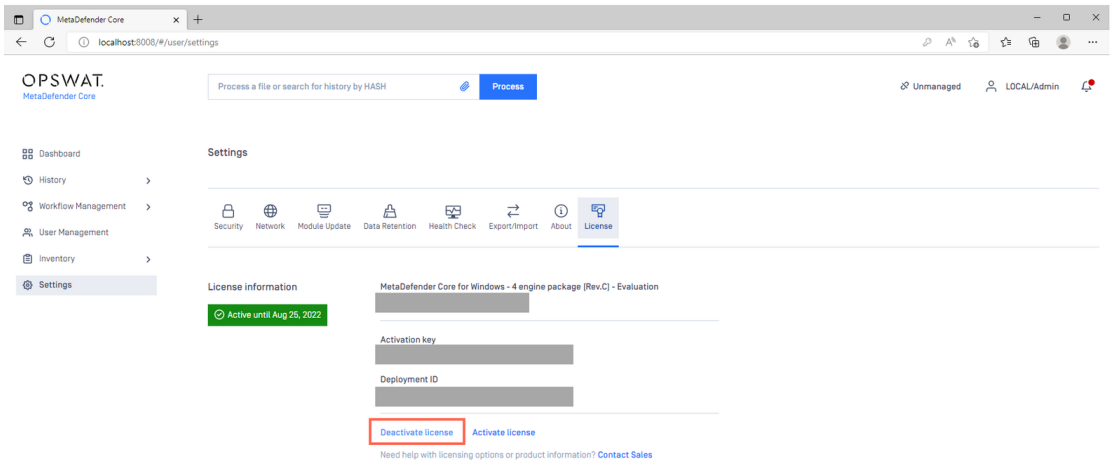
Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

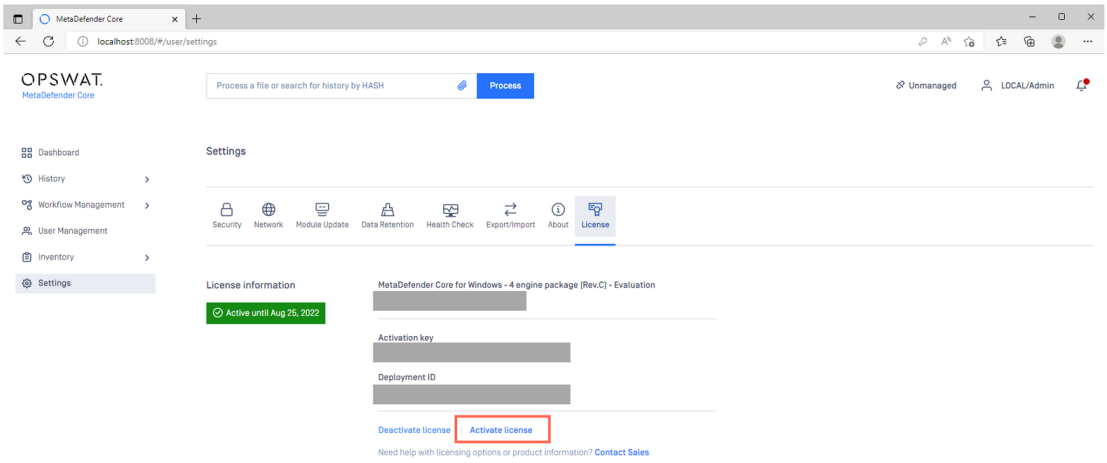
To update to a new license key, you will first need to deactivate your existing deployment (if it hasn't expired or been deactivated already), before activating again under a new license.

Online Activation

1. To do this, go to your **MetaDefender Core Management Console>Settings>License** tab, then click **Deactivate License**, as highlighted in the screenshot below.

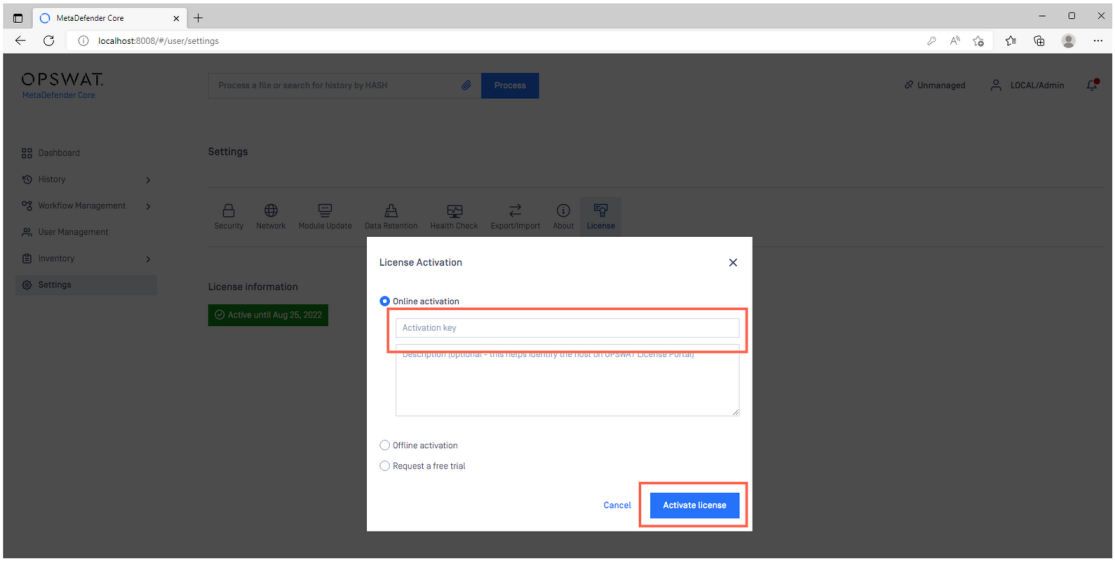


2. Next, click **Activate License**.



3. A pop-up will appear for you to enter your new **Activation Key** in the field provided, before selecting the appropriate **Online Activation/Offline Activation** option.

4. Then click **Activate License**, as highlighted in the screenshot below.



Offline Activation

For offline activation please follow the instruction here: How do I activate my offline deployment of MetaDefender Core?

Your new deployment should now be active.

Support:

If you have followed the instructions above but are **Unable To Update To a New License Key On a MetaDefender Core Server**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What types of licenses are available for MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

MetaDefender Core is available in a range of package options and highly customisable configurations, in order to cater to the diverse needs of our home, small to large business, offline/online platform and critical infrastructure clients.

Pro Tip:

Our standard packages differ in terms of the number of licensed AV scan engines they include, which in turn impacts other factors such as the requirements for the underlying system/s, including disk space, RAM, and the number of CPU cores, etc.

As a general rule, the scale and complexity of your digital infrastructure will dictate the size of your chosen packages, as well as the number and configuration of **MetaDefender Core** instances that need to be deployed under your license.

Product Options:

In the case of more advanced deployments, your infrastructure would likely benefit from integration with various other OPSWAT product options, such as **MetaDefender Kiosk**, **Vault** and more.

For more information on our products, please visit our Product Page, or reach out to OPSWAT support as instructed later in this article.

Support Options:

OPSWAT offers three software support plans that provide increasing levels of support and expert interaction, including **Standard, Gold and Platinum** options.

These options vary in terms of support availability, response time, price and other factors. For details about each option, please read our Support Page.

With that said, our package license options, along with their corresponding number of scan engines and their basic system requirements, are included in the table below.

MetaDefender Core Windows packages

Package	CPU cores	Free System RAM	Free Disk Space
MetaDefender Core 8	8	8 GB	16 GB
MetaDefender Core 12	16	16 GB	24 GB
MetaDefender Core 16	16	16 GB	32 GB
MetaDefender Core 20	32	16 GB	40 GB
MetaDefender Core 32	32	32 GB	120 GB

MetaDefender Core Linux packages

Package	CPU cores	Free System RAM	Free Disk Space
MetaDefender Core 5	4	4 GB	10 GB
MetaDefender Core 10	10	8 GB	20 GB

For more information on system requirements per package, please Read This.

Support:

If you need further guidance in **Choosing The Appropriate License Option For Your Enterprise**, feel free to contact OPSWAT support by creating a support case or chatting with our support engineer.

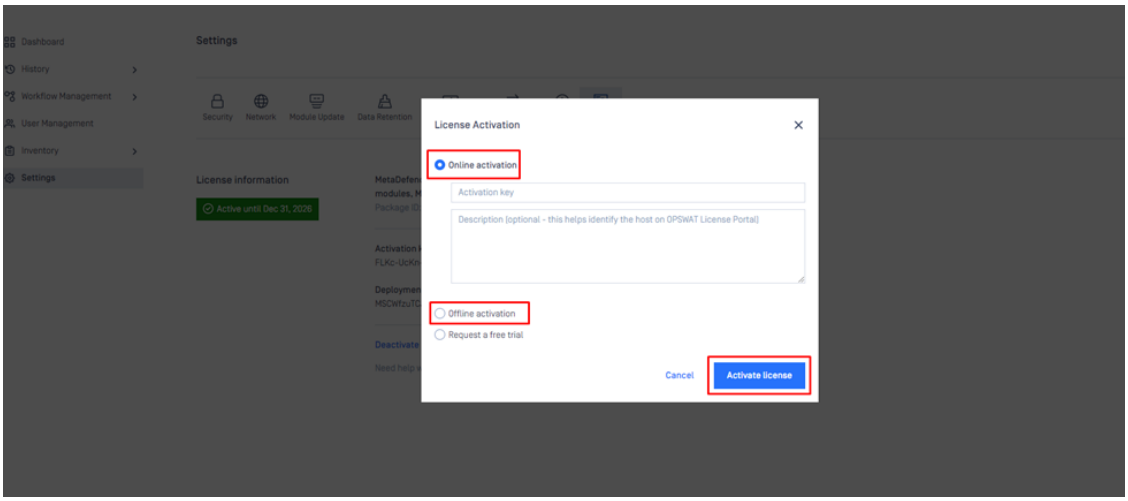
After license renewal does the product activation happen automatically (if the license key is the same)?

Check Your Version:

This article applies to all MetaDefender V5 releases deployed on Windows and Linux systems.

In special situations where the renewal process is completed and the license values regarding products and expiration dates are changed (but the license key remains the same), new license information will be obtained and applied when triggering an online engine update.

For offline environments, users will need to go through the activation steps and generate a new yml. file.



Support:

If **Further Assistance** is required, please follow the instructions on How to Create Support Package With Bundle Tools?, then proceed to log a support case or chat with our support engineer.

How to install .NET 8 Runtime dependencies on Linux Distributions?

Check Your Product:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Linux systems.

Pro Tip:

Linux Distributions usually have .NET dependencies installed in their system, if the engines failed to load, please follow the steps below to install

Red Hat Enterprise Linux 8.x

Code · bash

```
sudo yum install libnsl
sudo dnf install krb5-libs libicu openssl-libs zlib
```

Red Hat Enterprise Linux 9.x / Rocky Linux 9

Code · bash

```
sudo yum install libnsl
sudo dnf install krb5-libs libicu openssl-libs zlib
sudo dnf install compat-openssl11
```

Debian 11.x

Code · bash

```
wget https://packages.microsoft.com/config/debian/11/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt update
sudo apt install dotnet-runtime-deps-8.0
```

Debian 12.x

Code · bash

```
wget https://packages.microsoft.com/config/debian/12/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt update
sudo apt install dotnet-runtime-deps-8.0
```

Ubuntu 22.04

Code · bash

```
wget https://packages.microsoft.com/config/ubuntu/22.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt update
sudo apt install dotnet-runtime-deps-8.0
```

Amazon Linux 2003

Code · bash

```
sudo dnf install krb5-libs libicu openssl-libs zlib
```

Support:

If you have followed the instructions above and were unable to install .**NET Runtime dependencies on Linux Distributions**, please open a Support Case with the OPSWAT team via phone, online chat or form.

How to connect LDAP via SSL on Linux

Check Your Version:

This article applies to all MetaDefender V5 releases deployed on Linux systems.

There are 2 ways we can config to make MetaDefender Core read the cert using LDAP via SSL connection

Note:

Please use only one method. If you check with local trusted store, please don't use */etc/openldap/ldap.conf*. If you use */etc/openldap/ldap.conf* please check that there is no file *ldap.crt* file in local trusted store also, there is no content of *ldap.crt* file in *ca-bundle.crt*. Please use only one method.

1. We can use a local trusted store, here is how we add crt file to the local trusted store

    1. Deb/Ubuntu:

Code · bash

```
cp -f ldap.crt /usr/local/share/ca-certificates
update-ca-certificates
```

b. Centos/RHEL:

Code · bash

```
cp -f ldap.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust enable
update-ca-trust extract
```

2. If we do not want to use a local trusted store, we also can create a specific configuration file to tell MetaDefender Core where to look for cert

Code · bash

```
#Create file ldap.conf in /etc/openldap/ldap.conf
##Centos/RHEL
echo "TLS_CACERT /path/to/ldap.crt" >> /etc/openldap/ldap.conf
##Debian/Ubuntu
echo "TLS_CACERT /path/to/ldap.crt" >> /etc/ldap/ldap.conf
```

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to Backup and Restore standalone database for MetaDefender Core?

Info

This document based on PostgreSQL official documents for backup and restore here

Prerequisites

- No third-party application needed
- Windows OS
- PostgreSQL built-in (Metadefender Core standalone mode)

How to

1. Backup database

- **Please note** that you need to go to the folder `<installation_folder_MD_Core>/postgres/bin/` to execute the command below, you need to run it on CMD (Command Prompt) with Administrator privileges.
- To back up the database, you need to run the command below.

Code

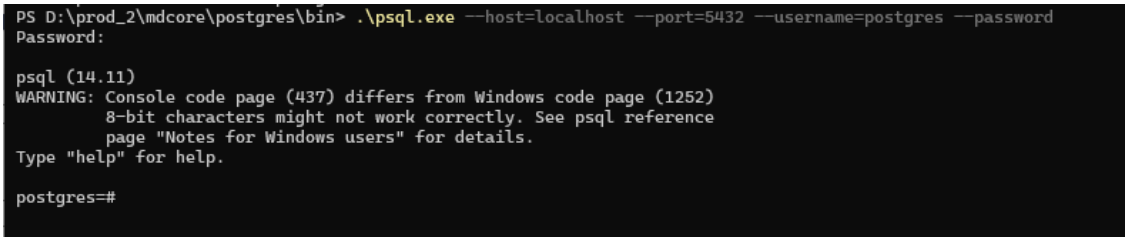
```
<installation_folder_MD_Core>/postgres/bin/pg_dump.exe -h <hostname> -p <port> -U <username> -W
<database_name> > <output_path>
```

- In order to use `pg_dump` to backup exactly database that you would like to backup, we need the database name. To get database name, you need to connect to PostgreSQL local:

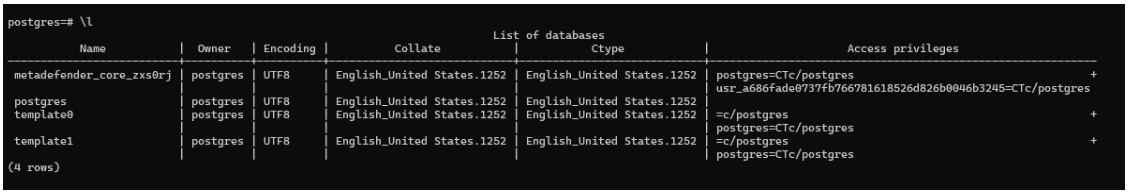
Code

```
psql --host=localhost --port=5432 --username=postgres --password
```

- It requires you to input the password for the user `postgres`. After you input the password, you will successfully log in to Postgres. For example:



- type `\l` to show all database instances in PostgreSQL.

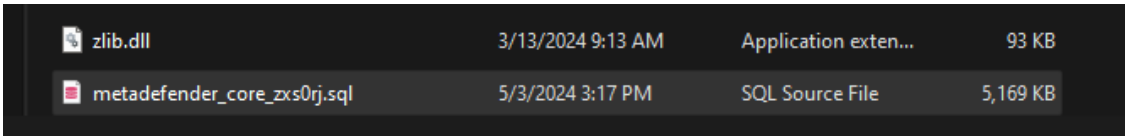


- To exit PostgreSQL, you can type `exit`.
- When you know your database instance name, you can use `pg_dump` to backup exactly database that you would like to backup.

Code

```
pg_dump.exe -h localhost -p 5432 -U postgres -W metadefender_core_zxs0rj >
metadefender_core_zxs0rj.sql
```

- After finished, you will see the file generated.



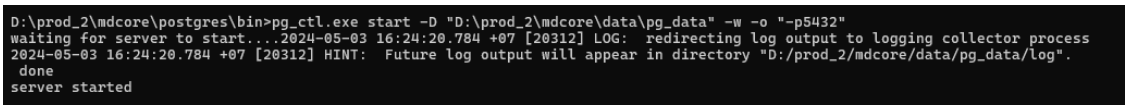
2. Restore database

- To backup database, you need to stop Metadefender Core first. Then, please start PostgreSQL service by command below.

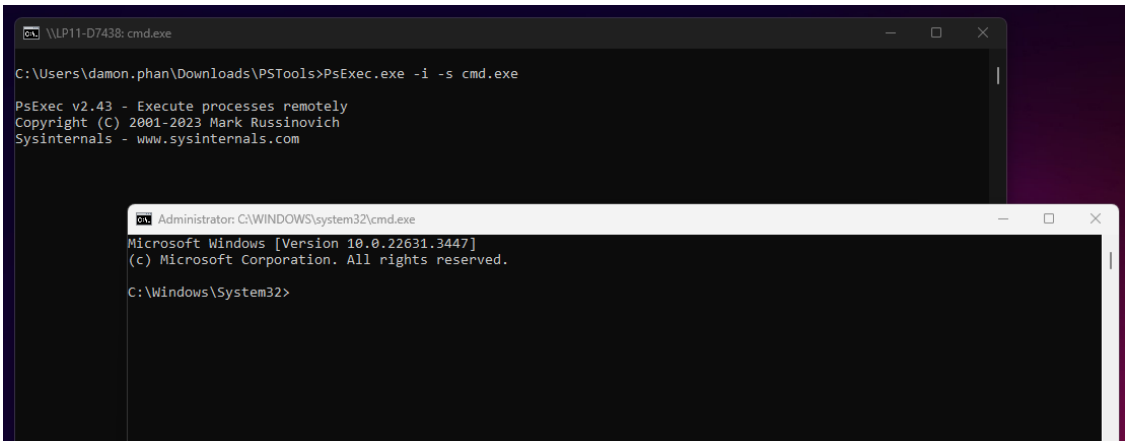
Code

```
<installation_folder_MDCore>\postgres\bin\pg_ctl.exe start -D "<PG_DATA_DIR>" -w -o -p5432
```

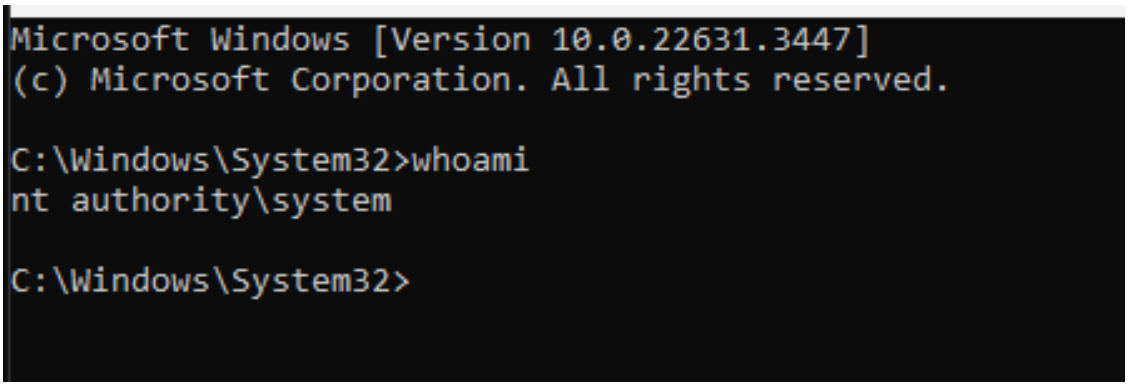
- For example:



- **NOTE**: If your MD Core is on the C drive (system drive), you'll need to utilize the system user to execute the command above for manually starting PostgreSQL.
  - To use system user, you need to download the PSTool provided by Microsoft PsTools - Sysinternals | Microsoft Learn
  - Once you've downloaded PSTool, to utilize it, you must open Command Prompt with Administrator privileges. Next, navigate to the PSTool folder and execute the command `PsExec.exe -i -s cmd.exe`. Following this command, a new Command Prompt window will appear, indicating successful execution.

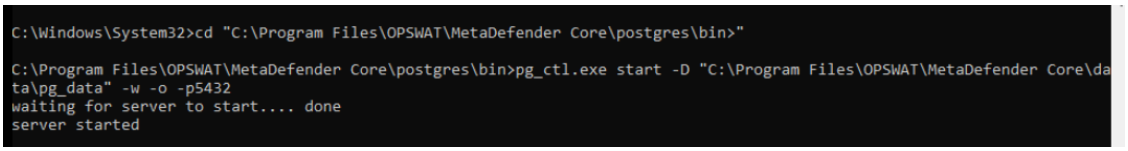


- To determine the user being utilized in the new CMD window, you can execute the command `whoami`. This will provide you with the user information.



- Then, we will run the command to manually start PostgreSQL. For example: "c:\Program Files\OPSWAT\MetaDefender Core\postgres\bin\pg_ctl.exe" start -D "c:\Program Files\OPSWAT\MetaDefender Core\data\pg_data" -w -o -p5432

- You will see the output below.

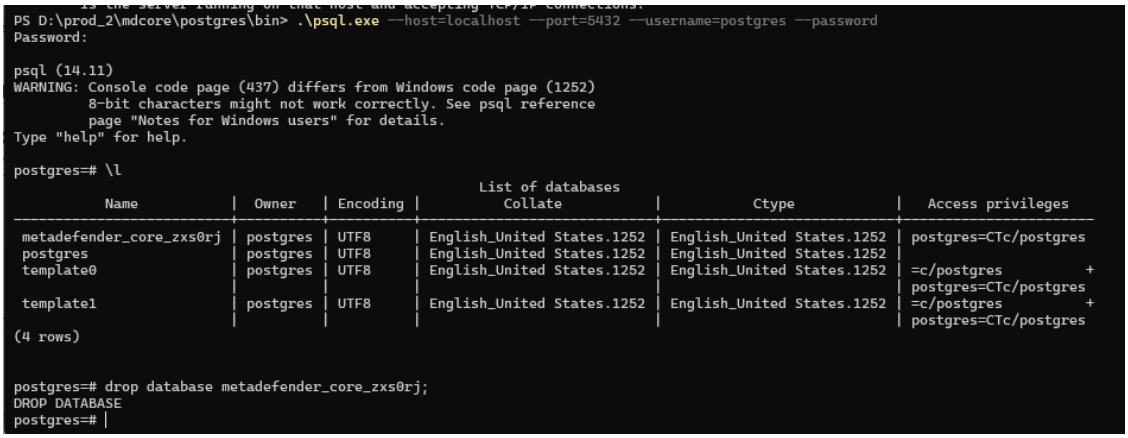


- Once you've successfully initiated PostgreSQL manually, you'll need to log in to PostgreSQL and delete the current MD Core database. To accomplish this, you can utilize the following command:

Code

```
drop database metadefender_core_zxs0rj;
```

- Here is example:

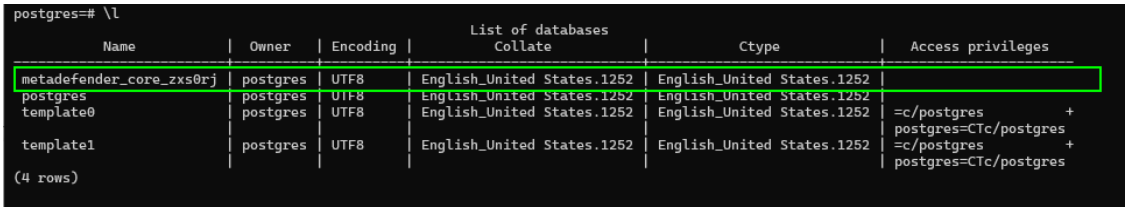


- When you dropped the current database of MD Core, you create the new database again with the same instance name.
- To create database, you can use the command below. Please note that the name of database should be the same.

Code

```
create database metadefender_core_zxs0rj;
```

- You can recheck by command `\l` to list all database.



- You can see that the new database with name "metadefender_core_zxs0rj" created, and currently, it does not have any access privileges.

- Next, exit PostgreSQL and import your data from the the dump file to the new database. You can use command below to import the database.

Code

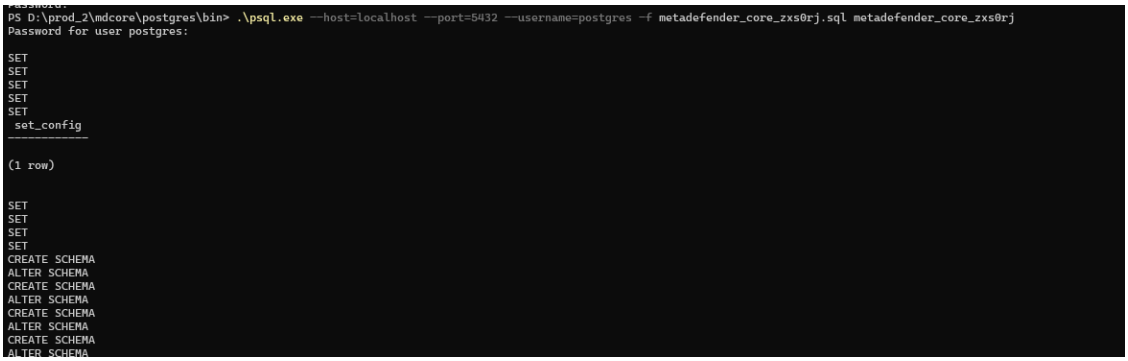
```
<installation_folder_MDCore>\postgres\bin\psql.exe --host=localhost --port=5432 --
username=postgres -f <YOU_DUMP_FILE> <database_name>
```

- For example:

Code

```
.\psql.exe --host=localhost --port=5432 --username=postgres -f metadefender_core_zxs0rj.sql
metadefender_core_zxs0rj
```

- You will the output like this



- After you finish import database, you need to stop PostgreSQL. You can use command below to stop

Code

```
<installation_folder_MDCore>\postgres\bin\pg_ctl.exe stop -D "<PG_DATA_DIR>" -w -o -p5432
```

- For example:

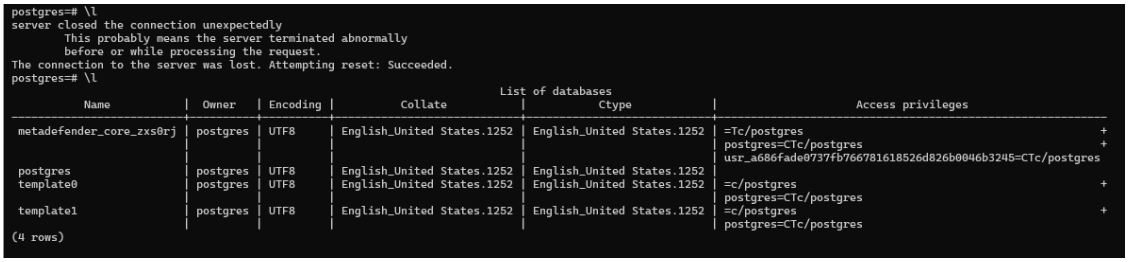


- Finally, you start Metadefender Core service. While starting, MD Core will create new internal user and grant access priviledge for the new database.
- You can recheck by access to PostgreSQL and show status all database, after MD Core service successfully start and running.

Code

```
psql --host=localhost --port=5432 --username=postgres --password
```

- For example:



- You can see that the new database has new access privileges for `internal_user` `usr_a686fade0737fb76678161 8526d826b0046b3245`

Support:

If you require further assistance, please log a support case or chat with our support engineer.

How to install msttcore-fonts on Linux systems?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Linux systems.

To apply non-English watermarks (e.g., Hebrew, Japanese) in P-DLP or to resolve issues where text becomes corrupted (e.g., characters overlapping) after OCR, additional fonts from Microsoft must be installed on your Linux system.

Installation Instructions

1. **Debian/Ubuntu-based Systems**

To install the required font, please issue the following commands.

Code · bash

```
sudo apt install fontconfig ttf-mscorefonts-installer
fc-cache -v -r
```

In case you get an error like `Package ttf-mscorefonts-installer is not available...`, please enable the contrib repository, which contains `ttf-mscorefonts-installer`, by adding **contrib** to the repository lines in **/etc/apt/sources.list** (see below) and then run `sudo apt update`.

Code · bash

```
deb http://deb.debian.org/debian/ bullseye main contrib
deb-src http://deb.debian.org/debian/ bullseye main contrib
deb http://security.debian.org/debian-security bullseye-security main contrib
deb-src http://security.debian.org/debian-security bullseye-security main contrib
deb http://deb.debian.org/debian/ bullseye-updates main contrib
deb-src http://deb.debian.org/debian/ bullseye-updates main contrib
```

2. **RHEL based systems**

Please first take a look at the package and its content/dependencies to ensure it complies with your regulations.

First, you need to install the dependencies:

Code · bash

```
sudo yum install curl cabextract xorg-x11-font-utils fontconfig
```

Once the dependencies have been installed, you can install the .rpm package itself:

Code · bash

```
sudo rpm -i https://downloads.sourceforge.net/project/mscorefonts2/rpms/msttcore-fonts-
installer-2.6-1.noarch.rpm
fc-cache -v -r
```

3. **RHEL 9 system**

First, install the required dependencies:

Code · bash

```
dnf install curl fontconfig cabextract
```

If you get the following error: `No match for argument: cabextract` you need to enable the EPEL repository:

Code · bash

```
subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

Then, try to install The `cabextract` again:

Code · bash

```
dnf install cabextract
```

The `xorg-x11-font-utils` is also a dependency for the msttcore-fonts-installer, but it is not available in any base RHEL 9 repository, so you need to install the following tools:

Code · bash

```
dnf install mkfontscale mkfontdir xset
```

When you install the msttcore fonts RPM, you need to use the `--nodeps` option so the installation won't fail because of the missing `xorg-x11-font-utils`:

Code · bash

```
sudo rpm -i https://downloads.sourceforge.net/project/mscorefonts2/rpms/msttcore-fonts-installer-2.6-1.noarch.rpm --nodeps
fc-cache -v -r
```

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to upgrade Amazon Corretto JRE?

Check Your System:

If the Sandbox engine is installed and relies on Amazon Corretto JRE, we strongly recommend upgrading to Amazon Corretto 17 to ensure system security and compatibility.

Upgrade Instructions:

1. Download the latest version of Amazon Corretto 17 from the official AWS website: Amazon Corretto 17
2. Follow the installation instructions provided on the download page.
3. For detailed guidance, refer to the MetaDefender Core Sandbox engine system requirements documentation: Sandbox Dependencies
4. Upon upgrading, the preflight script of the Sandbox engine will no longer display a warning about using an outdated Corretto version.

Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

Can I activate the product license before the start date?

No, product licenses can only be activated when the **set start date and time** are reached. Activation will **not be possible** before this period as the license is time-restricted based on the agreement.

There are three different options for the product activation:

- Online activation: MetaDefender will connect directly to the OPSWAT licensing server online, and acquire its license based on your Activation key and its Deployment ID.
- Offline activation: Requiring you to upload a license file (.yml). That license file could be retrieved by activating your license via OPSWAT Portal. Follow the displayed instructions for details.
- Request trial key online: For evaluation purpose, you can receive a trial Activation key via email. Follow the displayed instructions for details.v

If you require **early activation before the official start date**, please **contact our sales team** for further assistance.

Support:

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

How to completely uninstall MetaDefender Core?

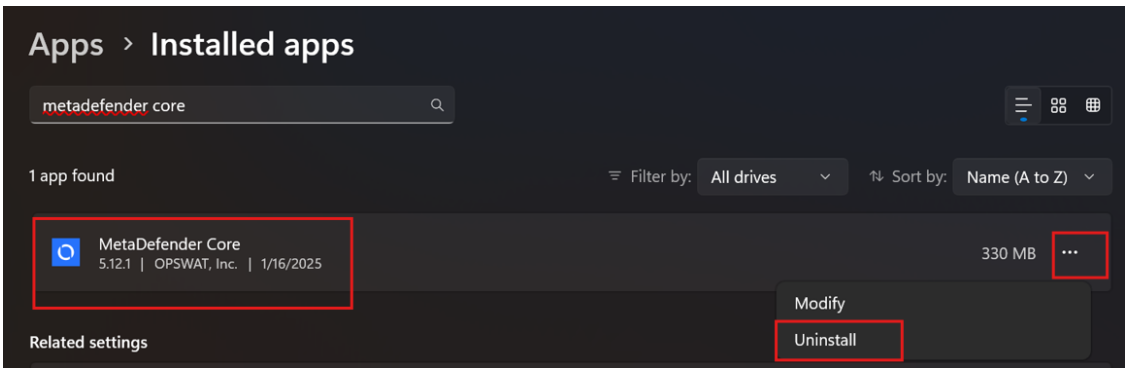
Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

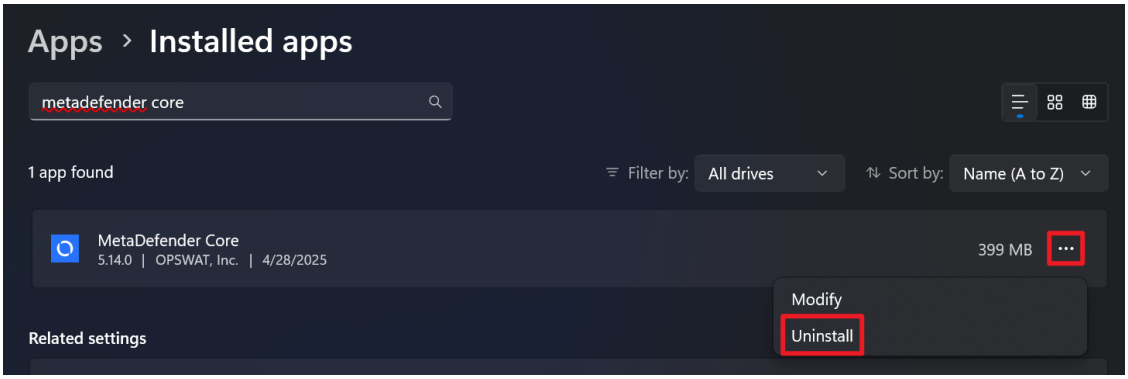
This article provides comprehensive steps to completely uninstall MetaDefender Core, including removal of the application, services, configuration files, logs, and database files on both Windows and Linux platforms.

Uninstall Microsoft Windows Deployments

1. Open the **Add or Remove Programs** settings in Windows, then search for **MetaDefender Core** in the list of installed applications.



2. Click the three-dot menu next to MetaDefender Core, and select **Uninstall**.



3. Delete Remaining Files and Directories

To completely remove all MetaDefender Core-related data, logs, and configuration files, please refer to this article: What are the List of folders that should be removed after MetaDefender Core uninstallation on Windows?

Uninstall Linux Deployments

1. Stop and Disable MetaDefender Core Services

Before uninstalling, stop the service to ensure no processes are running:

```
sudo systemctl stop ometascan
sudo systemctl disable ometascan
```

2. Remove the Core Software Package

For RPM-based systems (Rocky Linux, CentOS, RHEL):

```
sudo dnf remove ometascan
```

For Debian-based systems (Ubuntu, Debian):

```
sudo apt-get remove --purge ometascan
```

3. Delete Remaining Files and Directories

To completely remove all MetaDefender Core-related data, logs, and configuration files, please refer to this article: What are the List of folders that should be removed after MetaDefender Core uninstallation on Linux?

Navigate to the File Explorer and remove the following folders if they exist:

> **Support:**
>
> If Further Assistance is required, please proceed to create a support case or chat with our support engineer.

How to Trust MetaDefender Core's Self-Signed Certificate When HTTPS Is Enabled?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

When enabling HTTPS for MetaDefender Core, most of the times a self-signed Security Certificate is being used. In these scenarios, other products integrated directly with it, in a 1-to-1 connection (without any intermediary proxy and/or network appliance), will fail to connect due to an SSL Trust Relation not establishing anymore. To address this, the Security Certificate needs to be installed in the Trusted Root Certification Authorities store of the machine hosting the application integrated with Core. Please find below the steps to achieve this for each Operating System:

Windows:

1. Double-click the .crt file opens "Certificate".
2. Click "Install Certificate...".
3. Choose "Local Machine" (You will need Administrator rights).
4. Choose "Place all certificates in the following store"
5. Click Browse select Trusted Root Certification Authorities.
6. Click Next, then Finish.

Debian:

1. Copy your certificate to the trusted directory:

```
sudo cp mycert.crt /usr/local/share/ca-certificates/
```

2. Update CA store:

```
sudo update-ca-certificates
```

RedHat Enterprise Linux:

1. Copy the certificate:

```
sudo cp mycert.crt /etc/pki/ca-trust/source/anchors/
```

2. Update CA Trust:

```
sudo update-ca-trust extract
```

*The application service/machine needs to be restarted for the changes to take place and the SSL Trust Relation to establish.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How are MetaDefender Core AWS AMI images secured and updated?

Patching & Security Updates:

1. How quickly are critical security patches applied to the AMI?

```
For vulnerabilities detected from the MDCore application level, they will be patched in the next release of MD Core version. For vulnerabilities detected from the OS level, they will be patched and re-published within AWS policy.
```

2. Is the AMI hardened for secure configuration?

```
Yes. If the AMIs don't match the security policy from AWS, they won't let us release them to the marketplace
```

3. Is there a predefined SLA for security fixes?

```
We follow the SLA from AWS. If we don't fix the issue within the AWS SLA, they will remove our AMIs from the marketplace
```

AMI Lifecycle & Versioning:

How frequently is a new AMI released?

It will be released with every version of MD Core. After MetaDefender Core new version is released on My OPSWAT , it takes us approximately 1 week after public release to publish on AMI.

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

How to handle high CPU load on MetaDefender Core server?

Check Your Version:

This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems.

Hardware factor

When encountering high CPU utilization on your MetaDefender Core server, first verify that the system meets the minimum hardware requirements for your respective engine package:

- Windows system requirements
- Linux system requirements

As a reference, you can see Performance and Load Estimation metrics (throughput, CPU utilization, processing time), to estimate required hardware resource that could meet your throughput and processing time requirements.

Software factor

If the system meets the requirements, there are other software factors that can affect CPU utilization, as outlined in the MetaDefender Core Performance Tuning Guide:

- MetaDefender Core version: older versions may not have the best performance optimization.
- Engines and database: processing with more engines and using local PostgreSQL database can increase CPU usage on the server.
- Configuration: more thorough scan settings (such as deep scans, archive scanning) require more hardware resources.
- File characteristics: more complex file types require more resources to scan and process.
- Concurrent scans: higher concurrent scans require more resources.
- Integration method: how MetaDefender Core is integrated into other applications (e.g. via APIs) can influence performance.

Details on these software factors, and software tuning guide to avoid high CPU utilization can be found in the MetaDefender Core Performance Tuning Guide.

Issue Example:

When resource utilization is very high (i.e. 100% CPU utilization) and files contain many archived files.

You can export MetaDefender Core's processing history as CSV format, and analyze which file types are being processed the most, for example:

Row Labels	Count of filetype
Adobe Portable Document Format	5
Algol 68 source	1
ASCII Text	15562
assembler source	1
data	135
DOS executable (COM)	2
Dynamic Link Library	25
Executable File	1
Extensible Markup Language	15
Graphics Interchange Format	622
Hypertext Markup Language	7593
ISO Media	1384
Java Server Pages	11
JavaScript Object Notation	5187
JPEG Image	2717
MP3 Audio	7
MPEG Transport Stream	2802
OpenPGP Public Key	1
OpenPGP Secret Key	1
OpenType font data	36
Portable Network Graphics	2472
Scalable Vector Graphics	779

If you see that there are many archive file types, you may need to adjust archive handling limits (max recursive level, max total extracted files, max total size of extracted files) in MetaDefender Core workflow.

- A higher limit means MetaDefender Core needs to dig deeper to process these files, so incrementally reduce the limit and observe CPU utilization.

Hardware limitation:

If all software tuning options have been exhausted and CPU utilization remains high, you might need to consider increasing the hardware resources in order to meet the demanding processing load.

Support:

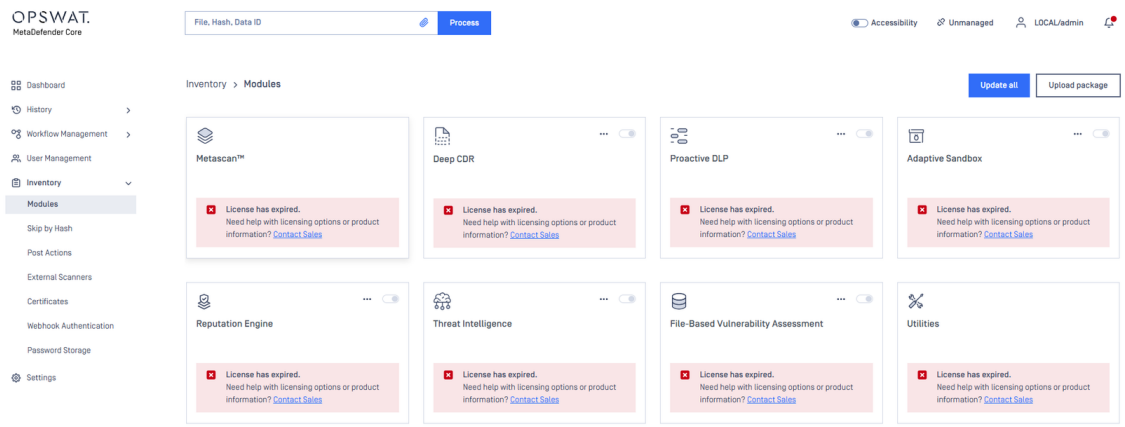
If you have followed the instructions above but are still encountering **High CPU load on MetaDefender Core server**, please follow these instructions on How to Create Support Package With Bundle Tools?, before **creating a support case or chatting with our support engineer**.

What will happen to the MD core after the license is expired?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

When the license associated with MetaDefender Core reaches its expiration date, the licensed modules and engines will stop receiving updates. This may impact the effectiveness and security posture of your system, as engine definitions and threat intelligence become outdated.



Once the License Has Expired:

- **Engine Updates Will Cease** All associated engines—such as Metascan, Deep CDR, Threat Intelligence, and others—will no longer receive updates.
- **License Expiration Banner** A red warning banner will appear in the MetaDefender Core UI (as shown in the image), indicating that the license has expired and prompting you to contact Sales.
- **Manual Updates Restricted** Even if updates are attempted manually, engine or module updates will fail until a valid license is restored.
- **Unable to Scan Files** Once the engines have expired, MetaDefender Core will be unable to scan any files until the license is renewed.

What to Do Next:

- Please contact our Sales team to renew or extend your license based on your usage needs.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What are the List of folders that should be removed after MetaDefender Core uninstallation on Linux?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Linux systems.

Please find below the list of folders that should be removed after performing a clean uninstallation of MetaDefender Core on Linux

List of folders · bash

```
/var/lib/ometascan
/var/log/ometascan
/var/tmp/ometascan
/var/run/ometascan
/usr/lib/ometascan
/run/ometascan
/etc/ometascan
```

Support

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

How can I fix a failed or looping MetaDefender Core module update by clearing the package cache?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

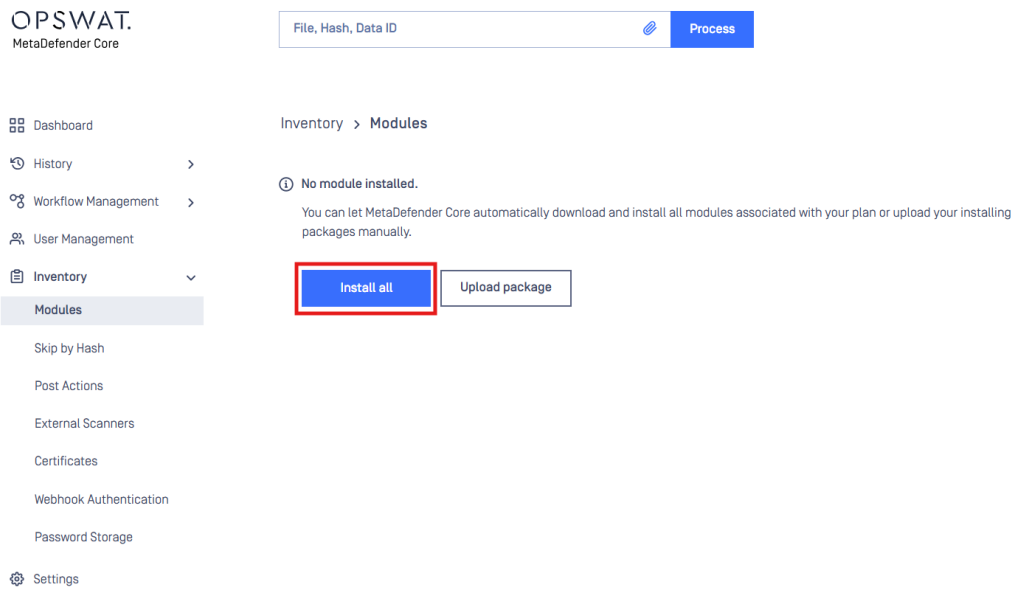
When should you use this?

- Engine/module updates fail, are stuck, or engines behave inconsistently.
- You need a **full clean reset** of engine and update data to force fresh downloads.
- In some cases, after a license undergoes modifications, remnants of old engines that are no longer reflected in the new license can still exist on the drive.

Option A - Standard reset (cached modules/engines)

Stop the service (elevated Command Prompt)`net stop ometascan`*(Or via **Services** → **OPSWAT MetaDefender Core** → **Stop**.)*

2. **Delete update cache** Delete all contents of: C:\Program Files\OPSWAT\MetaDefender Core\data\updates *(Delete files inside only; keep the folder.)*
3. **Delete engine folders** Delete all contents of: C:\Program Files\OPSWAT\MetaDefender Core\data\engines *(Delete files inside only; keep the folder.)*
4. **Start the service (elevated Command Prompt)**`net start ometascan`*(Or via **Services** → **Start**.)*
5. **Trigger module updates** In Core UI go to **Inventory** → **Modules** and click the **Install all** (blue) button.



6. **Verify** Ensure all modules show **Up to date** and sample scans succeed.

Linux Systems steps:

1. **Stop the service (elevated Command Prompt)**`systemctl stop ometascan`
2. **Delete update cache** Delete all contents of: /var/lib/ometascan/updates *(Delete files inside only; keep the folder.)*
3. **Delete engine folders** Delete all contents of: /var/lib/ometascan/engines *(Delete files inside only; keep the folder.)*
4. **Start the service (elevated Command Prompt)**`systemctl start ometascan`
5. **Trigger module updates** In Core UI go to **Inventory** → **Modules** and click the **Install all** (blue) button.
6. **Verify** Ensure all modules show **Up to date** and sample scans succeed.

Option B - Full reset including old-license leftovers *(recommended if engines don't match entitlements and option A fails to clean the unlicensed engines)*

1. **Deactivate the Core license** In the Core UI, open the **License** page and **Deactivate**. *(If OCM manages your license, temporarily prevent OCM from auto-pushing an old license until step 3 is complete.)*
2. **Stop the service (elevated Command Prompt)** `net stop ometascan` *(Or via **Services** → **OPSWAT MetaDefender Core** → **Stop**.)*
3. **Delete update cache** Delete all contents of: C:\Program Files\OPSWAT\MetaDefender Core\data\updates *(Delete files inside only; keep the folder.)*
4. **Delete engine folders** Delete all contents of: C:\Program Files\OPSWAT\MetaDefender Core\data\engines *(Delete files inside only; keep the folder.)*
5. **Start the service (elevated Command Prompt)** `net start ometascan` *(Or via **Services** → **Start**.)*
6. **Re-activate the license** In the Core UI, **Activate** the correct license (online or offline as applicable).
7. **Trigger engine updates** Go to **Inventory** → **Modules** and click **Install all** (blue button). Core will now download the engine set that matches your license.
8. **Verify** Confirm engines match your entitlements and show **Up to date**.

Linux Systems steps:

1. **Deactivate the Core license** In the Core UI, open the **License** page and **Deactivate**. *(If OCM manages your license, temporarily prevent OCM from auto-pushing an old license until step 3 is complete.)*
2. **Stop the service (elevated Command Prompt)** `systemctl stop ometascan`
3. **Delete update cache** Delete all contents of: /var/lib/ometascan/updates *(Delete files inside only; keep the folder.)*
4. **Delete engine folders** Delete all contents of: /var/lib/ometascan/engines *(Delete files inside only; keep the folder.)*
5. **Start the service (elevated Command Prompt)** `systemctl start ometascan`
6. **Re-activate the license** In the Core UI, **Activate** the correct license (online or offline as applicable).
7. **Trigger engine updates** Go to **Inventory** → **Modules** and click **Install all** (blue button). Core will now download the engine set that matches your license.
8. **Verify** Confirm engines match your entitlements and show **Up to date**.

Notes

- This resets **local engine/update data only**; Core configuration remains intact.
- For offline sites, prepare the required offline update packages and license files before you start.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

If a New AV Engine Is Released for MetaDefender Core, Is It Automatically Included in My License?

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Not automatically. A new AV engine being released doesn't mean it's instantly available to you without any action.

MetaDefender Core only downloads engines that are included in your license. When OPSWAT releases a new engine, whether it becomes available to you depends on whether your license entitles you to it.

If your license covers the engine:

- MetaDefender Core will automatically (or next time update is triggered manually) download all engines covered by your license from OPSWAT's online update infrastructure
- New engine versions (updates to existing engines) are also downloaded automatically, based on your Module Update configuration (Settings > Module Update)

However, if a brand-new engine is added to the OPSWAT catalogue and it's not part of your current license tier, it will not be included automatically. Your license would need to be updated to cover it.

If you're unsure whether a specific new engine is covered by your license, check your license details or reach out to your OPSWAT account team.

Support:

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

How can performance of MetaDefender Core be increased when deployed on K8S environments?

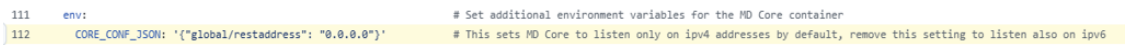
This article applies to all supported versions of MetaDefender Core

Background

While resource allocation influence the performance of MetaDefender Core, there are also internal environmental variables specific to the MetaDefender Core application that can be updated to increase performance on a Kubernetes deployment as well as we do on regular package installs.

Steps

For Kubernetes deployments, you can make updates to performance parameters using the CORE_CONG_JSON variable locates in the values.yaml file, like is done in metadefender-k8s/helm_charts/mdcore/values.yaml at f13e458b58b730d0735d2624abc4aef655c4487e · OPSWAT/metadefender-k8s



Using this environment variable, different values detailed in MetaDefender Configuration can be set. In particular the ones related to performance like parallecount, db_connection, etc. Keep in mind that the format for these entries should follow the format below:

```
[Section]/[parameter]: [value]
```

Example

For example, to set the parallel_count from its default, you can add an entry like: "**internal/parallelcount": 40**. This line would be updated as follows:

```
CORE_CONF_JSON: '{"global/restaddress": "0.0.0.0", "internal/parallelcount": 40, "internal/next_extraction_polling_interval": 1000}'
```

Support:

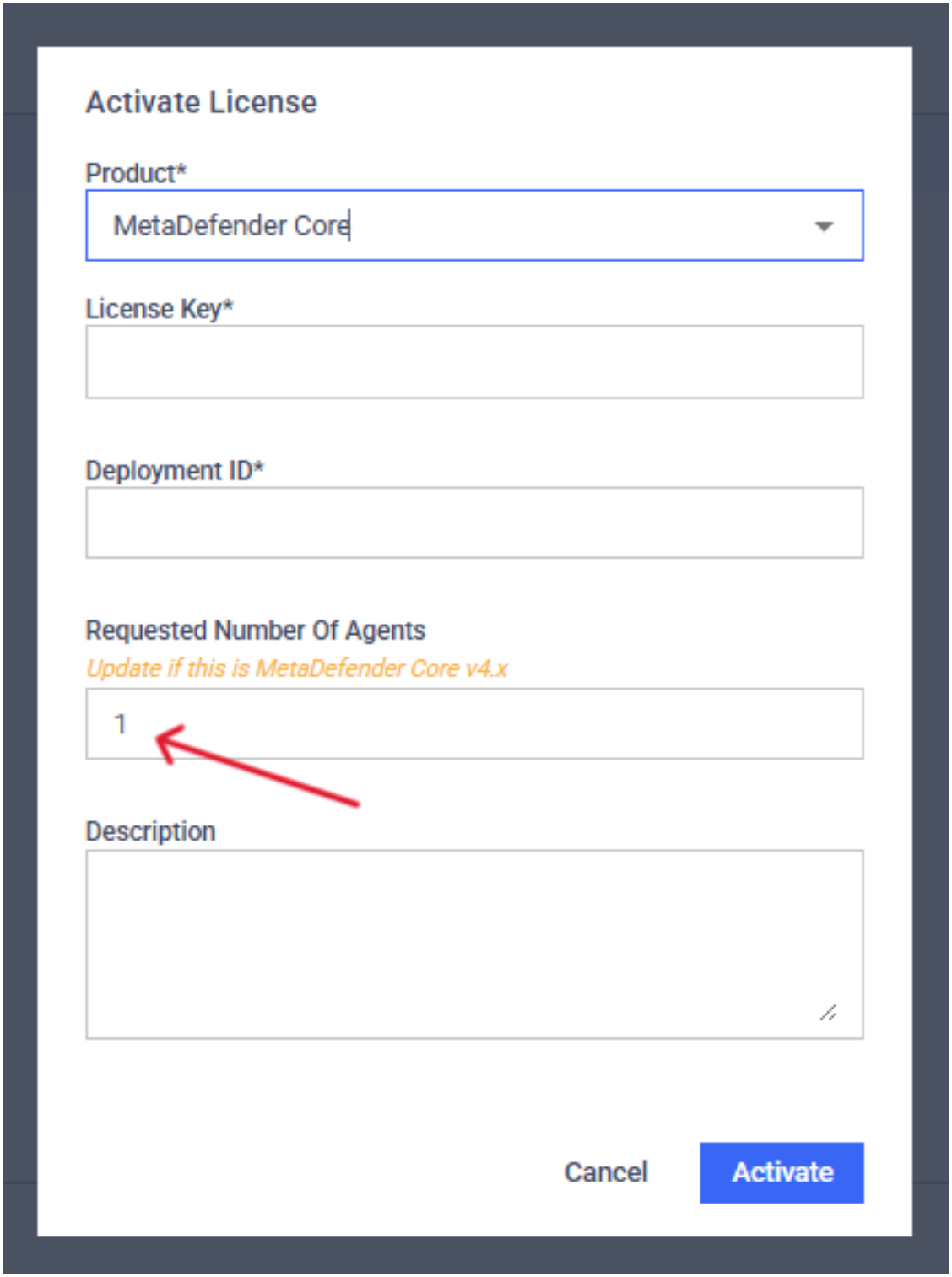
If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

Why Is MetaDefender Core Consuming More License Slots Than Expected After Activation?

Check your version:

This article applies to MetaDefender Core and is applicable to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

During product activation through the MyOPSWAT Portal, the number of agents requested may be set higher than intended. To ensure that only a single license slot is utilized per MetaDefender Core deployment, the **Requested Number Of Agents** field should be set to **1** (default).



Ensure the Number of Agents is set to 1 unless using Core v4.x deployments.

If a higher number is selected, multiple license slots may be allocated to a single deployment, even if only one instance is in use. The only exception to this is MetaDefender Core deployments running version **v4.x**, where multiple agents may be required depending on the configuration.

If you notice that more product license slots are being consumed than expected, please reach out to OPSWAT Customer Support. Our team can review your activation and adjust the license allocation on our end to ensure it accurately reflects your deployment.

Support:

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

Can MetaDefender Core use Azure SQL Managed Instance instead of PostgreSQL?

Check your version:

This article applies to MetaDefender Core and is applicable to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

MetaDefender Core **requires** PostgreSQL as its database. Other database engines, including Microsoft Azure SQL Managed Instance, are not supported.

If you are planning a deployment, you will need to provision a PostgreSQL database for MetaDefender Core to function correctly. **Support for alternative database platforms is not currently available and is not on the product roadmap.**

For best results, follow the official deployment documentation and ensure your PostgreSQL instance meets the recommended configuration and performance guidelines.

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

Can I delete the ignition file after MetaDefender Core is deployed?

Check Your Version: This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Answer

Yes. The ignition file (`ometascan.conf`) is used only during initial deployment. Core reads it to accept the EULA (End User License Agreement), import configuration, and create first user account, among other things.

Before deleting it, please confirm that MetaDefender Core is fully initialized and that you can log in normally. If initialization has not yet completed, the file is still in use and should be left in place.

File location

- **Linux:** `/etc/opswat/ometascan.conf`
- **Windows:** `C:\OPSWAT\ometascan.conf`

Because the file is never consulted again on a fully deployed system, removing it has no effect on MetaDefender Core's operation. Deleting it is also good security practice since some of the sensitive information is stored in plain text.

Support:

If you have questions, concerns or issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

How to migration MD Core postgresql database deployed on dedicated PostgreSQL server?

Check Your Version:

This article applies to all MD Core releases deployed on Windows and Linux systems. This guide walks you through migrating your MetaDefender Core PostgreSQL database to a new server.

Prerequisites

- Set up a new PostgreSQL server that is network-reachable by all MD Core servers.

Migration Steps

Step 1: Stop MD Core Services

Stop all MD Core instances that are connected to the shared MD Core database.

Step 2: Back Up the Existing Database

On the **original** PostgreSQL server, back up the metadefender_core database:

pg_dump.exe --file "C:\Core_DB_backup\metadefender_core.backup" --host "127.0.0.1" --port "5432" --username "postgres" --format=c --blobs --no-owner --no-privileges --verbose "metadefender_core"

Note: Adjust --host, --port, and --username if your setup differs from the defaults shown above.

Step 3: Create the New Database

On the **new** PostgreSQL server, create an empty database using the **same name** as the original (metadefender_core).

Step 4: Restore the Backup

Restore the backup file onto the newly created database:

pg_restore --host "127.0.0.1" --port "5432" --username "postgres" --dbname "metadefender_core" --verbose "C:\Core_Db_backup\metadefender_core.backup"

Note: Adjust --host, --port, and --username if your setup differs from the defaults shown above.

Step 5: Update MD Core Configuration

Point MD Core to the new PostgreSQL server by running(default path):

C:\Program Files\OPSWAT\MetaDefender Core\ometascan-change-db-credentials.exe

Follow the prompts to enter the new server's connection details.

Actions required after migration

Once configuration is updated:

1. **Start** your MD Core service.

2. **Login to MD Core UI**

3. **Verify** connectivity to the new database: “Settings” option à “Network” Tab à “Database” section

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Is there a virus test I could use to test MetaDefender Core engines?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Please Note:

Tests to determine an engine's operation should not be run with live malware. The suggested approach is to test using an industry standard test file called an **EICAR Test File**, which most antivirus engines detect as positive - even though no threat exists.

An **EICAR Test File** can be downloaded from here. You can also create your own version of the file by copying the following string into a file and renaming the file ‘**eicar.com**’.

Code

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Pro Tip:

For more information about the EICAR Test File, please visit the European Expert Group for IT-Security's Website.

Please take note that the EICAR file could be blocked by the localAV, so for testing purposes you should exclude it as a threat.

Support:

If you're having any trouble testing MetaDefender Core's included **Scan Engines**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How do I check if "noexec" flag exists on a Linux OS?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Linux systems.

Pro Tip:

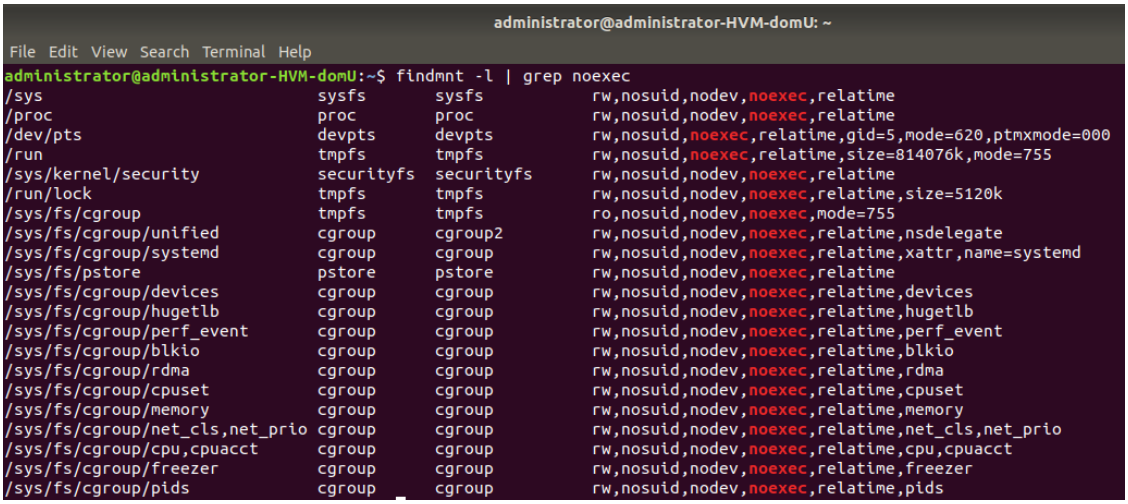
On Linux, MetaDefender Core deploys its engines and files in the **/var/lib** and **/usr/lib** folders. Depending on your file mount security policy, you may or may not have mounted **/var** or **/usr** with the flag **noexec**.

Essentially, the **noexec** flag will not allow any direct execution of binaries from the mounted filesystem/folder. In essence, if this flag exists, MetaDefender Core will not be able to launch its engine processes.

This will result in an engine with a **permanently_failed** module.

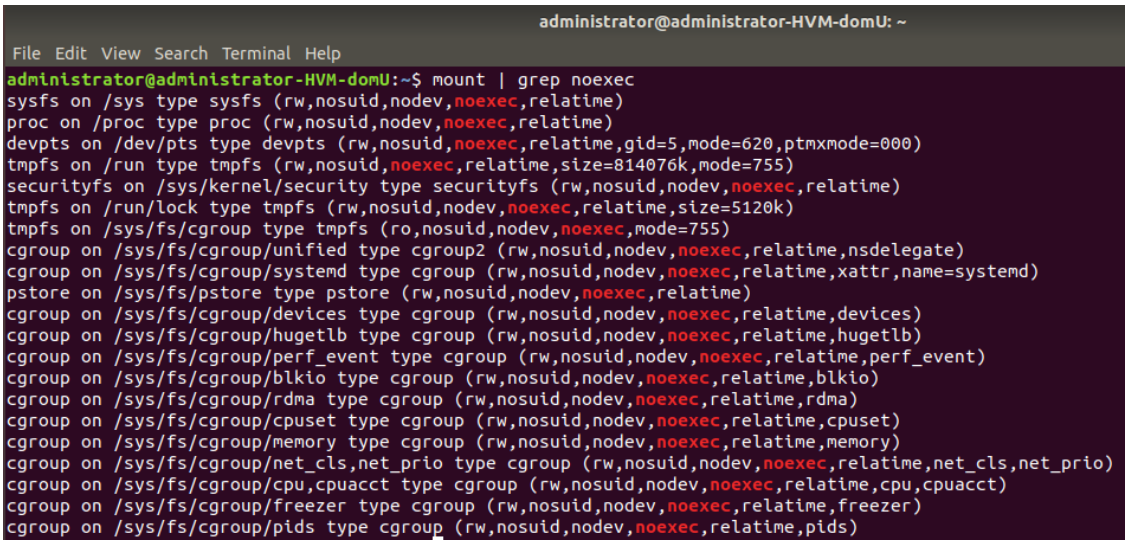
To c heck if the 'noexec' flag exists on /var or /usr simply do the following:

- Run Terminal and use one of the following commands:
  - **findmnt -l | grep noexec**



OR:

- **mount | grep noexec**



- Using the commands above will reveal whether there is a mount point with the **noexec** flag.
- If **/var** or **/usr** exist on the list, you must remove the **noexec** flag with the relevant following command:
  - **mount -o remount,rw,exec /var**
  - **mount -o remount,rw,exec /usr**

Support:

If you have followed all of the steps above but still have a **permanently_failed** module, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

How can I run tests to see the different scan results on

MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

The following test cases explain how to obtain the possible scan results from MetaDefender Core:

- **No Threat Detected:** Test this result by scanning any file you are certain is clean (e.g., a newly created text file)
- **Infected/Known :**
  1. Download an EICAR test file from here.
  2. Scan the file.
- **Suspicious :** This result is usually caused by an engine's heuristic algorithm. (We can actually simulate to produce this result if we write a script with external scanner.)
- **Blocklisted :** Test this result by adding the file to be tested to the **Blocklist**. For instructions on how to add files to the **Blocklist**, please refer to the Workflow Configuration Template.
- **Allowlisted:** Test this result by adding a file by its name or its mime-type to the **Skip** option and then scanning it. For more instructions on how to **Allowlist** files, please refer to the Workflow Configuration Template.
- **Exceeded Archive Size :**
  1. Configure "**Max total size of extracted files**" to a small value (i.e. 5 MB). This setting can be found on the MetaDefender Core Management Console under Policies Workflow Rules Select Workflow Rule Archive Tab.
  2. Create an archive file with a total size greater than 5 MB (after extraction).
  3. Scan the file.
- **Exceeded Archive File Number :**
  1. Configure "**Max number of files extracted**" with a small value (i.e. 10). This setting can be found on the MetaDefender Core Management Console under Policies Workflow Rules Select Workflow Rule Archive Tab.
  2. Create an archive file that contains more than 10 files (after extraction).
  3. Scan the file.
- **Password encrypted document/archive:** Scanning a password protected/encrypted document will produce this result. Currently, MetaDefender Core supports decryption of encrypted files both directly via the **Management Console** (as illustrated below) and via **REST API**.



- **Exceeded Archive Depth:** Test this result by configuring a lower recursion level than the current archive depth settings (Policies → Workflow Rules → Select Workflow Rule → Archive Tab)
- **Failed to scan:** Test this result by sending a file to scan which has no read permissions or is invalid. Alternatively, if no engine is in the MetaDefender Core installation and scan is enabled through the Security Rule configurations, this will be the final result
- **Mismatch:**
  1. Test this result by enabling the "**Detect File Type Mismatch**" option from the "**Policies**", tab "**File Type Analysis**" under the section inside of the "**Workflow Rule**" you are using.
  2. Alternatively, this result can be tested by changing the original extension of a file to different extension (i.e. test.docx → test.pdf) and scanning the file.
  3. Note that the option "**Detect File Type Mismatch**" only applies to workflows.
- **Potentially Vulnerable File:** Clean files can be marked as vulnerable if the Vulnerability Engine identifies known application vulnerabilities which are then reported by severity level. For more information on the **Vulnerability Engine**, please refer here

Support:

If you have any difficulty running tests to view scan results on MetaDefender Core, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Are scan engine updates automatic?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

By default, engines and their dependent databases are downloaded and updated automatically and on schedule thanks to MetaDefender Core update settings.

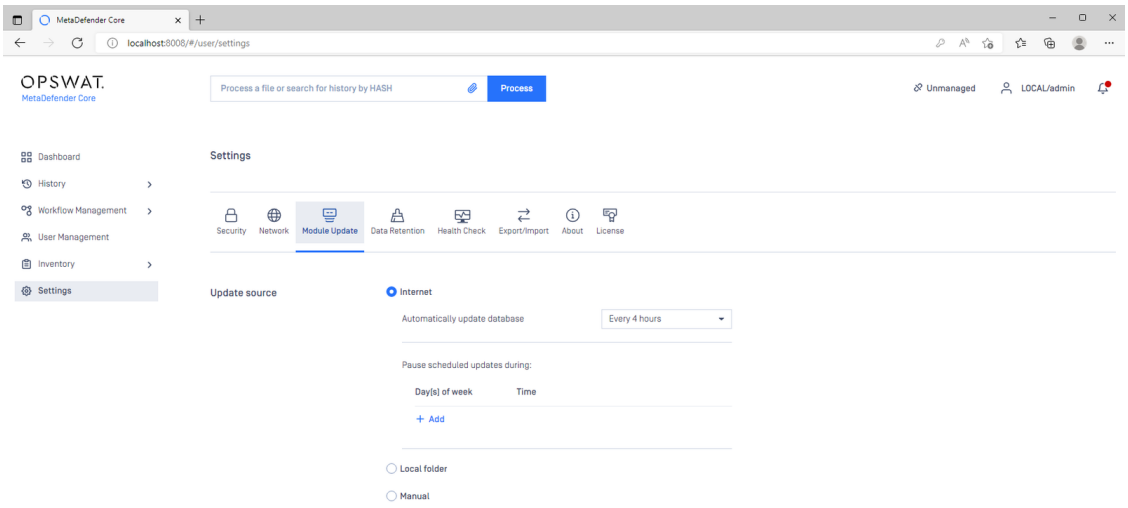
When applicable, customers can configure MetaDefender Core update settings to manually update them via the

Internet (on trigger), or designated local folder source, or upload engine packages manually.

MetaDefender Core update settings also supports to control update window time.

A side note that, MetaDefender Core supports to pin each engines and databases separately to avoid automatic update happening, and ensure only packages well-tested on customers' staging will be deployed in the production.

Most MetaDefender Core V5 scan engines (and their signature/definition databases) can be configured to either **Internet** (Automatic), **Local** (Automatic) or **Manual** updates, as illustrated in the screenshot below.



To learn about configuring updates for MetaDefender Core V5, visit Engines Update Configuration.

When configured for Automatic updates, MetaDefender Core V5 will check for available scan engine and signature/definition database updates on a regular basis. Visit Regular Maintenance to learn more.

Pro Tip:

OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate updated scan engines, state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

Support:

If you have any issues configuring your **Scan Engine Updates**, or for further guidance on the subject, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What operating system patches should be applied to the system hosting MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

The operating systems in OPSWAT's labs are kept updated with the latest patches. As a result, MetaDefender Core is tested for use with only optimized (patch-updated) operating systems.

Pro Tip:

MetaDefender Core does not require specific Windows patches to be operational, but it is best practice to patch the system as soon as possible to avoid security vulnerabilities.

OPSWAT highly recommends that you keep the operating system hosting MetaDefender Core completely updated with the latest OS version updates, patches and bug fixes.

Support:

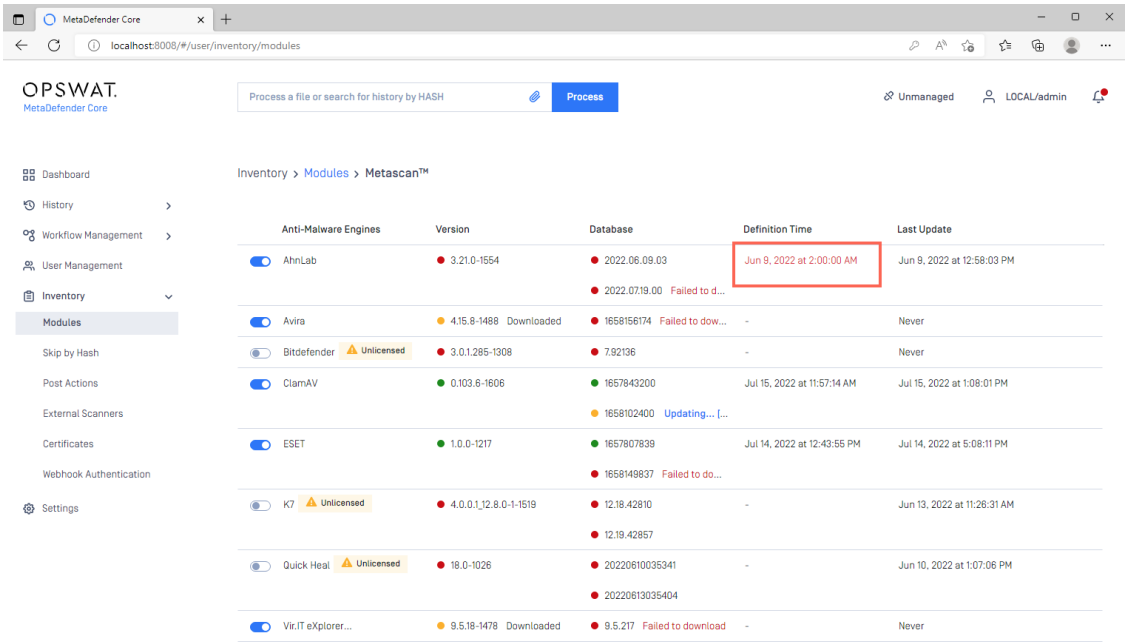
If your **Operating System** is fully updated but MetaDefender Core is not run optimally, please follow these instructions on How to Create Support Package With Bundle Tools?, before logging a Support Case with the OPSWAT team.

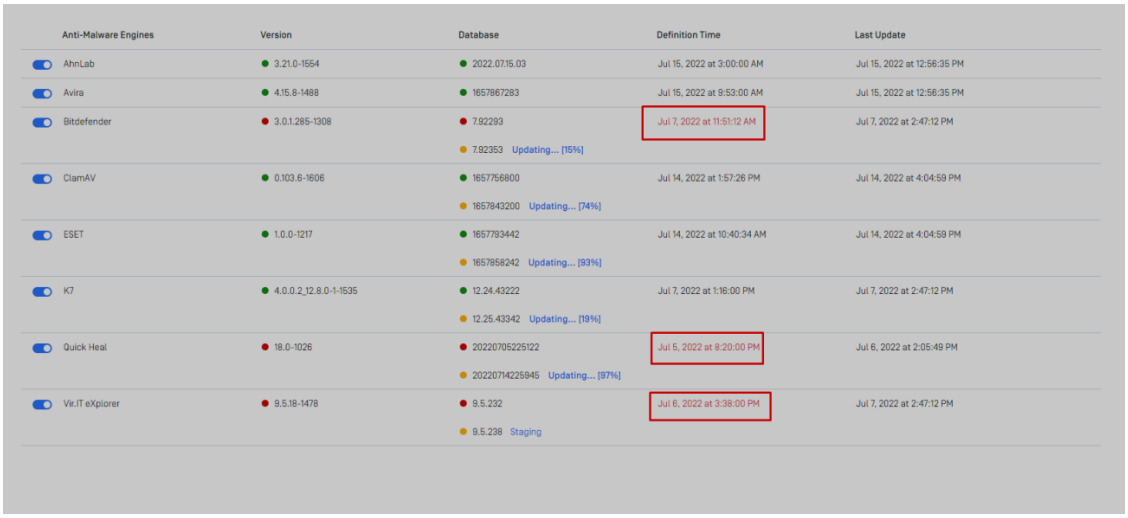
How do I check the update status of MetaDefender Core's licensed AV scan engines?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

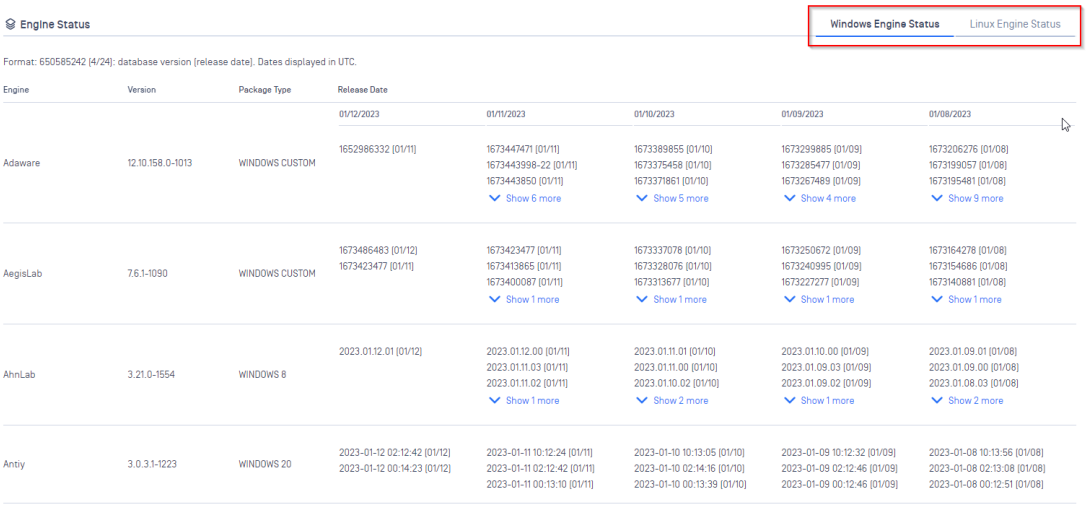
To check whether all of your licensed scan engines are downloaded and up-to-date, please go to your **MetaDefender Core Management Console>Inventory>Modules>Metascan** section, where you will see the current activation and update status of all engines in your deployment, along with associated details.





If certain engines appear to be out-of-date, as highlighted in the screenshot above, please compare these with the current engine update statuses from OPSWAT MetaDefender Core's licensed AV scan engines online here.

The webpage linked above lists, in real time, the latest updates OPSWAT received from each of our licensed MetaScan engine vendors. On this page, customers can also select the Operating System of the MetaDefender Core they are using, for a clearer view of the associated engines, as highlighted in the image below.



You can also confirm engine updates in MetaDefender Core by checking specific log entries:

1.Locate the Log Files

Navigate to the following directory on the MetaDefender Core server:

```
C:\Program Files\OPSWAT\MetaDefender Core\data\log\rotate\core
```

Inside this directory, you'll find subfolders for each engine. Select the folder corresponding to the engine you want to check. **Example:** For the FileType engine, open the folder named `oms_filetype`.

2.Confirm the Engine is Running Properly

Look for a log entry similar to the following:

```
[INFO] YYYY.MM.DD hh:mm:ss.mmm: (engine_process) Connected and initialized successfully [msgid: 228]
```

Explanation: This message confirms that the engine has started, connected, and initialized successfully. This is your key indicator that the engine is up and running.

3.Verify Successful Engine Update

To confirm the engine was updated and is now running the new version, search for the following sequence of log entries:

a. Engine Shutdown (Old Version)

```
[INFO] YYYY.MM.DD hh:mm:ss.mmm: (engine_process) ==== Engine End ====, engineName='FileType', engineVersion='7.4.2-8214', dbVersion='7.4.3-8347' [msgid: 5589]
```

Explanation: This entry shows that the previous engine version `7.4.2-8214` was shut down.

b. Engine Startup (New Version)

```
[INFO] YYYY.MM.DD hh:mm:ss.mmm: (engine_process) ==== Engine Start ====, engineName='FileType', engineVersion='7.4.3-8347', dbVersion='7.4.3-8347', pid='6400' [msgid: 5585]
```

Explanation: This log confirms that the new engine version `7.4.3-8347` has started successfully.

c. Logging Reconfiguration

```
[INFO] YYYY.MM.DD hh:mm:ss.mmm: (engine_process) Swap log config done, filePath='E:/OPSWAT/MetaDefenderCore/data/logs/oms_filetype/oms_filetype.log', engineName='FileType', engineVersion='7.4.3-8347', dbVersion='7.4.3-8347' [msgid: 5586]
```

Explanation: This confirms that the system updated the logging configuration to reflect the new engine version.

Support:

If you have any difficulty **Updating or Checking the Update Status of MetaDefender Core's Licensed AV Scan Engines**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What is the frequency of signature/definition updates on MetaDefender Core?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

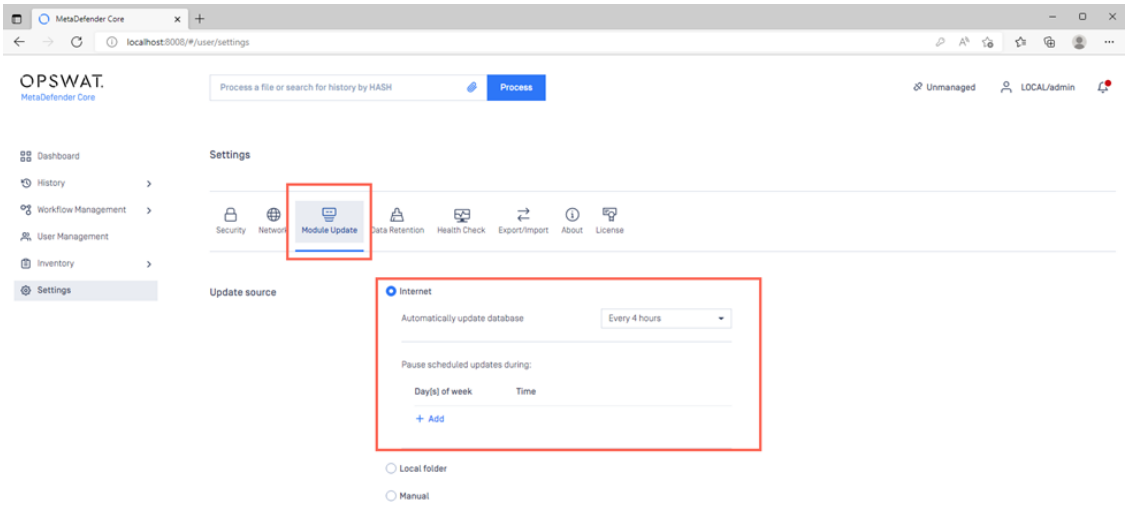
There are two parameters that determine the frequency of signature/definition updates:

- The frequency with which each antivirus vendor releases updates
  - Windows
  - Linux
- The configuration setting on your MetaDefender Core or MetaDefender Kiosk installation that specifies the time interval between system checks and applying new updates.

Please Note:

Most antivirus vendors release definitions at least once per day. Many have multiple daily releases. Some vendors release updates on weekends while others do not.

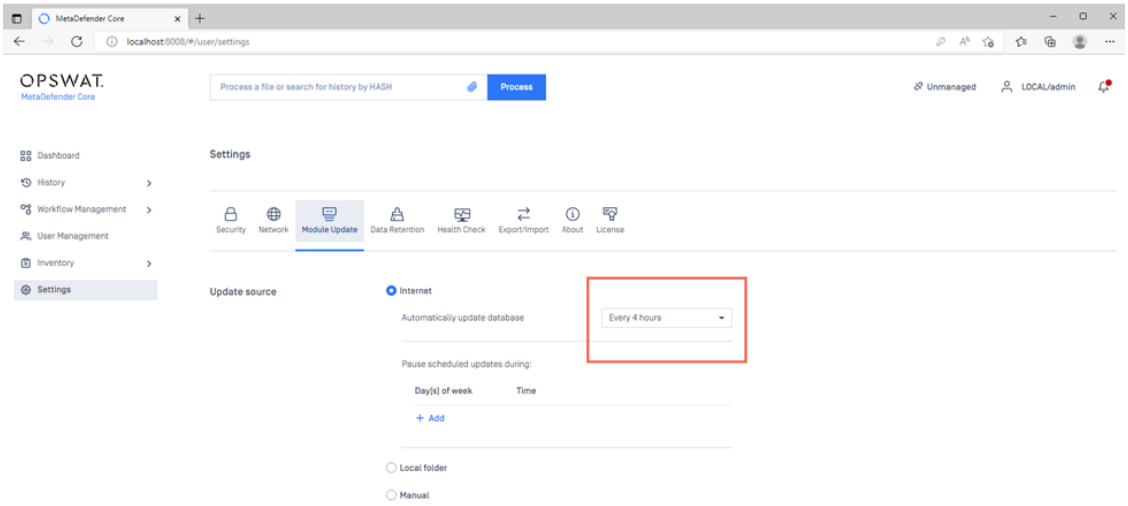
If you have configured OPSWAT's **Online Update** feature to auto-apply updates (i.e. via direct internet connection), you can then also configure the update interval to suit your preference.



When configured for **Automatic Updates**, MetaDefender Core will check for available scan engine (and signature/definition database) updates on a regular basis. To learn more about engine and database update checks, visit Regular Maintenance.

Please Note:

The default interval setting for a new installation of MetaDefender Core or MetaDefender Kiosk is every 4 hours (once per day).



If you have toggled settings to **Manual Update** (aka **Offline Updates**) the frequency is controlled by how often you download and apply the offline update package from OPSWAT.

Pro Tip:

Most of MetaDefender Core's included scan engines (and their signature/definition databases) can be configured to either **Internet** (**Automatic**), **Local** (**Automatic**) or **Manual** updates:

- To learn about configuring updates for MetaDefender Core V5, visit Engines Update Configuration.

For engines where update settings cannot be toggled, updates will be incorporated into new MetaDefender Core version releases.

Support:

For further guidance regarding **Scan Engine Signature/Definition Updates**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Why does the Deep CDR engine keep failing to update?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Please Note:

Please ensure that **Microsoft Visual C++ 2017 Redistributable Package** is installed.

If this third-party requirement is not met, it could result in Deep CDR's failure to activate, update and execute.

Linux dependencies:

CentOS: libgomp, ncurses-compat-libs

RedHat 7.x: libgomp, ncurses

RedHat 8.x: libgomp, ncurses-compat-libs, libnsl

Ubuntu/Debian: libgomp1, libncurses5

libgdiplus 6.0.5 or above (only need if you have file type conversion from documents to images such as DOCX to JPG, ...)

Another factor that could result in Deep CDR's failure (and other AV engines) to update is **inadequate network bandwidth**.

In an online environment where MetaDefender Core is configured to download engine update packages directly from an OPSWAT update server source, using a gigabit network speed (125MB/sec) is highly recommended to avoid potential network related issues during file downloads.

At the very least, the requirement is a network speed of 100Mbps (12.5MB/sec).

Support:

If you have followed the instructions above but **Your Deep CDR Engine Still Keeps Failing To Update**, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

What Happens After the Discontinuation of Kaspersky Support for Windows and Linux?

OPSWAT remains committed to providing the most comprehensive and cutting-edge threat detection capabilities within MetaDefender Core. We continuously evaluate and optimize our engine portfolio in Metascan to ensure optimal coverage and performance for our valued users.

What is changing?

Metadefender Core:

- Windows 20 Engines: Kaspersky will be replaced with Xvirus
- Windows 16 Engines: Kaspersky will be replaced with Comodo
- No immediate replacements are available for: Linux and Windows Max package, Custom 12 Engines, and Custom 8 Engines
- All bundle products that include Kaspersky
  - Metadefender Drive Enterprise Package: Kaspersky will be replaced with ESET
  - Metadefender Managed File Transfer:
    - 20 Engines: Kaspersky will be replaced with Xvirus
    - 16 Engines: Kaspersky will be replaced with Comodo
  - Metadefender Core for Kiosk:
    - 20 Engines: Kaspersky will be replaced with Xvirus
    - 16 Engines: Kaspersky will be replaced with Comodo
    - 4 Engines: Kaspersky will be replaced with ESET
  - Email Security:
    - Kaspersky will be replaced with Xvirus

Metadefender Cloud:

- Commercial, Professional and Enterprise: No Immediate replacement

Action Required

Choice of Engine Replacement:

- If you are using one of the packages mentioned above, you can request your replacement engine before April 1, 2024. If you do not take any action, Kaspersky will be automatically replaced by the replacement engines on April 1, 2024.

- If you have Kaspersky as a custom engine in your bundle, please contact your OPSWAT sales representative or our dedicated Support team for personalized guidance and assistance. You may choose any engine from the custom engine list to replace Kaspersky in your custom engine package.

Activating new license

- **Online deployments:** To receive the latest updates regarding this change, please reactivate your Metadefender Core license by navigating to 'Settings' > 'License', then select 'Deactivate' followed by 'Activate'.
- **Offline deployments:** To receive the latest update regarding this change, please contact your OPSWAT sales representative or our dedicated Support team to receive the new license key and activate the license.

What is the Timeline?

Effective January 1, 2024, we have ceased offering the Kaspersky anti-malware engine for both Windows and Linux platforms and have also stopped including it in our bundles. We will maintain support for existing customers until April 1, 2024. After this date, Kaspersky will be automatically replaced with the aforementioned engine replacement. If you wish to receive the replacement earlier, please contact your sales representative or OPSWAT Support.

What is the Next Step?

OPSWAT is committed to providing top-tier threat detection solutions. We will keep assessing and incorporating leading anti-malware engines into Metadefender Core to boost its Multiscanning capabilities and safeguard your critical infrastructure.

Further to Kaspersky Discontinuation

We will be removing the Antiy anti-malware from Windows 20 Engine packages in MetaDefender Core and replacing it with LIONIC anti-malware. This change is scheduled to occur starting April 1st, 2024. The update will be carried out automatically. During this transition, the MetaDefender Core UI may display an "unlicensed" label next to the Antiy Engine. Please disregard this message, as the Antiy engine will no longer be included in the Windows package 20, having been replaced by LIONIC.

Please Note:

OPSWAT will no longer provide any support or information regarding Kaspersky.

Support

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

How to Troubleshoot an Error related to Connection to Database Failing after an Upgrade to v5.11.0?

Check Your Version:

This article applies to all MetaDefender V5 releases deployed on Windows and Linux systems.

This knowledge base article addresses concerns regarding a database error encountered after upgrading to v5.11.0.

When this error occurs, the MD Core UI displays a database configuration error message:

“There’s a problem when connecting to the database. Please contact your administrator.”

The Core fails to connect to the database. The PostgreSQL logs indicate an issue with password authentication:

DETAIL: User "postgres" does not have a valid SCRAM secret.

Note

This issue does not affect all cases when upgrading to v5.11.0.

It specifically occurs when the customer has previously upgraded from version 5.5.1 or older to version 5.6.0 or newer, and then again to version 5.11.0.

Remediation Steps

In the short term, the engineering team has created a workaround tool to address this issue. See the steps below for both Windows and Linux platforms.

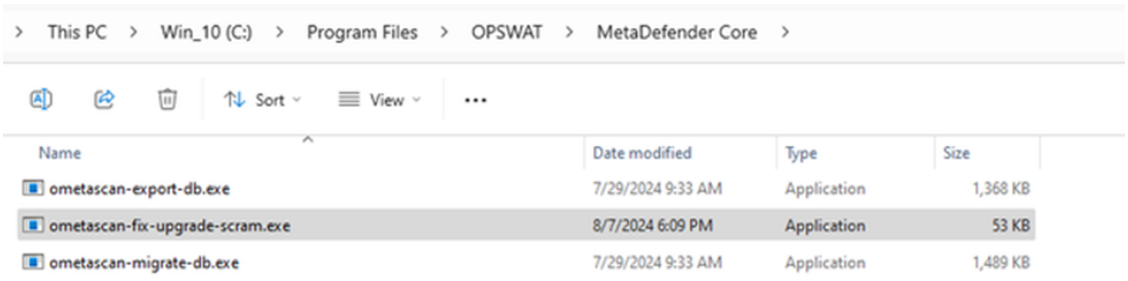
In the long term, a fix will be included in the v5.11.1 release.

Steps to Run the Tool (Password to unzip: **opswat123**)

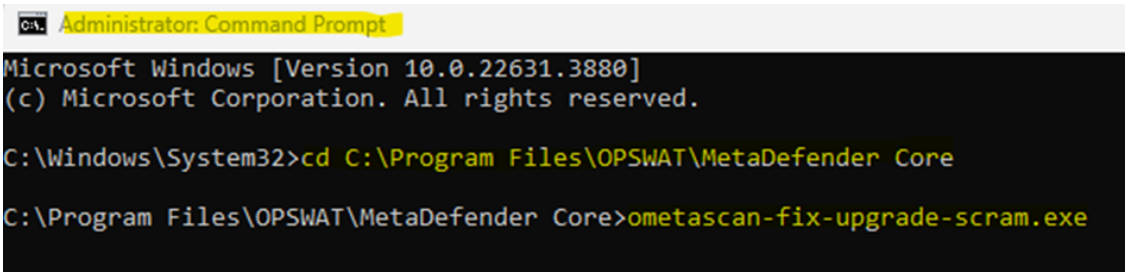
Windows:

Link to download: ometascan-fix-upgrade-scram-Windows.zip (SHA1: F6F0C8444CB99C903B52E0435A088EC398A5483E)

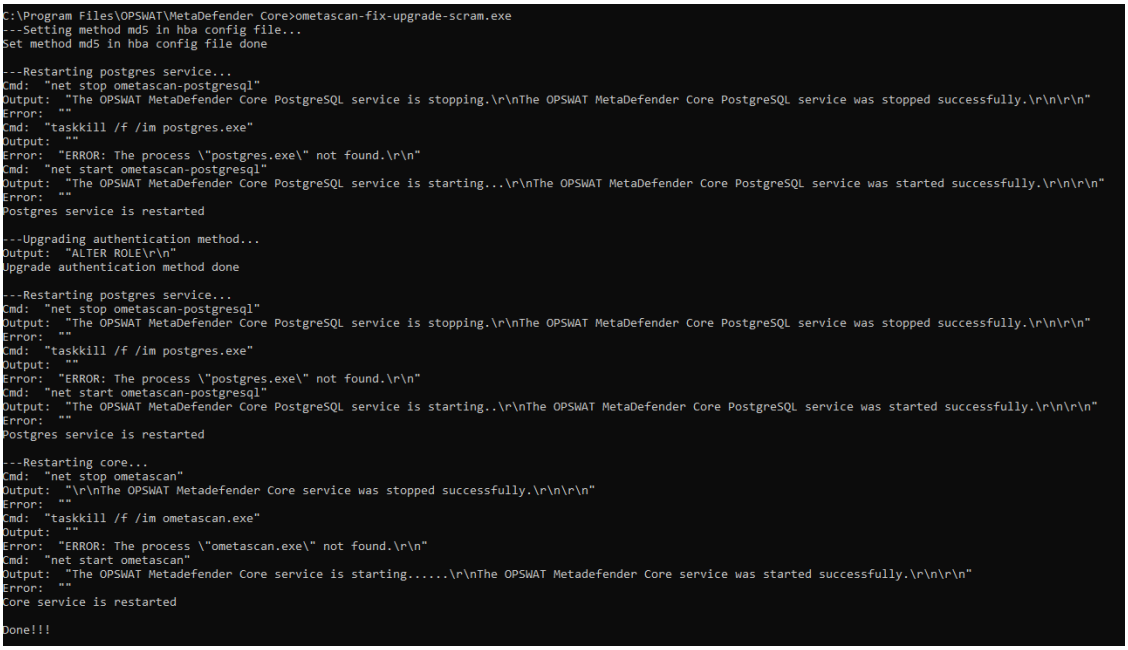
1. Extract and copy the tool to Core’s installation folder on Windows.



2. Open Command Prompt as Administrator, navigate to Core’s installation directory, and run the tool.



3. Check the output; it should look like the image below.



> **Note**
>
> If the script didn't run successfully, please try restarting the system and then run the script again.

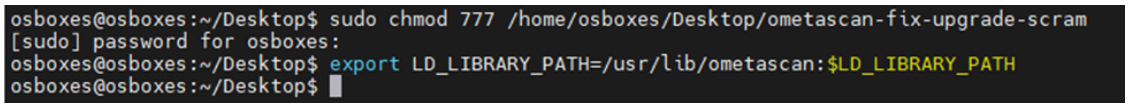
Linux:

Link to download: ometascan-fix-upgrade-scram-Linux.zip (SHA1: F8B0AA96A844EC9C455125007D673CD5F5B7C410)

1. Run the below commands:

Code · bash

```
chmod 777 /path_to_the_tool/ometascan-fix-upgrade-scram
export LD_LIBRARY_PATH=/usr/lib/ometascan:$LD_LIBRARY_PATH
```



2. Run the tool with *sudo*

Code · bash

```
sudo /path_to_the_tool/ometascan-fix-upgrade-scram
```

3. Check the output, it should looks as below

```
osboxes@osboxes:~/Desktop$ sudo ./ometascan-fix-upgrade-scram
---Setting method md5 in hba config file...
Set method md5 in hba config file done

---Restarting postgres service...
Cmd:  "systemctl stop ometascan-pg"
Output:  ""
Error:  ""
Cmd:  "killall -s KILL postgres"
Output:  ""
Error:  "postgres: no process found\n"
Cmd:  "systemctl start ometascan-pg"
Output:  ""
Error:  ""
Postgres service is restarted

---Upgrading authentication method...
Output:  "ALTER ROLE\n"
Upgrade authentication method done

---Restarting postgres service...
Cmd:  "systemctl stop ometascan-pg"
Output:  ""
Error:  ""
Cmd:  "killall -s KILL postgres"
Output:  ""
Error:  "postgres: no process found\n"
Cmd:  "systemctl start ometascan-pg"
Output:  ""
Error:  ""
Postgres service is restarted

---Restarting core...
Cmd:  "systemctl stop ometascan"
Output:  ""
Error:  ""
Cmd:  "killall -s KILL ometascan"
Output:  ""
Error:  "ometascan: no process found\n"
Cmd:  "systemctl start ometascan"
Output:  ""
Error:  ""
Core service is restarted

Done!!!
```

After running the tool, open the Core Management Console and try to log in.

If the issue persists, please capture a screenshot of step #3 and send it to Support for further analysis.

Support:

If **Further Assistance** is required, please follow the instructions on How to Create Support Package With Bundle Tools?, then proceed to create support case or chatting with our support engineer.

Does a CVE affect MetaDefender Core?

In today's rapidly evolving digital landscape, understanding the impact of security vulnerabilities is more crucial than ever. The Common Vulnerabilities and Exposures (CVE) list is an invaluable resource for identifying potential security risks associated with software products. This article provides a comprehensive overview of various CVEs, helping you determine whether MetaDefender Core is affected. By staying informed about these vulnerabilities, you can take proactive measures to protect your systems and data from potential threats.

CVE	Summary
CVE-2024-32113	CVE-2024-32113 is the vulnerability of Apache OFBiz which we don't use in MetaDefender Core
CVE-2024-38856	CVE-2024-32113 is the vulnerability of Apache OFBiz which we don't use in MetaDefender Core
CVE-2024-4367	CVE-2024-4367 does not impact MetaDefender Core
CVE-2024-34342	CVE-2024-34342 does not impact MetaDefender Core
CVE-2024-40725	MetaDefender Core does not use Apache HTTP Server and is not affected by CVE-2024-40725
CVE-2024-40898	MetaDefender Core does not use Apache HTTP Server and is not affected by CVE-2024-40898
CVE-2024-27348	MetaDefender Core does not utilize Apache HugeGraph-Server in its architecture. As a result, it is not susceptible to the CVE-2024-27348 vulnerability
CVE-2018-25103	MetaDefender Core does not use lighttpd, it is not affected by the vulnerabilities identified in CVE-2018-25103
CVE-2024-4603	MetaDefender Core does not use either

CVE	Summary
	EVP_PKEY_param_check() or EVP_PKEY_public_check() hence it's not affected by CVE-2024-4603
CVE-2023-46589	MetaDefender Coredoes not use Tomcat, so the product is not impacted by the CVE
CVE-2023-50164	MetaDefender Coreis not built on Apache Struts which is affected by this CVE
CVE-2022-21724	CVE-2022-21724 is not used by MetaDefender Core to make connections to Postgres, so the product is not impacted by the CVE
CVE 2018-2894	MetaDefender Core doesn't use Java, so this CVE doesn't impact the product
CVE-2024-38819	MetaDefender Core doesn't use this framework, so this CVE doesn't impact the product
CVE-2024-7348	Since MetaDefender Core 5.11.1, we upgraded Postgres to a newer version to address this CVE
CVE-2024-3566	Although MetaDefender Core executes another programs (Nginx, engine processes, engine installation scripts), we do not pass any user-input arguments to these programs. We are not affected by this CVE.
CVE-2024-27980	MetaDefender Core does not use Node.js so it's not affected
CVE-2024-24576	MetaDefender Core does not use Rust in its code so it's not affected
CVE-2024-10979	MetaDefender Core does not use PL/Perl and PL/Python so it's not affected
CVE-2018-15133	MetaDefender Core does not use PHP and Laravel so it's not affected
CVE-2023-40581	CVE-2023-40581 affects **yt-dlp**, it does not impact

CVE	Summary
	MetaDefender Core functionality
CVE-2024-1874	MetaDefender Core does not use PHP so it's not affected
CVE-2025-0411	This vulnerability only affects the GUI version of 7-Zip. Exploitation requires a user to manually open a malicious archive in 7-Zip File Manager and execute a file inside. Since MetaDefender Core's Archive Engine does not utilize the GUI version of 7-Zip, it is not affected by this CVE
CVE-2024-4577	MetaDefender Core does not use PHP-CGI so it's not affected
CVE-2019-9082	MetaDefender Core does not use ThinkPHP so it's not affected
CVE-2024-21235 CVE-2024-21210 CVE-2024-21217 CVE-2024-21208 CVE-2024-21147 CVE-2024-21144 CVE-2024-21145 CVE-2024-21138 CVE-2024-21131 CVE-2024-21140	MetaDefender Core does not rely on Amazon Corretto JRE or Java as a dependency; therefore, these vulnerabilities do not affect the product. If the Sandbox engine is installed and relies on Amazon Corretto JRE, we strongly recommend upgrading to Amazon Corretto 17 to ensure system security and compatibility. Please follow this article for more information : How to upgrade Amazon Corretto JRE?
CVE-2025-21298	MetaDefender Core itself is not affected by this CVE, but we cannot confirm the same for the engines running inside MetaDefender Core. According to National Vulnerability Database, this CVE affects Windows OS. It is recommended that the users apply the latest security updates to patch this CVE if the user is on the following OS: • Windows 10 • Windows 11 23H2 • Windows Server 2016, 2019, 2022

Support:

If **Further Assistance** is required, please proceed to log a support case or chat with our support engineer.

End-of-Life Support for CentOS 7 and RedHat 7 in MetaDefender Core and Engines

In line with industry standards and our commitment to providing secure and efficient solutions, OPSWAT will discontinue support for CentOS 7 and RedHat 7 in MetaDefender Core and its associated engines. This article details the implications of this change, outlines the operating systems that will be supported moving forward, and provides essential guidance for users who may need to continue using CentOS 7 or RedHat 7 temporarily.

CentOS 7 and RedHat 7 has been a reliable platform for many users, but it has now reached its end-of-life phase. Supporting an outdated operating system increases security risks and prevents access to new features and improvements. Therefore, MetaDefender Core and its engines will no longer support CentOS 7 and RedHat 7 after December 2024.

Supported Operating Systems Moving Forward

To ensure optimal performance and security, we encourage all users to transition from CentOS 7 or RedHat 7 to the following supported operating systems for MetaDefender Core and its engines:

- Red Hat Enterprise Linux (RHEL) 8, 9
- Debian 11
- Ubuntu 20.04, 22.04
- Amazon Linux 2 is supported via Amazon Machine Image (AMI)

Important Notice

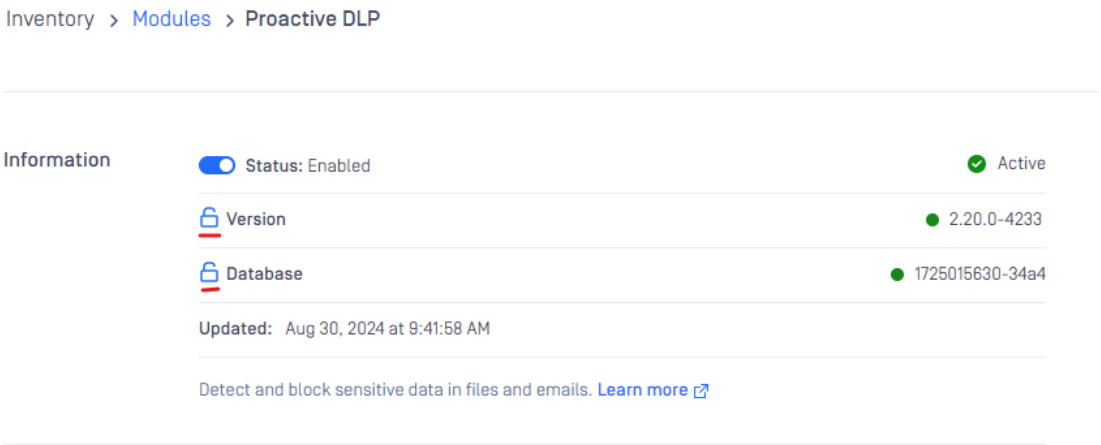
If you do not transition to a supported operating system, you will no longer receive the latest updates and features from OPSWAT MetaDefender Core and its engines. This could significantly impact the functionality and security of your system.

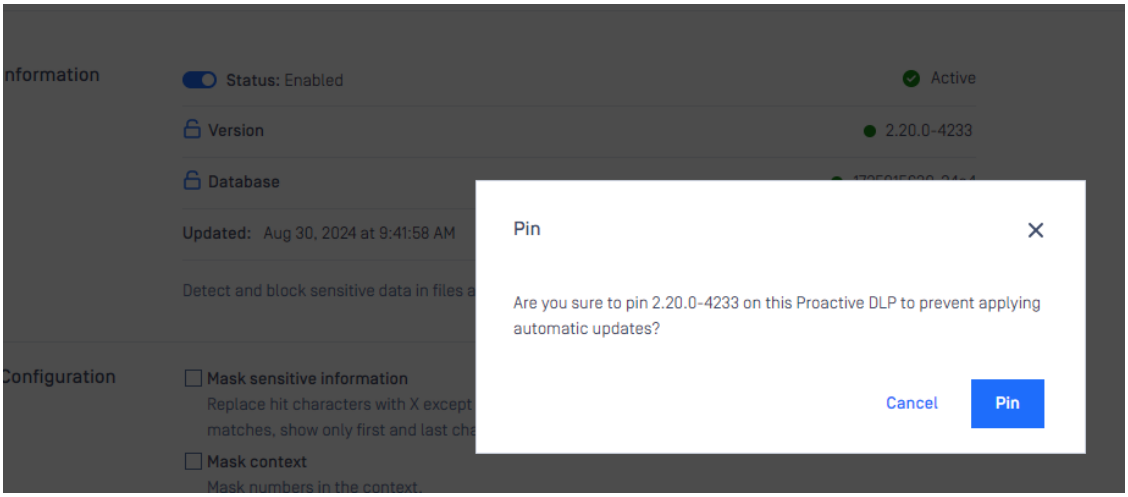
Continuing with CentOS 7/RedHat 7 - Pinning Your Engine [Not Recommended]

We understand that some users may need additional time to transition from CentOS 7/RedHat 7. If you choose to continue using these engines temporarily, it is crucial to take the following steps **before December 2024**:

- **Pin Your Engine Version**: MetaDefender Core engines are configured by default to receive automatic updates. While this ensures systems stay up-to-date with the latest features, it can cause issues on unsupported operating systems like CentOS 7 after it reaches End-of-Life. By pinning your engine version, you can prevent automatic updates that may not be compatible with these operating systems, thus maintaining system stability during your transition period.
  - How to Pin the Engine:
    - Access your MetaDefender Core configuration settings.

- Go to the Inventory section and click on the Modules.
- For each available engine, click on it and pin the current version and database to prevent automatic updates.





- **Plan Your Migration**: Begin planning your transition to a supported operating system as soon as possible. Delaying this process can expose your system to security vulnerabilities and compliance risks.

Conclusion

The End-of-Life support for CentOS 7 and RedHat 7 in MetaDefender Core and its engines is a necessary step towards providing a more secure and efficient experience for our users. We strongly recommend upgrading to one of the supported operating systems listed above to continue receiving updates and support. If you must remain on CentOS 7 or RedHat 7 temporarily, ensure you pin your engine version before December 2024 and expedite your migration plan to a supported platform.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

End of Life Support for Ubuntu 20.04 and Amazon Linux 2 in MetaDefender Core

As of April 2025, Ubuntu 20.04 LTS has reached its End of Life (EOL), meaning it no longer receives security updates, patches, or support from Canonical. Amazon Linux 2 support for MetaDefender Core and its engines will also be discontinued after June 30, 2025, aligning with the initial EOL date set by MetaDefender Core, despite Amazon Linux 2's extended EOL to 2026. This article outlines the implications, risks, and recommendations for continuing to use these operating systems with MetaDefender Core.

Background

MetaDefender Core relies on the underlying operating system for security updates and stability. With Ubuntu 20.04 and Amazon Linux 2 reaching EOL, Canonical and Amazon Web Services (AWS) have ceased providing updates, leaving systems vulnerable to new security threats. To ensure the security and reliability of MetaDefender Core deployments, OPSWAT strongly recommends upgrading to supported operating systems.

Impact on MetaDefender Core

Running MetaDefender Core on Ubuntu 20.04 or Amazon Linux 2 after their EOL introduces significant risks, including:

- **Security Vulnerabilities**: Without security patches, these operating systems are exposed to new exploits and threats.
- **Compatibility Issues**: MetaDefender Core updates and new features may not function correctly on unsupported platforms.
- **Compliance Risks**: Many regulatory standards require systems to run on actively supported operating systems.
- **Lack of Support**: OPSWAT technical support may be limited or unavailable for issues related to EOL operating systems.

Recommendations

To maintain a secure and supported MetaDefender Core environment, OPSWAT recommends the following:

1. **Upgrade to a Supported Operating System** Transition to a supported operating system, such as Ubuntu 22.04 LTS, Ubuntu 24.04 LTS, or Amazon Linux 2023, which receive active security updates and are fully compatible with the latest MetaDefender Core versions. Refer to the MetaDefender Core System Requirements for a complete list of supported platforms.
2. **Plan Your Migration**
   - **Assess Your Environment**: Identify all systems running Ubuntu 20.04 or Amazon Linux 2.
   - **Test Compatibility**: Deploy MetaDefender Core on the new operating system in a staging environment to ensure compatibility with your workflows.
   - **Execute Migration**: Follow OPSWAT's migration guides to transfer configurations, policies, and data to the new platform.

- **Validate Post-Migration**: Verify that MetaDefender Core operates as expected after the upgrade.

3. **Extended Security Maintenance (Optional)** For Ubuntu 20.04, Canonical offers an Extended Security Maintenance (ESM) program, which provides additional security updates for a fee. This may serve as a temporary solution for organizations unable to migrate immediately. However, ESM does not guarantee full compatibility with MetaDefender Core, and OPSWAT recommends upgrading to a fully supported OS as soon as possible. Amazon Linux 2 does not offer an equivalent extended support program, making migration to a supported OS even more critical.

Continuing with Ubuntu 20.04/Amazon Linux 2 - Pinning Your Engine (Not Recommended)

If migration is not immediately feasible, you may choose to pin your MetaDefender Core engine to a specific version compatible with Ubuntu 20.04 or Amazon Linux 2. This approach, however, is **not recommended** due to the following risks:

- **Security Exposure**: The operating system will remain unpatched, increasing the risk of exploitation.
- **Limited Functionality**: Pinned engines do not receive new features, bug fixes, or performance improvements.
- **Unsupported Configurations**: OPSWAT may not provide support for issues arising from EOL operating systems, even with pinned engines.
- **Compliance Violations**: Running an EOL operating system may violate organizational or regulatory compliance requirements.

To pin your engine:

1. Access the MetaDefender Core management console.
2. Navigate to **Settings > Update Settings**.
3. Select the option to pin the engine to the current version.
4. Save the configuration.

Note: Pinning should only be a temporary measure while planning your migration to a supported operating system.

Conclusion

The EOL of Ubuntu 20.04 and Amazon Linux 2 poses significant risks to the security and functionality of MetaDefender Core deployments. OPSWAT strongly urges users to upgrade to a supported operating system, such as Ubuntu 22.04 LTS or Amazon Linux 2023, to ensure continued security, compatibility, and support. For assistance with migration or additional guidance, contact OPSWAT Support or refer to the MetaDefender Core Documentation.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How will End of Life Support for Windows Server 2016 and Windows 10 versions before 21H2 in MetaDefender Core affect my deployments?

Overview

Microsoft has announced the end-of-life (EOL) for Windows Server 2016 and Windows 10 versions prior to 21H2, marking the end of official support and security updates for these platforms. As a result, OPSWAT will no longer support MetaDefender Core installations on these operating systems starting October 2025. To maintain platform compatibility, security, and compliance, organizations should begin planning a transition to a supported operating system well ahead of this date.

This article outlines the impact of these EOL dates on OPSWAT MetaDefender Core deployments, including supported operating systems, recommended actions, and upgrade paths to ensure continued security and compliance.

Affected Operating Systems

- **Windows Server 2016**: Mainstream support ended on January 11, 2022, with extended support ending on **January 12, 2027**. After this date, Microsoft will no longer provide security updates, non-security updates, or technical support for Windows Server 2016.
- **Windows 10 Versions Prior to 21H2**: All Windows 10 versions before 21H2 (e.g., 20H2, 2004, 1909, etc.) have reached or will soon reach their EOL. For example:
  - Windows 10 20H2 reached EOL on **May 10, 2022** (Enterprise and Education editions on **May 9, 2023**).
  - Windows 10 21H1 reached EOL on **December 13, 2022**.
  - Other earlier versions (e.g., 1909, 2004) are already out of support.

After these EOL dates, Microsoft will no longer provide security patches or updates, increasing the risk of vulnerabilities and compatibility issues with MetaDefender Core.

Impact on MetaDefender Core

MetaDefender Core relies on the underlying operating system for stability, security, and compatibility with its advanced threat **protection features including MetaScan Multiscanning, Deep CDR, Proactive DLP, Adaptive Sandbox, and File-based Vulnerability Assessment.** Running MetaDefender Core on unsupported operating systems like Windows Server 2016 or Windows 10 versions prior to 21H2 may result in:

- **Security Risks**: Lack of security patches from Microsoft increases exposure to malware, zero-day attacks, and other cyber threats.
- **Compatibility Issues**: MetaDefender Core updates and new features may not function correctly on unsupported operating systems, leading to degraded performance or unexpected behavior.
- **Compliance Violations**: Failure to update operating systems can lead to regulatory and compliance violations (e.g., PCI-DSS, HIPAA).

Supported Operating Systems

To ensure optimal performance and security, OPSWAT recommends deploying MetaDefender Core on the following supported Windows operating systems:

- **Windows Server**: Windows Server 2019, Windows Server 2022, Windows Server 2025
- **Minimum System Requirements**:
  - CPU: 8 vCPUs
  - RAM: 8 GB (16 GB or more recommended)
  - Disk Space: 50 GB free disk space for installation and operation
  - .NET Framework: Version 4.8 or later

For the latest supported operating systems and system requirements, refer to the MetaDefender Core Documentation.

Recommended Actions

To maintain a secure and compliant MetaDefender Core deployment, OPSWAT strongly advises the following steps:

1. **Upgrade Operating Systems**:
   - For Windows Server 2016, migrate to **Windows Server 2019** or **Windows Server 2022 or 2025**. Microsoft provides upgrade paths via in-place upgrades or clean installations. Refer to Microsoft's Windows Server Upgrade Documentation for guidance.
   - For Windows 10 versions prior to 21H2, upgrade to **Windows 10 22H2** or **Windows 11** to receive ongoing security updates and support. Use the Windows Update feature or download the latest ISO from Microsoft's Download Center.

2. **Plan Ahead for EOL**:
   - Develop a migration plan well in advance of the EOL dates to avoid disruptions. Test MetaDefender Core on the target operating system in a staging environment before production deployment.
   - Update any scripts, automation, or integrations that rely on the current operating system to ensure compatibility with the new OS.

3. **Leverage MetaDefender Cloud Image Deployment Options**:
   - For AWS users, consider deploying MetaDefender Core using the latest AMIs (Amazon Machine Images) available on the AWS Marketplace, which support Windows Server 2019 and 2022. Refer to the How to deploy using AWS services.
   - Use CloudFormation scripts for streamlined deployment, which can reduce setup time to approximately 3–12 minutes depending on the deployment model (single or distributed).

4. **Contact OPSWAT Support**:

- If you need assistance with upgrading or migrating MetaDefender Core, contact the OPSWAT support team by **creating a support case or chatting with our support engineer**.
- For licensing inquiries or custom deployment options, reach out to sales-inquiry@opswat.com.

Upgrade Process

1. **Backup Configuration**: Before upgrading the operating system, back up your MetaDefender Core configuration, including scan engine settings, policies, and logs. Refer to the MetaDefender Core Operating Guide for backup instructions.
2. **Test Compatibility**: Install MetaDefender Core on a test instance running the target operating system (e.g., Windows Server 2022 or Windows 11) to verify compatibility with your workflows and integrations.
3. **Perform Upgrade**:
   - For Windows Server, follow Microsoft's in-place upgrade process or deploy a new instance with the updated OS and reinstall MetaDefender Core.
   - For Windows 10, use the Windows Update Assistant or a clean installation to upgrade to 22H2 or Windows 11.
4. **Validate Deployment**: After upgrading, verify that MetaDefender Core functions correctly, including file scanning, Deep CDR, and integration with services like Amazon S3 or SQS (if applicable).
5. **Monitor and Update**: Regularly check for MetaDefender Core updates and apply them to ensure compatibility with the latest security features and threat intelligence.

Additional Resources

- Microsoft Windows Server 2016 Lifecycle
- Microsoft Windows 10 Release Information
- MetaDefender Core Release Notes
- OPSWAT MetaDefender Core Documentation

Conclusion

The end-of-life for Windows Server 2016 and Windows 10 versions prior to 21H2 requires proactive planning to maintain a secure and compliant MetaDefender Core deployment. By upgrading to supported operating systems like Windows Server 2019/2022, organizations can ensure continued protection against malware, zero-day attacks, and data breaches. For further assistance, contact OPSWAT's support or sales teams to explore deployment options and licensing tailored to your needs.

Will MetaDefender Core Experience Downtime During Engine Updates?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

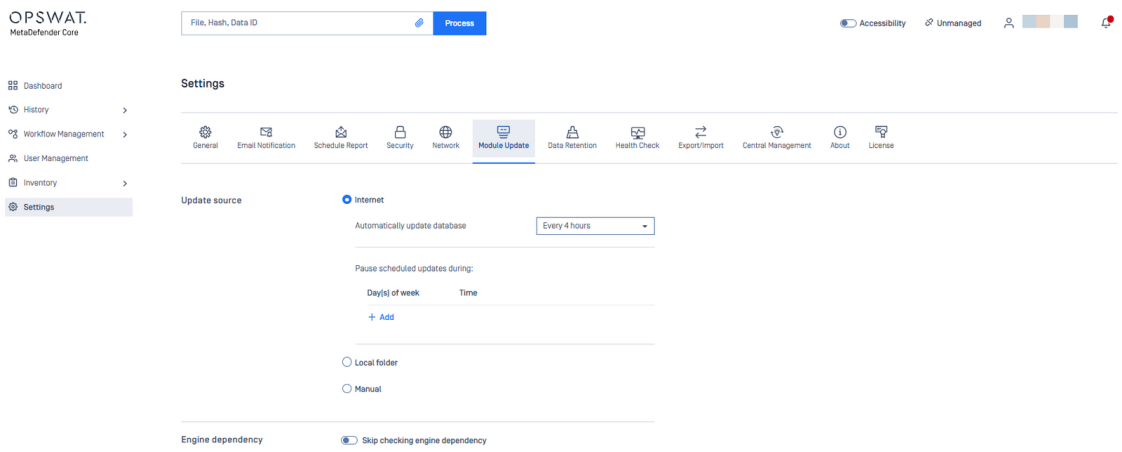
MetaDefender Core is built to ensure high availability and minimize system downtime, even during antivirus engine updates.

What Happens During an Engine Update? MetaDefender Core includes a built-in engine failover mechanism. When multiple scanning engines are in use, this ensures uninterrupted scanning: if one engine is temporarily disabled for an update, the system continues scanning using the remaining active engines. Once the update completes, the updated engine is automatically re-enabled.

Does the System Shut Down? No, MetaDefender Core does **not** shut down during engine updates. Scanning and file analysis continue without interruption, maintaining protection and system stability.

Key Takeaways:

- MetaDefender Core uses engine failover to ensure continuous operation when multiple engines are enabled.
- **Ongoing scans will continue using the current engine version and definitions without interruption. Any updated engine or database will be applied to new scan submissions as they become available.**
- **The update delay can be configured under Settings > Module Update. Locking the engine and database versions is available to accommodate your specific requirements.**



- It's recommended to update during low business hours to minimize interruption to daily operation.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

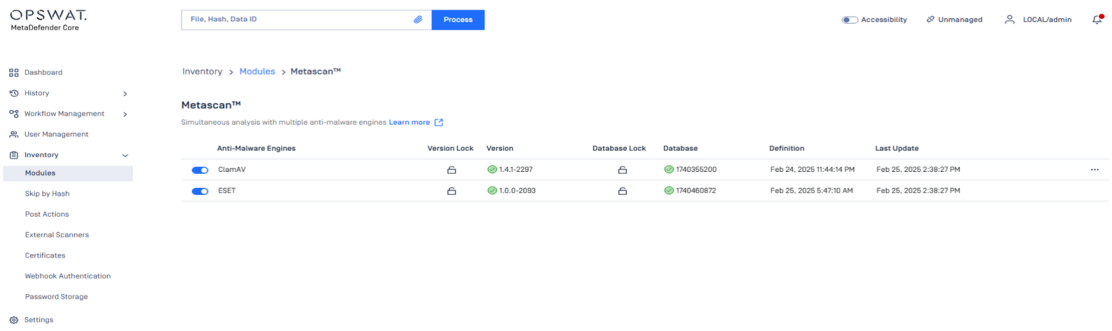
How to Update Antivirus Engines in MetaDefender Core Manually?

This KB's purpose is a POV of the user on how to download the Antivirus engine individually or a specific selection

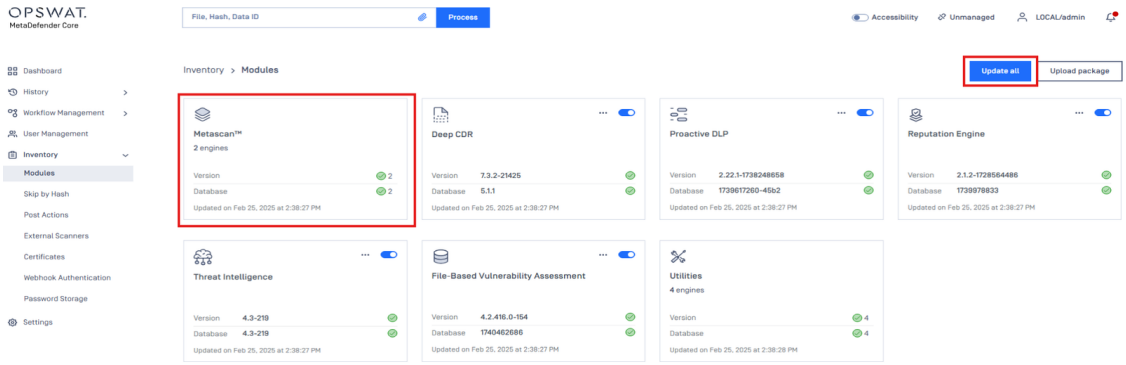
that the user wants.

Online download

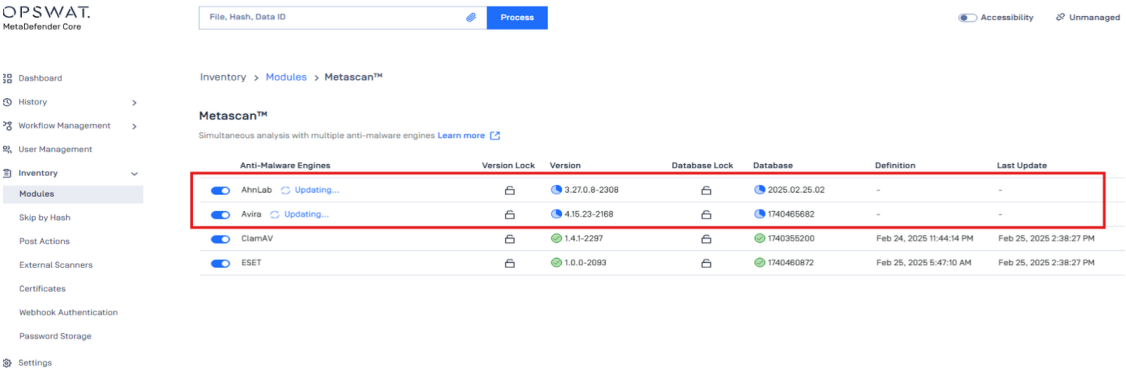
1. One of the issues that we commonly see is that the engine is unable to be uploaded, or a network error, which results in Metascan features missing engines. Specifically for this case, which is the **AhnLab** and **Avira**.



2. We will use the online download option, which is to click on the Upload all button to retrieve all the engines that were missing or unfinished downloads.

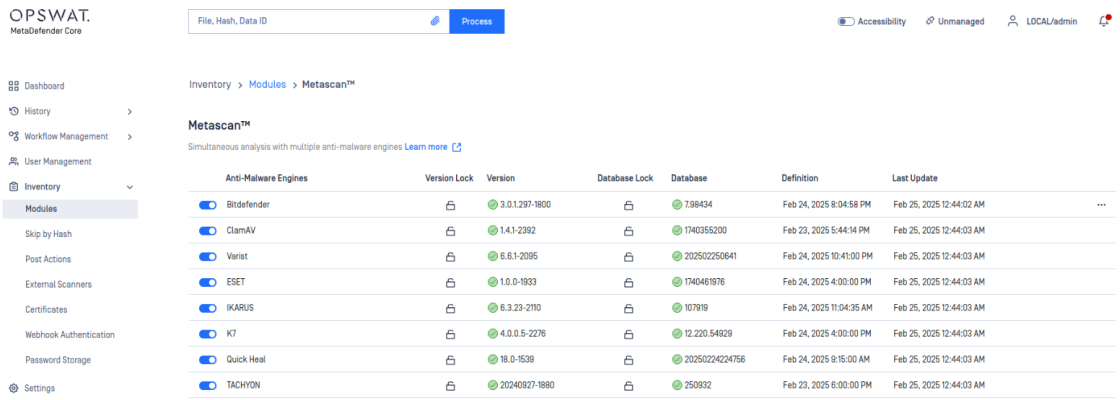


3. After hitting the button, MetaDefender Core will automatically find a suitable engine and download it to the Metascan tab.

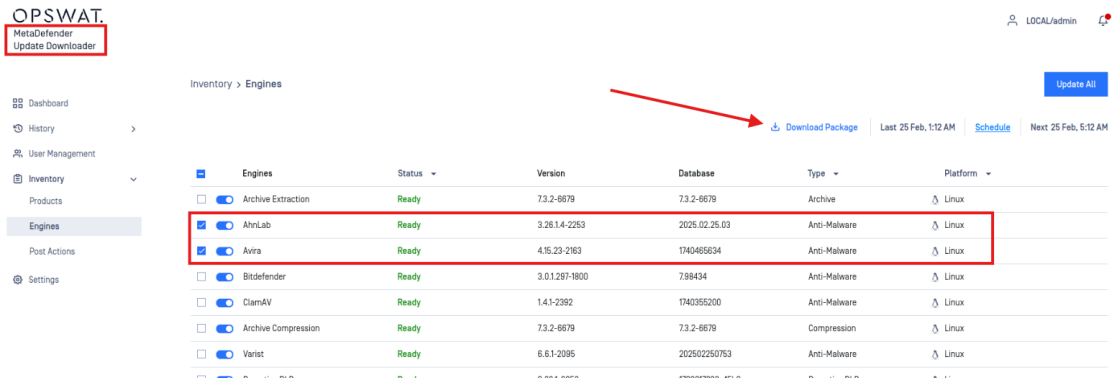


Offline download

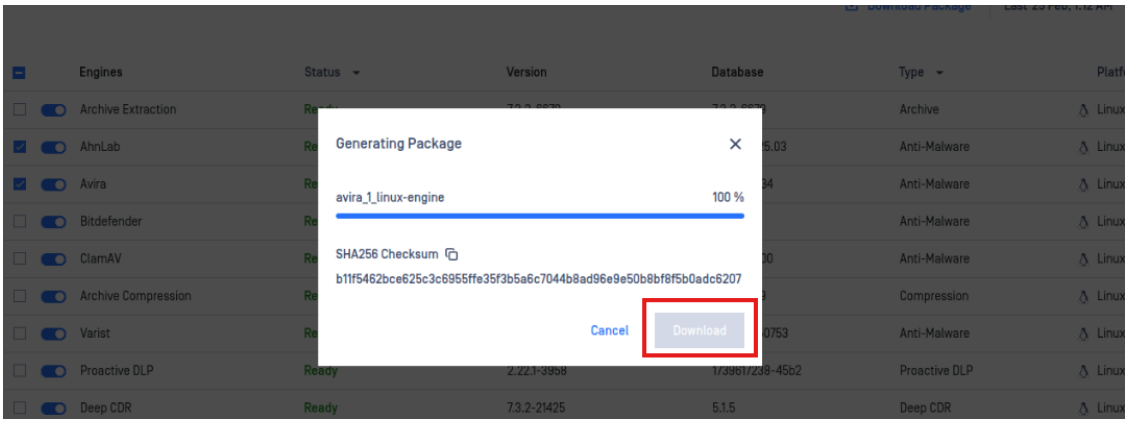
1. For the offline download, we will choose the Antivirus engine that is the same as the online downloader, which is **AhnLab** and **Avira**.



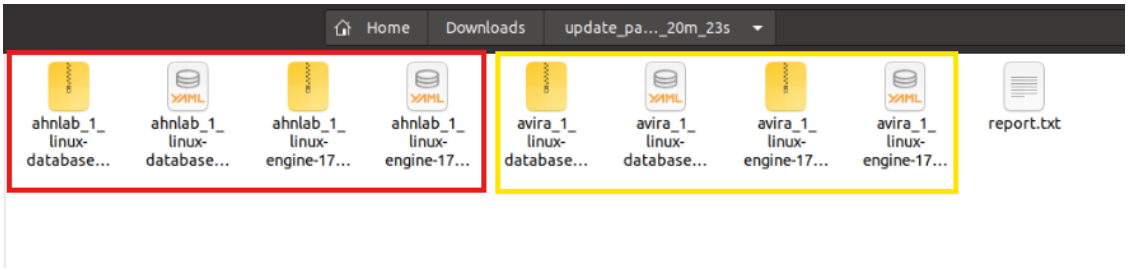
2. In this situation, we will utilize the tool MetaDefender Update Downloader. Please follow the link on how to download the tools here: How do I deploy MetaDefender Core to an offline Windows environment?



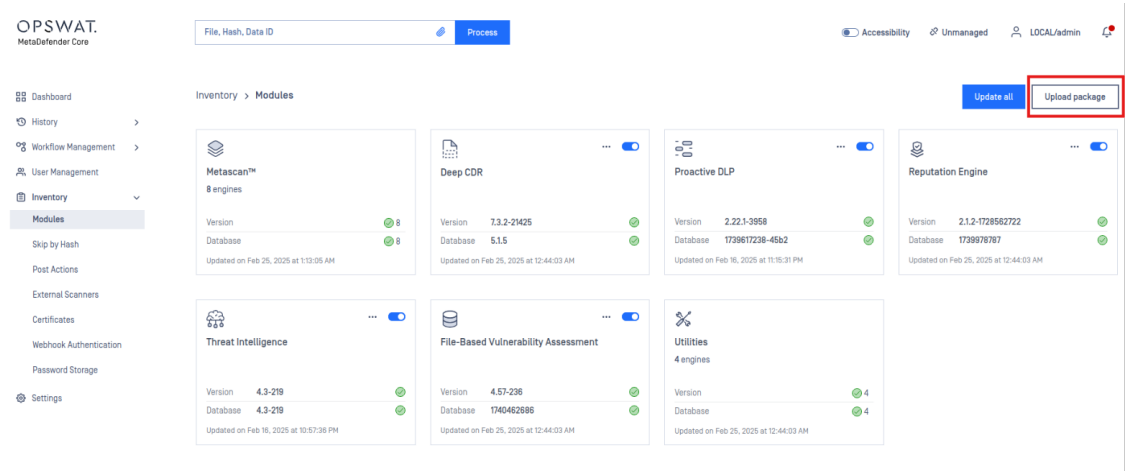
3. From this step, we choose the option that are needed from the tab and click on “Download Package” to download the zip that contains all the package we need for the Antivirus engines.

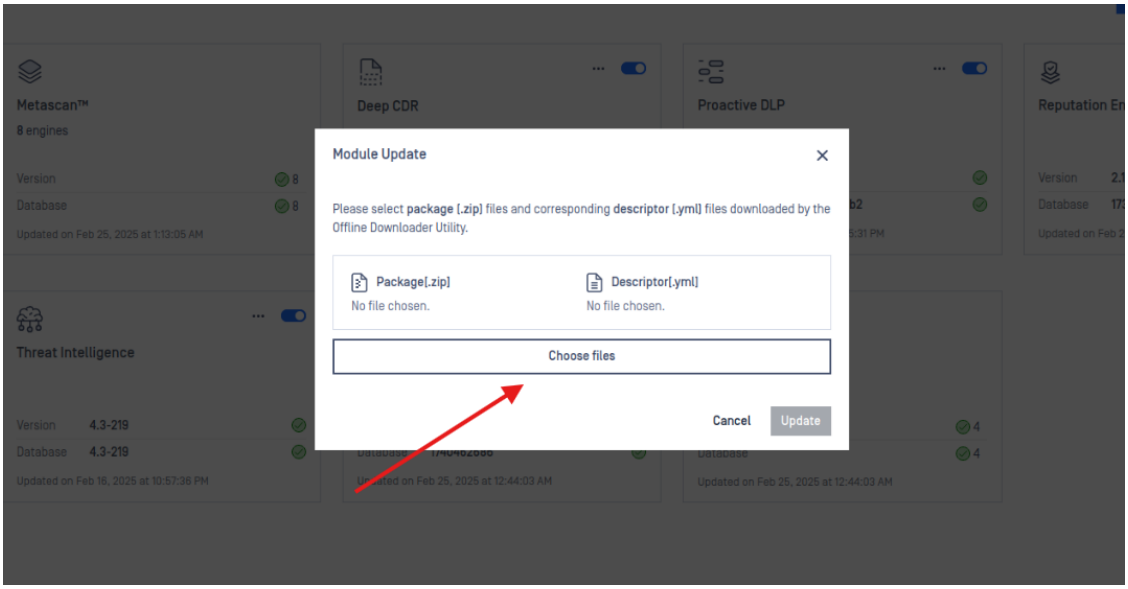


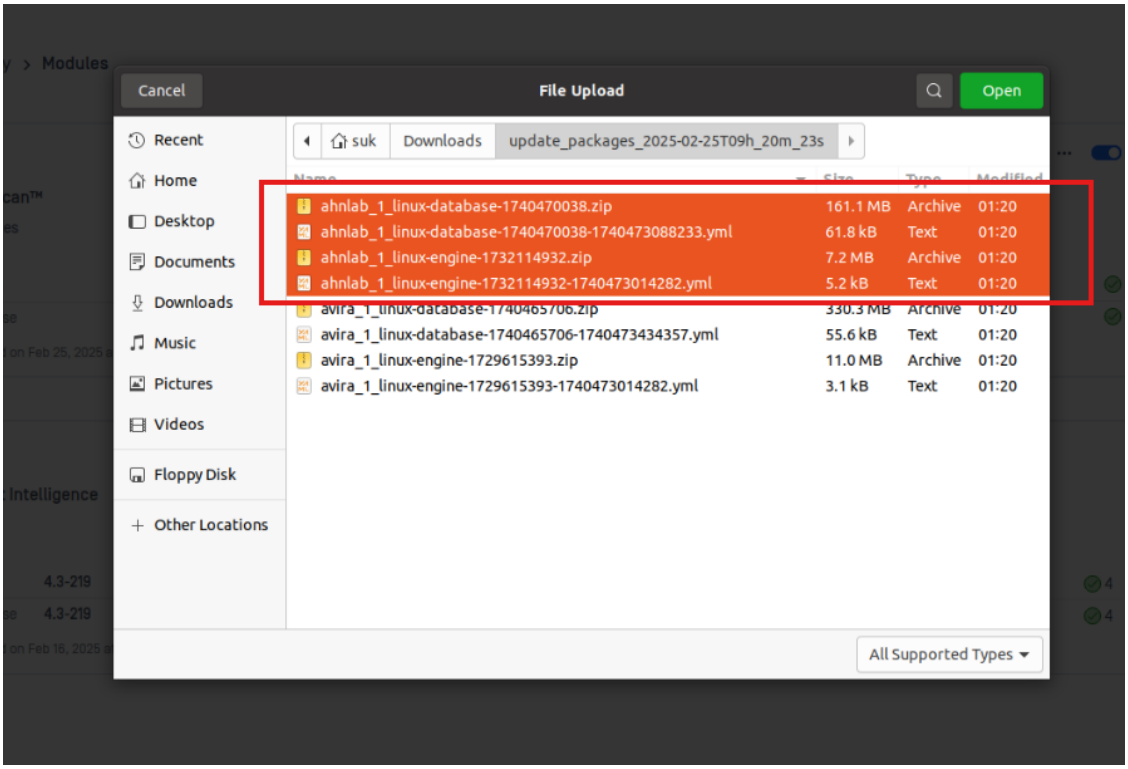
4. From the Zip that was downloaded, please extract it and prepare for uploading the packages to the MetaDefender Core. From the folder, the Engine and the databases of the engine can be seen from the folder.

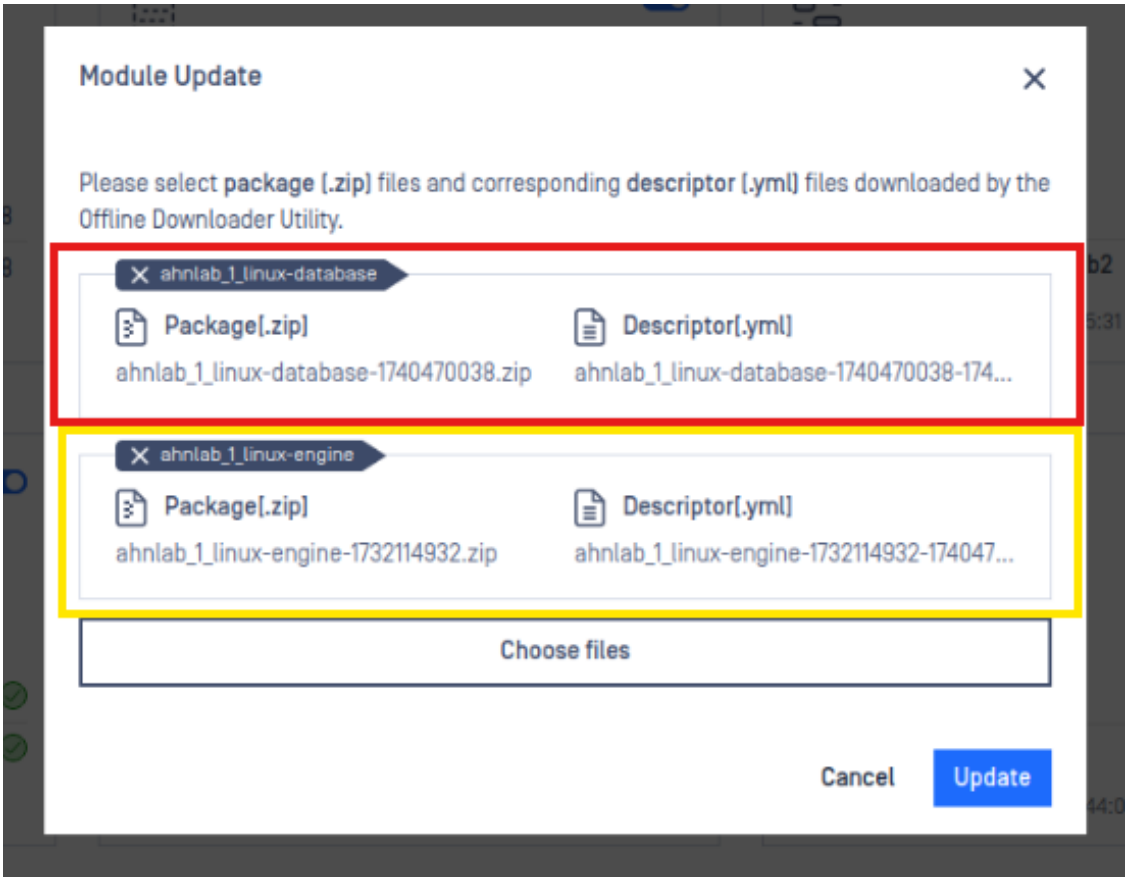


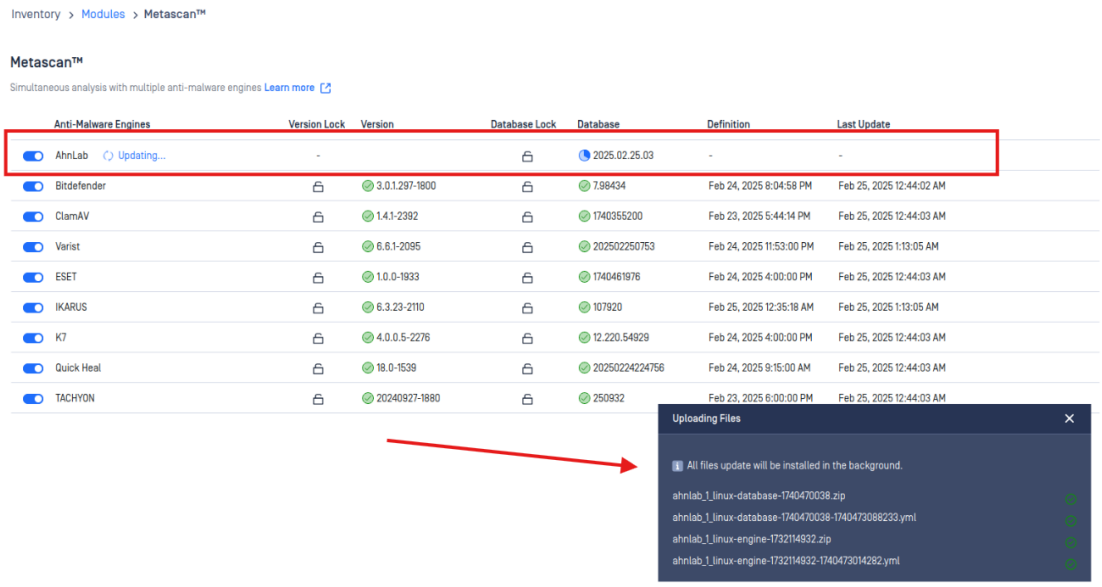
5. Navigate back to the MetaDefender Core Module tab, click on the “Upload Packages”. And manually upload 1 engine at a time as follows from the image below.






Cancel
File Upload
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Floppy Disk
Other Locations
suk
Downloads
update_packages_2025-02-25T09h_20m_23s
ahnlab_1_linux-database-1740470038.zip 161.1 MB Archive 01:20
ahnlab_1_linux-database-1740470038-1740473088233.yml 61.8 kB Text 01:20
ahnlab_1_linux-engine-1732114932.zip 7.2 MB Archive 01:20
ahnlab_1_linux-engine-1732114932-1740473014282.yml 5.2 kB Text 01:20
avira_1_linux-database-1740465706.zip 330.3 MB Archive 01:20
avira_1_linux-database-1740465706-1740473434357.yml 55.6 kB Text 01:20
avira_1_linux-engine-1729615393.zip 11.0 MB Archive 01:20
avira_1_linux-engine-1729615393-1740473014282.yml 3.1 kB Text 01:20
All Supported Types


Module Update
Please select package [.zip] files and corresponding descriptor [.yml] files downloaded by the Offline Downloader Utility.
ahnlab_1_linux-database
Package[.zip]
ahnlab_1_linux-database-1740470038.zip
Descriptor[.yml]
ahnlab_1_linux-database-1740470038-174...
ahnlab_1_linux-engine
Package[.zip]
ahnlab_1_linux-engine-1732114932.zip
Descriptor[.yml]
ahnlab_1_linux-engine-1732114932-174047...
Choose files
Cancel
Update



Once the Antivirus engine update process can be seen from the dashboard and verified that the update is complete. It will be the final steps on how to manually update Antivirus engines.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

What Happens After the Discontinuation of VirusBlokAda and NanoAV Support for Windows and Linux?

OPSWAT remains committed to providing the most comprehensive and cutting-edge threat detection capabilities within MetaDefender Core. We continuously evaluate and optimize our engine portfolio in Metascan to ensure optimal coverage and performance for our valued users.

What is changing?

Metadefender Core:

- Windows 20 Engines: VirusBlokAda will be replaced with CMC and NanoAv will be replaced by Bkav Pro
- Windows 16 Engines: VirusBlokAda will be replaced with CMC and NanoAv will be replaced by Bkav Pro
- No immediate replacements are available for: Linux and Windows Max package

Action Required

Choice of Engine Replacement:

- If you are using one of the packages mentioned above, you do not need to take any action, VirusBlokAda and NanoAV will be automatically replaced by the replacement engines on January 1, 2026.

- If you have NANOAV as a custom engine in your Linux bundle, please contact your OPSWAT sales representative or our dedicated Support team for personalized guidance and assistance. You may choose any engine from the custom engine list to replace NanoAV in your custom Linux engine package.

Activating new license

- **Online deployments:** To receive the latest updates regarding this change, please reactivate your Metadefender Core license by navigating to 'Settings' > 'License', then select 'Deactivate' followed by 'Activate'.
- **Offline deployments:** To receive the latest update regarding this change, please contact your OPSWAT sales representative or our dedicated Support team to receive the new license key and activate the license.

What is the Timeline?

Effective **January 1, 2026**, we have ceased offering the VirusBlokAda and NanoAV anti-malware engine for both Windows and Linux platforms and have also stopped including it in our bundles. After this date, VirusBlokAda and NanoAV will be automatically replaced with the aforementioned engine replacement. If you wish to receive the replacement earlier, please contact your sales representative or OPSWAT Support.

What is the Next Step?

OPSWAT is committed to providing top-tier threat detection solutions. We will keep assessing and incorporating leading anti-malware engines into Metadefender Core to boost its Multiscanning capabilities and safeguard your critical infrastructure.

Please Note:

OPSWAT will no longer provide any support or information regarding VirusBlokAda and NanoAV.

Support

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

What Happens After the Discontinuation of RocketCyber Support for Windows and Linux?

OPSWAT remains committed to providing the most comprehensive and cutting-edge threat detection capabilities within MetaDefender Core. We continuously evaluate and optimize our engine portfolio in Metascan to ensure optimal coverage and performance for our valued users.

Background

RocketCyber has officially announced the End-of-Life (EOL) for the anti-malware engine currently integrated into OPSWAT MetaDefender Core. As a result of this vendor-driven discontinuation, OPSWAT will be removing the

RocketCyber engine from all applicable MetaDefender Core and cloud packages and replacing it with an alternative engine to maintain uninterrupted multiscanning coverage.

This change is being made to ensure that our customers continue to benefit from fully supported, actively maintained, and high-performing anti-malware engines within the Metascan portfolio.

What is Changing?

MetaDefender Core:

- **Linux and Windows Max Package:** No immediate replacement is available at this time

Action Required

Choice of Engine Replacement:

- If you are using one of the packages mentioned above, no action is required on your part. The RocketCyber engine will be automatically removed on **August 1, 2026**.
- If you have **RocketCyber as a custom engine** in your Windows or Linux bundle, please contact your OPSWAT sales representative or our dedicated Support team for personalized guidance and assistance. Y**ou may choose any engine from the custom engine list to replace RocketCyber in your custom Windows or Linux engine package**.

Activating new license

- **Online deployments:** To receive the latest updates regarding this change, please reactivate your Metadefender Core license by navigating to 'Settings' > 'License', then select 'Deactivate' followed by 'Activate'.
- **Offline deployments:** To receive the latest update regarding this change, please contact your OPSWAT sales representative or our dedicated Support team to receive the new license key and activate the license.

What is the Timeline?

Effective **August 1, 2026**, OPSWAT will cease offering the RocketCyber anti-malware engine for both Windows and Linux platforms and will no longer include it in our bundles. Before this date, please contact your sales representative or OPSWAT Support to select your replacement engine.

What is the Next Step?

OPSWAT is committed to providing top-tier threat detection solutions. We will keep assessing and incorporating leading anti-malware engines into Metadefender Core to boost its Multiscanning capabilities and safeguard your critical infrastructure.

Please note

OPSWAT will no longer provide any support or information regarding RocketCyber.

Support

If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

Is MetaDefender Core affected by NGINX CVEs?

Check Your Version:

- This article applies to MetaDefender Core prior to 5.18.0

Issue

Impact assessment for six NGINX vulnerabilities against MetaDefender Core:

- CVE-2026-42945: Not affected. MD Core's rewrite rules do not use nested or overlapping capture groups, capture substitutions such as $1/$2, or the vulnerable rewrite patterns described in the advisory.
- CVE-2026-42946: Not affected. MD Core does not use the scgi_pass or uwsgi_pass modules in the NGINX configuration.
- CVE-2026-40460: Not affected. MD Core does not use the HTTP/3 QUIC module.
- CVE-2026-42926: Not affected. The NGINX configuration does not use proxy_http_version 2.
- CVE-2026-40701: Low risk. The vulnerability only affects deployments using specific mTLS and OCSP validation settings in NGINX. Please verify whether ssl_verify_client on|optional and ssl_ocsp on are enabled in the active NGINX configuration.
- CVE-2026-42934: Not affected. The vulnerable configuration combination (charset/source_charset/charset_map together with proxy_pass and proxy_buffering off) is not used in the NGINX configuration.

Resolution

- Upgrade to MetaDefender Core 5.19.0 or later for NGINX 1.30.1.
- If you must stay on 5.18.0 temporarily, please log a support case

Support:

If you require further assistance, please follow these instructions on How to Create Support Package?, before creating a support case or chatting with our support engineer.

End of Life Support for Debian 11 in MetaDefender Core

As of August 31, 2026, Debian 11 (Bullseye) will reach its End of Life (EOL), meaning it will no longer receive security

updates, patches, or support from the Debian project. In alignment with this timeline, OPSWAT will discontinue MetaDefender Core support for Debian 11 starting **July 1, 2026** — ahead of the official Debian EOL date. This means MetaDefender Core v5.20.0 and later will not support Debian 11. This article outlines the implications, risks, and recommendations for continuing to use this operating system with MetaDefender Core.

Background

MetaDefender Core relies on the underlying operating system for security updates and stability. With Debian 11 approaching EOL, the Debian project will cease providing updates, leaving systems vulnerable to new security threats. To ensure the security and reliability of MetaDefender Core deployments, OPSWAT strongly recommends upgrading to a supported operating system before July 1, 2026.

Impact on MetaDefender Core

Running MetaDefender Core on Debian 11 after its EOL introduces significant risks, including:

- **Security Vulnerabilities:** Without security patches, the operating system is exposed to new exploits and threats.
- **Compatibility Issues:** MetaDefender Core updates and new features may not function correctly on unsupported platforms.
- **Compliance Risks:** Many regulatory standards require systems to run on actively supported operating systems.
- **Lack of Support:** OPSWAT technical support may be limited or unavailable for issues related to EOL operating systems.

Recommendations

To maintain a secure and supported MetaDefender Core environment, OPSWAT recommends the following:

Upgrade to a Supported Operating System Transition to a supported operating system, such as Debian 12 (Bookworm) or Debian 13 (Trixie), which receive active security updates and are fully compatible with the latest MetaDefender Core versions. Refer to the MetaDefender Core System Requirements for a complete list of supported platforms.

Plan Your Migration

1. **Assess Your Environment:** Identify all systems running Debian 11 with MetaDefender Core installed.
2. **Test Compatibility:** Deploy MetaDefender Core on the new operating system in a staging environment to ensure compatibility with your workflows.
3. **Execute Migration:** Follow OPSWAT's migration guides to transfer configurations, policies, and data to the new platform.
4. **Validate Post-Migration:** Verify that MetaDefender Core operates as expected after the upgrade.

Continuing with Debian 11 (Not Recommended)

If migration is not immediately feasible, you may choose to remain on MetaDefender Core v5.19.0, the last release to support Debian 11, and pin your engines to specific versions compatible with Debian 11. This approach, however, is not recommended due to the following risks:

- **Security Exposure:** The operating system will remain unpatched after August 31, 2026, increasing the risk of exploitation.
- **Limited Functionality:** You will not receive new features, bug fixes, or performance improvements introduced in later MetaDefender Core versions.
- **Unsupported Configurations:** OPSWAT may not provide support for issues arising from EOL operating systems.
- **Compliance Violations:** Running an EOL operating system may violate organizational or regulatory compliance requirements.

To pin your engine:

1. Access the MetaDefender Core management console.
2. Navigate to **Inventory > Modules**.
3. Select the option to pin the engine to the current version.
4. Save the configuration.

Note:

Staying on v5.19.0 and pinning engine version should only be a temporary measure while planning your migration to a supported operating system.

Conclusion

The upcoming EOL of Debian 11 poses significant risks to the security and functionality of MetaDefender Core deployments. OPSWAT strongly urges users to upgrade to a supported operating system — such as Debian 12 or Debian 13 — before July 1, 2026, to ensure continued security, compatibility, and support. For assistance with migration or additional guidance, contact OPSWAT Support or refer to the MetaDefender Core Documentation.

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with our support engineer**.

Is MetaDefender Core Affected by PHP2Shell (CVE-2026-109618 and CVE-2026-109619)?

Check Your Version

Based on the product-level security assessment, MetaDefender Core does not use PHP.

Security Assessment: Not Affected

MetaDefender Core is not affected by CVE-2026-109618 or CVE-2026-109619 within the assessed product scope.

Overview

PHP2Shell refers to two security vulnerabilities in the PHP interpreter: **CVE-2026-109618** and **CVE-2026-109619**. Under certain conditions, an affected PHP runtime may allow remote code execution.

Product Assessment

MetaDefender Core does not contain PHP. The assessed product is therefore not susceptible to these PHP interpreter vulnerabilities.

Is Any Action Required?

No product-specific patch, hotfix or configuration change is required for the assessed product to address these two PHP vulnerabilities. PHP installed separately as part of another application or service must be assessed independently.

Note: This article addresses only the two CVEs above. It does not assess other vulnerabilities or independently installed third-party software.

Additional Resources

- PHP Security Release Information — October 21, 2026

Support

If you have questions about this security assessment or a reported vulnerability finding, please contact OPSWAT Support.

This article was last updated on 2026-10-22.

How do I manually upgrade to the latest MetaDefender Core version?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems, and will cover deployments in both Standalone and Shared Database mode.

Please scroll down to find the instructions applicable to your OS and mode.

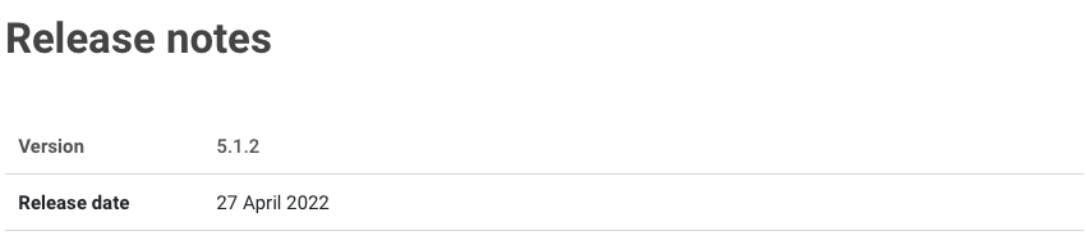
Please Note:

MetaDefender Core version upgrades are always **Manual**; only certain feature updates (such as engine updates) can be configured to **Automatic**.

Windows (Standalone DB mode)

To perform a **MetaDefender Core Deployment Upgrade**, please follow the steps below:

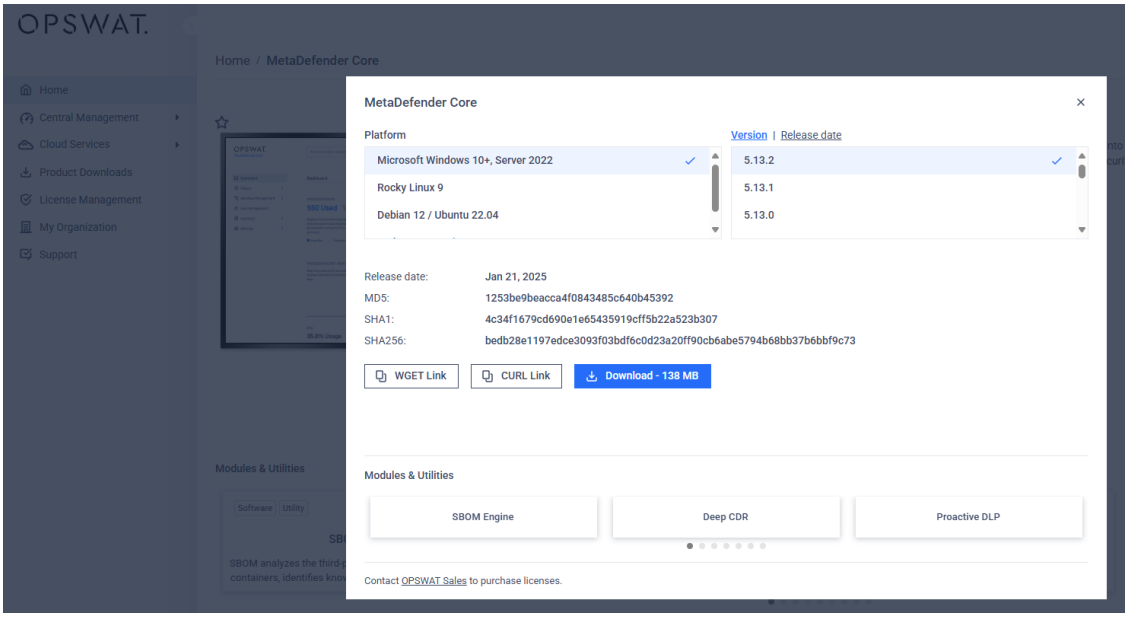
1. Browse Release Notes for details on the latest MetaDefender Core version release.



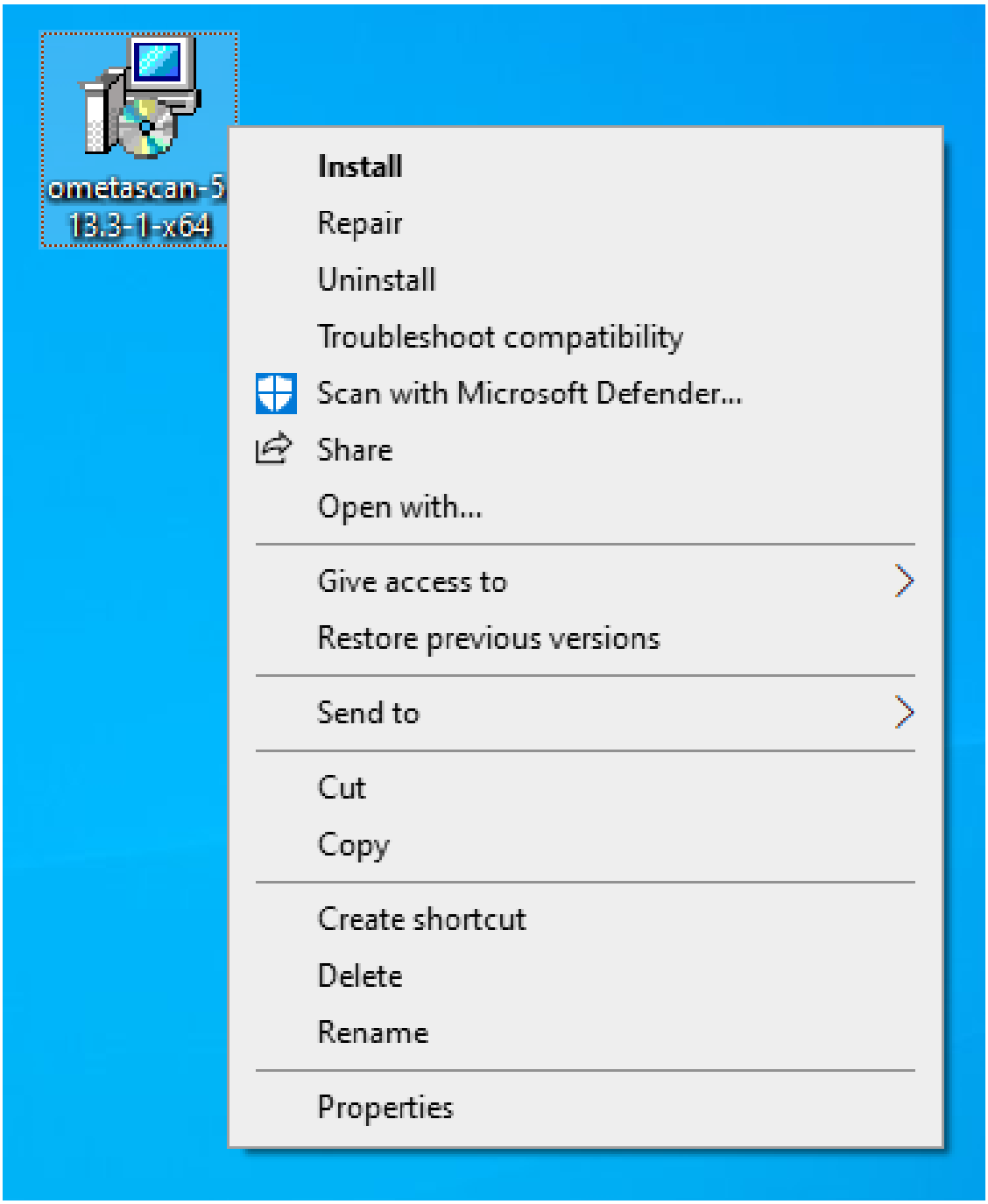
Pro Tip:

OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

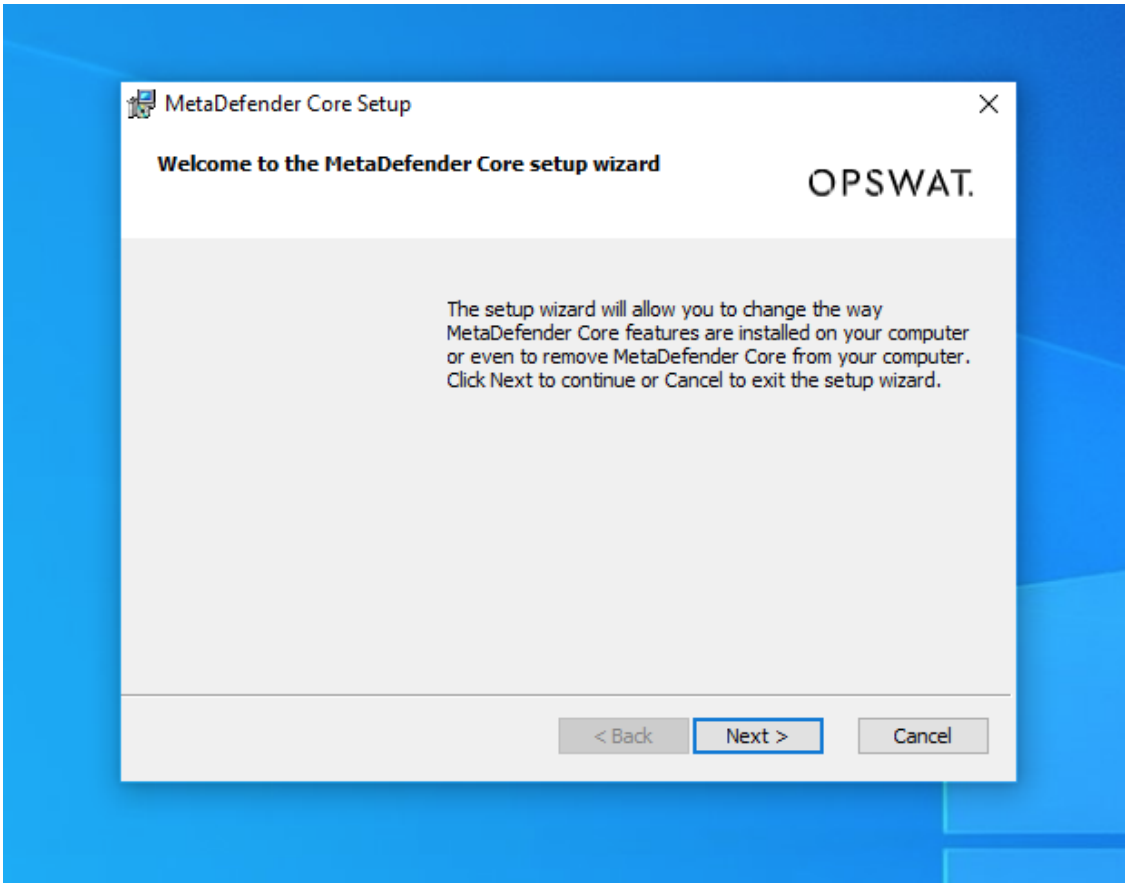
2. Visit My OPSWAT to download the latest MetaDefender Core version for Windows.



3. Log in as Admin user on the MetaDefender Core server, then **Run** the installer file.

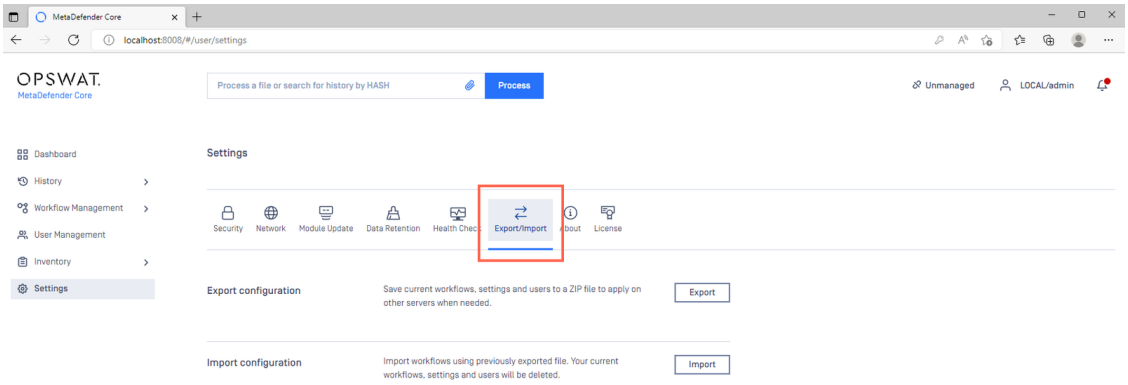


4. Let the **MetaDefender Core Setup Wizard** guide you through the process.



5. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more.

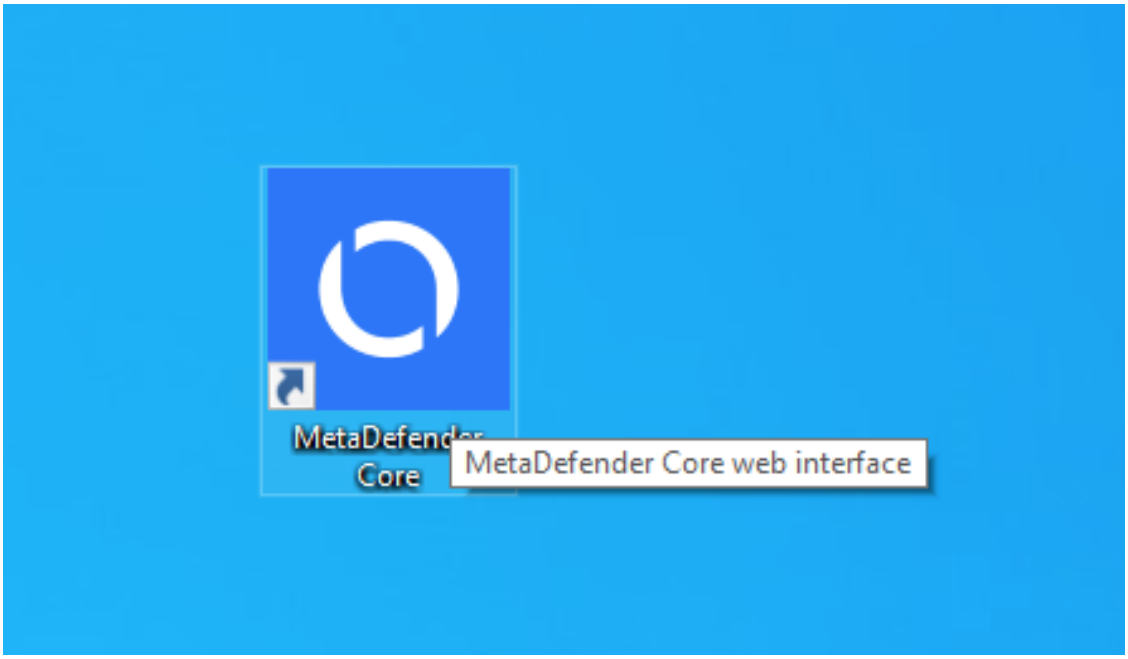
With the help of this feature, and provided no errors occur during the upgrade process, your configurations and licensing information will be fully preserved through the upgrade process.



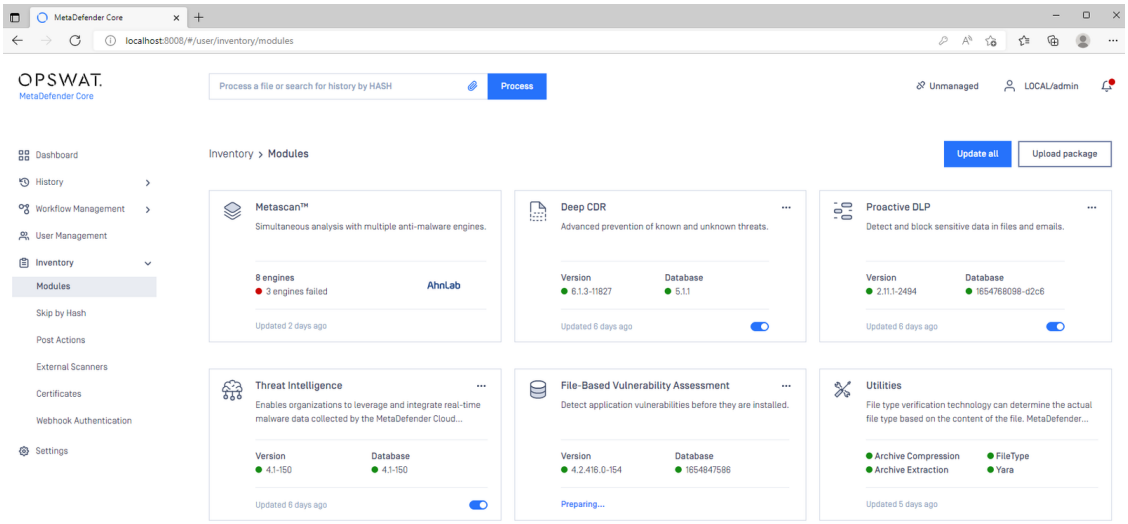
Please Note:

Once the installer has finished, your system will be upgraded to the latest version.

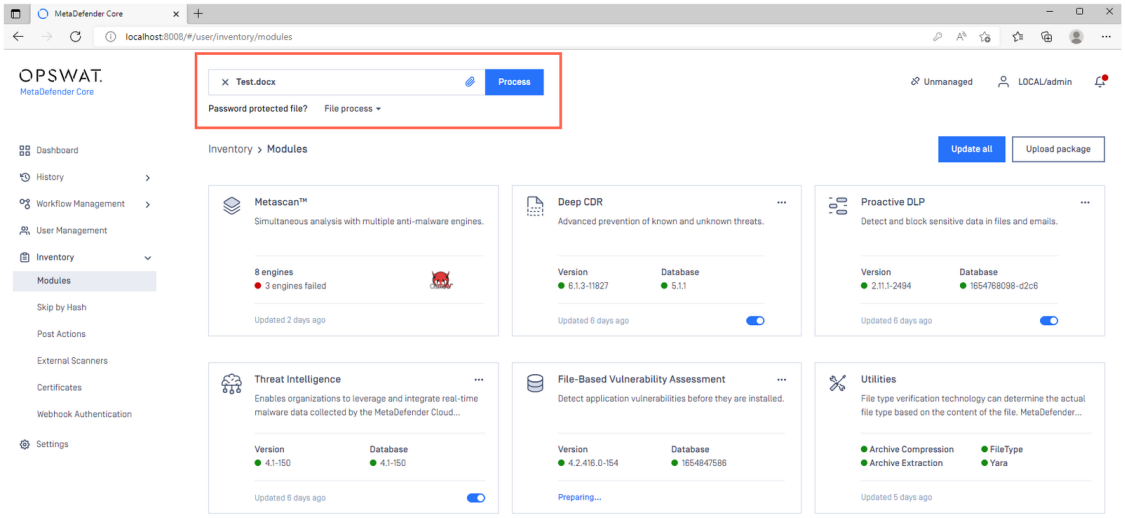
6. Once installed, click the MetaDefender Core icon to log into the **Management Console** and verify that the system is functioning as expected.



7. In the Console, open **Inventory>Modules** to view the status of your module and engine downloads.



8. Once all your required modules are up-to-date and enabled, and your **Security Policies** configured, drop a test file in the **Process** field to run your first scan.



Windows (Standalone DB mode) command line

> **Please Note:**
>
> For command line installation in Windows, it is mandatory to use a MetaDefender Core ignition file to add PostgreSQL server information ***before*** installing MetaDefender Core 4.19.0 or later releases.

MetaDefender Core supports two modes to setup PostgreSQL server:

1. Local PostgreSQL Server
2. Existing PostgreSQL Server

Local PostgreSQL Server

Here, MetaDefender Core will install a new PostgreSQL server locally, alongside the product.

A sample ignition file for PostgreSQL server's information is shown below. Note that only non-Unicode characters are supported for both **user** and **password**.

ometascan.conf

```
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
```

On Windows systems, it is possible to install MetaDefender Core by running the corresponding *.msi* file. This can be done via command line by executing the following command.

Code

```
msiexec /i <msi file name> <option key>=<option value>
```

Where the <option keys>, their default values and their descriptions are as follows:

Key	Default Value	Description
INSTALLFOLDER	C:\Program Files\OPSWAT\MetaDefender Core	Customize installation folder for product. Example: INSTALLFOLDER="D:\Products"
RESTADDRESS	•	REST interface binding IPv4 or IPv6 address ('*' means that the service listens on all IPv4 and IPv6 interfaces).
RESTPORT	8008	REST interface binding port
EXTERNALNODE		Whether to enable external processing nodes or not. To enable external processing nodes, set EXTERNALNODE=1 . ADDRESS and PORT values below are admitted only if EXTERNALNODE=1 is set.
ADDRESS		Address of the computer to accept external scan node connections. ADDRESS value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but ADDRESS is not, then ADDRESS defaults to * (all IPv4 and IPv6 interfaces).
PORT		Designated port number to accept external scan node connections. PORT value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but PORT is not, then PORT defaults to 8007.

Key	Default Value	Description
REPORT_ENGINE_ISSUE	true	Enable reporting of engine issue count (possible values: 'true' or 'false').

Existing PostgreSQL Server

Here, MetaDefender Core will connect to a pre-installed PostgreSQL server, running remotely, before setting up its database.

Be sure to follow the Open Connection steps on PostgreSQL server first, so that your existing remote PostgreSQL server is ready to accept the upcoming connection from MetaDefender Core.

A sample ignition file for PostgreSQL server's information is shown below.

ometascan.conf

```
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
```

On Windows systems, it is possible to install MetaDefender Core by running the corresponding *.msi* file. This can be done via command line by executing the following command.

Code

```
msiexec /i <msi file name> <option key>=<option value>
```

Where the <option keys>, their default values and their descriptions are as follows:

Key	Default Value	Description
INSTALLFOLDER	C:\Program Files\OPSWAT\MetaDefender Core	Customize installation folder for product. Example: INSTALLFOLDER="D:\Products"
RESTADDRESS	•	REST interface binding IPv4 or IPv6 address ('*' means that the service listens on all IPv4 and IPv6 interfaces).

Key	Default Value	Description
RESTPORT	8008	REST interface binding port
EXTERNALNODE		Whether to enable external processing nodes or not. To enable external processing nodes, set EXTERNALNODE=1 . ADDRESS and PORT values below are admitted only if EXTERNALNODE=1 is set.
ADDRESS		Address of the computer to accept external scan node connections. ADDRESS value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but ADDRESS is not, then ADDRESS defaults to * (all IPv4 and IPv6 interfaces).
PORT		Designated port number to accept external scan node connections. PORT value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but PORT is not, then PORT defaults to 8007.
REPORT_ENGINE_ISSUE	true	Enable reporting of engine issue count (possible values: 'true' or 'false').

Windows (Shared Database mode)

To perform a **MetaDefender Core Deployment Upgrade**, please follow the steps below:

1. Browse Release Notes for details on the latest MetaDefender Core version release.

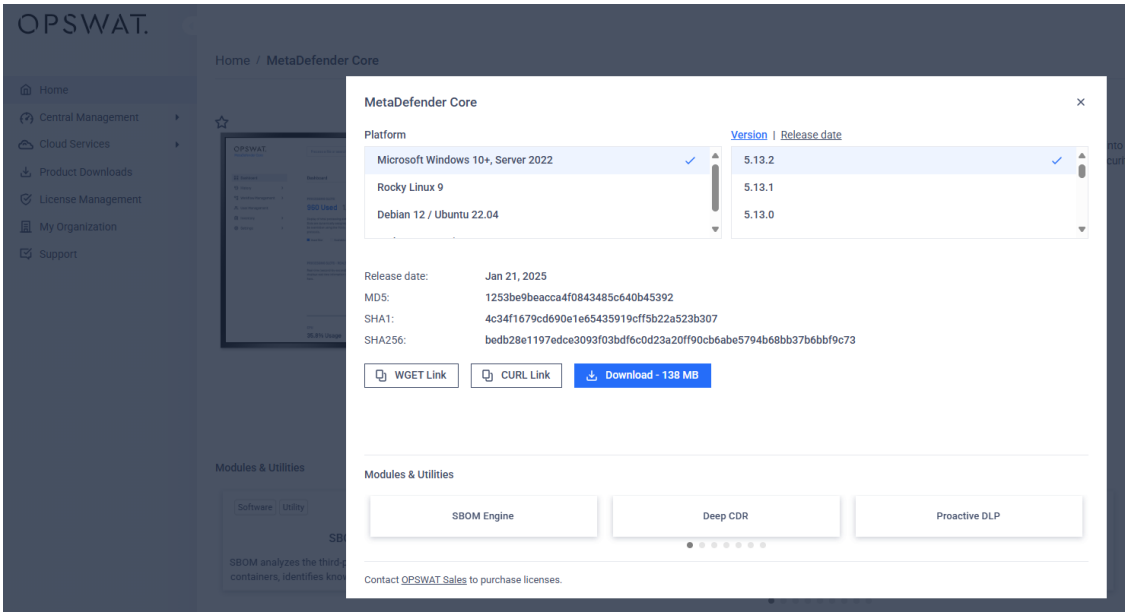
Release notes

Version	5.1.2
Release date	27 April 2022

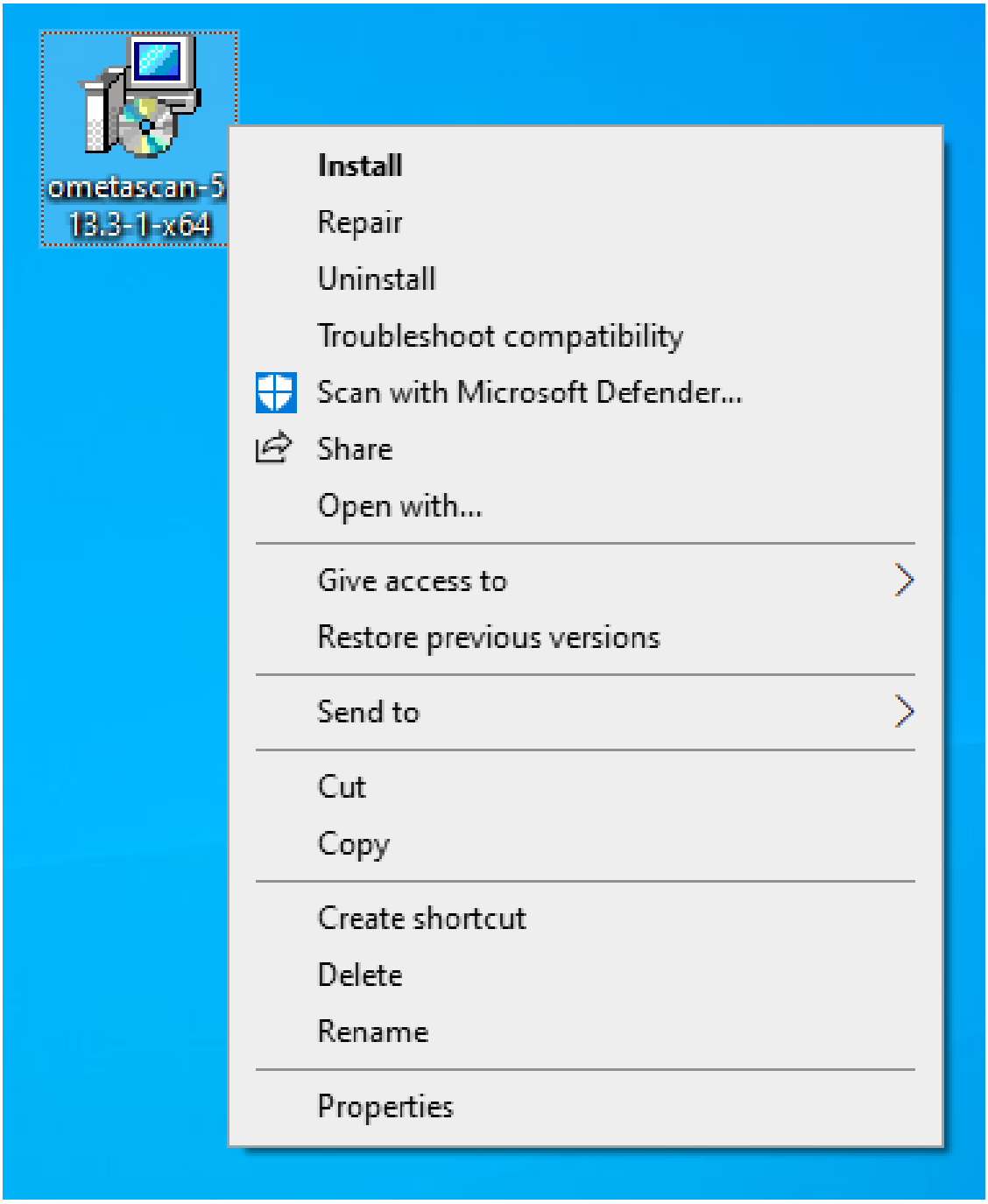
Pro Tip:

OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

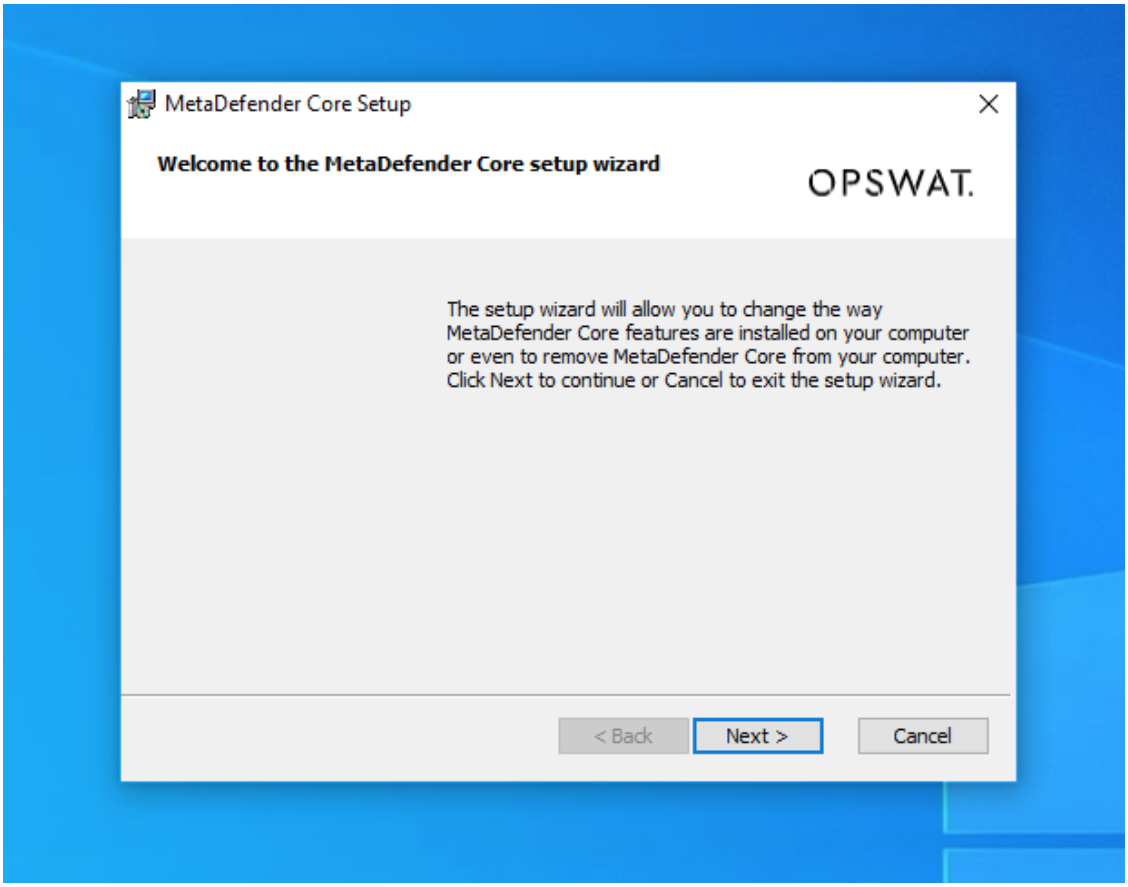
2. Visit My OPSWAT to download the latest MetaDefender Core version for Windows.



3. Log in as Admin user on the MetaDefender Core server, then **Run** the installer file.

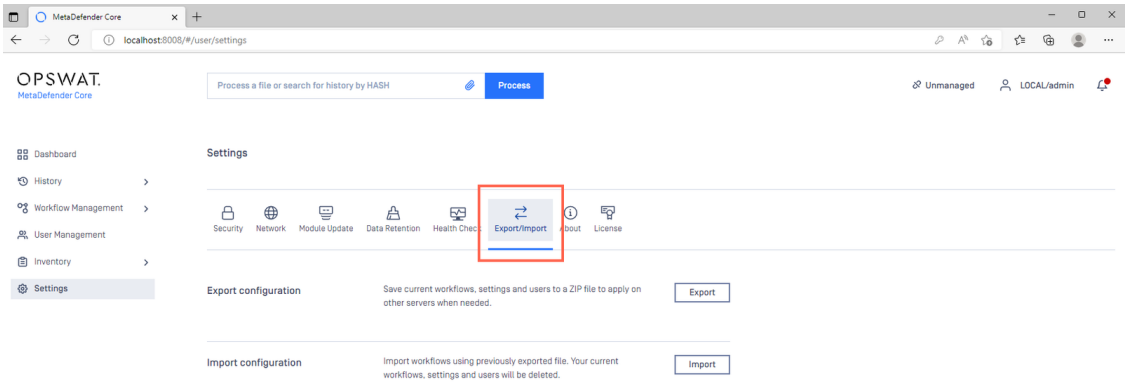


4. Let the **MetaDefender Core Setup Wizard** guide you through the process.

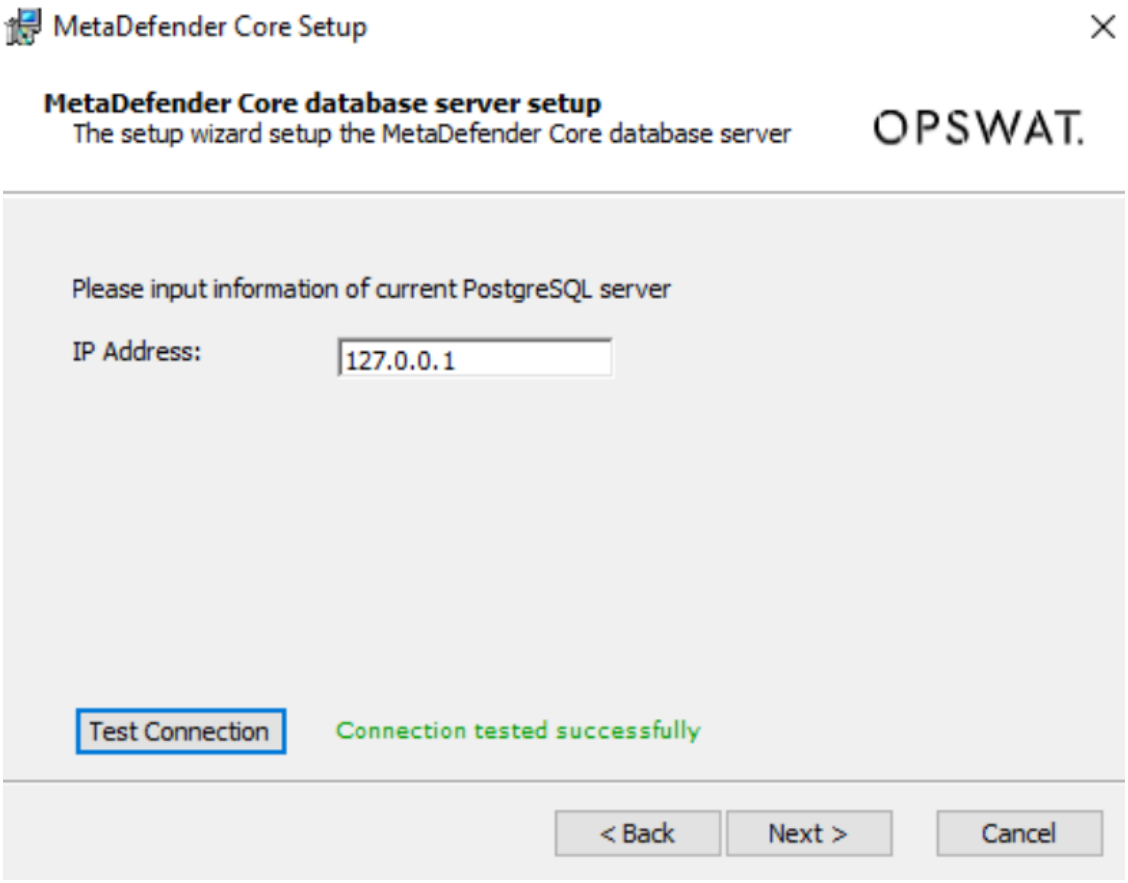


5. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more.

With the help of this feature, and provided no errors occur during the upgrade process, your configurations and licensing information will be fully preserved through the upgrade process.



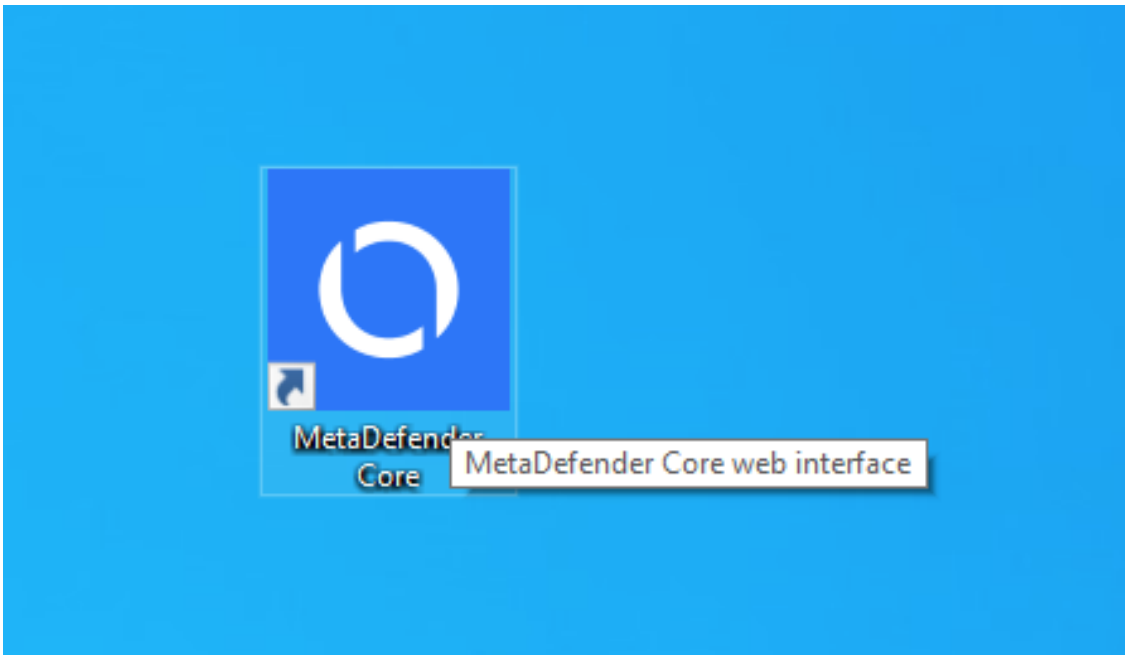
6. For **Shared Database** mode, the same upgrade instructions apply as described above for **Standalone** mode, with one exception: at the Setup stage illustrated in the screenshot below, you will need to specify your current PostgreSQL server IP address.



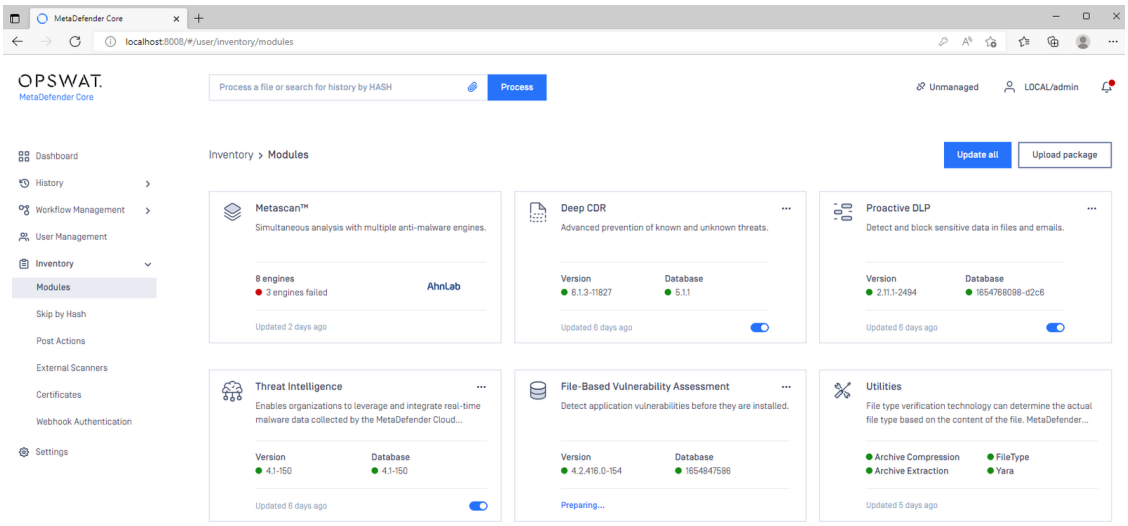
Please Note:

Once the installer has finished, your system will be upgraded to the latest version.

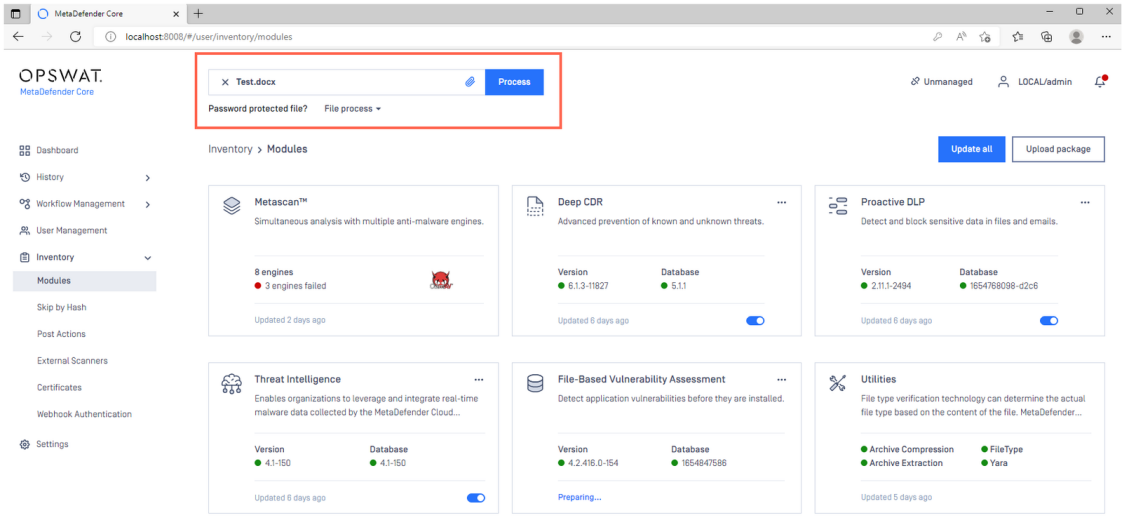
7. Once installed, click the MetaDefender Core icon to log into the **Management Console** and verify that the system is functioning as expected.



8. In the Console, open **Inventory>Modules** to view the status of your module and engine downloads.



9. Once all your required modules are up-to-date and enabled, and your **Security Policies** configured, drop a test file in the **Process** field to run your first scan.



Windows (Shared Database mode) command line

> **Please Note:**
>
> For command line installation in Windows, it is mandatory to use a MetaDefender Core ignition file to add PostgreSQL server information *before* installing MetaDefender Core 4.19.0 or later releases.

MetaDefender Core supports two modes to setup PostgreSQL server:

1. Local PostgreSQL Server
2. Existing PostgreSQL Server

Local PostgreSQL Server

Here, MetaDefender Core will install a new PostgreSQL server locally, alongside the product.

A sample ignition file for PostgreSQL server's information is shown below. Note that only non-Unicode characters are supported for both **user** and **password**.

ometascan.conf

```
[global]
dbmode=4

[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
```

On Windows systems, it is possible to install MetaDefender Core by running the corresponding *.msi* file. This can be done via command line by executing the following command.

Code

```
msiexec /i <msi file name> <option key>=<option value>
```

Where the <option keys>, their default values and their descriptions are as follows:

Key	Default Value	Description
INSTALLFOLDER	C:\Program Files\OPSWAT\MetaDefender Core	Customize installation folder for product. Example: INSTALLFOLDER="D:\Products"
RESTADDRESS	•	REST interface binding IPv4 or IPv6 address ('*' means that the service listens on all IPv4 and IPv6 interfaces).
RESTPORT	8008	REST interface binding port
EXTERNALNODE		Whether to enable external processing nodes or not. To enable external processing nodes, set EXTERNALNODE=1 .

Key	Default Value	Description
		ADDRESS and PORT values below are admitted only if EXTERNALNODE=1 is set.
ADDRESS		Address of the computer to accept external scan node connections. ADDRESS value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but ADDRESS is not, then ADDRESS defaults to * (all IPv4 and IPv6 interfaces).
PORT		Designated port number to accept external scan node connections. PORT value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but PORT is not, then PORT defaults to 8007.
REPORT_ENGINE_ISSUE	true	Enable reporting of engine issue count (possible values: 'true' or 'false').

Existing PostgreSQL Server

Here, MetaDefender Core will connect to a pre-installed PostgreSQL server, running remotely, before setting up its database.

Be sure to follow the Open Connection steps on PostgreSQL server first, so that your existing remote PostgreSQL server is ready to accept the upcoming connection from MetaDefender Core.

A sample ignition file for PostgreSQL server's information is shown below.

ometascan.conf

```
[global]
dbmode=4

[dbserver]
type=local
host=localhost
port=5432
```

```
user=postgres
password=non_Unicode_password
```

On Windows systems, it is possible to install MetaDefender Core by running the corresponding *.msi* file. This can be done via command line by executing the following command.

Code

```
msiexec /i <msi file name> <option key>=<option value>
```

Where the <option keys>, their default values and their descriptions are as follows:

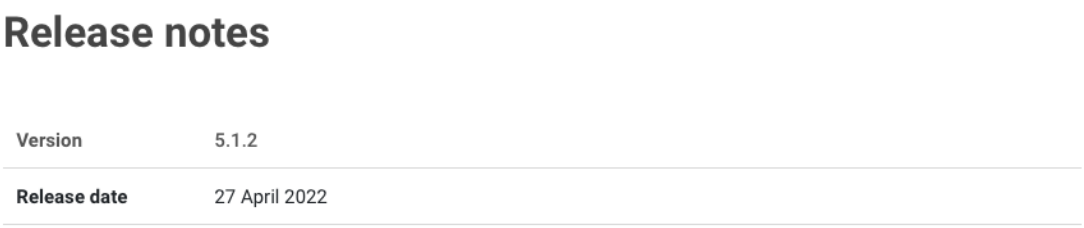
Key	Default Value	Description
INSTALLFOLDER	C:\Program Files\OPSWAT\MetaDefender Core	Customize installation folder for product. Example: INSTALLFOLDER="D:\Products"
RESTADDRESS	•	REST interface binding IPv4 or IPv6 address ('*' means that the service listens on all IPv4 and IPv6 interfaces).
RESTPORT	8008	REST interface binding port
EXTERNALNODE		Whether to enable external processing nodes or not. To enable external processing nodes, set EXTERNALNODE=1 . ADDRESS and PORT values below are admitted only if EXTERNALNODE=1 is set.
ADDRESS		Address of the computer to accept external scan node connections. ADDRESS value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but ADDRESS is not, then ADDRESS defaults to * (all IPv4 and IPv6 interfaces).
PORT		Designated port number to accept

Key	Default Value	Description
		external scan node connections. PORT value is only admitted if EXTERNALNODE=1 is set. If EXTERNALNODE=1 is set but PORT is not, then PORT defaults to 8007.

Linux (Standalone mode)

To perform a **MetaDefender Core Deployment Upgrade**, please follow the steps below:

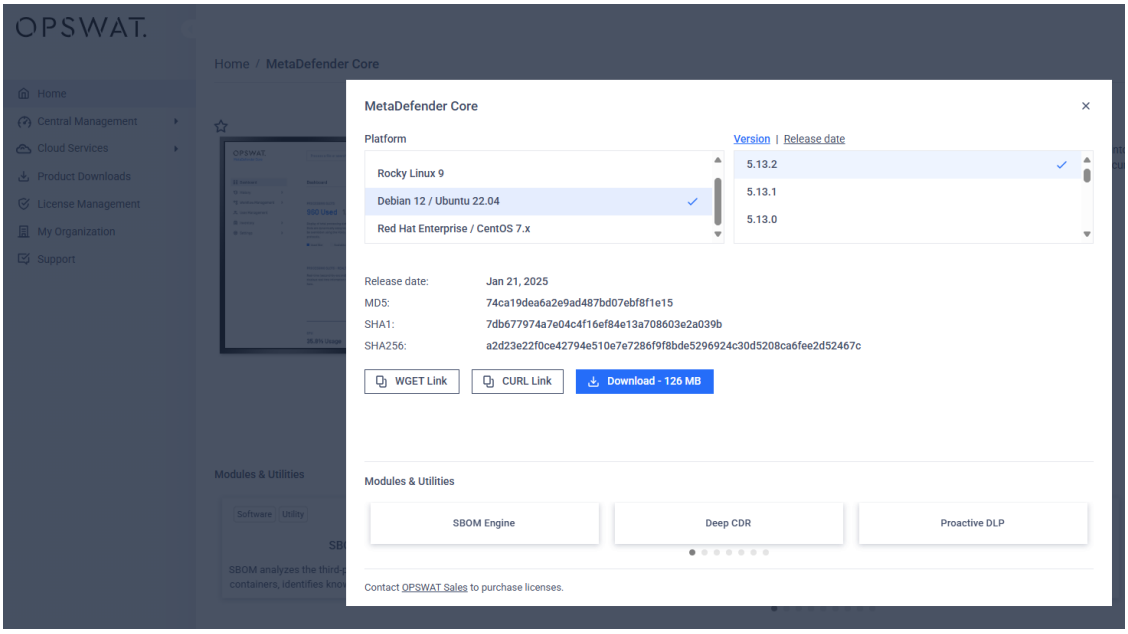
1. Browse Release Notes for details on the latest MetaDefender Core version release.



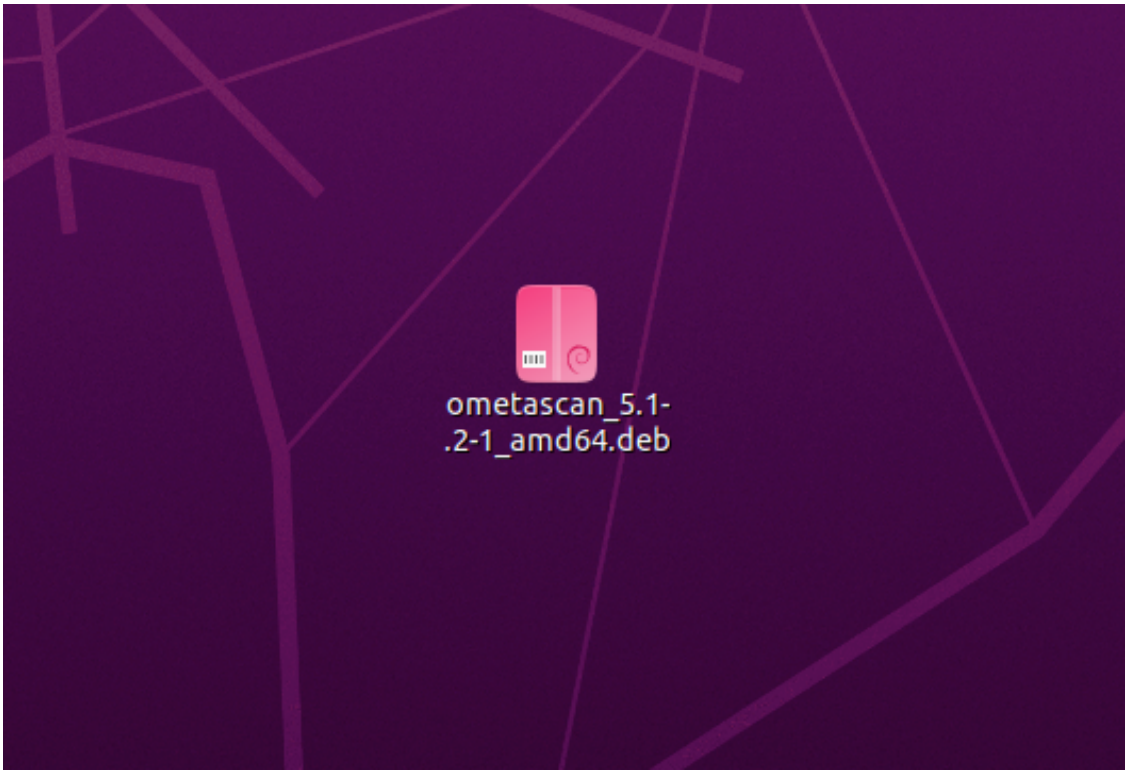
Pro Tip:

OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

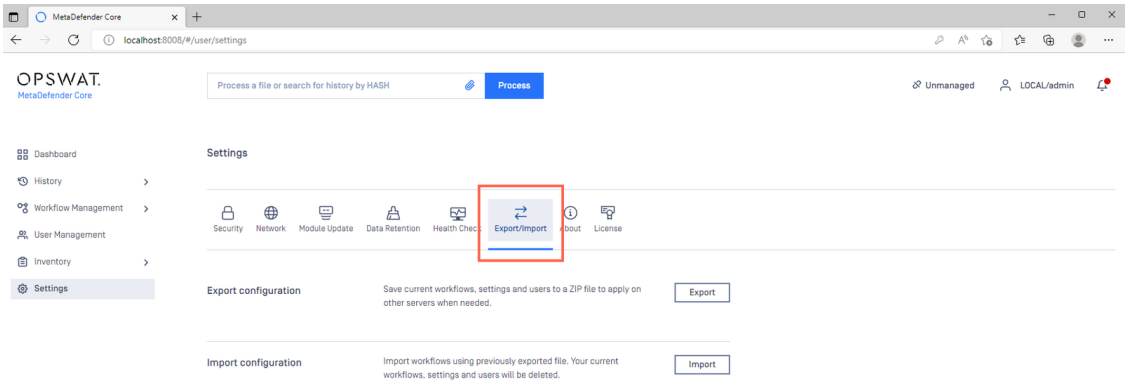
2. Visit My OPSWAT to download the latest MetaDefender Core version for your Linux distro.



3. Log in as **Root** user on the MetaDefender Core server, then **Run** the installer file.



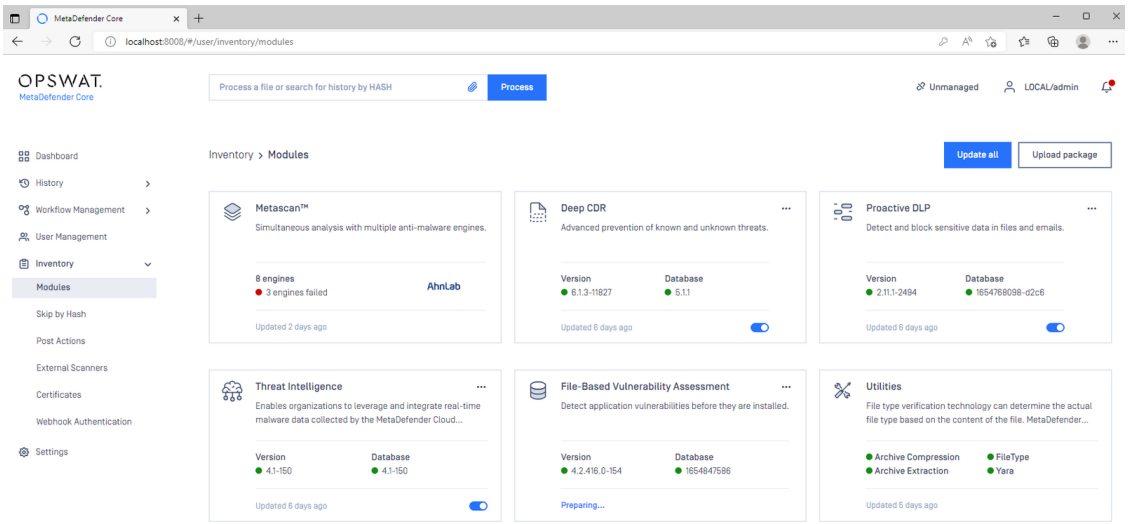
4. Depending on your distro, **Software Install** or a similar utility will guide you through the process.
5. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more. With the help of this feature, and provided no errors occur during the upgrade process, your configurations and licensing information will be fully preserved through the upgrade process.



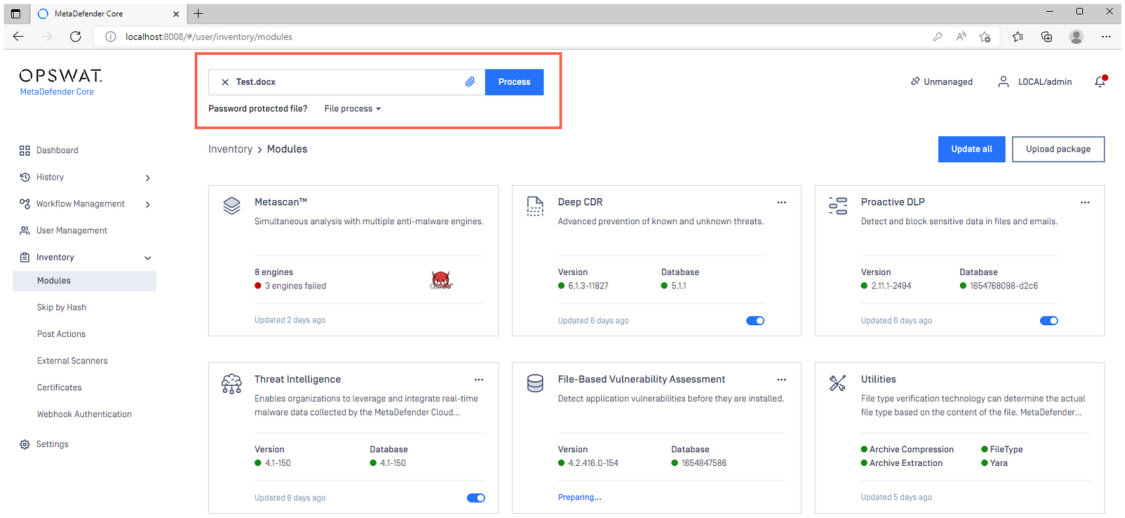
Please Note:

Once the installer has finished, your system will be upgraded to the latest version.

6. Once installed, click the MetaDefender Core icon to log into the **Management Console** and verify that the system is functioning as expected.
7. In the Console, open **Inventory>Modules** to view the status of your module and engine downloads.



8. Once all your required modules are up-to-date and enabled, and your **Security Policies** configured, drop a test file in the **Process** field to run your first scan.



Linux (Standalone mode) command line

Please Note:

For command line installation in Linux, it is mandatory to use a MetaDefender Core ignition file to add PostgreSQL server information *before* installing MetaDefender Core 4.19.0 or later releases.

1. Log into the terminal as **Root** user.
2. Create an ignition file (if one does not already exist) via command line by running the following command: **touch ometascan.conf**
3. Now, run: **nano ometascan.conf** to edit the file with the necessary information.
4. An example of an ignition file is shown below.

- *Linux:* ***/etc/opswat/ometascan.conf***

ometascan.conf

```
eula=true

[global]
dbmode=1
instance_name=MetaDefenderCoreA
storage_path=D:\path\

[user]
name=admin
password=admin
email=admin@local

[dbserver]
```

```
type=remote
host=192.168.1.1
port=5432
user=postgres
password=non_Unicode_password
current_host=192.168.163.1
```

MetaDefender Core supports two modes to setup PostgreSQL server:

1. Local PostgreSQL Server

2. Existing PostgreSQL Server

Local PostgreSQL Server

Here, MetaDefender Core will install a new PostgreSQL server locally, alongside the product.

A sample ignition file for PostgreSQL server's information is shown below. Note that only non-Unicode characters are supported for both **user** and **password**.

ometascan.conf

```
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
```

Finally, enter the relevant command from the table below to install MetaDefender Core.

Debian package (.deb)	*sudo dpkg -i <filename> || sudo apt-get install -f*
On Red Hat Enterprise 7.x Linux / CentOS package (.rpm)	*sudo yum install <filename>*
For systems which enabled GPD check flag	*sudo yum install --nogpgcheck <filename>*
Red Hat Enterprise Linux 8.x package (.rpm)	*sudo dnf install -y <filename>*

Existing PostgreSQL server

Here, MetaDefender Core will connect to a pre-installed PostgreSQL server, running remotely, before setting up its database.

Be sure to follow the Open Connection steps on PostgreSQL server first, so that your existing remote PostgreSQL

server is ready to accept the upcoming connection from MetaDefender Core.

A sample ignition file for PostgreSQL server's information is shown below.

ometascan.conf

```
[dbserver]
type=remote
host=192.168.86.32
port=5432
user=existingserver_admin_user
password=existing_server_password
```

Finally, enter the relevant command from the table below to install MetaDefender Core.

Debian package (.deb)	*sudo dpkg -i <filename> || sudo apt-get install -f*
On Red Hat Enterprise 7.x Linux / CentOS package (.rpm)	*sudo yum install <filename>*
For systems which enabled GPD check flag	*sudo yum install --nogpgcheck <filename>*
Red Hat Enterprise Linux 8.x package (.rpm)	*sudo dnf install -y <filename>*

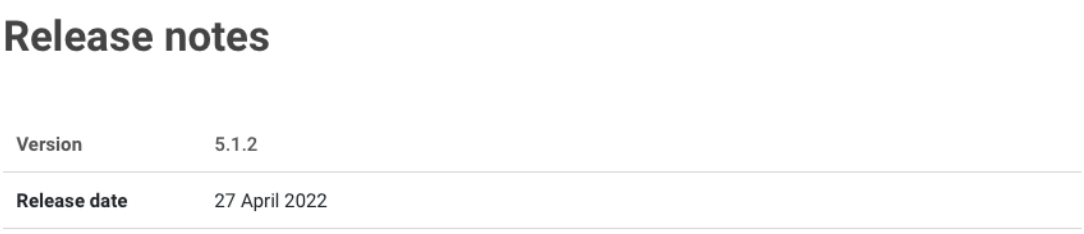
Pro Tip:

For security reasons, after installing MetaDefender Core successfully, you may want to remove all credentials information created for the **[dbserver]** section in the ignition file.

Linux (Shared Database mode)

To perform a **MetaDefender Core Deployment Upgrade**, please follow the steps below:

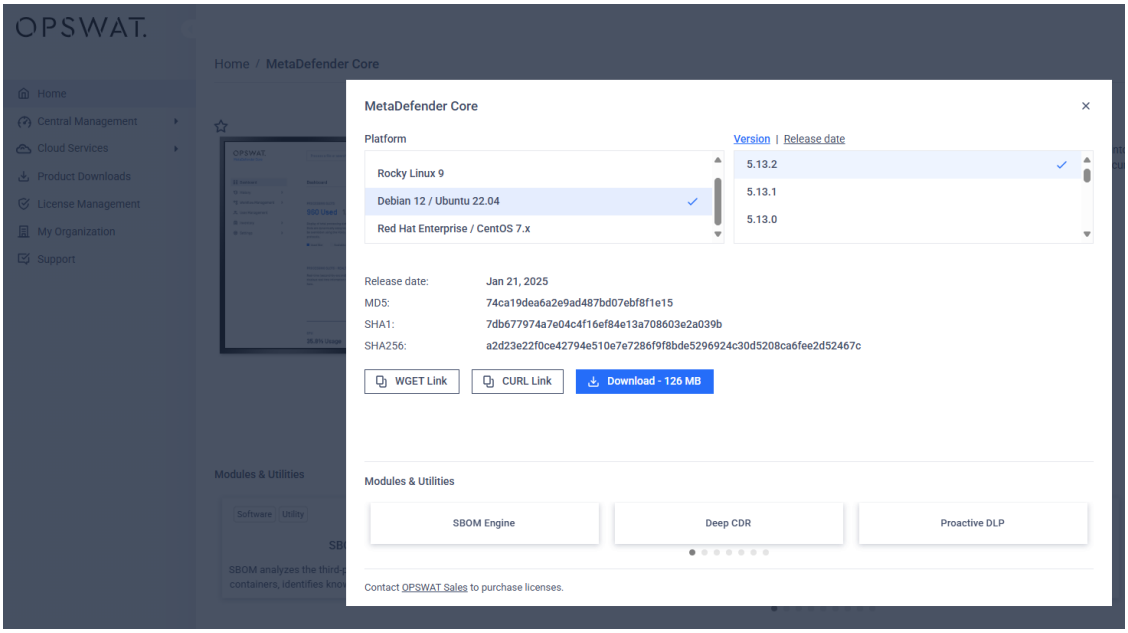
1. Browse Release Notes for details on the latest MetaDefender Core version release.



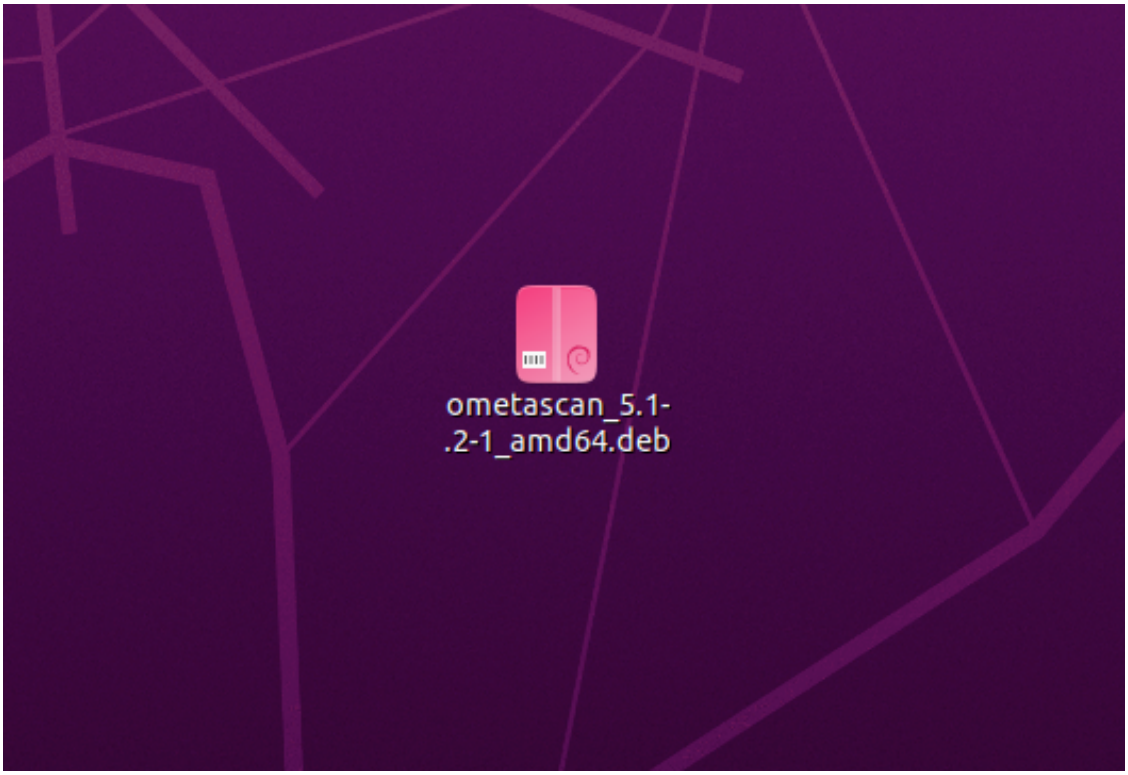
Pro Tip:

OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

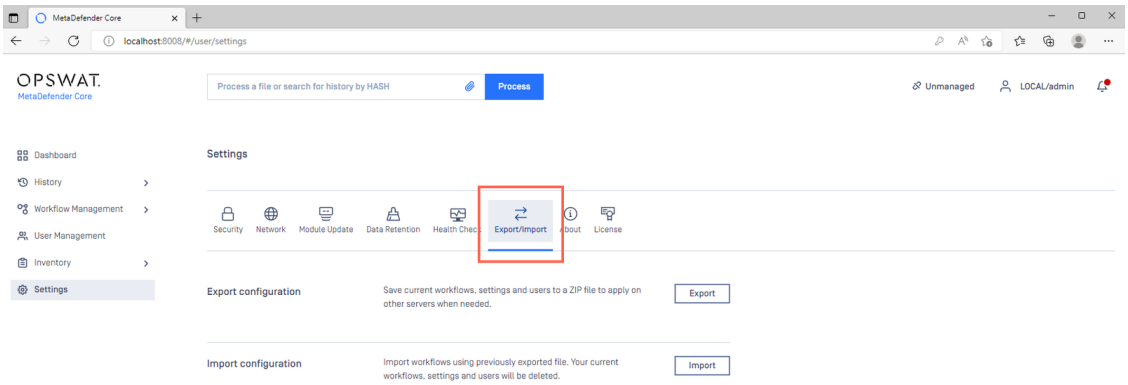
2. Visit My OPSWAT to download the latest MetaDefender Core version for your Linux distro.



3. Log in as **Root** user on the MetaDefender Core server, then **Run** the installer file.



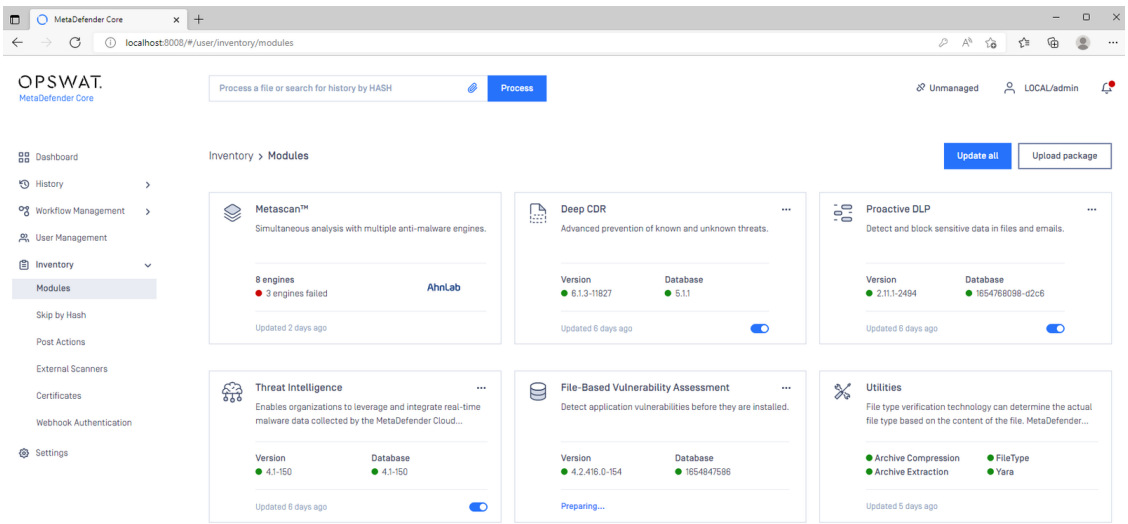
4. Depending on your distro, **Software Install** or a similar utility will guide you through the process.
5. For Shared Database mode, the same upgrade instructions apply as described above for Standalone mode, with one exception: during setup, you will need to specify your current PostgreSQL server IP address.
6. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more. With the help of this feature, and provided no errors occur during the upgrade process, your configurations and licensing information will be fully preserved through the upgrade process.



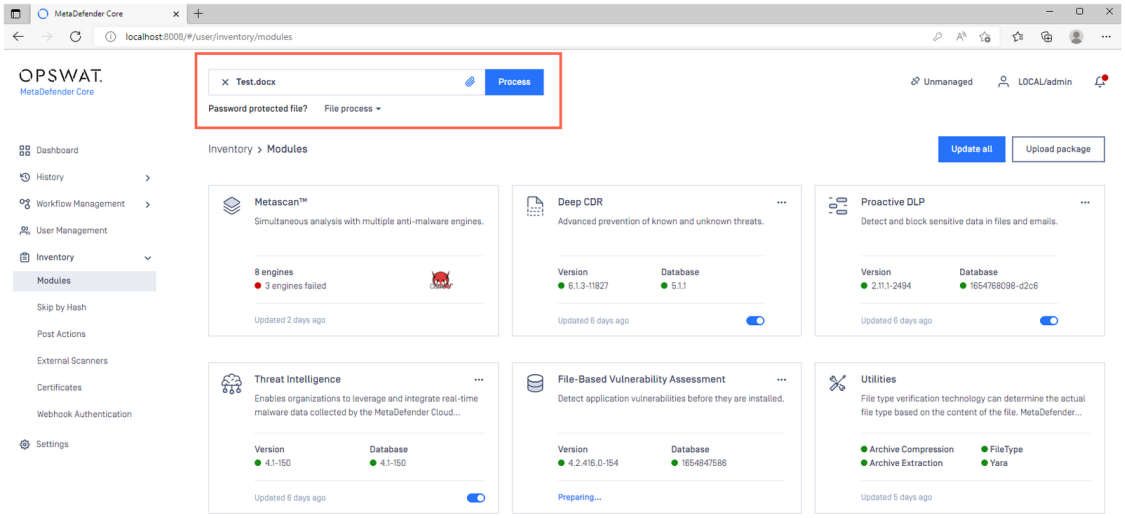
Please Note:

Once the installer has finished, your system will be upgraded to the latest version.

7. Once installed, click the MetaDefender Core icon to log into the **Management Console** and verify that the system is functioning as expected.
8. In the Console, open **Inventory>Modules** to view the status of your module and engine downloads.



8. Once all your required modules are up-to-date and enabled, and your **Security Policies** configured, drop a test file in the **Process** field to run your first scan.



Linux (Shared Database mode) command line

Please Note:

For command line installation in Linux, it is mandatory to use a MetaDefender Core ignition file to add PostgreSQL server information *before* installing MetaDefender Core 4.19.0 or later releases, as follows.

1. Log into the terminal as **Root** user.
2. Create an ignition file (if one does not already exist) via command line by running the following command: **touch ometascan.conf**
3. Now, run: **nano ometascan.conf** to edit the file with the necessary information.
4. An example of an ignition file is shown below.

Linux:/etc/opswat/ometascan.conf

ometascan.conf

```
eula=true

[global]
dbmode=4
instance_name=MetaDefenderCoreA
storage_path=D:\path\

[user]
name=admin
password=admin
email=admin@local

[dbserver]
```

```
type=remote
host=192.168.1.1
port=5432
user=postgres
password=non_Unicode_password
current_host=192.168.163.1
```

MetaDefender Core supports two modes to setup PostgreSQL server:

1. Local PostgreSQL Server

2. Existing PostgreSQL Server

Local PostgreSQL Server

Here, MetaDefender Core will install a new PostgreSQL server locally, alongside the product.

A sample ignition file for PostgreSQL server's information is shown below. Note that only non-Unicode characters are supported for both **user** and **password**.

ometascan.conf

```
[dbserver]
type=local
host=localhost
port=5432
user=postgres
password=non_Unicode_password
```

Finally, enter the relevant command from the table below to install MetaDefender Core.

Debian package (.deb)	*sudo dpkg -i <filename> || sudo apt-get install -f*
On Red Hat Enterprise 7.x Linux / CentOS package (.rpm)	*sudo yum install <filename>*
For systems which enabled GPD check flag	*sudo yum install --nogpgcheck <filename>*
Red Hat Enterprise Linux 8.x package (.rpm)	*sudo dnf install -y <filename>*

Existing PostgreSQL Server

Here, MetaDefender Core will connect to a pre-installed PostgreSQL server, running remotely, before setting up its database.

Be sure to follow the Open Connection steps on PostgreSQL server first, so that your existing remote PostgreSQL

server is ready to accept the upcoming connection from MetaDefender Core.

A sample ignition file for PostgreSQL server's information is shown below.

ometascan.conf

```
eula=true

[global]
dbmode=4
instance_name=MetaDefenderCoreA
storage_path=D:\path\

[user]
name=admin
password=admin
email=admin@local

[dbserver]
type=remote
host=192.168.1.1
port=5432
user=postgres
password=non_Unicode_password
current_host=192.168.163.1
```

Finally, enter the relevant command from the table below to install MetaDefender Core.

Debian package (.deb)	*sudo dpkg -i <filename> || sudo apt-get install -f*
On Red Hat Enterprise 7.x Linux / CentOS package (.rpm)	*sudo yum install <filename>*
For systems which enabled GPD check flag	*sudo yum install --nogpgcheck <filename>*
Red Hat Enterprise Linux 8.x package (.rpm)	*sudo dnf install -y <filename>*

Pro Tip:

For security reasons, after installing MetaDefender Core successfully, you may want to remove all credentials information created for the **[dbserver]** section in the ignition file.

Support:

For further guidance through the upgrade process, please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

Where can I get old engine versions?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Using older engine versions is **not recommended**, as they lack the latest features, detection capabilities, and security patches for third-party dependencies.

Exceptions:

- **Legacy OS compatibility**: If MD Core runs on an unsupported OS, an older engine version may be required and pinned to prevent auto-updates until an upgrade is possible.
- **Known issues in latest version**: If the latest engine has a confirmed issue with no available fix, a temporary rollback to an older version is acceptable.

Older versions should only be used temporarily. Upgrade to the latest supported version as soon as possible. If you need to **download previous versions** of the following engines:

- **File Type Engine**
- **Archive Engine**
- **Deep CDR Engine**
- **DLP Engine**
- **SBOM Engine**
- **COO Engine**

Please **contact the OPSWAT Support Team** for access to the installation files and further instructions.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

Are older antivirus engine module versions available for downgrade?

Check Your Version:

This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Overview

Customers may request to downgrade an antivirus engine module in MetaDefender Core for testing or validation purposes. However, **downgrading is not supported**, and older engine module versions are not maintained or distributed.

Attempting to revert to earlier versions may also introduce compatibility issues, stability concerns, and security risks, particularly when using outdated signatures.

Recommendation

For testing or validation scenarios, it is recommended to use the **latest available engine version** and ensure the system can successfully retrieve updates from the update server.

If short-term stability is required (for example, during troubleshooting or investigation), you can **temporarily prevent updates** by pinning the current engine version:

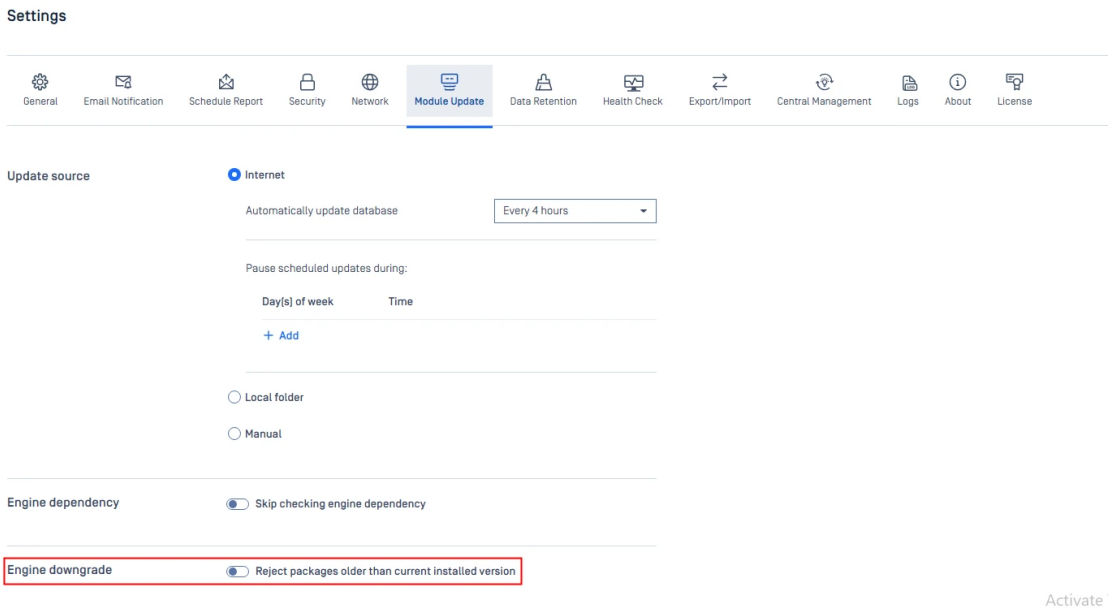
- Navigate to **Inventory > Modules** and confirm the engine and database version currently in use
- Go to **Settings > Module Update** and set updates to *manual* or adjust the schedule to pause updates during your maintenance window
- If using **OPSWAT Central Management**, ensure that central policies do not override these settings

This approach allows you to maintain a consistent environment for testing.

> This approach should only be used as a **temporary workaround**. Once testing or troubleshooting is complete, restore automatic updates to ensure the engine continues receiving the latest signatures and security fixes.

Starting from MetaDefender Core v5.19.0, a new **Prevent engine downgrade** setting is available to help protect deployments from unintentionally reverting to older engine or database packages.

When enabled, MetaDefender Core blocks installation of engine or database packages older than the versions currently installed. This helps reduce the risk of outdated threat definitions, compatibility issues, or accidental rollback scenarios.



Please note that intentional engine downgrades are still not officially supported, and older engine packages are not maintained or distributed for rollback purposes.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

[K8S] Upgrade of existing MD Core stack to a newer version by retaining existing shared database

Overview

This runbook documents the process used to upgrade the local MD Core deployment from `5.19.0` → `5.20.0` while keeping the existing PostgreSQL PVC (`DB_MODE=4`), database structure, and scan-processing history. The same re-deployment flow can be used to also upgrade to a higher version of the MD Core application. The below article would implement the steps using a local cluster created by Minikube, that hosts 3 pods in total:

Item	Value
Minikube profile	`md-core`
Application namespace	`md-core`
Helm release	`md-core`
PostgreSQL deployment	`postgres-core`

Item	Value
Existing PostgreSQL PVC	`postgres-core-20260722195541-60761`
Source database	`metadefender_core`
New database	`metadefender_core_5_20_0`
MD Core endpoints	http://192.168.59.107:8088/(POD1)http://192.168.59.107:8089/(POD2)

Safety Rules

Rule	Note
Do not delete active PVC	Do not delete the active PostgreSQL PVC before confirming which claim is mounted by `postgres-core`.
Pin PVC when redeploying	Do not run a normal app redeploy without pinning the existing PVC.
Avoid concurrent app pods	Do not run both MD Core application pods while doing a history migration.

Pre-Checks

Confirm the current context and live resources:

Code · bash

```
kubectl config current-context
kubectl get deploy,pod,svc,pvc -n md-core -o wide
helm history md-core -n md-core
```

Identify the active PostgreSQL PVC:

Code · bash

```
kubectl get deploy postgres-core -n md-core \
-o jsonpath='{range .spec.template.spec.volumes[*]}{.name}{"\t"}
{.persistentVolumeClaim.claimName}{"\n"}{end}'
```

For the current deployment, the active claim is: `postgres-core-20260722195541-60761`

Back Up The Source Database

Step 1 - Acquire pod name:

Code · bash

```
PGPOD="$(kubectl get pod -n md-core -l app=postgres-core -o
jsonpath='{.items[0].metadata.name}')"
```

Step 2 - Set backup path of choice:

Code · bash

```
BACKUP="/tmp/metadefender_core_pre_prod_5.20.0_history_migration_$(date -u +
%Y%m%dT%H%M%SZ).dump"
```

Step 3 - Generate the backup:

Code · bash

```
kubectl exec -n md-core "$PGPOD" -- sh -c 'PGPASSWORD="$POSTGRES_PASSWORD" pg_dump -U
"$POSTGRES_USER" -Fc -d metadefender_core' > "$BACKUP"
```

Step 4 - Ensure information has been stored:

Code · bash

```
ls -la /tmp/metadefender_core_pre_prod_5.20.0_history_migration_20260729T150523Z.dump
```

Stop MD Core During Migration

Scale the two application deployments down before running the database migration:

Code · bash

```
kubectl scale deploy/md-core-8088 deploy/md-core-8089 -n md-core --replicas=0
```

PostgreSQL must remain running !

Code · bash

```
kubectl rollout status deploy/md-core-8088 -n md-core --timeout=300s || true
kubectl rollout status deploy/md-core-8089 -n md-core --timeout=300s || true
kubectl get pods -n md-core -o wide
```

Run The 5.20.0 History Migration Pod

Create a temporary migration pod using the `5.20.0` image. This pod reads from `metadefender_core (source database)` and creates the versioned target database `metadefender_core_5_20_0`.

> This pod is a one-time migration runner and is sole purpose is to assist with the migration process - **this pod should be removed before the real Helm deployment is started again**.
>
> Deploy the migration pod separately using the below example temporary .YAML file - by adjusting the `image: opswat/metadefendercore-debian:5.20.0`:

The control flow of the migration pod executes the following:

1 - Start a temporary pod based on the `5.20.0` image.2 - Connect to the PG through the `postgres-core:5432`3 - Authenticate using the `mdcore-postgres-cred` K8S secret4 - Read the source database named `metadefender_core`5 - Create the versioned target database, which becomes `metadefender_core_5_20_0`6 - Copy/migrate config using the `MDCORE_UPGRADE_FROM_DB_NAME, UPGRADE_DB and MIGRATE_HISTORY` parameters and scan history into the new `metadefender_core_5_20_0`

For more information on the `values.yaml`:

metadefender-k8s/helm_charts/mdcore/values.yaml at main · OPSWAT/metadefender-k8s

Create the migration pod and apply the YAML:

Code · bash

```
kubectl apply -n md-core -f md-core-520-prod-history-migration.yaml
```

md-core-520-prod-history-migration.yaml

Code · yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: md-core-520-prod-history-migration
  labels:
    app: md-core-520-prod-history-migration
spec:
  restartPolicy: Never
  containers:
    - name: md-core
      image: opswat/metadefendercore-debian:5.20.0
      imagePullPolicy: IfNotPresent
```

```
      ports:
        - containerPort: 18191
          name: rest
      resources:
        requests:
          cpu: "2"
          memory: 6Gi
          ephemeral-storage: 10Gi
        limits:
          cpu: "6"
          memory: 12Gi
          ephemeral-storage: 40Gi
      env:
        - name: DB_MODE
          value: "4"
        - name: DB_TYPE
          value: remote
        - name: DB_HOST
          value: postgres-core
        - name: DB_PORT
          value: "5432"
        - name: DB_USER
          valueFrom:
            secretKeyRef:
              name: mdcore-postgres-cred
              key: user
        - name: DB_PWD
          valueFrom:
            secretKeyRef:
              name: mdcore-postgres-cred
              key: password
        - name: MDCORE_DB_NAME
          value: metadefender_core
        - name: MDCORE_UPGRADE_FROM_DB_NAME
          value: metadefender_core
        - name: UPGRADE_DB
          value: "true"
        - name: MIGRATE_HISTORY
          value: "true"
        - name: MD_USER
          valueFrom:
            secretKeyRef:
              name: mdcore-cred
              key: user
        - name: MD_PWD
          valueFrom:
            secretKeyRef:
              name: mdcore-cred
              key: password
        - name: APIKEY
          valueFrom:
```

```
            secretKeyRef:
              name: mdcore-api-key
              key: value
        - name: LICENSE_KEY
          valueFrom:
            secretKeyRef:
              name: mdcore-license-key
              key: value
        - name: MD_INSTANCE_NAME
          value: md-core-520-prod-history-migration
        - name: REST_PORT
          value: "18191"
        - name: CORE_CONF_JSON
          value: '{"global/restaddress": "0.0.0.0"}'
```

Watch for migration success:

Code · bash

```
kubectl logs -f -n md-core md-core-520-prod-history-migration
```

Expected success markers:

Code

```
Starting Migrate Database
Total migrate time
Total Upgrade schema time
Total Migrate and Upgrade schema time
Run Upgrade Successful
Upgrade successfuly
```

Verify History Was Preserved

Check the source and migrated database counts and confirm the migration of the scan-processing history:

Code · bash

```
PGPOD="$(kubectl get pod -n md-core -l app=postgres-core -o
jsonpath='{.items[0].metadata.name}')"

for db in metadefender_core metadefender_core_5_20_0; do
  echo "DB=$db"
  kubectl exec -n md-core "$PGPOD" -- sh -c \
    "PGPASSWORD=\"\$POSTGRES_PASSWORD\" psql -U \"\$POSTGRES_USER\" -d '$db' -Atc \"
      select 'scan.request', count(*) from scan.request
      union all select 'scan.scan_result', count(*) from scan.scan_result
      union all select 'warehouse.fact_engine_duration_by_rule_2026', count(*) from
```

```
warehouse.fact_engine_duration_by_rule_2026
      union all select 'warehouse.fact_scan_performed_by_user_2026', count(*) from
warehouse.fact_scan_performed_by_user_2026
      order by 1;
    \""
done
```

Update Runtime Values

After migration, normal MD Core runtime should use the migrated database and should not keep upgrade mode enabled.

In `mdcore-minikube-values.yaml`:

Code · yaml

```yaml
storage_provisioner: custom
pvc:
  enabled: false

env:
  MDCORE_DB_NAME: "metadefender_core_5_20_0"
  MDCORE_UPGRADE_FROM_DB_NAME: "metadefender_core"
  UPGRADE_DB: "false"
  MIGRATE_HISTORY: "false"
```

The host-networked core components + `init` containers should also use the `5.20.0` image:

Code · yaml

```yaml
#[POD1]
core_components:
  md-core-8088:
    image: opswat/metadefendercore-debian:5.20.0
    initContainers:
      - name: check-db-ready
        image: opswat/metadefendercore-debian:5.20.0

#[POD2]
 core_components:
  md-core-8089:
    image: opswat/metadefendercore-debian:5.20.0
        initContainers:
      - name: check-db-ready
        image: opswat/metadefendercore-debian:5.20.0
```

Render And Validate Helm

Render against the existing PostgreSQL PVC and dry-run to ensure configuration is accepted:

Code · bash

```
helm lint metadefender-k8s/helm_charts/mdcore -f mdcore-minikube-values.yaml
```

Then run the template against the `Existing PostgreSQL PVC`:

Code · bash

```
helm template md-core metadefender-k8s/helm_charts/mdcore \
  -n md-core \
  -f mdcore-minikube-values.yaml \
  --set-string core_components.postgres-core.storage_name=postgres-core-20260722195541-60761 > /
tmp/mdcore-520-render.yaml
```

Dry-run the template and ensure the new variables are set in place:

Code · bash

```
kubectl apply --dry-run=client -f /tmp/mdcore-520-render.yaml
```

Item	Expected
image	`opswat/metadefendercore-debian:5.20.0`
MDCORE_DB_NAME	`MDCORE_DB_NAME: "metadefender_core_5_20_0"`
UPGRADE_DB	`UPGRADE_DB: "false"`
MIGRATE_HISTORY	`MIGRATE_HISTORY: "false"`
claimName	`claimName: postgres-core-20260722195541-60761`

Finalize the the Helm Upgrade

Delete the temporary migration pod `md-core-520-prod-history-migration` before starting the real deployments! Remove the temporary migration pod:

`kubectl delete pod md-core-520-prod-history-migration -n md-core --ignore-not-found --wait=true`

Upgrade the release while **explicitly keeping the current PostgreSQL PVC**:

Code · bash

```
helm upgrade --install md-core metadefender-k8s/helm_charts/mdcore \
  --namespace md-core \
  --history-max 2 \
  -f mdcore-minikube-values.yaml \
  --set-string core_components.postgres-core.storage_name=postgres-core-20260722195541-60761
```

Await the uptime on both Core deployments to confirm the success of the upgrade:

Code · bash

```
kubectl rollout status deploy/md-core-8088 -n md-core
kubectl rollout status deploy/md-core-8089 -n md-core
kubectl get pods -n md-core -o wide
```

Confirm active PostgreSQL sessions are using the migrated database:

Code · bash

```
PGPOD="$(kubectl get pod -n md-core -l app=postgres-core -o
jsonpath='{.items[0].metadata.name}')"

kubectl exec -n md-core "$PGPOD" -- sh -c '
  PGPASSWORD="$POSTGRES_PASSWORD" psql -U "$POSTGRES_USER" -d postgres -Atc "
    select datname, count(*)
    from pg_stat_activity
    where datname like \$\$%metadefender_core\$\$
    group by datname
    order by datname;
  "'
```

If everything has been set properly in place and there haven't been no runtime error, the expected active database after upgrade should match: `metadefender_core_5_20_0`

Rollback Sequence

Data Divergence Warning: Any scan requests, audit records, or processing history generated while `5.20.0` was live exist solely in `metadefender_core_5_20_0`. Rolling back by pointing back to `metadefender_core` will **not** merge post-upgrade records back to the source database.

Because the original `metadefender_core` database remains intact, and assuming no pruning occurred in earlier

Helm revisions, you can run a `helm rollback`. There are two options to revert the changes.

> Keep the old source database `metadefender_core` until you are certain that the new `5.20.0` version is correct and properly deployed - only then it can be advisable to consider the option of removing any traces of the old one through a PG DROP statement and then removing obsolete not-bound PVCs.

Option 1: Change runtime values back to `5.20.0 -> 5.19.0` in the original configuration file

Configuration	Value / Details
env	`MDCORE_DB_NAME: metadefender_core UPGRADE_DB: "false" MIGRATE_HISTORY: "false"`
core_components	`#[POD1]core_components: md-core-8088: image: opswat/metadefendercore-debian:5.19.0 initContainers: - name: check-db-ready image: opswat/metadefendercore-debian:5.19.0 #[POD2] core_components: md-core-8089: image: opswat/metadefendercore-debian:5.19.0 initContainers: - name: check-db-ready image: opswat/metadefendercore-debian:5.19.0`

For a normal rollback sequence, `MDCORE_DB_NAME` is the important variable. `MDCORE_UPGRADE_FROM_DB_NAME` matters only during an upgrade; reverting it to `metadefender_core_5_20_0` while rolling back can be misleading and dangerous if someone later sets `UPGRADE_DB=true`.

> Since the original PVC would hold the information on both source and destination databases, you should keep it as it is - `postgres-core-20260722195541-60761`

Apply the re-adjustments through Helm:

Code · bash

```
helm upgrade --install md-core metadefender-k8s/helm_charts/mdcore \
  --namespace md-core \
  --history-max 2 \
  -f mdcore-minikube-values.yaml \
  --set-string core_components.postgres-core.storage_name=postgres-core-20260722195541-60761
```

Verify status:

Code · bash

```
kubectl rollout status deploy/md-core-8088 -n md-core
kubectl rollout status deploy/md-core-8089 -n md-core
```

Check if connections are on the original database:

Code · bash

```
PGPOD="$(kubectl get pod -n md-core -l app=postgres-core -o
jsonpath='{.items[0].metadata.name}')"

kubectl exec -n md-core "$PGPOD" -- sh -c '
  PGPASSWORD="$POSTGRES_PASSWORD" psql -U "$POSTGRES_USER" -d postgres -Atc "
    select datname, count(*)
    from pg_stat_activity
    where datname like \$\$%metadefender_core\$\$
    group by datname
    order by datname;
  "'
```

The rollback version `5.19.0` should point back to `metadefender_core`, not `metadefender_core_5_20_0`.

Option 2: Use helm rollback if previous stable revision is present

Find the last good 5.19.0 revision, for example revision 24:

Code · bash

```
helm history md-core -n md-core
```

The `helm rollback` restores the old release values exactly - if that old revision already pointed to the `5.20.0` or to a different PVC, rollback would not return you to `5.19.0` safely, so kindly consider verifying that before running it.

Inspect what that revision would restore:

Code · bash

```
helm get values md-core -n md-core --revision 24 --all
```

Confirm that the revision points to:

Code · bash

```
  image: opswat/metadefendercore-debian:5.19.0
```

```
  MDCORE_DB_NAME: metadefender_core
  PVC: postgres-core-20260722195541-60761
```

Rollback through Helm:

Code · bash

```
helm rollback md-core 24 -n md-core --wait --timeout 15m
```

Verify pods and PG status:

Code · bash

```
  kubectl rollout status deploy/md-core-8088 -n md-core
  kubectl rollout status deploy/md-core-8089 -n md-core
```

References

Reference	Link
OPSWAT MD Core Kubernetes upgrade documentation	https://www.opswat.com/docs/mdcore/container-deployment/upgrade-metadefender-core-version-on-k8s
OPSWAT MD Core Docker image documentation	https://www.opswat.com/docs/mdcore/container-deployment/docker-image-published-on-opswat-docker-hub

Support:

If **further assistance** is required, please log a **support case or chat with our support engineer**.

How do I plan and perform a MetaDefender Core version upgrade?

Overview

This article is the complete planning reference for upgrading MetaDefender Core. It covers how to choose the right upgrade path based on your current version, what to prepare before any upgrade, what happens to your database during an upgrade, and how to handle special situations such as End-of-Life versions, large version gaps, and limited disk space. The detailed step-by-step upgrade procedure for each deployment type is maintained in a separate

dedicated article, linked from the "Upgrade procedure" section below.

Quick answers

Question	Answer
Are version upgrades free?	Yes. The latest version is free for all active license holders to download and run.
Are version upgrades automatic?	No. Product version upgrades are always manual. Only feature updates within the product (such as engine updates, which are covered separately) can be set to automatic.
What is the latest version?	Check the Release Notes, or log in to My OPSWAT -> go to Product Downloads MetaDefender Core -> click on the Download button.
Can I downgrade after upgrading?	No. There is no in-place downgrade. The only rollback path is restoring the snapshot or backup taken before the upgrade.
Can I jump directly to the latest version?	It depends on the size of the version gap — see "Choosing your upgrade path" below.

OPSWAT recommends upgrading to the latest version as soon as practical and not waiting until the end of a release's support cycle. Current versions include the latest features, bug fixes, and security patches, and bug fixes are only ever applied to the next release — never backported to the release you are on.

Step 1 — Check your support status (End-of-Life policy)

Before planning any upgrade, confirm whether your current version is still supported.

OPSWAT supports each MetaDefender Core release for **18 months after the publication of the following release**. Versions not listed on the support lifecycle page are no longer supported, and **Version 3 and earlier releases are out of support entirely**.

See: How long is the support lifecycle for a specific version of MetaDefender Core?

If your current version is **End-of-Life (EoL)**:

- Upgrading from an EoL version is **not a supported upgrade path**. It may still succeed, but it is best-effort only and carries a meaningfully higher risk of database migration failure. Architectural differences accumulate over time — database schema changes, altered data relationships, and third-party dependency updates — and very old versions may not be able to upgrade directly at all.
- Running an unsupported version can cause **engine failures at any time**, independently of any upgrade.

- Installers for very old releases may **no longer be available for download** from My OPSWAT. If an intermediate version you were counting on is unavailable, an incremental upgrade path may not be completable — verify installer availability before committing to that path.
- If in doubt, contact OPSWAT Support for assistance planning the upgrade before you start.

Step 2 — Prepare (applies to every upgrade)

1. **Review the target version's requirements.** Check the Recommended System Configuration for the version you are upgrading **to**, not the one you are on. Note in particular:
   - **Windows:** Microsoft Visual C++ Redistributable for Visual Studio 2015–2022 (x64), **version 14.44 or later**, is required by current releases. An outdated redistributable will cause the installation to fail.
   - **PostgreSQL:** some MetaDefender Core versions are only compatible with certain PostgreSQL versions. Check the release notes and system requirements of the target version.
2. **Take a snapshot or full backup** of the machine(s) hosting the application and the database (for virtual machines, a disk snapshot; for physical systems, a full backup). This is the only reliable rollback path — there is no in-place downgrade.
3. **Export your configuration** from the Management Console using the built-in Config Export/Import feature (see Import/Export Configuration). Provided no errors occur during the upgrade, this preserves your configuration and licensing information through the process.
   - Note: a configuration package exported from a much older version may not import successfully into a current release, due to accumulated changes in the configuration schema. For large version gaps, also take screenshots or notes of your workflows, settings, and user management as a fallback for manual reconfiguration.
4. **Export your processing history** if you need it for your records: History > Processing > Export History.
5. **Check available disk space.** Major upgrades can require far more temporary space than the product uses day to day — see "Disk space requirements" below. As a rule of thumb, ensure available free space is **larger than your Core database size**.
6. **Plan a maintenance window.** Database migration time scales with the size of your scan history; a large processing history can extend the upgrade considerably.
7. **MetaDefender Core Custom installations** (installations provided by OPSWAT that include custom engines): contact OPSWAT Support before upgrading for guidance on upgrading the custom engines.

Step 3 — Choose your upgrade path

The deciding factor is the **size of the version gap** between your current version and the target. The wider the gap, the more you should break the upgrade into stages. This applies both when crossing major versions (4.x to 5.x) **and** when spanning many releases within the same major version (for example, 5.12.x to 5.21.x).

Your situation	Recommended path
1–2 releases behind the target	**Path A** — direct upgrade
3 or more releases behind, same major version (e.g. 5.12 → 5.21)	**Path B** — incremental upgrade, 2–3 versions per stage
Crossing major versions (4.19 or later → 5.x)	**Path B** — incremental upgrade through an early 5.x release
Running version 4.18 or older	**Path C** — clean installation (required; see database note below)
Current version is End-of-Life	**Path C** recommended; Path B is best-effort only
Historical processing data is not required	**Path C** — regardless of gap size, this is the faster and more reliable route

Version-specific notes:

- **Version 4.18 and older** use a SQLite database, whereas newer versions use PostgreSQL. There is no migration plan for these versions: a complete uninstall and fresh installation is required (Path C). Take screenshots and/or notes of your current settings, user management, and workflows before uninstalling, as they will be lost.
- **Version 4.19 and newer** use PostgreSQL and can follow the incremental path (Path B) or, for small gaps, the direct path (Path A).
- **Version 5.11.1:** upgrade directly to 5.14 or newer.

Path A — Direct upgrade (small version gap)

V5 releases support quick manual installation of the new version directly over the existing installation. Follow the steps in the "Upgrade procedure" section below.

Path B — Incremental upgrade (large version gap)

Use this when you need to preserve your existing database and processing history across a large gap.

1. Complete all preparation steps above (snapshot, config export, requirements check).
2. Divide the version gap into stages of roughly **2–3 versions** each. For example, starting from 4.19: 4.19 > 5.0 > 5.2 > 5.4 > 5.6 > 5.8, and so on, continuing in similar steps until you reach a supported version — then upgrade to the latest.
3. At each stage, review the Release Notes to confirm the version you plan to land on is a stable, suitable stopping point, and re-check the system requirements — a version in the middle of your ladder may introduce a requirement

(such as a PostgreSQL or Visual C++ Redistributable version) that your host does not yet meet.

4. On older versions you may need to stop the "OPSWAT MetaDefender Core" service before running the installer:
   - **Windows:** stop the service via services.msc. *[IMAGE — reuse from "How to upgrade an End-of-Life [EOL] MetaDefender Core?": Windows services screenshot]*
   - **Linux:** `sudo systemctl stop ometascan`
5. Run the installer for that stage and let the database migration complete fully. **Do not interrupt it** — on a large processing history, a long-running migration is expected behavior, not a hang.
6. After each stage, reboot the machine for the changes to take effect, then verify the deployment is healthy: the Management Console loads, services are running, and engines update successfully.
7. Repeat until you reach the target version.

Understand the trade-offs before choosing this path:

- It is time-intensive, and every stage requires validation.
- There is no formal, certified ladder of approved intermediate versions. Choosing the stopping points is done from the release notes on a case-by-case basis.
- If the source version is very old, there is no guarantee the path completes cleanly. It is possible to invest several hours and still end with a database error, at which point a clean installation (Path C) becomes the only remaining option.

Path C — Clean installation

Use this when: the current version is 4.18 or older, the current version is End-of-Life, the gap is very large and historical data is not required, or an incremental attempt has failed.

1. Export the configuration from Settings, and take screenshots and/or notes of your workflows, settings, and user management (for very old versions, the notes are the reliable record — the export may not import into the new release).
2. Export your processing history for your records: History > Processing > Export History.
3. Take a snapshot or backup of the existing instance.
4. Uninstall the current version.
5. Install the latest MetaDefender Core release with a clean database.
6. Import the configuration package, or reconfigure manually from your notes/screenshots.
7. Validate scanning behavior against the previous configuration before decommissioning anything.

Trade-off: you lose historical processing data, and settings may need to be rebuilt manually. In exchange, this is by a wide margin the cleaner, faster, and more predictable route for a large version gap.

Upgrade procedure

The full step-by-step upgrade procedure — GUI and command line, for Windows and Linux, in both Standalone and Shared Database modes — is maintained in a dedicated article:

How do I manually upgrade to the latest MetaDefender Core version?

The same procedure applies whether you are performing a direct upgrade (Path A) or one stage of an incremental upgrade (Path B).

Why the version gap matters (database migration)

The installer does not simply overwrite the previous version — it migrates the database schema forward **one version at a time**, from your current schema version through every intermediate revision to the target. For example, a database at schema version 1 (introduced with 4.19.0) upgrading to schema version 53 passes through all fifty-three steps in a single run.

Two consequences follow:

- **Duration.** The migration can take a long time, and it scales with the volume of scan history stored.
- **Risk.** Every additional schema step is another opportunity for the migration to fail. Failures mid-migration are what make very large single jumps unattractive.

Breaking the upgrade into stages reduces the number of schema transitions applied in any single run — which is exactly why the incremental path is more survivable than one large jump.

Disk space requirements

When upgrading to a much newer version, the process may report requiring **well over 150 GB** of available disk space even if the installation itself only occupies a modest amount (for example, 40 GB). This is because major upgrades may:

- Temporarily **duplicate data** for migration or backup — ensure available free space is larger than the Core database size;
- **Extract and stage** large upgrade packages;
- Create **fallback points** in case the upgrade needs to be rolled back.

If disk space is the barrier, options include:

1. Reduce existing data usage — see How to manage disk space usage in the pg_data\base directory
2. Adjust Data Retention settings — see What can I do when MetaDefender Core (Windows) has an insufficient disk space error?
3. Move the Temp and Quarantine folders to a drive with more space — see How can the Temp folder be changed?
4. Perform a fresh installation on a new machine with sufficient space (Path C).

If an upgrade has already failed on disk space, collect the following before retrying and attach it to your support case: total and available disk space on all drives, a screenshot of the installation folder Properties, a screenshot of the `data` folder Properties, the upgrade log mentioning the space requirement, and a support package.

Version-specific checkpoints

Always read the release notes for every version in your upgrade range. Known checkpoints:

- **Weak TLS ciphers (5.13.3 and later):** MetaDefender Core's NGINX web server will not start if weak cipher suites are configured for HTTPS. OpenSSL 1.x has been replaced with OpenSSL 3.x, affecting NGINX and PostgreSQL dependencies. Review and update your SSL/TLS configuration to remove unsupported cipher suites **before** upgrading, or the NGINX service will fail to start post-upgrade.
- **Syslog Server Configuration API (5.13.0 and later):** Syslog settings can be configured via API in addition to the GUI. Existing configurations are unaffected, but changes made through the API require a restart of the MetaDefender Core service to take effect.
- **Visual C++ Redistributable (Windows):** current releases require version 14.44 or later of the Microsoft Visual C++ Redistributable 2015–2022 (x64). Verify before upgrading.
- **PostgreSQL compatibility:** some MetaDefender Core versions are only compatible with certain PostgreSQL versions — check the release notes and system requirements of every target version in your ladder.
- **Version 5.11.1:** upgrade directly to 5.14 or newer.

References

Item	Detail
Upgrade cost	Free for all active license holders
Upgrade trigger	Always manual — no automatic product upgrades
Recommended increment for large gaps	2–3 versions per stage
Gap rule applies to	Major-version crossings (4.x → 5.x) and wide same-major gaps (e.g. 5.12 → 5.21)
Database boundary	4.18 and older: SQLite (fresh install required); 4.19.0 and newer: PostgreSQL (schema migrated sequentially)
Support lifecycle	18 months after the following release is published
Unsupported releases	Version 3 and earlier; any version not listed on the support lifecycle page
Windows prerequisite	Microsoft Visual C++ Redistributable 2015–2022 (x64), 14.44 or later

Item	Detail
Mandatory safeguard	Snapshot or full backup of application and database hosts
Rollback method	Restore from snapshot or backup — no in-place downgrade exists
Disk space rule of thumb	Free space larger than the Core database size; major upgrades may require 150+ GB temporarily

Notes

- The incremental path is guidance, not a certified upgrade matrix. OPSWAT does not publish a fixed list of approved intermediate versions; the correct landing points are determined from the release notes for the versions in your specific range.
- Confirm that installers for your chosen intermediate versions are still available on My OPSWAT before starting — an unavailable intermediate build will block the ladder partway through.
- Do not interrupt a database migration that appears to be taking a long time. On a large processing history this is expected behavior, not a hang.
- Testing the upgrade in a staging environment before the production rollout is strongly recommended, especially for large gaps.
- If your current version is End-of-Life, or you are unsure which path applies to your deployment, open a support case before starting.

Support: If you have questions, concerns or want to report issues regarding **MetaDefender Core**, please open a Support Case with the OPSWAT team via phone, online chat or form, or feel free to ask the community on our OPSWAT Expert Forum.

How to recover MetaDefender Core administrator access using an existing admin API key

Check Your Version:

This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Overview

If you have forgotten your MetaDefender Core administrator password, you can use the **Forgot password?** option when SMTP is configured and the account has a valid, accessible email address.

If SMTP is not configured, but you still have a valid administrator API key, you can create a temporary administrator account through the API. You can then sign in with this account and reset the original administrator's password.

If you have also forgotten the original administrator username, the procedure below explains how to retrieve it.

Prerequisites

- A valid administrator API key with permission to manage users.
- Access to the MetaDefender Core API and Web Management Console.
- A MetaDefender Core version that provides the **Admin direct reset** option.

The commands below use the default address `http://localhost:8008`. Replace the address with your MetaDefender Core endpoint and HTTPS if applicable. Use the Linux/macOS examples in Bash-compatible shells. On Windows, use the separate `curl.exe` PowerShell examples because PowerShell uses different line continuation and quoting rules.

Procedure

Step 1: Retrieve the original administrator username

If you have forgotten the original administrator username, run the following request to list existing users.

Linux/macOS:

Code

```
curl -sS --request GET \
  --url "http://localhost:8008/admin/user" \
  --header "apikey: <ADMIN_API_KEY>"
```

Windows PowerShell:

Code

```
curl.exe -sS --request GET `
  --url "http://localhost:8008/admin/user" `
  --header "apikey: <ADMIN_API_KEY>"
```

Replace `<ADMIN_API_KEY>` with your existing administrator API key.

Locate the original administrator account in the response and record its `name` value. This is the username used to sign in.

If you still know the password, you can use this username to sign in without completing the remaining recovery steps.

Step 2: Create a temporary administrator account

Run the following request.

Linux/macOS:

Code

```
curl -sS --request POST \
  --url "http://localhost:8008/admin/user" \
  --header "apikey: <ADMIN_API_KEY>" \
  --header "Content-Type: application/json" \
  --data '{
    "name": "recovery_admin",
    "display_name": "Recovery Administrator",
    "email": "recovery@example.com",
    "password": "<TEMPORARY_ADMIN_PASSWORD>",
    "roles": ["1"],
    "directory_id": 1
  }'
```

Windows PowerShell:

Code

```
$body = @'
{
  "name": "recovery_admin",
  "display_name": "Recovery Administrator",
  "email": "recovery@example.com",
  "password": "<TEMPORARY_ADMIN_PASSWORD>",
  "roles": ["1"],
  "directory_id": 1
}
'@

Invoke-RestMethod -Method Post -Uri "http://localhost:8008/admin/user" `
  -Headers @{ apikey = "<ADMIN_API_KEY>" } `
  -ContentType "application/json" `
  -Body $body
```

Before running the request:

- Replace `<ADMIN_API_KEY>` with your existing administrator API key.
- Replace `<TEMPORARY_ADMIN_PASSWORD>` with a unique password that meets the configured password policy.
- Replace `recovery@example.com` with an appropriate administrator-controlled email address.
- Use a different temporary username if `recovery_admin` already exists.

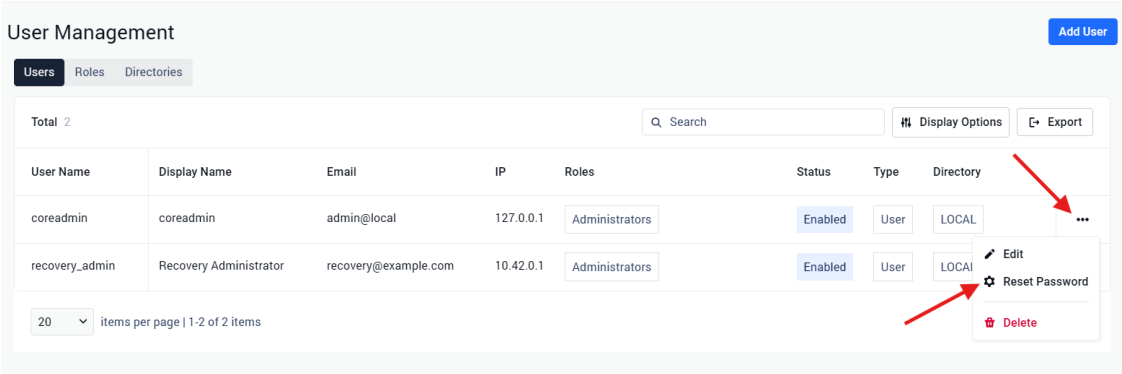
Review the API response and confirm that the account was created successfully.

Step 3: Sign in using the temporary administrator

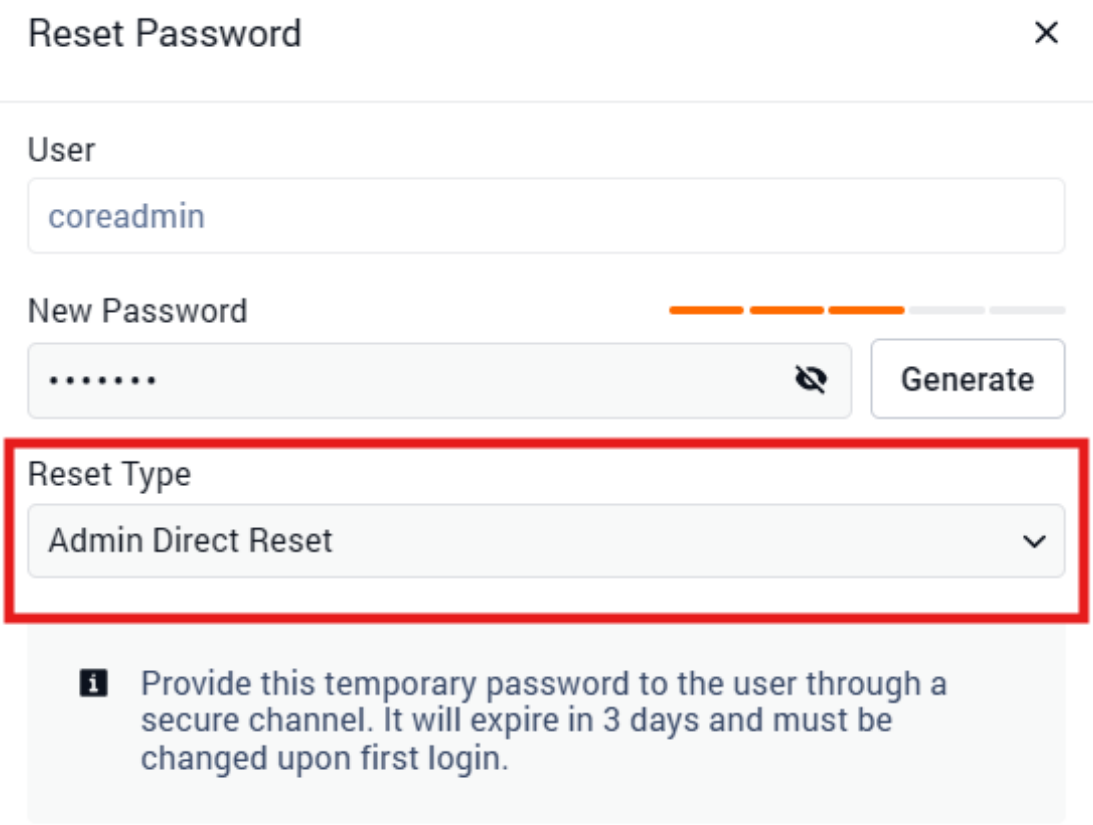
1. Open the MetaDefender Core Web Management Console.
2. Sign in using `recovery_admin` and the password specified in Step 2.
3. Navigate to **User Management > Users and groups**.

Step 4: Reset the original administrator's password

1. Locate the original administrator account.
2. Click the three-dot menu next to the account and select **Reset password**.



3. Enter a new temporary password that meets the configured password policy.
4. Set **Reset type** to **Admin direct reset**.



5. Click **Reset** to apply the change.

Step 5: Sign in using the original administrator account

1. Sign out of the temporary administrator account.
2. Sign in using the original administrator username and the temporary password set in Step 4.
3. When prompted, change the temporary password to a new password.
4. Confirm that you can access the Web Management Console and administrator functions.

Step 6: Remove the temporary administrator account

After confirming that access to the original administrator account has been restored, delete `recovery_admin` from **User Management > Users and groups**.

Support:

If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Troubleshooting MetaDefender Core

In this section you can find solutions for generic issues with MetaDefender Core

Installation issues

- Inaccessible Management Console

Where are the Metadefender Core logs located?

MetaDefender Core generates log files to **/var/log/ometascan** under Linux and to Windows Event Log under Windows.

The server and node logs are collected separately and are plain text files. For more information on how to read the logs, go to How to Read the Metadefender Core Log?

How can I create a support package?

To ensure the best help from OPSWAT support, you can create a support package with a tool that comes with MetaDefender Core.

For more information on how to create a support package, go to How to Create Support Package?

Which data is contains in a support package

This document describes the categories of data collected by the MetaDefender Core support package script (`ometascan-collect-support-data.bat/sh`). This package is generated on request to help OPSWAT technical support diagnose issues on a customer's system.

What Is Collected

1. Hardware Information

Data	Description
CPU details	Processor model, speed, core count, and other properties
Memory details	Physical RAM modules, capacity, and specifications
Disk drives	Physical disk drive model, size, and interface type
Logical disks	Drive letters, file system type, used/free space

2. Operating System Information

Data	Description
OS version	Windows/Linux edition, version number, build number
OS configuration	System locale, install date, registered organization

3. Network Information

Data	Description
Network interfaces	IP addresses, MAC addresses, DNS servers, DHCP settings
Routing table	All active network routes on the machine

4. Installation Directory Listing

A recursive listing of all files and folders under the MetaDefender Core installation directory. This includes file names, sizes, and timestamps — **not** file contents.

5. Configuration Data

Common configuration: via Registry (Windows) or Configuration file (Linux)

6. Database Information

Data	Description
Exported database schema	A structured export of the MetaDefender Core internal database
PostgreSQL configuration	Configuration files (`*.conf`) from the embedded PostgreSQL instance
PostgreSQL data files	Non-binary configuration/settings files from the `pg_data` directory
PostgreSQL logs	Files related to any database upgrade operation (`pg_upgrade*`) and operational logs

Note: The database export may contain scan job metadata and product configuration. It does **not** contain the actual

file content of scanned files.

7. Application Logs

Data	Description
MetaDefender Core application logs	Operational events, errors, warnings from the core service
Nginx web server logs of MetaDefender Core	HTTP request/error logs for the management interface
Engine logs (AV engine, Filetype, Deep CDR, etc)	Diagnostic output from individual anti-malware engines and other engines
Windows event or Syslog (Linux)	We only collect windows event or syslog belong to the application, **not all of the event/logs**

8. Crash Dump Files

If any of the following processes have previously crashed, their crash dump files are included:

- `ometascan.exe`
- `engineprocess.exe`
- `ocmdatatransporter.exe`

What Is NOT Collected

- The **actual content of files** that were submitted for scanning
- **User credentials** or passwords (these are not stored in any of the collected sources)
- Data from **outside the MetaDefender Core installation directory** (except process info, network info, and OS info described above)

Privacy Considerations

Customers who have concerns about specific data categories can use the following options before generating the package:

- Use `-enable-redact` to redact potentially sensitive values from log files.
- Use `-lite` to limit the volume of data collected (fewer logs, most recent crash dump only, no warehouse database).
- Use `-start-date` and `-end-date` to restrict log collection to a specific time window.

- Contact OPSWAT support to discuss which specific components are needed for your case before generating a full package.

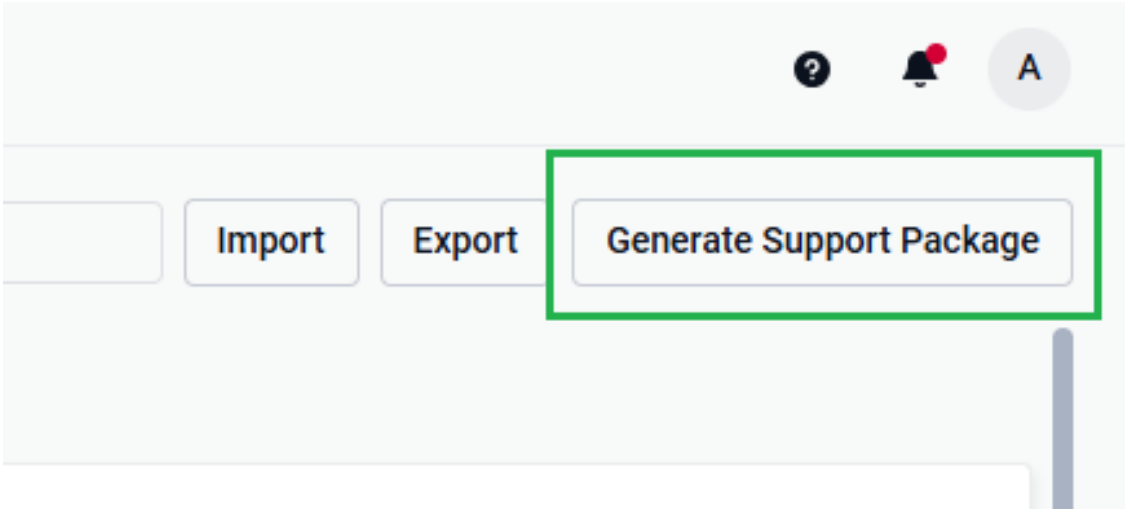
How to Create Support Package With Web Console

Check Your Version:

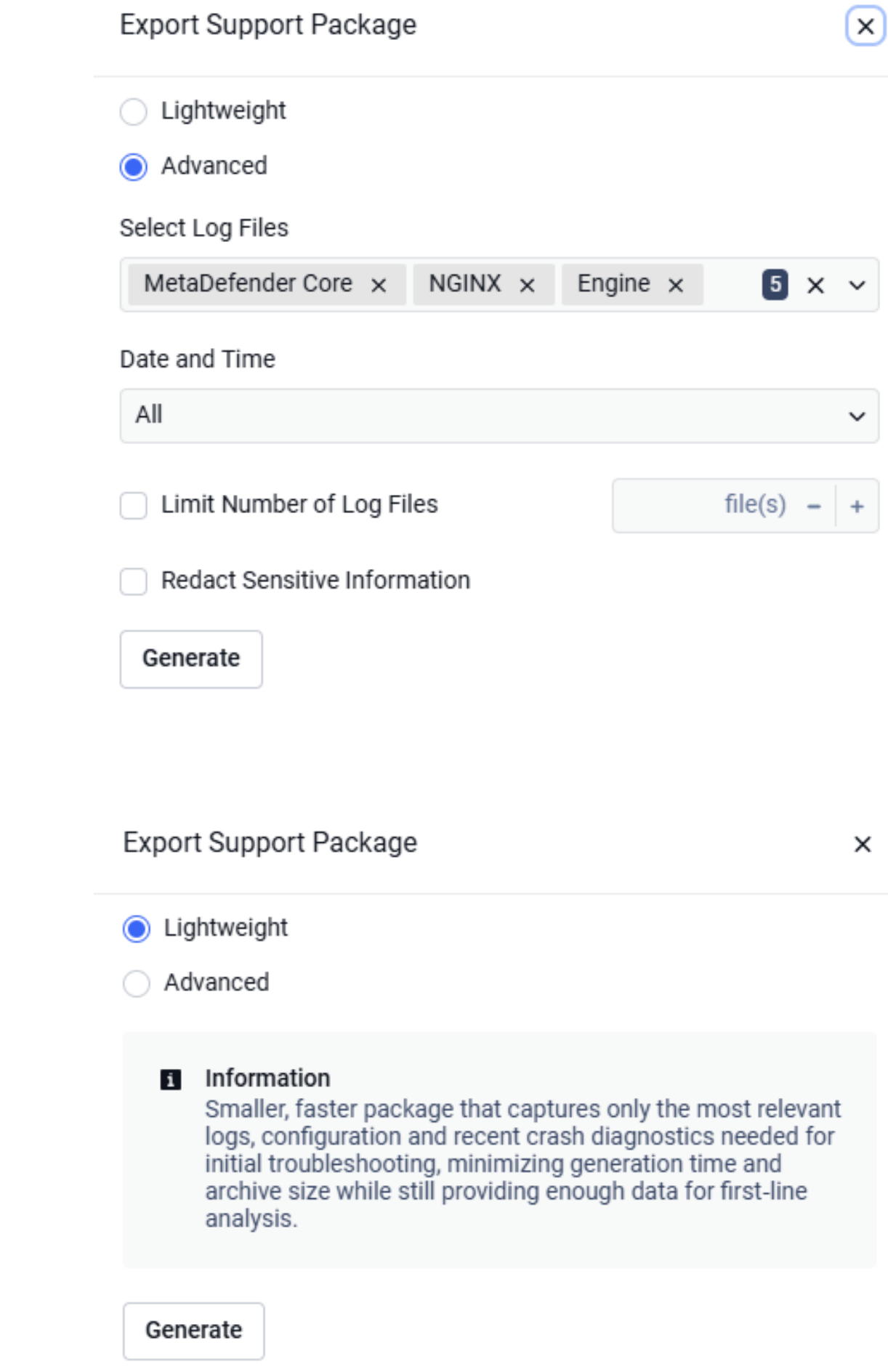
This article applies to all MetaDefender Core releases **after version 5.13.0**, deployed on both Windows and Linux systems.

Kindly note that the 'Generate Support Package' button is not available in versions earlier than 5.13.0.

Please go to Settings > On the top right click "Generate Support Package"



After that, a sliding windows will open > "Generate"


Export Support Package
Lightweight
Advanced
Select Log Files
MetaDefender Core
NGINX
Engine
5
Date and Time
All
Limit Number of Log Files
file(s)
Redact Sensitive Information
Generate
Export Support Package
Lightweight
Advanced
Information
Smaller, faster package that captures only the most relevant logs, configuration and recent crash diagnostics needed for initial troubleshooting, minimizing generation time and archive size while still providing enough data for first-line analysis.
Generate

Advance

Info

Users can select which kind of log you want to collect:

- Core: MetaDefender Core's logs
- Nginx: log of nginx
- Engine: log files belong the engines (AV engine, Deep CDR, etc)
- Database: log of PostgreSQL database (if using bundled)
- System: Windows event logs or Linux syslog

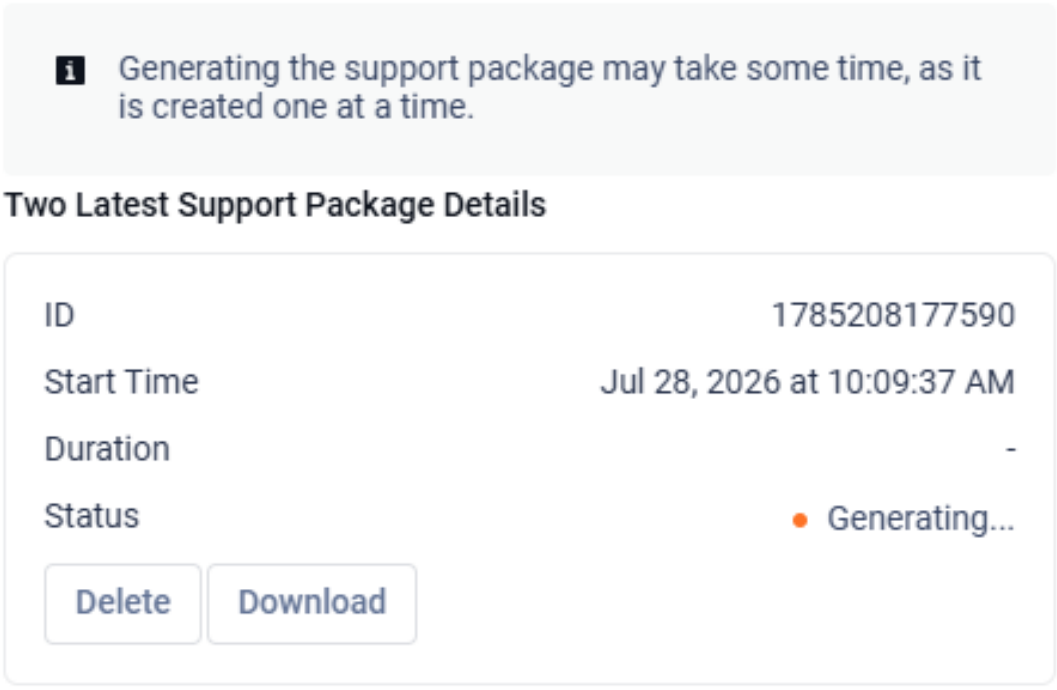
The "Limit number of log file" option allows users to choose how many log files the supported package should collect.

Date and Time: Users can select the logs within a specific date and time range.

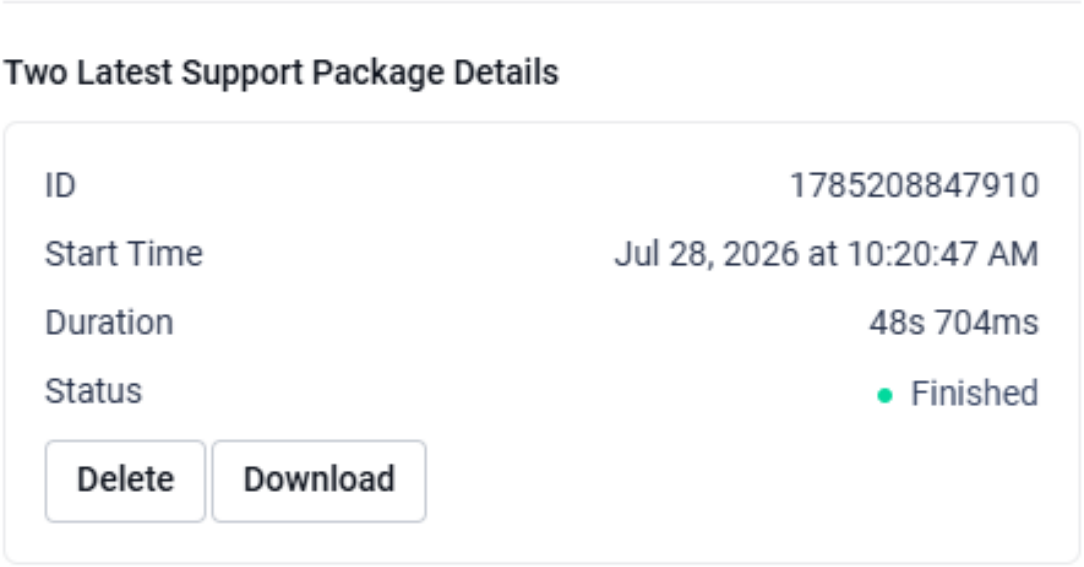
Lightweight

Smaller, faster package that captures only the most relevant logs, configuration and recent crash diagnostics needed for initial troubleshooting, minimizing generation time and archive size while still providing enough data for first-line analysis. With this option, the support package will be generated with predefine options, users don't need to care about the options.

After click `Generate` Please wait for awhile for generating process



After the generating process was done, you can select the support package and download it



You can also find the support package on: Installation folder/data/sp_package.

But we recommend that you should download support package from the UI to avoid break the file.

Support:

If Further Assistance is required, please proceed to **create a support case or chat with our support engineer**.

How to Create Support Package With Bundle Tools?

A support package contains essential information regarding the operating system and OPSWAT software found on the machine.

Code

```
#Windows
ometascan-collect-support-data.bat

#Linux
ometascan-collect-support-data.sh
```

Parameters (optional):

-max-log N

- Define a number of log files to be gathered and added into support package.
- The tool sorts log files by Date Modified with descending order.
- If not set, or set 0, or set a negative number, the tool will ignore it and collect all log files.

Warning

The `-max-log` parameter is calculated based on the number of log files, not the number of days. Since MetaDefender Core can generate multiple log files in a single day, please ensure that the calculation accurately reflects the total number of log files.

-enable-redact 1|0

- Hide sensitive information in the support package.
- 1 is enable, 0 is disable. The default is 0 and will not redact sensitive information.
- The information will be redact: IP address, MAC, file path.

-start-date and -end-date

These 2 options will accept either *Epoch timestamp (in seconds)* or *ISO 8061 date format* `YYYY-MM-DD` with local time zone.

Example:

- Epoch timestamp (second) format `ometascan-collect-support-data.bat -start-date 1773766800 -end-date 1773853200`
- ISO 8061 timestamp format `ometascan-collect-support-data.bat -start-date 2026-03-18 -end-date 2026-03-19` -> collect logs from *2026-03-18 00:00:00* to *2026-03-19 00:00:00 (UTC+7:00)* Note: We support this format to make our script easier to use for CX / customer.

-lite

Use this option when users want to create a lightweight support package, reduce size and generation time. When use this option, users do not need to care about other options.

For example (Linux)

Code · bash

```
/usr/bin/ometascan-collect-support-data.sh -lite
```

Creating the package on Linux

Run **/usr/bin/ometascan-collect-support-data.sh**

As the script processes the necessary information, the script generates the support package output.

The package files are tar.gz archive with the following name:

Code

```
ometascan-<VERSION>-support-<TIMESTAMP>.tar.gz
```

Where the timestamp is the date when the package was generated.

Example:

```
Code

ometascan-5.14.1-1-support-1439983514.tar.gz
```

The generated package will be placed in the current working directory.

Creating the package on Windows

Run script found under the installation directory of the product, default is **C:\Program Files\OPSWAT\Metadefender Core\ometascan-collect-support-data.bat**

As the script processes the necessary information, the script generates the support package output.

The package files is a zip archive with the following name:

```
Code

ometascan-<VERSION>-support-<TIMESTAMP>.zip
```

Where the timestamp is the date when the package was generated.

Example:

```
Code

ometascan-5.14.1-1-support-1439983514.zip
```

The generated package will be placed in the same location as the script that was called.

> **Info**
>
> If the support package too big in size, we can also input -max-log N for to collect logs for last N logs. For example
>
> **ometascan-collect-support-data.bat -max-log 15**
>
> Will collect logs for last 15 newest logs.

Support Package contents

The support package contains the following elements:

- **configuration** : the configuration files of OPSWAT software found on machine
- **log** : the log files of OPSWAT software found on machine
- **system information** : system information stored in file named os.info

- **hardware information:** hardware information stored in file named hw.info
- **network information:** network information stored in file named network.info
- **directory information:** OPSWAT software directory information stored in file named files.info
- **copy of config database** : config database **WITHOUT** user data

You can check the content of the generated package to make sure it does not contain any confidential information.

Windows Temporary Folder Path Changes for SYSTEM Account

Overview

Microsoft has recently (Feb 2026) introduced security improvements that change the default temporary folder location for services and applications running under the **SYSTEM** account.

This is a Windows operating system change and is not related to any changes in MetaDefender Core product.

What Changed?

Microsoft introduced a new secure temporary folder location called `C:\Windows\SystemTemp` specifically for processes running as the SYSTEM account. This folder has stricter security permissions to prevent path redirection attacks.

Previous Behavior

- SYSTEM account temp folder: `C:\Windows\Temp`

New Behavior

- SYSTEM account temp folder: `C:\Windows\SystemTemp`

Note: This change only affects applications running as the **SYSTEM** account. Applications running under user accounts are not affected and will continue to use their existing temp folder locations.

What You Might Observe

If your environment is affected, you may notice:

- Temporary files being created in `C:\Windows\SystemTemp` instead of `C:\Windows\Temp` .
- Applications that hardcode the path to `C:\Windows\Temp` may not find expected temporary files.
- Increased disk usage in the `C:\Windows\SystemTemp` directory.

For example, the temporary folder for NGINX now will be in `C:\Windows\SystemTemp\ometascan\`

Impact on MetaDefender Core

Currently, there is no known impact on MetaDefender Core services. However, if you experience any issues related to

this Windows change, please contact OPSWAT Support for assistance.

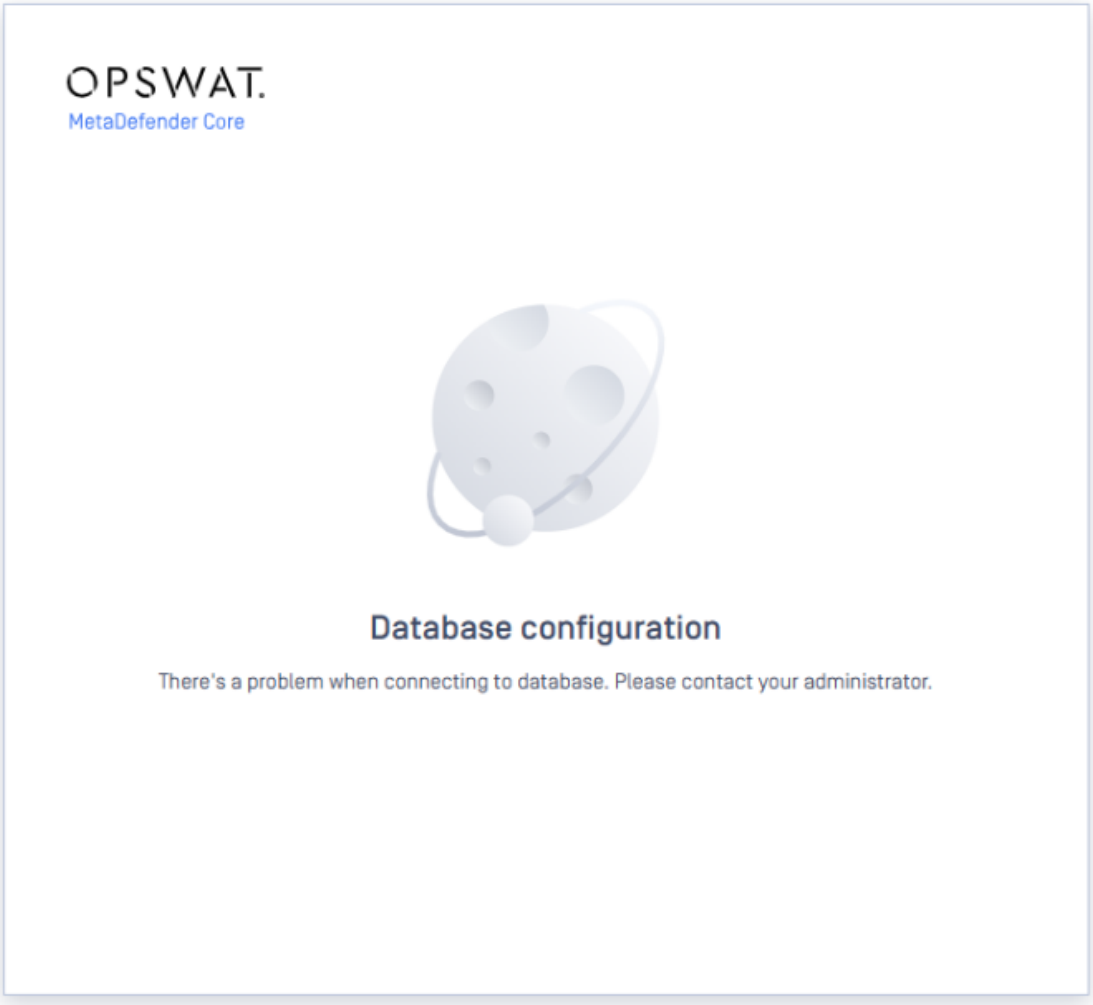
PostgreSQL Connection Loss After Upgrade on CIS-Hardened Ubuntu

Overview

MetaDefender Core may lose PostgreSQL connectivity following an upgrade on Ubuntu systems with CIS (Center for Internet Security) hardening enabled.

Symptoms

- Web Console becomes inaccessible after completing a MetaDefender Core upgrade
- Error page displayed when attempting to access the Web Console



- The following error appears in the application logs:

```
Code · bash
```

```
[ERROR  ] 2025.08.14 03:20:11.504: (core.postgres) Postgres issue,
error='Starting postgres service failed',
param='Warning: Tt file, source configuration file or drop-ins of ometascan-pg.service changed
on disk.
Run 'systemctl daemon-reload' to reload.
' [msgid: 4084]
```

Root Cause

This issue appears to be related to a systemctl behavior when CIS hardening is applied to the system. The exact cause is currently under investigation.

Solution

Restart the MetaDefender Core service:

Code · bash

```
# Use systemctl
systemctl restart ometascan
# Or use service
service ometascan restart
```

Additional Notes

- This issue is specific to CIS-hardened Ubuntu environments
- The service restart resolves the connectivity issue immediately
- No data loss or configuration changes are required

Debug logging

Warning

Enabling debug logging level on MetaDefender Core should only be used when absolutely necessary, and advised by the OPSWAT team.

Letting MeteDefender Core operate for a long time under debug logging mode will likely cause performance degradation, out of disk space, and possibly an IO bottleneck issue on the system.

Tips: We should only enable the debug logging level when:

- We already know the steps to reproduce or trigger the issue symptom.
- Or, we already know the issue will very likely come back very soon.
- Or, the OPSWAT team checked and confirmed there should be more useful information tracked with the debug log and relevant to the issue.

1. To provide debug logs for the OPSWAT support team, the level of the logfile for the given service ometascan must be set to `debug` (see how at How to enable debug logging for MetaDefender Core).
2. Execute the scenarios requested by the support team
3. Collect the generated log files by following the steps at
   - **How to Create Support Package With Web Console**
   - **How to Create Support Package With Bundle Tools**
4. **After collecting the debug logs, the log level should be set back to `info`, as the size of the logfile will increase significantly in debug mode.**

For information on how to interpret the log files, consult: Log Message Format

Support:

If **Further Assistance** is required, please proceed to log a **support case or chat with one of our support engineers**.

How to Read the Metadefender Core Log?

The log files are plain text files that can be opened with any text editor.

Files

Under Linux the server and nodes generate separate log files under **/var/log/ometascan**. The **ometascan.log** file (if present) belongs to MetaDefender Core server.

Under Windows there is no default logging into file unless otherwise specified. For details see MetaDefender Configuration.

Format

In the log, each line represents a log message sent by the server or node. Depending on the log file, the format of the line is as follows:

Code

```
[LEVEL] TIMESTAMP (COMPONENT) MESSAGE [msgid: MESSAGE ID]
```

Example:

Code

```
[INFO   ] 2015.08.19 09:40:27.941: (core.workflow) Scan finished,
dataId='c35a190681944380a52efb9ef32ef509', overallResult='No Threat Detected',
totalResultCount='5', infectedResultCount='0' [msgid: 82]
```

Where the different values are:

- **LEVEL** : the severity of the message
- **TIMESTAMP** : The date value when the log entry was sent
- **COMPONENT** : which component sent the entry
- **MESSAGE** : the verbose string of the entry's message
- **MESSAGE ID** : the unique ID of this log entry

Severity levels of log entries

Depending on the reason for the log entry, there are different types of severity levels.

Based on the configuration, the following levels are possible:

- **DEBUG** : Debuggers severity level, mostly used by support issues.
- **INFO** : Information from the software, such as scan results.
- **WARNING** : A problem occurred needs investigation and OPSWAT support must be contacted, however the product is supposed to be operational.
- **ERROR** : Software error happened, please contact support if the issue is persist. Software functionality may be downgraded in these cases.

Inaccessible Management Console

Problem: You cannot access the MetaDefender Core Management Console from your browser e.g. http://localhost:8008

Symptom

After you enter the Metadefender Core Management Console address (e.g. http://localhost:8008) you get an error message (connection refused) or your browser is waiting for reply.

Troubleshooting tips

1. Check if your computer can access the Metadefender Core IP address / domain name. Verify that you opened the firewall port on the Metadefender Core server for the Management Console.
2. Check if you are using proper protocol HTTP / HTTPS.
3. Check if you are using proxy for the connection.
4. Check if MetaDefender Core service is still running.
5. Check MetaDefender Core's Nginx web server log for a clue.

Slow or Inaccessible Management Console

Overview

This guide addresses performance issues with the MetaDefender Core Web Management Console, including slow loading times and unresponsive pages.

Symptom

- The Web Management Console loads significantly slower than normal
- The following pages may become unresponsive or fail to load: Workflow Management:
  - Workflows
  - Templates
  - Security Zones
- Check the Nginx logs for related error messages or timeouts

Code · bash

```
2025/09/09 13:10:11 [error] 12300#7664: *534823988 lua tcp socket read timed out, client: ::1,
server: , request: "GET /admin/schema HTTP/1.1", host: "localhost:8008", referrer: "http://
localhost:8008/"
2025/09/09 13:10:11 [error] 12300#7664: *534823988 [lua] resourcehandler.raw:0: ():
resourcehandler.lua: Request timed out: timeout, client: ::1, server: , request: "GET /admin/
schema HTTP/1.1", host: "localhost:8008", referrer: "http://localhost:8008/"
::1 - - [09/Sep/2025:13:10:11 +0530] "GET /admin/schema HTTP/1.1" 500 47 "http://
localhost:8008/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/139.0.0.0 Safari/537.36 Edg/139.0.0.0" 150.000
```

- The scan function (API POST /file) responds slowly or becomes unresponsive

Remediation

- This is a know issue of MetaDefender Core version 5.12.0
- The issue has been fixed from MetaDefender Core version 5.13.2
- The customer is recommend to upgrade MetaDefender Core to latest version (as of writing is **5.15.2**) to fix the issue

Open Connection On PostgreSQL Server

Applicable to use cases where users want to use a PostgreSQL server sitting on a different server from where MetaDefender Core resides.

Info

Just in case the firewall is enabled, please also make sure to configure your firewall rule properly for the connections between PostgreSQL server and MetaDefender Core.

Info

The guide here assumes we are using an SSL connection with PostgreSQL. With a non-SSL connection, please use host instead of hostssl

Windows

1.) Modify **C:\Program Files\OPSWAT\MetaDefender Core\data\pg_data\pg_hba.conf**

(**If that's a separate PostgreSQL server, then you should find it in the PostgreSQL data directory instead**)

```
Code

hostssl    all             all             0.0.0.0/0           scram-sha-256
```

In the above example, it means to accept any MetaDefender Core machine source address. Learn more at: https://www.postgresql.org/docs/9.1/auth-pg-hba-conf.html

2.) Create a new file **C:\Program Files\OPSWAT\MetaDefender Core\postgres\postgresql.conf**

(**If that's a separate PostgreSQL server, then you can find it in the PostgreSQL data directory instead**)

```
Code

listen_addresses = '*'
```

In the above example, it means to ask the PostgreSQL server to start listening for all MetaDefender Core source connections. Learn more at https://www.postgresql.org/docs/9.1/runtime-config-connection.html

3.) Restart **ometascan-postgres** service

Linux

1.) Modify **/var/lib/ometascan/pg_data/pg_hba.conf**

(**If that's a separate PostgreSQL server, then you can find it in the PostgreSQL data directory instead**)

```
Code

hostssl    all             all             0.0.0.0/0           scram-sha-256
```

2.) Create a new file **/usr/lib/ometascan/postgres/postgresql.conf**

(**If that's a separate PostgreSQL server, then you can find it in the PostgreSQL data directory instead**)

```
Code

listen_addresses = '*'
```

3.) Restart **ometascan-postgres** service

Too Many Sockets or Files Open

Only on Linux systems: if too many sockets or files are open by the process this can cause problems.

How to detect

Check the **file descriptor** limit:

```
Code

ulimit -n
```

Check the used file descriptor count on a running process:

```
Code

watch -n 1 "ls /proc/\`ps -eo comm,pid | awk '\$1 == \"ometascan\" { print \$2 }'\`/fd | wc -l"
```

If the count is close to the limit this will cause problems.

Rule of thumb: 1 scan workflow requires 2-3 file descriptors.

Solution

Increasing the number of file descriptors

The command **ulimit -n** displays the current set number of maximum file descriptors. In order to increase this number follow the next steps:

Append this line to **/etc/sysctl.conf**

```
Code

fs.file-max = 65535
```

Add the following lines to **/etc/security/limits.conf**

```
Code

*   soft   proc    65535
*   hard   nproc   65535
*   soft   nofile  65535
*   hard   nofile  65535
```

Restart the system to apply the new configuration. After restart you can check the changed limit by issuing **ulimit -n**

```
Code

> ulimit -n
> 65535
```

Too Many TIME_WAIT Socket

This trouble is **only on Linux** systems.

If TCP connections are in use the port limit can be reached. In this case, no new connection can be created.

How to detect

Kernel message:

```
Code

kernel: TCP: request_sock_TCP: Possible SYN flooding on port 8009. Sending cookies.  Check SNMP
counters.
```

Check the `TIME_WAIT` sockets count:

```
Code

watch -n 1 "netstat -nt | grep TIME_WAIT | wc -l"
```

If it is close to the available port range then your system is suffers from this issue:

```
Code

cat /proc/sys/net/ipv4/ip_local_port_range
```

Solution

You should enable socket reusing.

By default Linux selects a port from an ephemeral port range, which by default is a set to range from 32768 to 61000.

A TCP local socket address that has been bound is unavailable for some time after closing, unless the `SO_REUSEADDR` flag has been set. Care should be taken when using this flag as it makes TCP less reliable.

To avoid waiting on closed sockets and enable reusing them set tcp_tw_reuse sysctl to enable reusing of `TIME_WAIT` sockets by appending the following line to file **/etc/sysctl.conf**:

Code

```
net.ipv4.tcp_tw_reuse = 1
```

After this, sockets in state `TIME_WAIT` will be reused when necessary.

Technical Insights

Connect function error value in these cases is `EADDRNOTAVAIL`.

PIN Code Reset Tool

Overview

The `ometascan-reset-pin-code` utility is a command-line tool designed to manage PIN codes for OPSWAT MetaDefender Core database configuration. This tool allows administrators to either remove existing PIN code or configure new one for database access security.

Purpose

This tool manages the PIN code protection for database credentials in MetaDefender Core. The PIN code is stored in encrypted database credential files and provides an additional layer of security for database access.

Usage

Windows

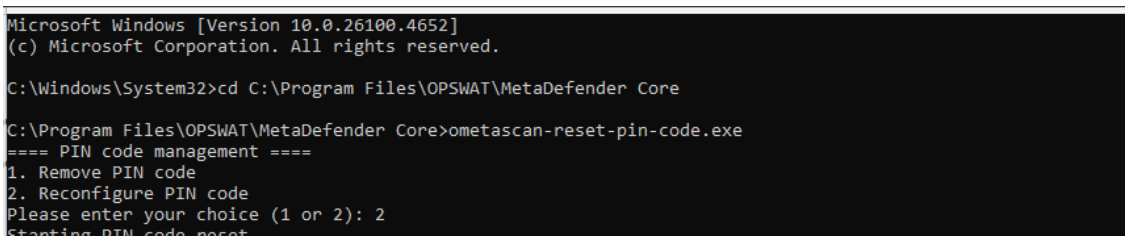
Code · bash

```
# Goto installation folder, for example
cd c:\Program Files\OPSWAT\MetaDefender Core\
#Run the tool
ometascan-reset-pin-code.exe
```

Linux

Code · bash

```
/usr/sbin/ometascan-reset-pin-code
```



Engine Clean-up Tool

Info

This article is only applicable to MetaDefender Core 4.19.0 and 4.19.1.

For MetaDefender Core 4.19.2 or newer, all engines could be removed directly on the MetaDefender Core management console UI under Modules page.

For MetaDefender Core 4.18.0 or older, please follow instructions here What are the engine clean-up instructions?.

Sometimes, during the engines downloading/deployment process, some of them may remain in "**failed**" or "**permanently failed**" status. In that case, you could perform an engine clean-up by using MetaDefender Core built-in engine sweeper tool:

- Locate the tool in MetaDefender Core installation folder named ometascan-engine-sweeper

Usage: `ometascan-engine-sweeper [options]`

Code · bash

```
MetaDefender Core v4 Engine Sweeper
Options:
?, -h, --help Displays this help.
e, --engine <engine> Engine name to clean. ('*' for matching any string, '?' for matching any
character)
l, --list List installed engines.
d, --retain-database Retain engine\'s database.
H, --host <host> Host of PostgreSQL server.
P, --port <port> Port of PostgreSQL server.
U, --username <username> Username for PostgreSQL server.
D, --database <database> Database name to connect.
B, --builtin-postgres Only use if that is a built-in PostgreSQL server (installed on the same
machine with MetaDefender Core)
```

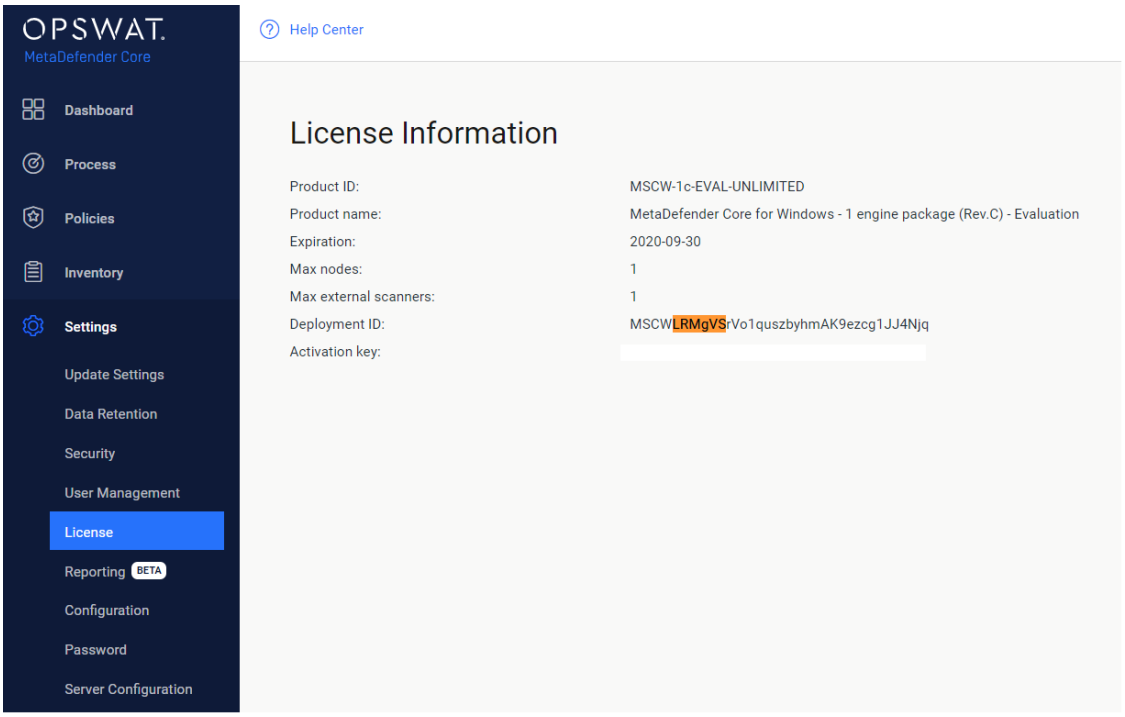
MetaDefender Core 4.19.x database information to connect

- **Host and port:** Specified while installing Core 4.19.0. In default installation option, MetaDefender Core setups local and connect to PostgreSQL server at localhost via port 5432
- **Username:** Specified while installing Core 4.19.0. In default installation option, MetaDefender Core uses username postgres
- **Database name:** MetaDefender Core metadefender_core_xxxxxx - whereas xxxxxx is the 6 characters in lower case following product code name (Windows: MSCW, Linux: MSCL) in your current Deployment ID.

Info

Making sure to lower case on all 6 characters

For example: database name should be metadefender_core_lrmgvs in the sample screenshot below



Example usages

Assuming we will connect to a local PostgreSQL server with following info:

- localhost via port 5432
- username = postgres
- database name = metadefender_core_lrmgvs

Example 1: List all installed engines

Code · bash

```
ometascan-engine-sweeper.exe -B -H localhost -P 5432 -U postgres -D metadefender_core_lrmgvs -l
```

note

Making sure to use “-B” parameter option if that is a local PostgreSQL server

Then the tool requires you to input PostgreSQL server password for MetaDefender Core (specified while installing MetaDefender Core)

Example output:

Code · bash

```
MetaDefender Core v4 Engine Sweeper.


Data directory detected: "C:/Program Files/OPSWAT/Metadefender Core/data/"

Enter PostgreSQL password:
Querying installed engines in database...
  Installed engines:
   "oesis_3_windows"            | "Vulnerability Scanning"
   "mdcloud_5_windows"          | "Threat Intelligence"
   "7z_13_windows"              | "Archive engine"
   "ds_3_windows"               | "Deep CDR"
   "yara_5_windows"             | "Yara"
   "dlp_8_windows"              | "Proactive DLP"
   "clamav_1_windows"           | "ClamAV"
   "filetype_1_windows"         | "FileType"
```

Example 2: Remove engine package and database package of Archive engine (e.g. 7z_13_windows)

Code · bash

```
ometascan-engine-sweeper.exe -B -H localhost -P 5432 -U postgres -D metadefender_core_lrmgvs -e
7z_13_windows
```

Note: Making sure to use “-B” parameter option if that is a local PostgreSQL server

Example output:

Code · bash

```
MetaDefender Core v4 Engine Sweeper.


Data directory detected: "C:/Program Files/OPSWAT/Metadefender Core/data/"

Enter PostgreSQL password:
Engine(s) to clean: ("7z_13_windows")


The OPSWAT Metadefender Core service was stopped successfully.


The OPSWAT Metadefender Core Node service was stopped successfully.

PostgreSQL connection has lost. Please check the parameters.
```

```
The OPSWAT Metadefender Core service is starting.....
The OPSWAT Metadefender Core service was started successfully.

The OPSWAT Metadefender Core Node service is starting...
The OPSWAT Metadefender Core Node service was started successfully.
```

Example 3: Remove engine only (retain its database)

Code · bash

```
ometascan-engine-sweeper.exe -B -H localhost -P 5432 -U postgres -D metadefender_core_lrmgvs -e
7z_13_windows -d
```

Note: Making sure to use “-B” parameter option if that is a local PostgreSQL server

Example output:

Code · bash

```
MetaDefender Core v4 Engine Sweeper.


Data directory detected: "C:/Program Files/OPSWAT/Metadefender Core/data/"

Enter PostgreSQL password:
Engine(s) to clean: ("7z_13_windows")


The OPSWAT Metadefender Core service was stopped successfully.


The OPSWAT Metadefender Core Node service was stopped successfully.

PostgreSQL connection has lost. Please check the parameters.

The OPSWAT Metadefender Core service is starting.....
The OPSWAT Metadefender Core service was started successfully.

The OPSWAT Metadefender Core Node service is starting...
The OPSWAT Metadefender Core Node service was started successfully.
```

Example 4: Remove all engines

Code · bash

```
ometascan-engine-sweeper.exe -B -H localhost -P 5432 -U postgres -D metadefender_core_lrmgvs -e
"*"
```

Note: Making sure to use “-B” parameter option if that is a local PostgreSQL server

Example output:

Code · bash

```
MetaDefender Core v4 Engine Sweeper.


Data directory detected: "C:/Program Files/OPSWAT/Metadefender Core/data/"

Enter PostgreSQL password:
Engine(s) to clean: ("ds_3_windows", "7z_13_windows", "clamav_1_windows", "mdcloud_5_windows",
"oesis_3_windows", "dlp_8_windows", "filetype_1_windows", "yara_5_windows")


The OPSWAT Metadefender Core service was stopped successfully.


The OPSWAT Metadefender Core Node service was stopped successfully.

PostgreSQL connection has lost. Please check the parameters.

The OPSWAT Metadefender Core service is starting.....
The OPSWAT Metadefender Core service was started successfully.

The OPSWAT Metadefender Core Node service is starting...
The OPSWAT Metadefender Core Node service was started successfully.
```

TCP/IP Port Exhaustion

Issue symptom

`Only one usage of each socket address (protocol/network address/port) is normally permitted` error appears in Nginx web server access log while MetaDefender Core is operating on Windows OS. There could possibly be a few reasons causing:

- Running out of available ports for new connections.
- Each closed port has a delay time (`WAIT_TIME`) before being re-used for another new connection, but in some cases, the delay time is too long. As the result we don't have port available for new connections.

Solution

A. Increasing port range

On Windows, by default the number of available ports is 5000 and maximum can be set is 65534.

1. Use command line

Check how much port can be used by running command:

```
Code

netsh int ipv4 show dynamicport tcp
```

Example output:

```
Code

Protocol tcp Dynamic Port Range
---------------------------------
Start Port      : 1024
Number of Ports : 64511
```

To adjust default values, use command:

```
Code

netsh int ipv4 set dynamicport tcp start=<starting_range> num=<total_number>
```

Restart Windows OS to apply new settings

2. Use registry

- Follow the path `HKLM\System\CurrentControlSet\Services\Tcpip\Parameters`
- Choose Edit New DWORD (32-bit) Value and enter `MaxUserPort`
- Right click and choose Modify on the new key
- Set your desired number
- Restart Windows OS to apply new settings

B. Decreasing TCP wait time

Default value is 4 minutes and can be set to lower e.g. 30 seconds (***however, if you are using synchronous scan mode, then decreasing TCP wait time shall not be ideal***)

- Follow the path `KLM\System\CurrentControlSet\Services\Tcpip\Parameters` in Registry
- Choose Edit New DWORD (32-bit) Value and enter `TcpTimedWaitDelay`
- Right click and choose Modify on the new key
- Set number of seconds to wait (e.g. 30)
- Restart Windows OS to apply new settings

References

- https://docs.microsoft.com/en-us/troubleshoot/windows-server/networking/default-dynamic-port-range-tcpip-chang
- https://support.jda.com/articles/Question/What-does-the-error

Archive Compression Engine Permanently Failed on MetaDefender Core v4

Starting from MetaDefender Core version 5.0.0, we have a new archive compression engine to handle archive sanitization feature on MetaDefender Core v5 whenever you have Deep CDR licensed, and in order to support customers not to use another product license when upgrading to v5, so we had to make a change on all existing MetaDefender Core v4 license keys to include this engine.

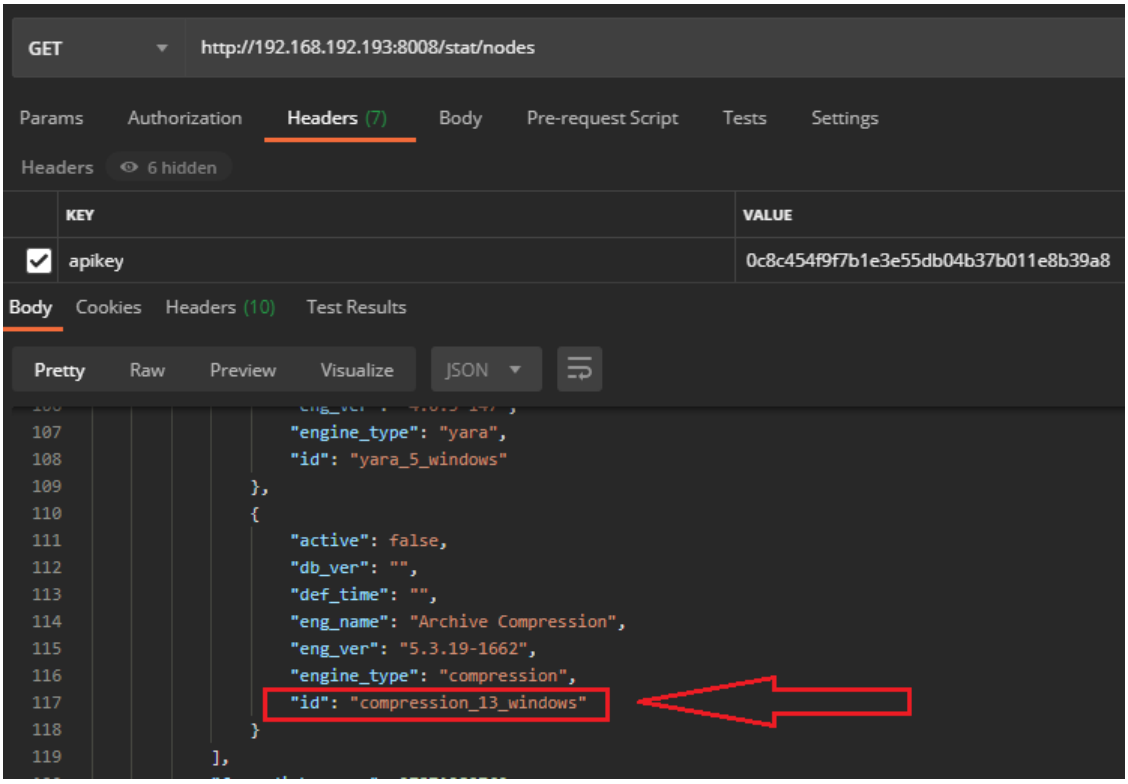
As a result, in MetaDefender Core v4 instances, when you have Deep CDR licensed, then you will have archive compression engine downloaded but that engine is not supported on Core v4, so it will become permanently failed (which is expected). However, that behavior does not cause any impact to MetaDefender Core v4 processing flow nor impact to performance.

For whom has implemented your own health-check mechanism for engine status check, to avoid being alerted for archive compression engine failure on Node, then please do following to disable that engine on MetaDefender Core v4:

Retrieve exact engine ID of archive compression engine

- `GET /stat/nodes`
- `apikey` header is required
- No payload in request

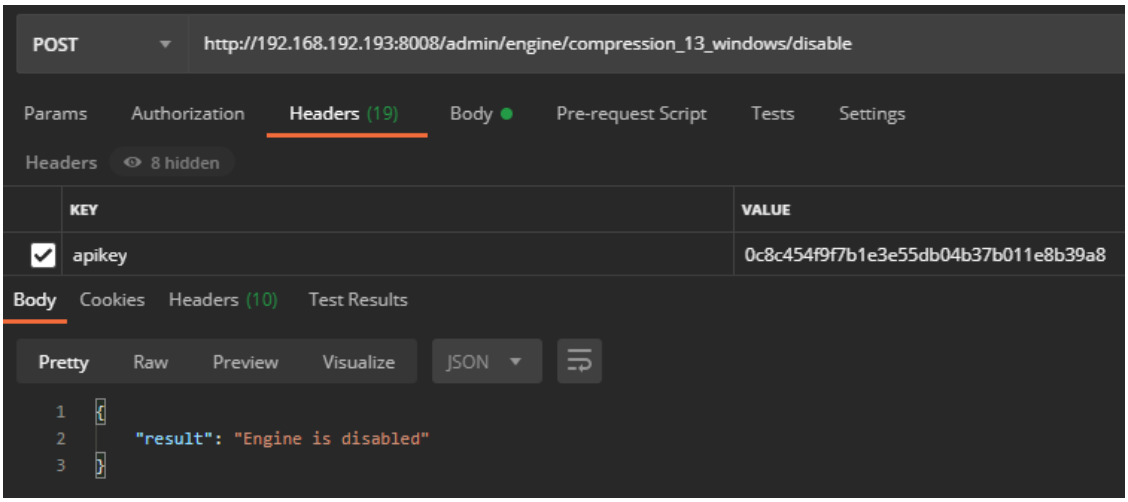
For example:



Disable archive compression engine based on retrieved engine ID

- `POST /admin/engine/<compression_engine_id>/disable`
- `apikey` header is required
- `<compression engine id>` should be the actual retrieved value above
- No payload in request

For example:



After that, the Node on MetaDefender Core v4 will no longer report issue with this engine, and using `GET /stat/nodes` will no longer include archive compression engine in the JSON repsonse.

Note

When you later upgrade to MetaDefender Core v5 from v4, then you might need to manually re-enable "Archive Compression" engine on the product UI to make it active and running on MetaDefender Core v5.

Enabling metadata Display in Logs

Overview

By default, request metadata (transmitted via metadata headers) is not displayed in the application logs. Metadata information appears in the logs as "Request metadata" entries at the `DEBUG` level only.

When DEBUG logging is enabled, metadata entries appear in the following format:

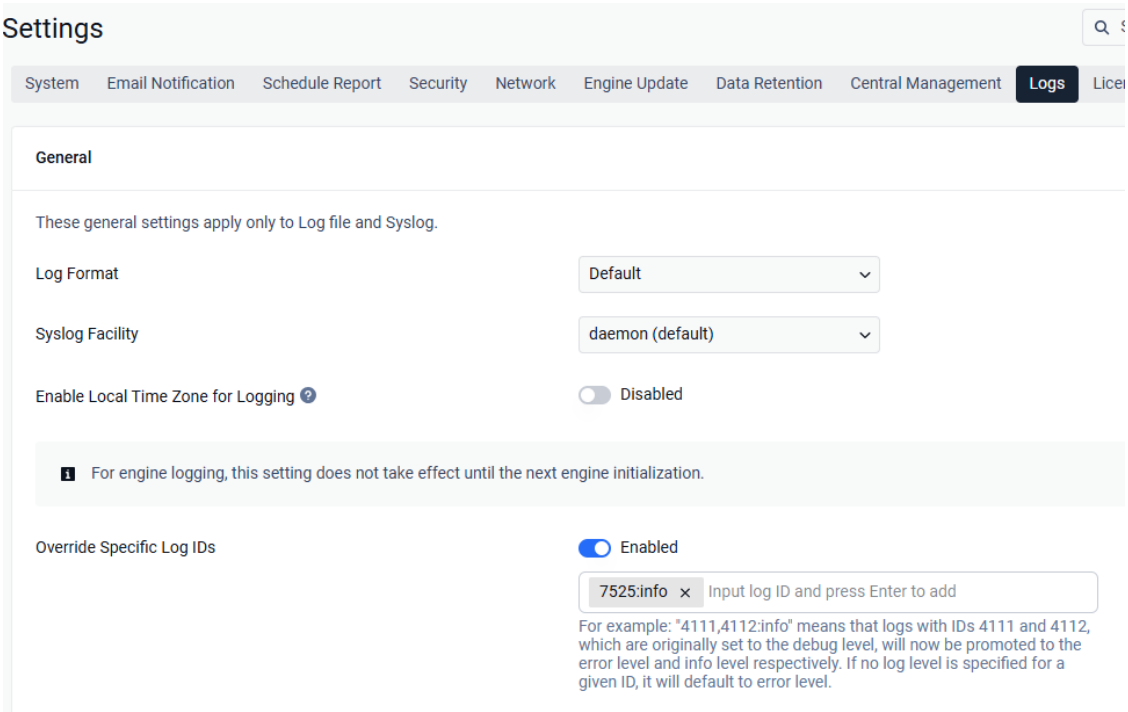
Code · bash

```
[DEBUG   ] 2025.08.25 18:47:04.253: (core.workflow) Request metadata, root_data_id='',
parent_data_id='', data_id='8613d9b8fa684b8ea47f5a697b41620e', metadata='{"client": "localhost",
"IP": "192.168.1.1"}', totalProcessingTime='9' [msgid: 7525]
```

Configuration for Higher Log Levels

To display metadata information at higher log levels (such as INFO), you can override the default log level settings:

1. Navigate to Settings > Logs
2. Enable the "Override specific log IDs" option
3. Enter `7525:info` in the configuration field
4. Click "Save changes"



Troubleshooting: Scan history not updating after upgrading MetaDefender Core - Linux

Who this is for

Administrators who upgraded MetaDefender Core (for example from an older release such as 5.1.7.x to a newer one such as 5.18.x) on Rocky Linux 9 using the **shared / bundle** installation, and then notice that new scans no longer appear in history (or related features that use the database fail).

What you might see

- Scans seem to run, but results are missing from scan history.
- Logs may contain messages similar to:
  - `could not load library ".../pg_trgm.so"`
  - `invoke failed, out='QMap(("error", QVariant(QString, "ERROR:  could not load library`
  - `Failed to insert new request`
  - Other database-related errors that appear right after an upgrade, without you having changed configuration on purpose.

Recommended fix

After a major upgrade, the database and application must be using the same updated components. Restart the services in this order:

1. Restart the PostgreSQL service used by MetaDefender Core (the bundled database).
2. Restart the MetaDefender Core service.

Wait until PostgreSQL has fully started before starting or restarting Core if your environment requires a strict order.

Example (systemd — names can differ on your system):

Code · bash

```
sudo systemctl restart ometascan-pg
sudo systemctl restart ometascan
```

HTTPS Failure on MetaDefender Core 5.2.0 (or newer)

Info

This article is only applicable when:

- You are planning to use (upgrade / fresh install) MetaDefender Core 5.2.0 or newer

- You already currently configured HTTPS (or planning to do so) on MetaDefender Core

Issue symptom

After upgrading to MetaDefender Core 5.2.0 or newer (or runing a fresh install), found that MetaDefender Core service failed to start with following error found in NGINX log (for example):

Code

```
“38772#35704: SSL_CTX_use_certificate("C:/Windows/Temp/ometascan/https_cert.pem") failed (SSL: error:0A00018E:SSL routines::ca md too weak)”
```

Reason

Starting MetaDefender Core version 5.2.0, the OpenSSL 1.x is replaced by OpenSSL 3.x within the product and other dependencies (PostgreSQL, NGINX) as a security improvement, and prevent potential known vulnerabilities on OpenSSL 1.x.

NGINX's OpenSSL 3.x on MetaDefender Core has the enforcement in place to reject all very weak cipher suites, and prevent being exploited. It only accepts "HIGH" encryption cipher suites https://www.openssl.org/docs/man1.1.1/man1/ciphers.html (MD5 and SHA1 hashing based will not be accepted as well).

As a result, if you already configured MetaDefender Core for HTTPS connection, but using a weak SSL cipher in your certificate, then MetaDefender Core will not be able to start due to NGINX's OpenSSL 3.x enforcement.

Solution

If you are already on MetaDefender Core 5.2.0 or newer, and hitting the issue

The ultimate solution is replacing your certificate used for HTTPS configure by another strong one, so that it is created based on strong cipher suites to strenghthen your own security.

If that is not possible now, then following steps below as a workaround:

- Just in case you currently configured HTTPS on MetaDefender Core UI (and being blocked to make any change because MetaDefender Core service is not running): you need to disable HTTPS configuration on MetaDefender Core using a dedicated CLI tool - see details at HTTPS Configuration Removal Tool
- Switch to configure HTTPS on MetaDefender Core via a NGINX custom config file - Enabling HTTPS
- On NGINX custom config file, specify that NGINX should accept any ciphers except ones with no encryption, or no authentication (`ssl_ciphers ALL:!aNULL;`)

ssl.conf

```
ssl on;
ssl_certificate "C:/Program Files/OPSWAT/Metadefender Core/nginx/your.crt";
ssl_certificate_key "C:/Program Files/OPSWAT/Metadefender Core/nginx/your.key";
```

```
ssl_ciphers ALL:!aNULL;
```

If you are not on MetaDefender Core 5.2.0 or newer yet

Before you decide to upgrade your MetaDefender Core to the version 5.2.0 or newer, check and ensure you are using a strong cipher suite for HTTPS connection with your certificate - Ref: https://helpcenter.gsx.com/hc/en-us/articles/207831828-How-to-identify-the-Cipher-used-by-an-HTTPS-Connection

Otherwise you might need to follow workaround steps mentioned above to force NGINX to accept any ciphers

HTTPS Configuration Removal Tool

Info

This article is only applicable when:

- You have configured HTTPS on MetaDefender Core via UI options (not via NGINX custom config file).
- Unexpected failure with your certificate causing MetaDefender Core service failed to start, and thus you have no option to revert the configuration change already made on the UI.

The removal tool can be located and run as following:

Windows

Code

```
<MetaDefender Core installation folder>\ometascan-reset-ssl-conf.exe
```

Restart MetaDefender Core service upon done.

Linux

Code

```
export LD_LIBRARY_PATH=/usr/lib/ometascan/

/usr/sbin/ometascan-reset-ssl-conf
```

Restart MetaDefender Core service upon done.

Customize Internal PostgreSQL User

Info

In public cloud environments, or when subject to customer-specific security requirements, password complexity policies may be enforced. This guide explains how to customize the internal PostgreSQL user passwords to ensure compliance with those security requirements.

MetaDefender Core once installed, it will automatically create an internal PostgreSQL user for operation purpose, and so far that user name is auto randomized by MetaDefender Core and non adjustable (`usr_<random characters>`).

Now MetaDefender Core supports database system admin to define and change to their own PostgreSQL user instead.

That can be done in both deployment scenarios:

1. Product installation (pre-setup)
2. Product configuration (post-setup)

Notes and limits

1. Internal PG username has to be matched with Postgresql naming conventions https://www.postgresql.org/docs/7.0/syntax525.htm.
2. We support backup/restore credential files mechanism. In case of customizing PostgreSQL user failure, the old credential file will be restored.
3. In SharedDB mode, each MetaDefender Core instance must have a different `private_username`. Otherwise, those instances cannot be started in a parallel way.
4. Currently we do not support internal PostgreSQL user contains special characters, space, uppercase. And it cannot begin with digits (0-9). Hyphens is supported.

Product installation (pre-setup)

In the ignition file, there are 2 keys needed for this user setup:

- dbserver/private_username
- dbserver/private_password

Info

1. If `private_username` is not on the Postgres server, MetaDefender Core creates an internal user.
2. If `private_username` matches the user of the Postgres server. The user on the Postgres is overridden, MetaDefender Corre doesn't create a new internal user.

Product configuraton (post-setup)

Using the CLI tool bundled with MetaDefender Core to modify PostgreSQL internal user while MetaDefender Core is running:

- Windows: C:\Program Files\OPSWAT\MetaDefender Core\ometascan-custom-internal-pg-user.exe
- Linux: /usr/sbin/ometascan-custom-internal-pg-user

You can switch to other user with password optionally:

Code · bash

```
ometascan-custom-internal-pg-user [-c <PATH TO CONFIG DB>] -nu <NEW USERNAME> [-np <NEW PASSWORD>]
```

or you can drop the current user:

Code · bash

```
ometascan-custom-internal-pg-user [-c <PATH TO CONFIG DB>] -r
```

All options and arguments:

- -h\t: (--help) instruction
- -c\t: (--configuration-path) default value: HKEY_LOCAL_MACHINE\Software\OPSWAT\Metascan on Windows or /etc/ometascan/ometascan.conf on Linux
- -nu\t: (--new-username) New username must be started with usr_
- -np\t: (--new-password) new password
- -r\t: (--remove-current-user) remove your current user

PostgreSQL User Privilege Requirements

Scope

You are setting up a dedicated remote PostgreSQL database server for MetaDefender Core to operate on, and your own PostgreSQL server has a strict security policy, then this article is relevant to your deployment.

MetaDefender Core will need to create and use dedicated PostgreSQL users for database setup, upgrade and its own operations. Those users requires certain privileges configured on PostgreSQL server, otherwise expecting failures to occur while installing and operating MetaDefender Core.

MetaDefender Core requires 2 types of PostgreSQL users:

	Purposes

	Purposes
Main PostgreSQL user	Create databases, schemas, roles, PostgreSQL extensions, internal PostgreSQL users, etc.
Internal PostgreSQL user	• Uses to invoke almost query executions during MetaDefender Core runtime. MetaDefender Core uses this internal PostgreSQL users instead of the main PostgreSQL user to narrow down permissions leveraged by MetaDefender Core while operating. • The internal PostgreSQL user will be automatically generated by MetaDefender Core (in the format of `usr_xxxxxxx`) while starting up the product service, and will be also automatically dropped when MetaDefender Core service is stopped. • You can also configure MetaDefender Core to designate your own PostgreSQL user Customize Internal PostgreSQL User. If configured then MetaDefender Core will no longer drop and create a new internal PostgreSQL user when restarting its service. • In the shared database deployment model, it requires at least one main PostgreSQL user for all MetaDefender Core instances, and one internal PostgreSQL user per each MetaDefender Core instance.

PostgreSQL User Privilege Requirements

Main PostgreSQL User

Privilege	Description
SUPERUSER	Create PostgreSQL extensions, create MetaDefender Core databases, create internal PostgreSQL user, terminate connections and rename database while upgrading MetaDefender Core (in case upgrading standalone mode to shared mode only). **Notes:** • If this privilege is granted, needless to grant the other privileges (see below in the table).

Privilege	Description

	• If you **revoke** this privilege:
	  • MetaDefender Core could not be upgraded from standalone to share mode.
	  • MetaDefender Core could not drop the default generated internal users `usr_xxxxxxx`

CREATEDB	Create MetaDefender Core databases (fresh install, and upgrade from standalone to share mode). **Notes:** • If you **revoke** this privilege and drop MetaDefender Core database, then MetaDefender Core will not be able to create a new one.
CREATEROLE	Create internal PostgreSQL user (in the format of `usr_xxxxxxx`). **Notes:** • If you're using your own custom PostgreSQL internal user, you will not need this privilege (refer to this guideline Customize Internal PostgreSQL User). • If you're using the default generated user `usr_xxxxxxx` and you **revoke** this privilege, Core could not generate sub PostgreSQL users `usr_xxxxxxx`.
LOGIN	Access the client connection command.

Internal PostgreSQL User

Privilege	Description
LOGIN	Access the client connection command.

Privileges Setting Up Guidelines

Before MetaDefender Core Installation

Prepare Main PostgreSQL User

You can create a new main PostgreSQL user, or use your existing one, for example:

Code · sql

```
create role mdcore with SUPERUSER CREATEDB CREATEROLE LOGIN password 'mdcorepassword';
```

Configure PostgreSQL .conf

Configure your `pg_hba.conf` to allow the access on main PostgreSQL user.

- Just in case you use the customized internal PostgreSQL user Customize Internal PostgreSQL User, then you should also configure the access for this user as well.
- For example:

Code

```
host    all             mdcore              0.0.0.0/0                   md5
host    all             mdcore              ::/0                        md5

host    all             yourinternaluser    0.0.0.0/0                   md5
host    all             yourinternaluser    ::/0                        md5
```

Restart PostgreSQL server service to take effect.

After MetaDefender Core Installation

Configure PostgreSQL .conf (for auto generated internal PostgreSQL user)

Just in case you are letting MetaDefender Core to auto create and use internal PostgreSQL user (`usr_xxxxxxx`), then configure your `pg_hba.conf` to allow the access on this auto generated internal PostgreSQL user.

Note: To find out the exact username of the default generated user `usr_xxxxxxx`: https://www.postgresqltutorial.com/postgresql-administration/postgresql-list-users/

Code

```
host    all             usr_<>              0.0.0.0/0                   md5
host    all             usr_<>              ::/0                        md5
```

—

Restart PostgreSQL server service to take effect.

Revoke some privileges from the main PostgreSQL user (If needed)

If you need to revoke some privileges from the main PostgreSQL user for any security reason, you can do that for these privileges: **SUPERUSER, CREATEDB, CREATEROLE**

Please read carefully these notes to make sure you can revoke privileges:

- Without SUPERUSER privilege, MetaDefender Core could not drop the internal user `usr_xxxxxxx`automatically. You should use a custom internal PostgreSQL user instead Customize Internal PostgreSQL User. MetaDefender Core also could not create PostgreSQL extensions in case the database needed to be recreated.
- Without CREATEDB, MetaDefender Core could not create a new one in case the old database was dropped.
- Without CREATEROLE, MetaDefender Core could not generate a new one in case you let the internal PostgreSQL `user usr_xxxxxxx` is automatically generated by MetaDefender Core.

Steps to revoke SUPERUSER privilege on the main PostgreSQL user:

1. Install MetaDefender Core and set it up successfully.
2. Use another SUPERUSER and log in to the correct MetaDefender Core database.

`metadefender_core` for the shared deployment model.

`metadefender_core_xxxxxx` for standalone deployment model.

3. Then grant access to these 2 schemas `pg_catalog` and `information_schema` to the main PostgreSQL user.

Code · sql

```
GRANT USAGE ON SCHEMA pg_catalog TO mdcore;
GRANT ALL ON ALL TABLES IN SCHEMA pg_catalog TO mdcore;
GRANT USAGE ON SCHEMA information_schema TO mdcore;
GRANT ALL ON ALL TABLES IN SCHEMA information_schema TO mdcore;
```

4. Revoke the SUPERUSER privilege from the main PostgreSQL user.
5. Restart MetaDefender Core service.

Code · sql

```
alter user mdcore NOSUPERUSER CREATEDB CREATEROLE LOGIN;
```

Steps to revoke other privileges on the main PostgreSQL user:

1. Install MetaDefender Core and set it up successfully.
2. Revoke the CREATEDB CREATEROLE privilege from the main PostgreSQL user.

Code · sql

```
alter user mdcore NOSUPERUSER CREATEDB CREATEROLE LOGIN;
```

3. Restart MetaDefender Core service.

Remote PostgreSQL User Privileges Checking Tool

PostgreSQL User Privileges Requirements

Check out PostgreSQL User Privilege Requirements to learn more all detailed requirements while setting up remote PostgreSQL database server for MetaDefender Core.

This tool is to help your database system admin quickly determine whether or not your remote PostgreSQL database have sufficient user privileges as described at PostgreSQL User Privilege Requirements to work properly with MetaDefender Core application.

Download links:

- **Windows:** https://metascanbucket.s3.amazonaws.com/Metadefender/Core/utilities/PGComplianceCheck/windows/check-db-user-compliance.exe
- **CentOS / Red Hat:** https://metascanbucket.s3.amazonaws.com/Metadefender/Core/utilities/PGComplianceCheck/centos/check-db-user-compliance
- **Debian / Ubuntu:** https://metascanbucket.s3.amazonaws.com/Metadefender/Core/utilities/PGComplianceCheck/debian/check-db-user-compliance

Tool dependencies

Ensure that the tool is executed on a host where the psql executable is present and can be invoked from the command line.

Usage:

Code · bash

```
check-db-user-compliance -h <HOSTNAME> -p <PORT> -u <DATABASE USER> -m <MAIN USER NEED TO CHECK>
-i <INTERNAL USER NEED TO CHECK>
```

Options and arguments:

- -h (--help) instruction
- -h (--hostname) database server host or socket directory
- -p (--port) database server port
- -u (--pguser) database user name
- -m (--mainuser) PostgreSQL main user
- -i (--internaluser) PostgreSQL internal user

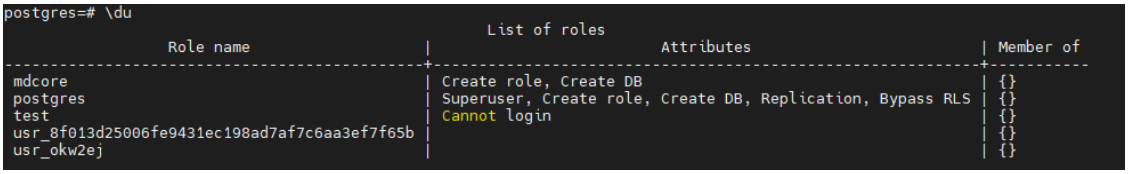
Set environmental path

If the PostgreSQL own `psql` executable file has not been assigned to the Windows environmental path yet,

and PostgreSQL's registry value does not exist, user has to input their own psql executable path.

Examples:

A. Inspecting PostgreSQL main user



Example 1: PostgreSQL main user with SUPERUSER privilege

Code · bash

```
check-db-user-compliance -h 172.168.1.196 -p 54321 -u postgres -m postgres

==== COMPLIANT RESULTS ====
Validating main user...
Main PostgreSQL user: OK
```

Example 2: PostgreSQL main user with CREATEDB and CREATEROLE privilege

Code · bash

```
check-db-user-compliance -h 172.168.1.196 -p 54321 -u postgres -m mdcore

==== COMPLIANT RESULTS ====
Validating main user...
Main PostgreSQL user: OK
WARNINGS:
 - Your user doesn't have SUPERUSERTRUE privilege
 - MetaDefender Core could not be upgraded from standalone to share mode
 - MetaDefender Core could not drop the default generated internal users usr_xxxxxxx
```

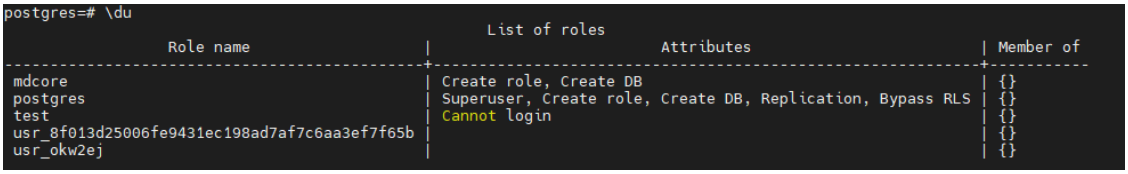
Example 3: PostgreSQL main user without privilege

Code · bash

```
check-db-user-compliance -h 172.168.1.196 -p 54321 -u postgres -m test

==== COMPLIANT RESULTS ====
Validating main user...
Main PostgreSQL user: FAILED
WARNINGS: Your user doesn't have SUPERUSER privilege
```

B. Inspecting PostgreSQL internal user



Example 1: PostgreSQL internal user with LOGIN privilege

Code · bash

```
check-db-user-compliance -h 172.168.1.196 -p 54321 -u postgres -m postgres -i
usr_8f013d25006fe9431ec198ad7af7c6aa3ef7f65b

==== COMPLIANT RESULTS ====
Validating main user...
Main PostgreSQL user: OK

Validating internal user...
Internal PostgreSQL user: OK
```

Example 2: PostgreSQL internal user without LOGIN privilege

Code

```
check-db-user-compliance -h 172.168.1.196 -p 54321 -u postgres -m postgres -i test

==== COMPLIANT RESULTS ====
Validating main user...
Main PostgreSQL user: OK

Validating internal user...
Internal PostgreSQL user: FAILED
WARNINGS: Your user doesn't have LOGINTRUE privilege
```

Database Cleanup Tool in Shared DB Mode

Info

This tool is prepared for the customers when they want to clean all the scan data from the dead instance of Core in the shared database model.

This tool can be found:

- Windows: C:\Program Files\OPSWAT\MetaDefender Core\ometascan-db-cleaner.exe
- Linux: /usr/sbin/ometascan-db-cleaner

Warning

This tool can be used for MetaDefender Core 5.2.1 or above.

How to run

Find the ometascan-db-cleaner.exe(window) or ometascan-db-cleaner(Linux) to the installation folder:

- Default Window path: C:\Program Files\OPSWAT\MetaDefender Core\
- Default Linux path: /usr/sbin/

Warning

With Linux, change permission for ometascan-db-cleaner with command:

sudo chmod 755 ometascan-db-cleaner

Run the cmd or Terminal as Administrator or root.

Info

If there is a mistake in the cleaning database process (clean data of a running instance), the user should restart MetaDefender Core service to re-create all its tables of scanned data again.

Otherwise the error will happen, and the error message will be saved on the core.log

Parameters

Parameter	Description	Example
`-- help` or `-h`	Display all the options and flags associated with the guides	`ometascan-db-cleaner --help` `ometascan-db-cleaner -h`

`--delete / -d <INSTANCE NAME>`
or
`--delete / -d --keep-instance-name/-k <INSTANCE NAME>`
or
`-dk <INSTANCE NAME>`
	Clean the database (required to indicate the instance's name concretely).
	Indicate the instance's name for cleaning the database or display the list of tables with suffix _<instance_id>
	Notes: if using the option --keep-instance-name/-k or -dk, this option means only cleaning data but **keeping** the instance name.
		`ometascan-db-cleaner --delete "fn2xjs"`
		`ometascan-db-cleaner -d "fn2xjs"`
		`ometascan-db-cleaner --delete --keep-instance-name "fn2xjs"`
		`ometascan-db-cleaner -d -k "fn2xjs"`
		`ometascan-db-cleaner -dk "fn`

Parameter	Description	Example
	Otherwise, the tool will delete the instance name with data.	`2xjs"`
`--list` or `-l`	List all instance's names in database	`ometascan-db-cleaner -l` `ometascan-db-cleaner --list`

Troubleshooting

If there is any problem with the tool while using, please collect all the log files in this folder and send us for investigating.

File log: `db_cleaner-yyyyMMdd.log`

- Windows: `C:\Program Files\OPSWAT\MetaDefender Core\data\logs\`
- Linux: `/var/log/ometascan/`

Rectify malformed FileType data in PostgreSQL database

FileType engine v6.0.2 sometimes creates malformed data after analyzing files. When being written into PostgreSQL database, those malformed data cause MetaDefender Core warehouse stuck and result in negative impacts:

- Executive Report in Dashboard gets frozen and changed back to zero.
- CPU usage will go too high.
- A dramatical decrease in file processing performance.

Even though a recently released version of FileType engine v6.1.1 fixes the malformed data issue, there might be a few remaining data stored in PostgreSQL database.

This tool will handle those data by rectifing and converting all the malformed data of `filetype_details` in PostgreSQL database to a true format.

Note: If you do fresh installation of MetaDefender Core v5.4.1, you can skip the following content since we already enhance and harden the service to prevent malformed data issue.

Parameters

Parameter	Mandatory	
`--help`/`-h`	No	display all the options and flags associated with the guides.

Usage

Besides an enhancement, the upcoming version 5.4.1 of MetaDefender Core will also bundle the tool `ometascan-data-normalizer` into its installation folder. For older versions, we provide following separate tools:

MetaDefender Core version	CentOS	Debian	Windows
5.2.0 or newer	download link	download link	download link
5.0.0 - 5.1.2	download link	download link	download link
4.21.x	download link	download link	download link

Step-by-step

1. Copy a relevant `ometascan-data-normalizer.exe`(Windows) or `ometascan-data-normalizer` (Linux) into the installation folder of MetaDefender Core
    1. Default Windows path: `C:\Program Files\OPSWAT\MetaDefender Core\`
    2. Default Linux path: `/usr/sbin/`
3. Stop posting files to MetaDefender Core.
4. Run the `cmd` or `Terminal` as Administrator or root.
5. Execute the command:
    1. Windows: `ometascan-data-normalizer.exe`
    2. Linux: `./ometascan-data-normalizer`
6. Start MetaDefender Core service. A few components might take time to start and refresh, so let's wait for a couple minutes.

Notes:

- After running the `ometascan-data-normalizer`, it might take considerable extra time to update data if database size is large.
- Ensure that MetaDefender Core server is running after the tool finishes. If it is stopped, please start it and wait for a while
- If you encounter error or warning during the execution, please wait for a couple of minutes before executing it again.

Log file

Filename format: `data_normalizer-yyyyMMdd.log`

- On Windows, log file will be placed at `C:\Program Files\OPSWAT\MetaDefender Core\data\logs\`

- On Linux, log file will be placed at `/var/log/ometascan/`

Log level	Message	Troubleshooting
INFO	[INFO]: Data normalization is done, data: filetype, instance id: 1	The process is completed successfully.
WARNING	[WARNING]: Data normalization is failed, data: filetype, instance id: 1,...	The data cannot be updated. Collect all the log files and send the support package to us for troubleshooting.
ERROR	[ERROR]: Select all last_request___id unsuccessfully....	The data cannot be updated. Collect all the log files and send the support package to us for troubleshooting.
ERROR	[ERROR]: Cannot connect to Postgres server: metadefender_core... [ERROR]: Failed to init Postgres service	1. Check the privilege when executing the tool and make sure that it has the true privilege to read the `.dat` file in the installation path. 2. Collect all the log files and send the support package to us for troubleshooting.
ERROR *on console*	ERROR: Cannot open file log: <logpath>/data normalizer-20230128.log	1. Check the log path and make sure that its existence is valid. 2. Check the privilege when executing the tool and make sure that it has the true privilege to create and edit the log file in the log path.

Migrate MetaDefender Core from one machine to another machine

Environment: Windows - Standalone mode - Local (Bundle) PostgreSQL server

Goal: Move as much as possible (the data) from one machine to another machine

- PG database data
- Log files
- Sanitized/DLP processed files
- Quarantine files
- etc

Some aspects need to be checked before and after the upgrade

- Registry
- Sanitized/DLP/Quarantine path
- Custom directory
- Custom log directory
- External scanner/ Post-action configuration file paths
- Certificated paths
- SSL PostgreSQL: certificate path
- Configure HTTPS via ssl.conf file
- Sanitized/DLP/Quarantined files: can be downloaded
- Custom local folder for module updates
- Try to restart MetaDefender Core service

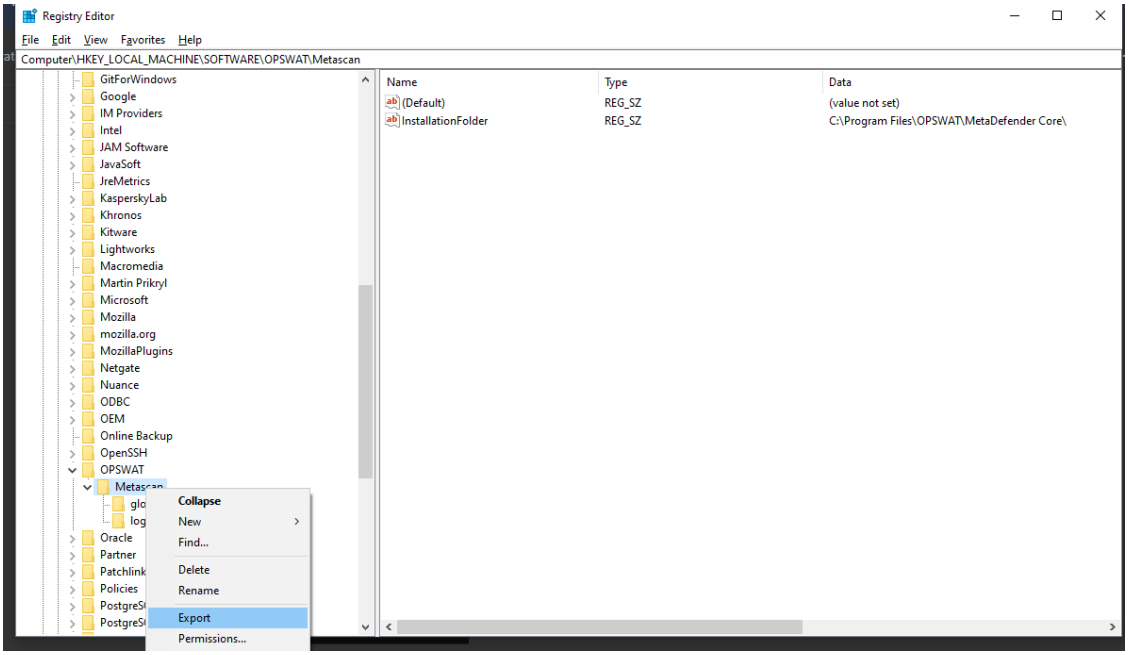
Steps by steps

Let's call the old (current server) Server A and the new server will be Server B

Have tested with MetaDefender Core 5.6.0

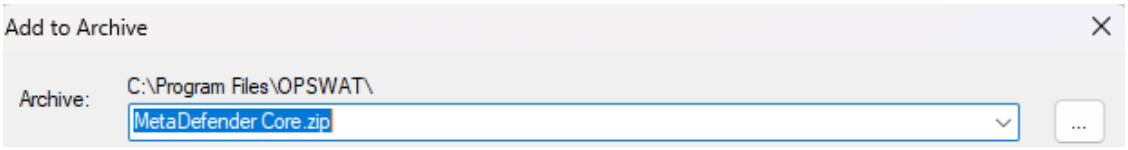
I. Server A

1. Stop MetaDefender Core service)and make sure MetaDefender Core is successfully stopped.
2. If you have any custom configuration on the registry (custom log file, custom temp dir etc): you need to export the registry config (if not please ignore this step):
    1. Open the registry editor, go `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan`
    2. Export the registry config



3. Compress MetaDefender Core's installation directory to a zip file named *MetaDefender Core.zip*

- For example: Compress whole folder `c:\Program Files\OPSWAT\MetaDefender Core\`



4. You also might need to compress/backup the data (if have any) in the custom folder:

> **Info**
>
> If you do not have any custom configured path, please ignore this step

- Custom log files folder
- Custom quarantine folder
- Custom sanitized files folder
- etc

II. Server B

1. Install MetaDefender Core with the same version and same installation path as MetaDefender Core A.
2. Stop MetaDefender Core service and make sure MetaDefender Core is successfully stopped.
3. Import registry that is exported from Server A (Step I.2).
4. Remove the MetaDefender Core's installation directory: `rd /s /q "C:\Program Files\OPSWAT\MetaDefender Core"`

5. Copy and extract the zip package of MetaDefender Core A (*MetaDefender Core.zip*) to *the new* MetaDefender Core's installation path (default is `C:\Program Files\OPSWAT`)
6. Copy/move the backup files (which were backup at step I.4) to the desired location.
7. Start MetaDefender Core service, login, and activate the license.

Working with self-signed certificate

When setting up the webhook server for the callback URL, customers may want to configure HTTPS and use a self-signed certificate.

Since self-signed certificates are not trusted by the system by default, this error might occurs

```
Code · markdown

The certificate is self-signed and untrusted
```

The certificate must first be added to the Trusted Root Certificate store - both Windows and Linux:

1. Add the certificate to the Trusted Root Certificate store.
2. Restart the MetaDefender Core service for the changes to take effect.

Authentication Failure After Upgrading From Core 5.10.1/5.11.0

Issue Description

After upgrading from MetaDefender Core 5.10.1/5.11.0 to more recent versions, users may experience authentication failures when attempting to log in using SSO or AD/LDAP credentials.

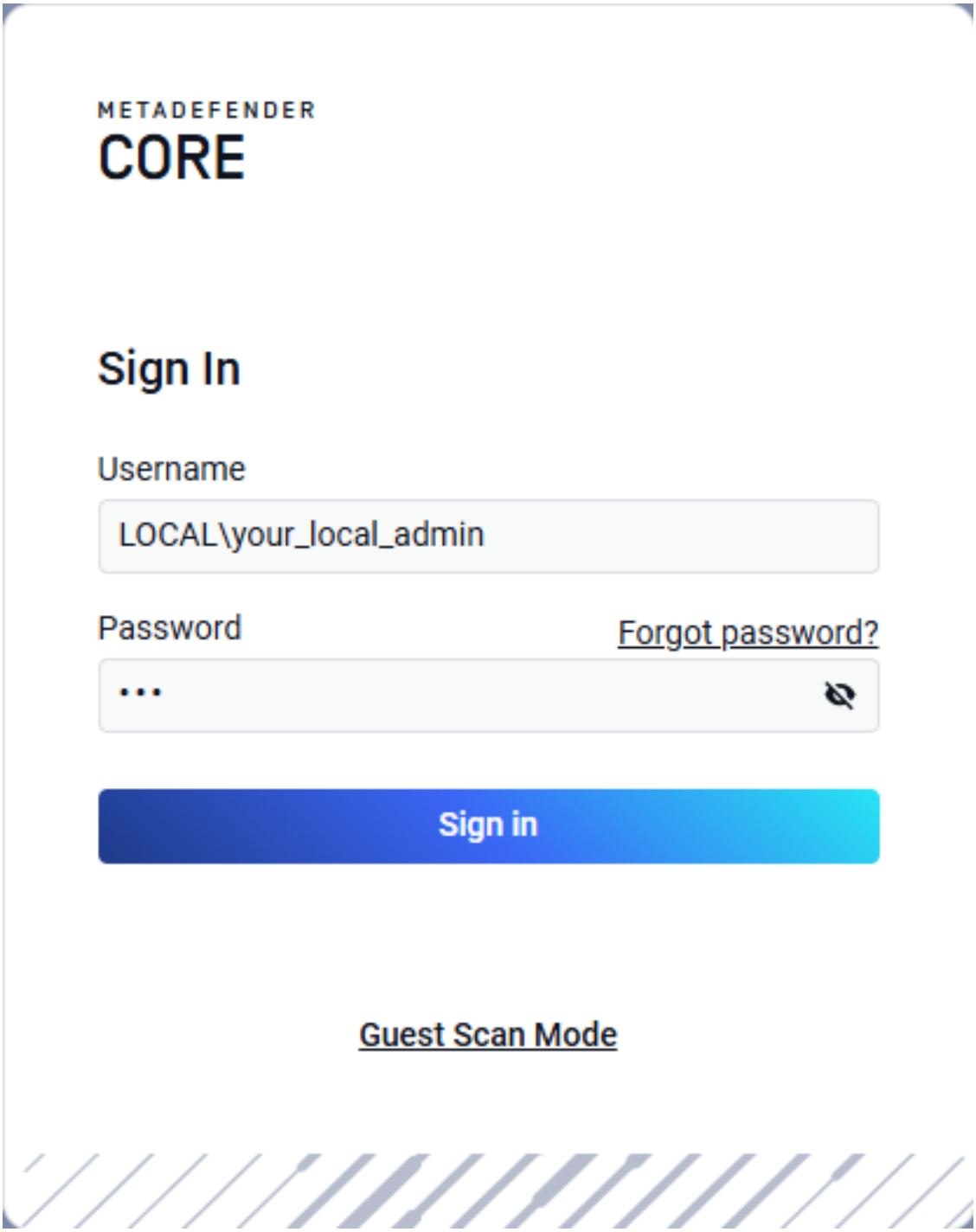
This issue occurs due to an encryption key incompatibility that may have affected your previous upgrade to 5.10.1/5.11.0. If the initial upgrade was completed with incompatible encryption keys, subsequent upgrade will encounter this authentication problem.

Resolution Path

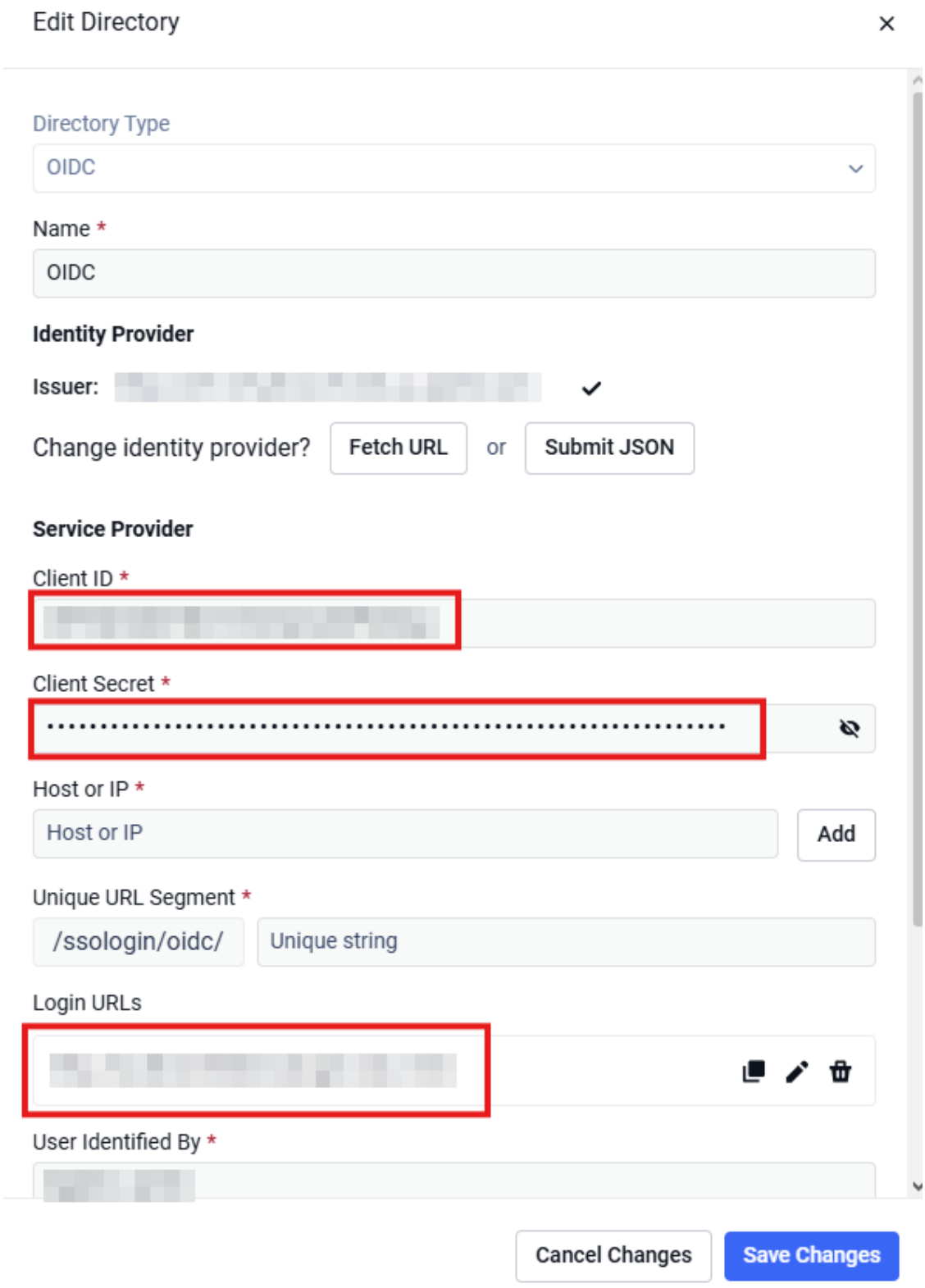
Please follow these steps to resolve the authentication issue:

- **Upgrade to MetaDefender Core 5.14.0** or newer version if you haven't already done so.

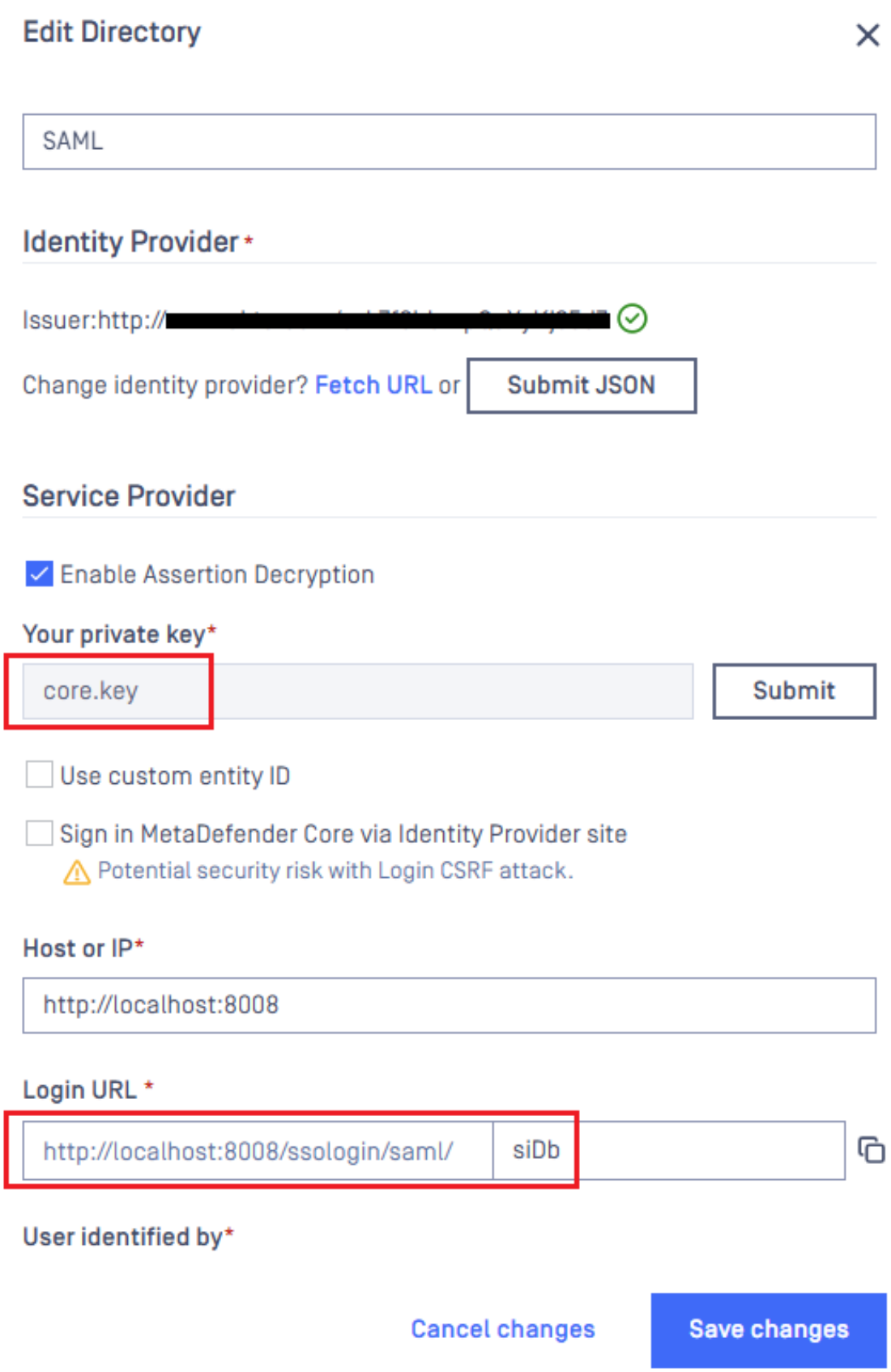
- **Log in using a LOCAL administrator account**.



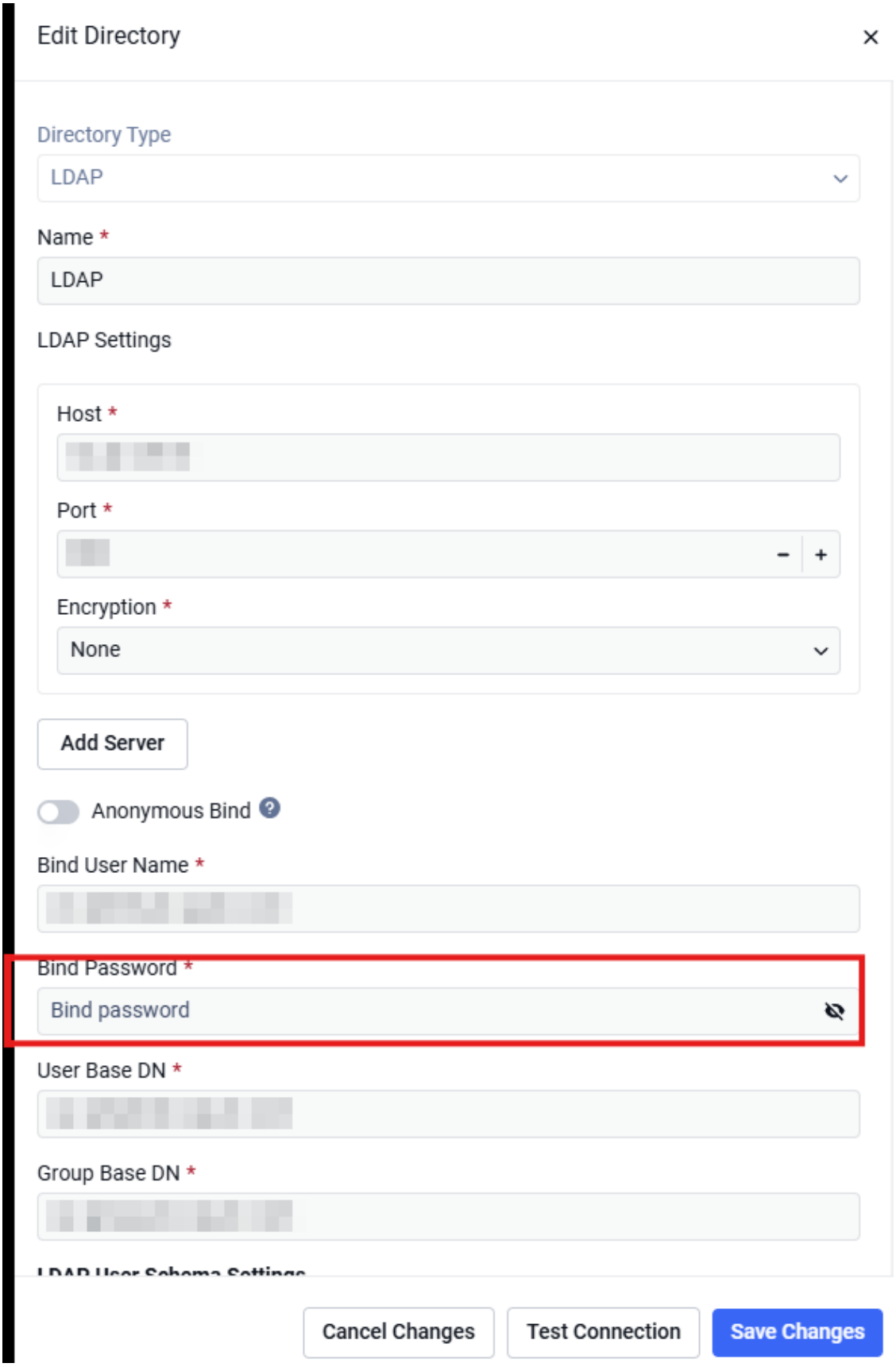
- Reconfigure your identity provider settings
- **OIDC directories**:
  - Re-enter your Client ID
  - Re-enter your Client Secret
  - Verify and update the Login URL



- With **SAML** directory: Customer has to re-enter
  - **Private Key** - If option Enable Assertion Decryption is in used.
  - **Login URL.**



With **AD/LDAP** directory: Customer has to re-enter **Bind password**

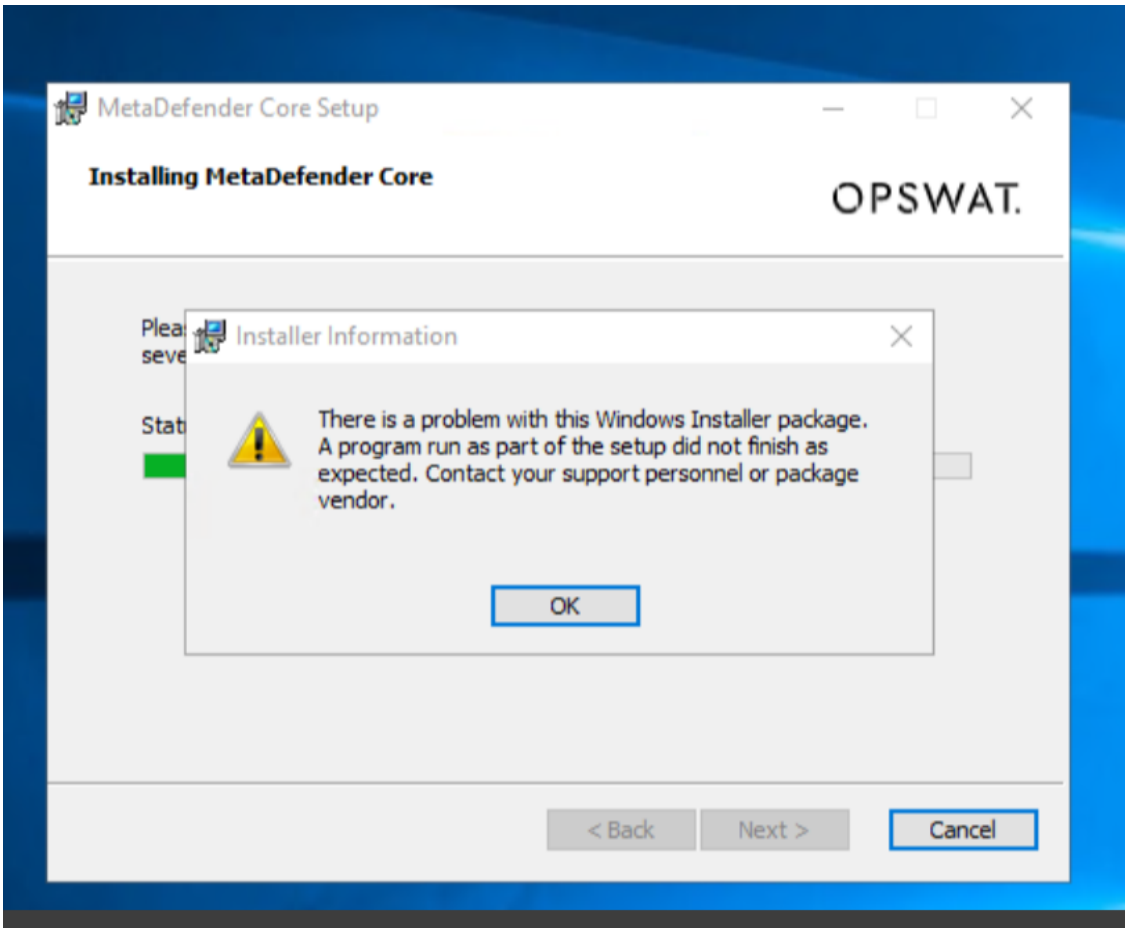


Once reconfigured, your SSO and AD/LDAP authentication should function properly again.

If you encounter any difficulties during this process, please contact OPSWAT Technical Support for assistance.

MetaDefender Core and Out-of-Support Windows OS

Beginning with version **5.17.0**, MetaDefender Core **no longer supports Windows Server 2016, Windows 10 LTSB 2016, and Windows 10 version 1607.** As a result, attempting to install MetaDefender Core on these OS will trigger an error message:



After users click OK button, the installer will automatically roll back to the previously installed version, and the system should continue to function normally after the rollback.

However, please note that **Windows Server 2016, Windows 10 LTSB 2016, and Windows 10 version 1607 are no longer supported**, and MetaDefender Core 5.17.0 or later must not be installed or upgraded on these operating systems. Please prepare a migration to newer OS such as Windows Server 2019, 2022, 2025.

Please check here for more details: How will End of Life Support for Windows Server 2016 and Windows 10 versions before 21H2 in MetaDefender Core affect my deployments?

SSO Login Failed: Domain or IP Not Allowed

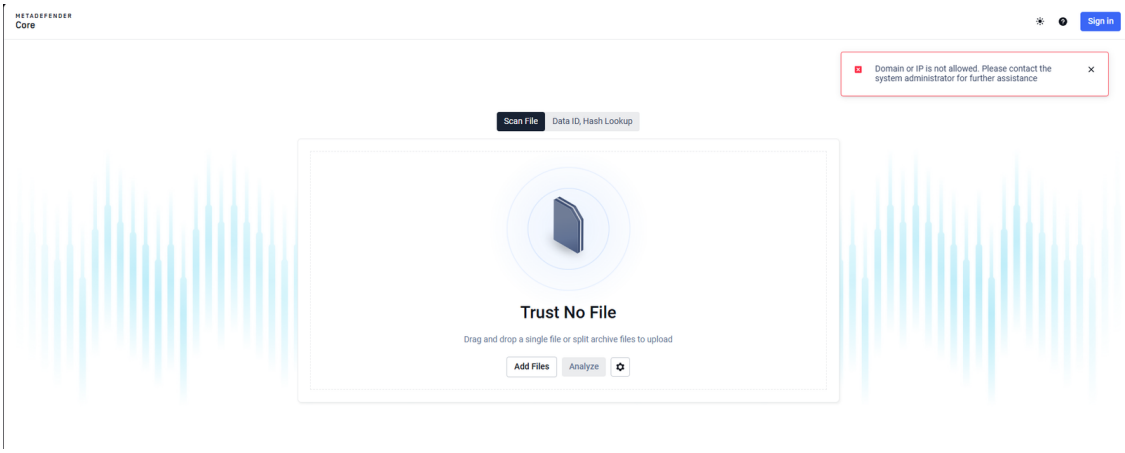
Info

This article applies to **MetaDefender Core version 5.17.1 and later** (Windows and Linux).

Problem

After an SSO directory is configured and enabled, MetaDefender Core automatically redirects login requests to the configured Identity Provider (IdP). If authentication is successful, users should be logged in and redirected to the MetaDefender Core dashboard.

In some cases, the login fails and the following error is displayed:



Why This Happens

This error means that the URL used to access MetaDefender Core does not match any of the login URLs configured in the SSO directory. For security reasons, MetaDefender Core only allows SSO authentication from explicitly configured domains or IP addresses.

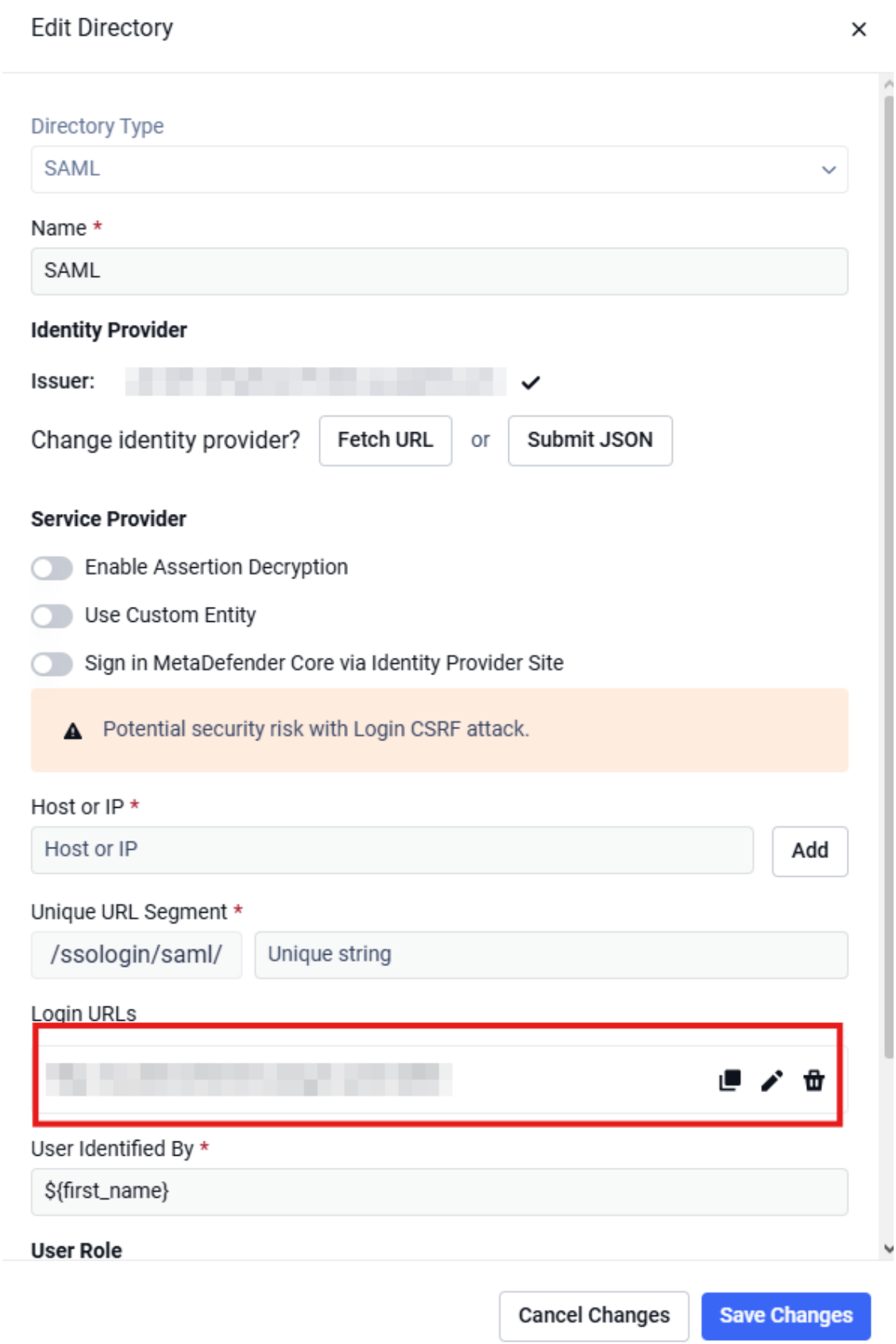
Common Causes

This issue is usually caused by one of the following:

- The login URL is configured in the IdP but is missing from the MetaDefender Core SSO directory.
- The URL scheme does not match (for example, `http` instead of `https` or vice versa)
- A mismatch in trailing slashes (for example, `/core` vs `/core/`).
- Accessing MetaDefender Core using a different domain, IP address, port or alias that is not registered in the SSO configuration.

How to Fix It

1. Sign in to MetaDefender Core using the `<mdcore-host>#/public/backuplogin` (non-SSO access).
2. Go to **User Management > Directories**.
3. Select the active SSO directory.
4. Navigate to **Login URLs** configuration.
5. Add or update the URL used to access MetaDefender Core, making sure it matches exactly.



6. Save the changes.

7. Log out and try signing in again using SSO.

What to Expect

Once the correct login URL is configured, authentication requests are redirected to the Identity Provider. After successful authentication, the user is granted access to the MetaDefender Core dashboard.

> **Info**
> If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

SSL Handshake Failures when fetching Azure Entra URL in SSO configuration

Problem

When deploying MetaDefender Core with Docker **Debian-based**, SSL handshake failures related to certificate chain issues may occur while accessing the fetching Azure Entra URL (https://login.microsoftonline.com)

The server (https://login.microsoftonline.com) uses SSL certificates issued by DigCert Certificate Authority (CA) typically **DigCert Global Root G2** and possibly intermediates like **DigCert Global G2 TLS RSA SHA256 2020 CA1**.

Since (https://login.microsoftonline.com) uses a **Content Delivery Network (CDN)**, the **certificate chain at your location may differ slightly** from the one seen at OPSWAT Support.

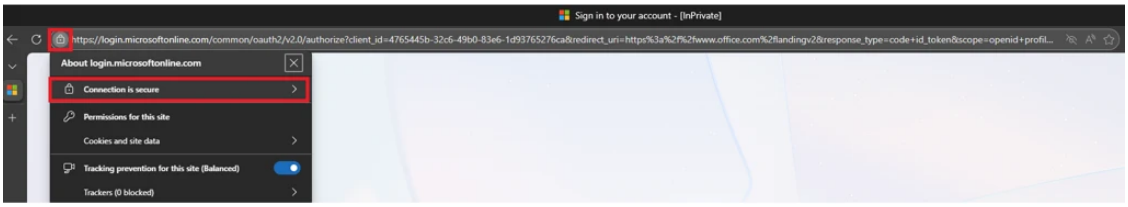
How to fix it

1. Access the Fetch URL in Your Browser

- Open a browser (e.g., Microsoft Edge or Chrome).
- Visit: https://login.microsoftonline.com

2. Export the Certificates

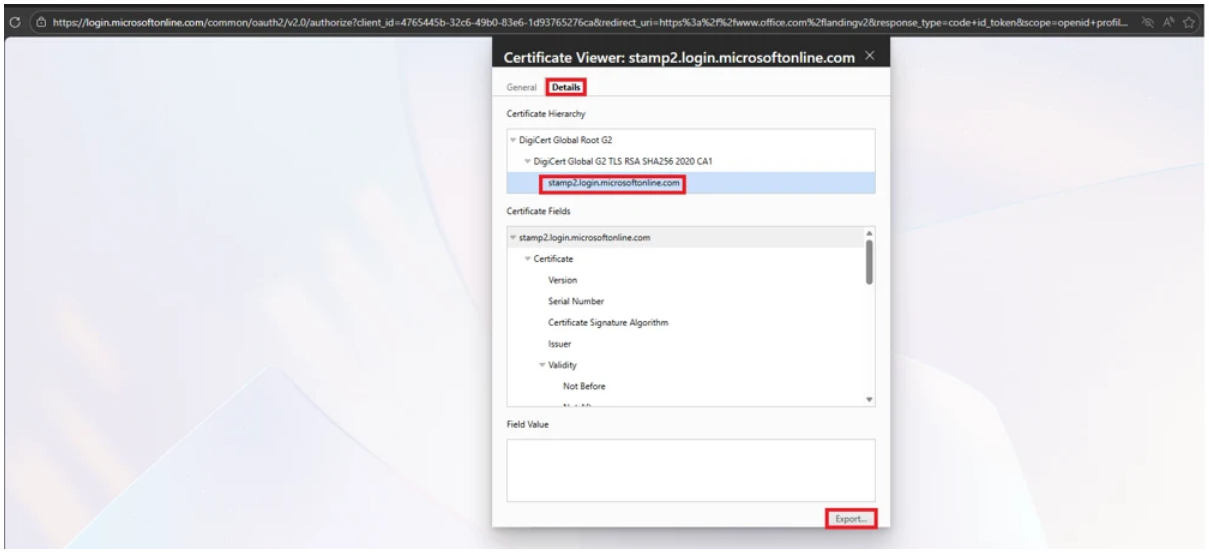
- Click on the security lock icon next to the address bar.



- View Certificate details.

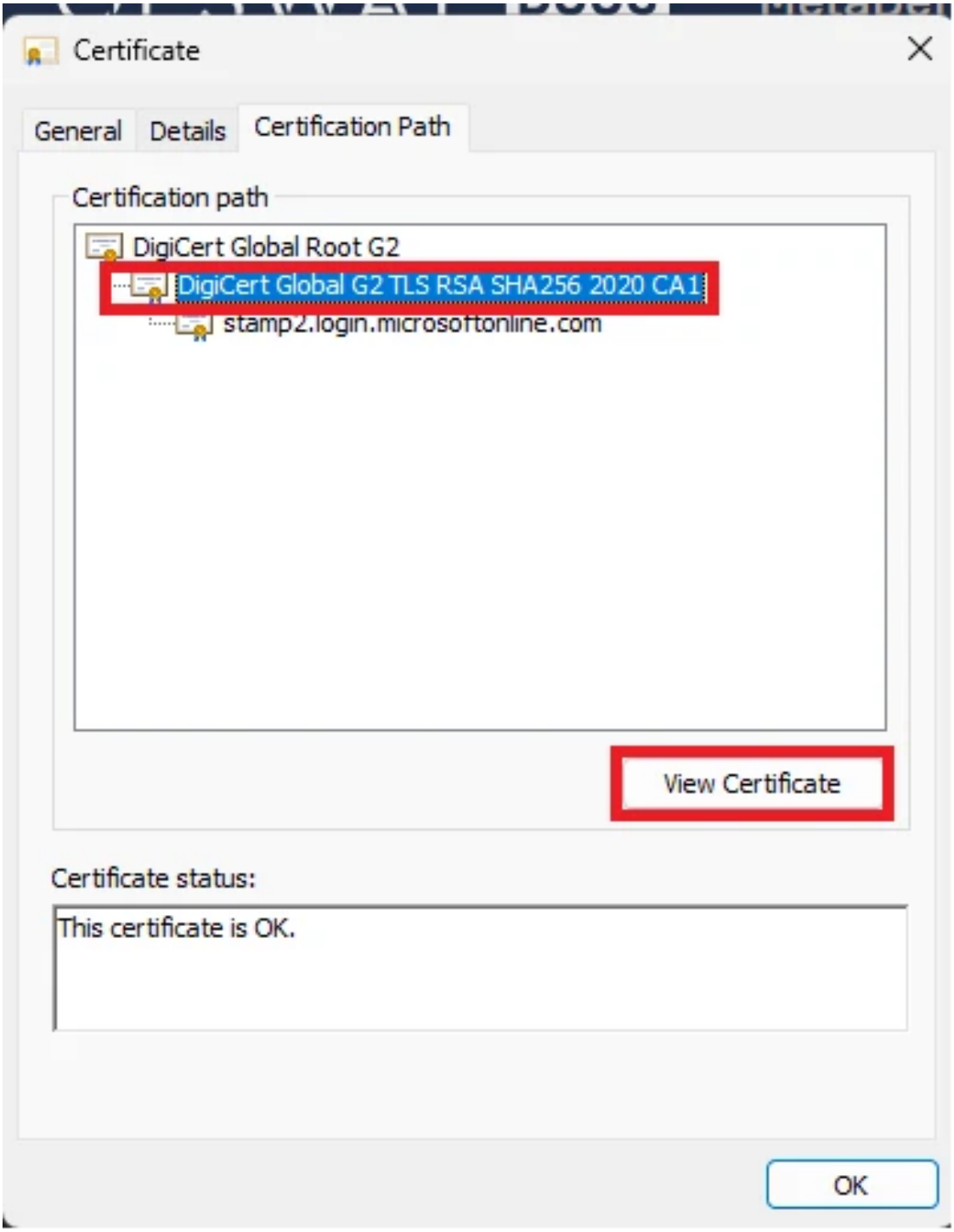


- Export the first certificate (subject certificate)
  - Select Base-64 encoded ASCII, single certificate (*.crt)
  - Save it as `stamp2.login.microsoftonline.com.crt`

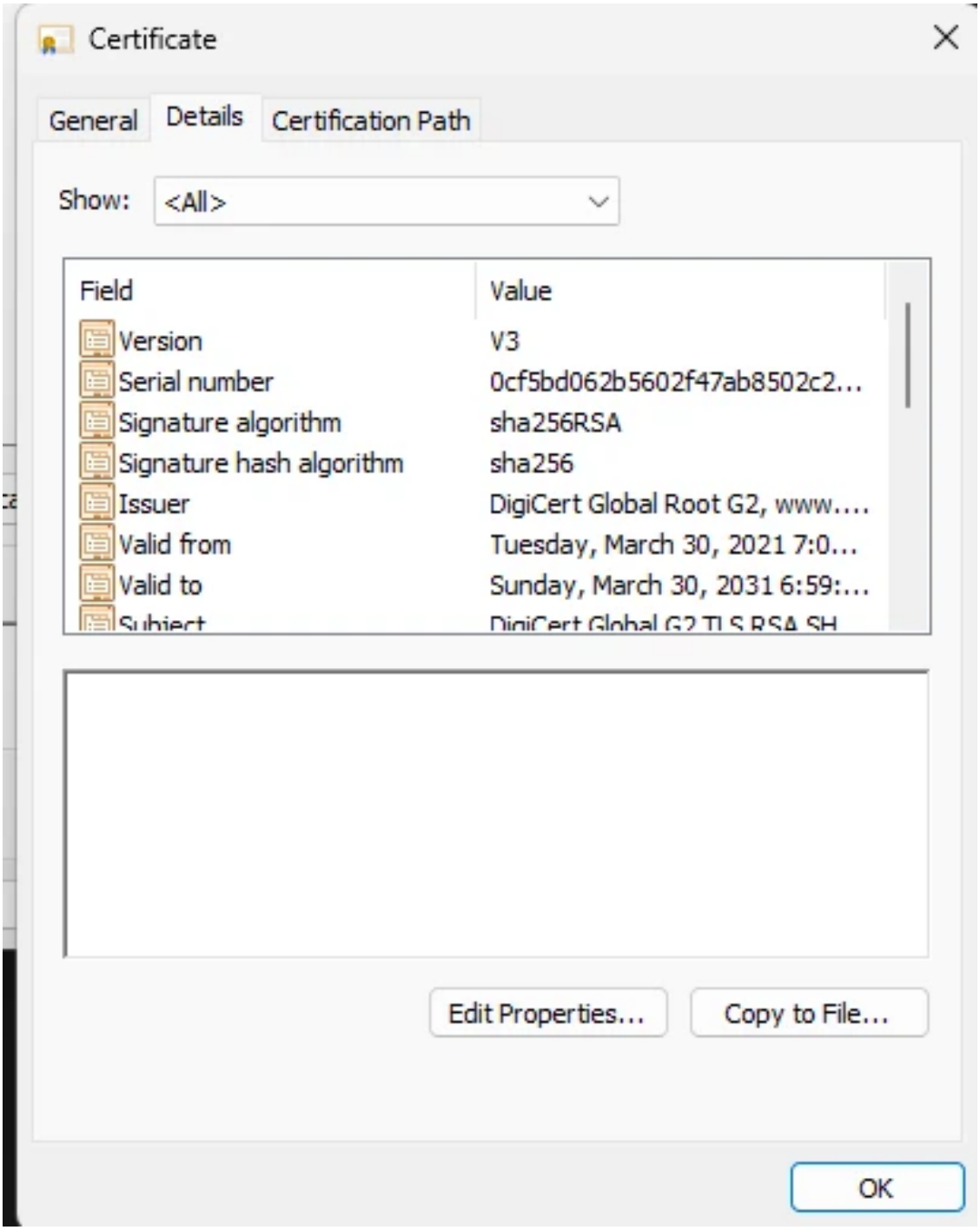


3. Export the first certificate (subject certificate)

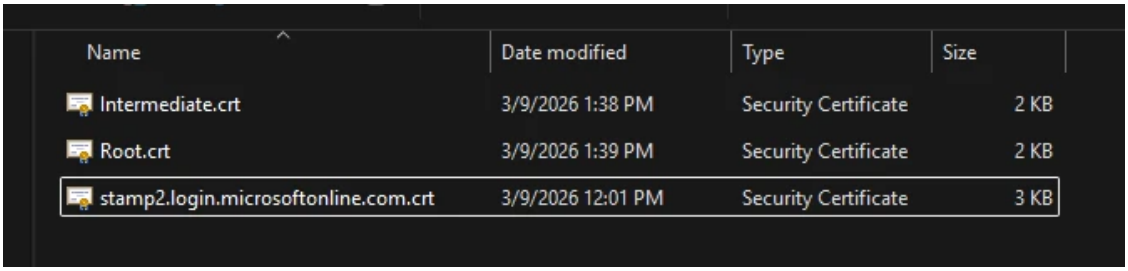
- In the certificate viewer, go to the Certification Path tab.



- Select the second certificate (the intermediate CA), then click View Certificate.
- Go to **Details** → **Copy to File**.
  - Select **Base-64 encoded X.509 (.CER)** format.
  - Save it as `Intermediate.crt` (removed `.cer`)



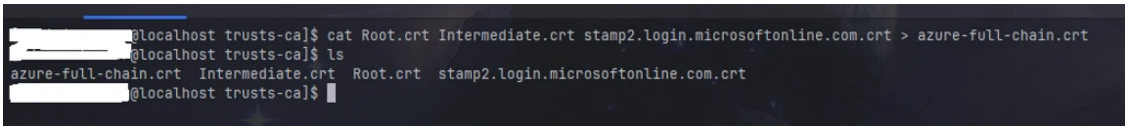
- Repeat the same process for the root certificate:
  - Save it as `Root.crt` (removed `.cer`) .
- You should now have three certificate files:
  - `stamp2.login.microsoftonline.com.crt` (subject/leaf)
  - `Intermediate.crt` (intermediate CA)
  - `Root.crt` (root CA)



- Create a fullchain cert with the linux command:

Code · bash

```
cat Root.crt Intermediate.crt stamp2.login.microsoftonline.com.crt > azure-full-chain.crt
```



4. Mount the volume for certificates into the trusted root certification authorities

- Given that the full chain certificate is stored at ./trusts-ca/azure-full-chain.crt, then we can mount it to the container with the following example:

Code · bash

```
docker run -d  \
   --name core \
   ...
   -v ./trusts-ca/azure-full-chain.crt:/etc/ssl/certs/azure-full-chain.crt \
   ...
   opswat/metadefendercore-debian-nonroot:5.17.1
```

Database dead-tuple alert: what it means and how to tune PostgreSQL autovacuum

Dead tuples (what they are)

In PostgreSQL, each row in a table is stored as a tuple. When a row is updated or deleted, PostgreSQL does not immediately remove the old copy from disk. The old version becomes obsolete and is referred to as a dead tuple (also called “dead rows”). Dead tuples remain until PostgreSQL runs VACUUM, which reclaims that space and keeps the database efficient.

If dead tuples accumulate, tables grow larger than necessary, queries and maintenance can take longer, and the database may use more disk I/O and storage than required.

How this relates to MetaDefender Core

During normal operation, MetaDefender Core creates, updates, and deletes database records as files are submitted and processed (for example, as scan state and results change over the lifecycle of a submission). Over time, this routine activity produces dead tuples in the same way any busy application database does.

Dead tuples are expected in a working system; the important part is that regular maintenance (autovacuum/VACUUM) keeps them under control so performance and storage use stay healthy.

When the system sends an email notification

Please check more details about email notification for dead tuples: Email Notification

The database monitoring feature sends an email when the estimated number of dead tuples exceeds a configured threshold (for example, 200,000).

> *Note: Thresholds, intervals, and severity may vary if your deployment uses non-default configuration. Use your administrator-defined settings as the source of truth.*

Bundled PostgreSQL compared with remote PostgreSQL

Whether you use the bundled PostgreSQL included with MetaDefender Core or a remote (external) self-managed PostgreSQL instance affects how often you are likely to see this alert.

- Bundled PostgreSQL: **The database is delivered with settings intended to keep dead tuples under control. In typical deployments, this alert is uncommon.**
- Remote PostgreSQL: Maintenance behavior depends on your organization's PostgreSQL configuration. If autovacuum is disabled, delayed, or tuned to run only at higher dead tuples levels than your alert threshold, you may see more frequent or recurring notifications until the database is tuned appropriately.

Bundled PostgreSQL (default installation)

When MetaDefender Core is installed with the bundled PostgreSQL database:

- Autovacuum is enabled by default, so PostgreSQL can reclaim dead tuples automatically during normal operation.
- The bundled configuration is aligned so that autovacuum is expected to address dead tuples before they reach typical alert thresholds (for example, autovacuum may begin cleanup around 100,000 dead tuples while an alert might be configured around 200,000—exact values depend on your environment).

Expected outcome: Dead tuples are routinely reclaimed, so the dead tuples count usually does not reach the email alert threshold. **As a result, administrators rarely receive this email on a correctly operating bundled database.**

Remote PostgreSQL (external / self-managed)

If MetaDefender Core uses a remote PostgreSQL instance that your organization hosts and maintains, **autovacuum behavior is determined by your database configuration**, not by the MetaDefender Core application alone.

Misalignment between your alert threshold (for example, 200,000 dead tuples) and your PostgreSQL maintenance

settings can cause repeated or intermittent email notifications.

Conditions that commonly lead to frequent alerts

The examples below assume an email alert threshold around 200,000 dead tuples (your environment may differ).

Scenario	What happens	Typical impact
Autovacuum is **disabled**	Dead tuples are not reclaimed automatically	Dead tuple counts can rise steadily; you may receive alerts on each evaluation cycle, subject to the 24-hour cooldown between emails
Autovacuum thresholds are higher than the alert threshold	Vacuum runs only after dead tuples exceed the database's configured trigger	The alert can trigger before autovacuum begins cleanup, which may produce recurring notifications until settings are aligned
Autovacuum is enabled but runs too slowly or too infrequently for workload	Dead tuples accumulate between successful vacuum cycles	Counts may sporadically cross the alert threshold during busy periods

Recommended action for remote PostgreSQL administrators

Work with your database administrator to ensure PostgreSQL autovacuum is enabled and tuned so that routine cleanup begins before dead tuples reach your MetaDefender Core alert threshold.

A practical approach is to align autovacuum settings with the bundled PostgreSQL defaults supplied with MetaDefender Core (or your organization's approved baseline). This typically reduces unnecessary dead tuples growth and helps prevent routine alert emails caused by maintenance lag.

> *Note: PostgreSQL tuning should consider server CPU, memory, disk performance, and database workload. Apply changes through your standard change-management process, validate in a non-production environment when possible, and reload/restart PostgreSQL according to your configuration method (`ALTER SYSTEM` + `pg_reload_conf()`, or equivalent).*

auto vacuum metrics	value
`autovacuum`	true
`autovacuum_vacuum_threshold`	100000

auto vacuum metrics	value
`autovacuum_vacuum_scale_factor`	0

Guidance on the recommended settings

The settings described in this guide are intended to encourage PostgreSQL to run autovacuum before dead tuples reach levels that commonly trigger MetaDefender Core alerts (for example, by using a lower autovacuum trigger than a typical 200,000 dead-tuple alert threshold, such as a 100,000-tuple-oriented baseline when paired with appropriate scale factors).

These values are recommendations, not a universal standard. You should tune PostgreSQL to match your hardware capacity, workload patterns, maintenance windows, and organizational policies.

What matters most

- Autovacuum must remain enabled for routine dead tuples cleanup.
- Autovacuum vacuum thresholds and scale factors should be set to reasonable values for your database size and activity so cleanup runs predictably and before dead tuples accumulate to the point where they affect performance or generate frequent monitoring alerts.

If you are unsure, validate changes in a non-production environment first, then monitor dead tuples trends, autovacuum activity, and system resource usage after deployment.

MetaDefender Core Deployment Readiness Tool

Overview

MetaDefender Core Deployment Readiness Tool (`ometascan-readiness`) is a small standalone program that checks whether a host is ready to **install** or **upgrade** MetaDefender Core v5. It runs in a few seconds, makes **no changes** to your system, and tells you up front whether the host meets the official requirements — so you find problems before the installer does, not during.

The tool is **read-only**: it never modifies Core configuration, the database, services, firewall rules, packages, users, or permissions.

Download HERE

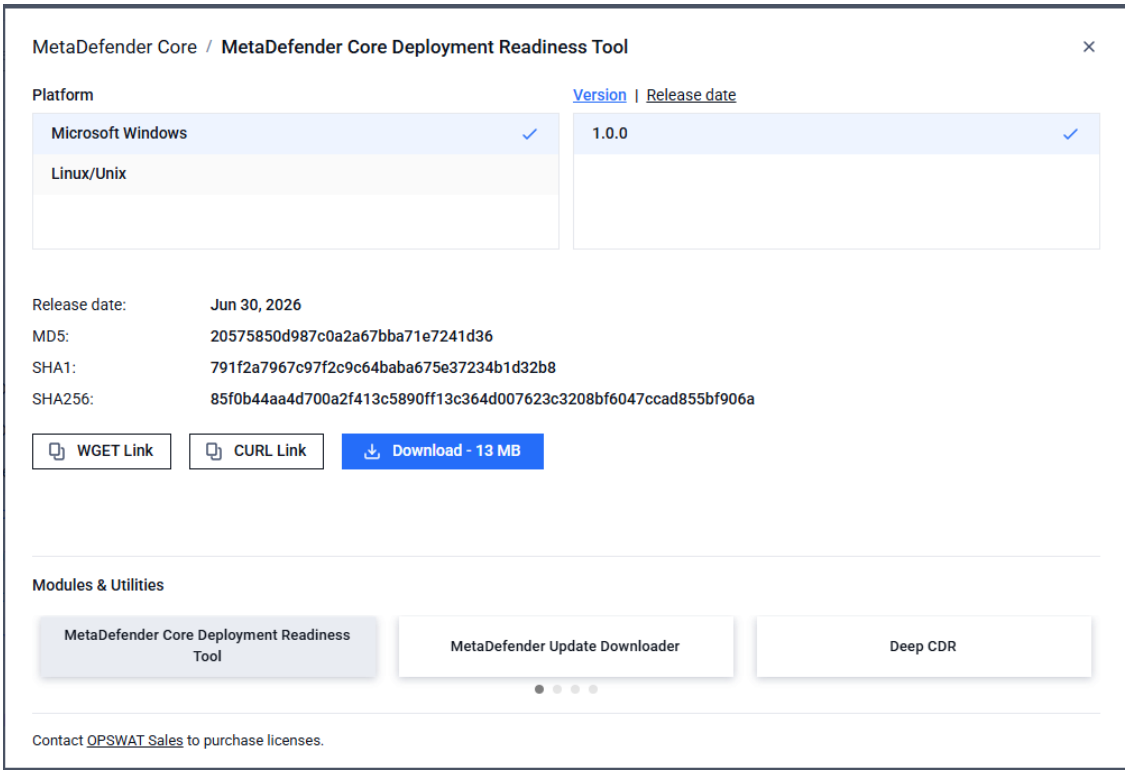
What It Checks

- Whether this is a fresh install or an upgrade (auto-detected)
- OS, architecture, CPU, RAM, disk space, and storage type
- Linux packages / Windows prerequisites
- REST API and PostgreSQL port availability
- Network reachability to OPSWAT update, download, and activation endpoints
- User, directory, file-limit, service-manager, and service-account readiness

- PostgreSQL connectivity, version, database size, and backup space
- Existing Core service or package conflicts before an upgrade

Before you run

The Deployment Readiness tool is available for download from My OPSWAT portal.



There are a few things before you run:

- **Run with administrator / root privileges.** On Linux, use `sudo`; on Windows, use an **elevated PowerShell** (Run as administrator). Without elevation, some checks (registry, service account, system directories) return *Unknown* because the OS denies the read.
- **For a fresh install:** know how many **AV engines** you plan to deploy. If you omit it, the tool prompts you interactively.
- **For an upgrade:** know the **target Core version** (e.g. `5.20.0`). Recommended but optional.

No prerequisites to install — the tool is a single self-contained binary.

How to run

Quick start

Linux:

Code · bash

```
chmod +x ometascan-readiness
sudo ./ometascan-readiness
```

Windows:

Code · powershell

```
.\ometascan-readiness.exe
```

The tool auto-detects an existing Core install. If found, it reads the current version and engine count automatically; if not, it prompts you for the engine count. When finished, it prints a console summary and writes timestamped `.html` and `.json` reports to the current directory.

Interactive mode & default values:

When you run the tool in a terminal without supplying all inputs on the command line, it **prompts you for the missing inputs**. Each prompt shows its **default value in brackets** — press **Enter** to accept the default, or type your own value to override it.

Example:

Code · bash

```
Number of AV engines planned? [10]:
Will you use Deep CDR or Proactive DLP? [y/N]:
PostgreSQL type? [bundled/remote] (default: bundled):
```

The defaults are:

Prompt	Default	Notes
Number of AV engines planned/ installed	**10**	Accepts `0`–`9999`. Override with `--av-engines N`.
Will you use Deep CDR or Proactive DLP?	**No**	Adds CPU/RAM sizing when Yes. Override with `--cdr` / `--dlp`.
Will you use Embedded Sandbox?	**No**	Adds sizing when Yes.
Will you use SBOM generation?	**No**	Adds sizing when Yes.
What type of PostgreSQL do you plan to use?	**bundled**	Choose `bundled` or `remote`. Override with `--db-type`.

Accepting the default: pressing Enter at any prompt selects the bracketed value. Invalid input **(a non-number, or a choice outside the listed options)** is rejected and the prompt repeats.

Info

When prompts are skipped (non-interactive mode): if you pass `--quiet` or `--report json`, the tool does **not** prompt. It uses these same defaults for the optional questions, but `--av-engines` becomes **required** for a fresh-install check and the tool exits with an error if it is missing.

Common scenarios

Fresh install:

Hardware is sized for the engine count plus any features you enable (`--cdr`, `--dlp`, `--sandbox`).

Code · bash

```
# 10 engines, with Deep CDR enabled
sudo ./ometascan-readiness --av-engines 10 --cdr
```

Code · powershell

```
# 8 engines, with Sandbox enabled
.\ometascan-readiness.exe --av-engines 10 --sandbox
```

Upgrade:

The tool detects your current version and engine count, validates the PostgreSQL version and database size, and checks that there is enough **backup space** for the pre-upgrade backup. It also warns if the Core service is still running.

Code · bash

```
# See which target versions this build knows about
./ometascan-readiness --list-versions

# Check readiness for upgrading to 5.20.0
sudo ./ometascan-readiness --target-version 5.20.0
```

Parameters

Parameters	Purpose
`--av-engines N`	Planned AV engine count (required for non-interactive fresh install).
`--cdr / --dlp / --sandbox`	Add CPU/RAM sizing for these features.
`--target-version X.Y.Z`	Validate against a specific Core version profile.

Parameters	Purpose
`--offline`	Air-gap mode: skip online checks; network failures become advisory INFO.
`--db-type remote`	Check a remote PostgreSQL (see below).
`--report all|html|json|none`	Choose which report files to write (default `all`).
`--output PATH`	Where to write reports. Use `-` to stream JSON to stdout.
`--quiet`	Suppress progress output (forces non-interactive).
`--strict-warnings`	Return exit code `1` when only warnings are present.
`--list-versions`	List bundled target-version profiles, then exit.

Limitation

- The tool only checks readiness — it does not install or upgrade Core.
- The schema check validates upgrade/downgrade *direction*; it does not prove every intermediate migration step is supported.
- Older target-version profiles may have incomplete historical schema data.

Read the report

After executing the binary, open the HTML/JSON report, or read the console summary:

Code · bash

```
==============================
 Readiness Summary
==============================
PASS: 18   WARN: 2   FAIL: 0   INFO: 11
```

Status	Meaning	What to do
PASS	Requirement met.	Nothing.
INFO	Informational context.	Read for awareness.

Status	Meaning	What to do
WARN	Below recommended, not blocking.	Review and fix before production load.
FAIL	A blocker.	Fix before running the installer.

Upgrade to MetaDefender Core 5.21.0 or later on Windows systems with a non-ASCII locale

Check your MetaDefender Core version and Operating System

This guide applies to MetaDefender Core on **Windows**, when **upgrading** to version **5.21.0 or later**.

Summary

Starting with MetaDefender Core 5.21.0, the product bundles **PostgreSQL 18**. PostgreSQL 18 cannot initialize its database cluster if the Windows system locale name contains non-ASCII characters. For example, Turkish Windows uses the locale `Turkish_Türkiye.1254`, which contains `ü`.

Check if your system is affected

Before upgrading, open PowerShell and run:

```powershell
Get-WinSystemLocale
```

If the returned locale name contains non-ASCII characters, your system is affected and you must follow the procedure below.

If the locale name contains only ASCII characters, no action is required — upgrade as usual.

Alternative: upgrade without preserving processing history

If your system is affected but you do not need to retain your processing history, you can use the following simpler procedure instead of the database backup and restore steps.

1. Export current configuration.
2. Fresh install new MetaDefender Core v5.21.0 or latest version
3. Import back the configuration to new installed MetaDefender Core.

How to upgrade safely

Follow these steps in order. Your scan history, license, and settings will be preserved at the end.

Step 1 — Back up your data from current installed Core

1. Open **Command Prompt as Administrator**.
2. Change to the PostgreSQL binary directory of the current installation: `<Installation Folder>\postgres\bin`. Example:

```
Code

cd "C:\Program Files\OPSWAT\MetaDefender Core\postgres\bin"
```

3. Export the database:

```
Code

pg_dumpall.exe -h <pg_host> -p <pg_port> -U <pg_user> -f <backup_location>\Core_backup.sql
```

   1. Example: `pg_dumpall -h localhost -p 5432 -U postgres -f C:\OPSWAT\Core_backup.sql`

4. If you are prompted for a password, enter the password configured for the bundled PostgreSQL instance.
5. Confirm that `Core_backup.sql` was created and is not empty.

Step 2 — Edit the backup file

1. Open `Core_backup.sql` in a text editor.
2. Remove **every occurrence** of the keyword `UNLOGGED`. This prevents errors when the dump is restored into the new database.
3. Save the file.

Example:

- **Before** (what's in the dump file):

```
Code · sql

CREATE UNLOGGED TABLE cached_scan.schema_version (
    version smallint,
    instance_id smallint
)
PARTITION BY LIST (instance_id);
```

- **After** (what it should look like):

Code · sql

```
CREATE TABLE cached_scan.schema_version (
    version smallint,
    instance_id smallint
)
PARTITION BY LIST (instance_id);
```

Step 3 — Uninstall current MetaDefender Core

Uninstall the current MetaDefender Core.

Step 4 — Install new MetaDefender Core version

Run the Core installer. Once installation completes, do not activate or configure anything yet.

Step 5 — Restore your data

1. Stop the **MetaDefender Core** service.

2. Following this guideline to manually start the bundled PostgreSQL service.

3. Open **Command Prompt** as Administrator and change to the PostgreSQL binary directory:

Code

```
cd "C:\Program Files\OPSWAT\MetaDefender Core\postgres\bin"
```

3. List all databases to find the MetaDefender Core database name:

Code

```
psql.exe -h localhost -p 5435 -U postgres -c "\l"
```

Look for the database whose name begins with `metadefender_core_` — for example, `metadefender_core_h21nqf`. Note this name; you need it in the following steps.

4. Drop the existing database:

Code

```
psql.exe -h localhost -p 5435 -U postgres -c "DROP DATABASE \"metadefender_core_h21nqf\";"
```

5. Create an empty database with the same name:

Code

```
psql.exe -h localhost -p 5435 -U postgres -c "CREATE DATABASE \"metadefender_core_h21nqf\";"
```

6. Restore the backup into the new database:

```
Code

psql.exe -h localhost -p 5435 -U postgres -d metadefender_core_h21nqf -f "C:
\OPSWAT\Core_backup.sql"
```

Depending on the size of the dump, the restore may take several minutes.

7. Start the **MetaDefender Core** service.

> ***Note:*** *Replace* `metadefender_core_h21nqf` *in the commands above with the actual database name you identified in step 3. The suffix is generated per installation and differs on every system.*

Step 6 — Verify

Open the MetaDefender Core web management console and confirm the following:

- **Processing History** shows your previous scan records.
- **Settings** retain your previous configuration.
- Your license is still active.

Need help?

If you encounter errors during the restore step or MetaDefender Core does not start after the upgrade, contact OPSWAT Support for help.

PostgreSQL fails to start after upgrade when SSL is enabled

> This issue may occur only when upgrading MetaDefender Core from a version **prior to 5.22.0 to version 5.22.0 or later**, on deployments **where SSL is enabled for the bundled PostgreSQL database**.
>
> It can be resolved with a configuration change, and no re-installation is required.

Overview

After upgrading MetaDefender Core from a version earlier than 5.22.0 to 5.22.0 or later, the bundled PostgreSQL service may fail to start if SSL was previously enabled for it without an `ssl_groups` setting. MetaDefender Core then cannot reach its database.

This applies only to standalone deployments that use the bundled PostgreSQL.

Symptoms

- The upgrade completes and the package reports as installed, but the Web Console is not usable after the upgrade
- Sign-in requests fail with HTTP `500`, while `GET /version` still answers with the new version
- The MetaDefender Core log contains the following errors:

```
Code

[ERROR   ] (core.postgres) Postgres issue, error='Starting postgres service failed',
param='pg_ctl: could not start server
Examine the log output.
'
[ERROR   ] (core.postgres) Postgres issue, error='Failed to restart PostgreSQL under FIPS after
cluster upgrade'
[ERROR   ] (common.base) Unhandled internal exception occurred in a slot, message='Failed to
re-engage FIPS for the bundled PostgreSQL after cluster upgrade
```

- Starting PostgreSQL manually reports the following error:

```
Code

FATAL:  could not set group names specified in ssl_groups: No valid groups found
HINT:  Ensure that each group name is spelled correctly and supported by the installed version
of OpenSSL.
```

Root Cause

From MetaDefender Core 5.22.0, the bundled PostgreSQL ships with a FIPS 140-3 validated OpenSSL module, and non-approved algorithms such as X25519 are not available to the database server. The PostgreSQL default for `ssl_groups` is `X25519:prime256v1`, so a PostgreSQL SSL configuration that does not set `ssl_groups` makes the service fail to start.

SSL configurations created for earlier versions do not contain `ssl_groups`. The upgrade keeps that configuration file, and PostgreSQL fails to start with it when the upgrade restarts the database.

Solution

1. Open the PostgreSQL SSL configuration file (the file that contains `ssl = on`) in the PostgreSQL folder:

   - **Windows:** `<installation folder>/postgres/`
   - **Linux:** `/usr/lib/ometascan/postgres/`

2. Correct the setting named in the error:`ssl_groups`: set a FIPS-approved groupFor example:

```
Code

ssl = on
ssl_groups = 'prime256v1'
```

```
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_cert_file = '<absolute_path_to_cert_file>'
ssl_key_file = '<absolute_path_to_key_file>'
ssl_ca_file = '<absolute_path_to_root_ca_cert_file>'
```

3. Restart the MetaDefender Core service:

Code

```
# Use systemctl
systemctl restart ometascan

# Or use service
service ometascan restart
```

License activation or engine download fails on a clean Windows Server environment

Applies to: MetaDefender Core installed on a clean Windows Server environment

Symptoms

- MetaDefender Core installs and starts normally.
- The server has internet access. Browsers and other tools on the same server can reach the internet, and the MetaDefender Core update server is reachable.
- One of the following issues occurs:
  - Online activation fails with the error: `Could not connect to the activation server. Check your Internet connection or use offline activation method`
  - Activation succeeds, but the engines are never downloaded.

Confirm the cause

The MetaDefender Core log contains the following errors:

Code

```
[WARNING] (common.communication) SSL errors while downloading file, url='https://
activation.dl.opswat.com/activation?...'
[WARNING] (common.communication) Certificate chain for the ssl error, ...
```

```
[WARNING] (common.communication) Certificate chain item, index='1', ...
issuerCommonName='QList("Amazon Root CA 1")', ... subjectCommonName='QList("Amazon RSA 2048 M01")'
[WARNING] (common.communication) Certificate chain item, index='2', ...
issuerCommonName='QList("Starfield Services Root Certificate Authority - G2")', ...
subjectCommonName='QList("Amazon Root CA 1")'
[WARNING] (common.communication) SSL error, error='...'
[WARNING] (common.update) Activation error, see error code, error='5'
[ERROR  ] (common.update) Error during activating the product, errorString='Could not connect to
the activation server...'
```

Root Cause

The activation server's certificate is issued by Amazon. It chains up to **Starfield Services Root Certificate Authority - G2**, a root certificate that Microsoft trusts.

Windows does not install all trusted root certificates up front. It downloads each one from Windows Update the first time a Windows program needs it. On a newly installed server, nothing has needed this root yet, so it is not in the certificate store.

MetaDefender Core does not trigger that download. It checks certificates against the roots already in the store, and it reads the store **when the service starts**. With the root missing, Core rejects the connection and reports it as a connection failure.

Solution

In most cases, restarting the MetaDefender Core service resolves the issue, because the service reloads the certificate store when it starts.

MDCLI User Guide

`mdcli` is a cross-platform command-line client for **MetaDefender Core (MD Core)**. It submits one file, or streams a list of up to millions of file paths, to MD Core's REST API and emits the analysis results as JSON — designed to be embedded in CI/CD pipelines, kiosks/airgap transfer stations, and high-volume batch jobs.

- Results go to **stdout**; progress and errors go to **stderr**.
- The batch mode emits **NDJSON** (one JSON object per line), streamed as each file completes, with **O(1) memory** regardless of list size.
- A deterministic **exit-code contract** makes it scriptable.

`mdcli` is provided by OPSWAT as a **free utility** for MetaDefender Core customers. Version 1.0.0 is **validated with MetaDefender Core 5.20.0 and later versions.** See the release notes for what is in each version.

---

1. System requirements

`mdcli` ships as a **single, statically-linked native binary** (built with `CGO_ENABLED=0`). It contains no interpreter and links no external shared libraries, so there is nothing to install alongside it.

Windows

Item	Requirement
OS	Windows 10 / 11, or Windows Server 2016 or later
Architecture	x64 (amd64)
Runtime	None. No .NET, no Visual C++ redistributable
Privileges	Standard user (no administrator rights required)
Certificate store	Uses the built-in Windows certificate store for HTTPS

Linux

Item	Requirement
Distributions (validated)	SUSE Linux Enterprise 15+, Red Hat Enterprise Linux 8+
Distributions (general)	Any Linux with kernel 3.2 or newer (Debian/Ubuntu,

Item	Requirement
	Rocky/Alma, Amazon Linux, etc.)
Architecture	x86-64 (amd64) and ARM64 (aarch64)
C library	**None required.** The binary is fully static — no glibc/musl dependency (`ldd ./mdcli` reports "not a dynamic executable")
Privileges	Standard user; must have read access to the files being scanned
CA certificates	For HTTPS to MD Core with certificate verification, the system CA bundle (`ca-certificates`) should be installed (see §2)

> **Note**
> Because there is no libc dependency, a binary built once runs unchanged across SUSE 15, RHEL 8, and newer distributions — the classic "built on a newer distro, won't run on an older one" glibc problem does not apply.

---

2. Dependencies

Runtime dependencies of the binary: none. It is self-contained.

The only external things `mdcli` needs are environmental, not packages:

1. **Network reachability to MD Core's REST API** — by default `http://localhost:8008`. Ensure the host/port is reachable and any firewall permits it.
2. **CA certificates, only when using HTTPS with verification.**
   - **Windows:** handled automatically by the OS certificate store — nothing to do.
   - **Linux:** install the distro CA bundle so Go can verify the server certificate:
     - RHEL/Rocky/Alma: `sudo dnf install ca-certificates`
     - SUSE: `sudo zypper install ca-certificates`
     - Debian/Ubuntu: `sudo apt-get install ca-certificates`
   - Plain `http://` (e.g., a co-located Core on localhost) needs no certificates.
   - For a private/self-signed CA, either add the CA to the system trust store, or (dev/test only) pass `--tls-skip-verify`.

3. **Read access** to every file path you submit. `mdcli` streams file contents to Core; it does not copy files elsewhere or require temp space.

---

3. Installation

`mdcli` is distributed as a prebuilt binary — there is nothing to compile.

Release binaries are version-stamped (e.g. `1.0.0`) so releases are easy to tell apart.

1. Download the appropriate binary from My OPSWAT and copy it to the target host:
   - Linux x64: `mdcli-1.0.0-linux-amd64`
   - Linux ARM64: `mdcli-1.0.0-linux-arm64`
   - Windows x64: `mdcli-1.0.0-windows-amd64.exe`
2. **Linux** — place it on the `PATH` and make it executable:

```sh
sudo install -m 0755 mdcli-1.0.0-linux-amd64 /usr/local/bin/mdcli
mdcli --version        # -> mdcli 1.0.0
```

3. **Windows** — copy `mdcli-1.0.0-windows-amd64.exe` to a folder (e.g. `C:\Program Files\mdcli\`), optionally rename to `mdcli.exe`, and add that folder to `PATH`:

```powershell
mdcli.exe --version    # -> mdcli 1.0.0
```

> **Note**
>
> **Upgrading:** always confirm `mdcli --version` reports the expected number after installing, and check `command -v mdcli` (Linux) in case an older copy earlier in `PATH` is shadowing the new one.

---

4. Quick start

```sh
# 1. Point at your Core (env vars keep the API key out of shell history).
export MDCORE_URL="http://localhost:8008"
export MDCORE_APIKEY="<your-api-key>"

# 2. Scan a single file -> one indented JSON object on stdout.
```

```
mdcli --file ./installer.msi

# 3. Scan a list of paths -> NDJSON, one result per line, streamed live.
mdcli --from-file ./paths.txt > results.ndjson

# 4. Feed paths from another command via stdin.
find /data/incoming -type f | mdcli --from-file -
```

5. Command reference

Code

```
mdcli [options]   (exactly one of --file, --from-file or --directory is required)
```

Input / output

Option	Description
`--file <path>`	Scan a single file. Emits one indented JSON object.
`--from-file <path>`	Scan every newline-separated entry in the file. `-` reads the list from stdin. Emits NDJSON. Each entry is either a local file path or an `http(s)://` download URL (see below); the two may be mixed.
`--directory <root>`	Recursively scan regular local files beneath this root, including hidden files. Emits NDJSON.
`-o, --output <path>`	Write results to a file instead of stdout (`-` = stdout).
`--unscannable-as-skipped`	Report files Core could not examine (encrypted/ password-protected, failed-to-scan, timeout, not-scanned) in the skipped report instead of counting them as detections. **Off by default.**
`--output-blocked`	Emit results only for files with a `Blocked` verdict (`MDCORE_OUTPUT_BLOCKED`). The run summary on stderr still counts every file.
`--version`	Print version and exit.

Note

`--output-blocked` filters the JSON that reaches stdout; it's independent of `--fail-on` (which sets the exit code). A Blocked result carries the full AV/Deep CDR/sandbox detail under `result` for the client to inspect. Files that aren't Blocked — including non-Blocked errors — are simply not emitted.

Recursive directory scanning

Code · sh

```
mdcli --directory /data/incoming --run-dir /var/tmp/mdcli-runs/{runid}
mdcli --directory /data/incoming --batch-name backup-scan --no-hash-lookup
```

On Windows, use a quoted path such as `--directory "C:\Backup Files"`. The directory is on the machine running mdcli; files are uploaded to the Core selected by `--url`. Discovery streams through the existing bounded worker pool, without building a complete path list. Results arrive in completion order and use absolute local paths in `cli.input_path`. Core receives the complete path relative to the scan root: scanning `D:\backup\Users\Alice\report.txt` with `--directory D:\backup` reports `Users\Alice\report.txt` in Core history. The backup container is omitted automatically; no stripping or reporting-prefix option is needed. Directory separators follow the scan host's path syntax. Spaces, `#`, and (on Linux) newlines in filenames are preserved. Hidden files and subdirectories are included.

Submitted filenames are URL-encoded so Core retains directory components, spaces, and literal percent signs. CR/LF characters are removed from the name sent to Core; the original path remains in `cli.input_path`. Hash-cache hits reuse an existing Core record and do not create a new history entry with a new name; use `--no-hash-lookup` when each path must be recorded by Core.

Symbolic links inside the tree and special files (such as sockets and named pipes) are excluded; links are not followed. A root supplied through a symlink is resolved first. Zero-byte files use the existing skipped-file behavior. An empty directory completes successfully with no result records. Discovery is not a filesystem snapshot: avoid modifying the tree during a scan.

`--directory` cannot be combined with `--file`, `--from-file`, or `--prefix` (including a prefix inherited from configuration or the environment). Like the other input selectors, it is command-line only. Existing concurrency, hash lookup, detection, filtering, reporting, and batch options apply. Use `--no-hash-lookup` when every nonempty file must be uploaded into a Core batch.

Keep all output artifacts outside the input tree. Paths supplied through mdcli options are checked before files are created, including with `--force`; shell redirection is outside mdcli's control. An invalid/unreadable root exits 2 before any Core request. A later directory traversal error stops discovery, completes in-flight work and batch cleanup, and exits 5 with an error on stderr. Individual file read errors retain the usual per-file error reporting and exit behavior.

Comments and blank lines. In a list file, blank lines and comment lines are ignored. A line is a comment when its first character is `#` **followed by a space**, or when the line is only `#`. A `#` that is not followed by a space is treated as part of the path, since filenames may contain `#` — so `#tag.txt` and `## draft.bin` are paths, while `# note` and a lone `#` are comments.

Code

```
# nightly set — updated 2026-07-30
/data/incoming/report.pdf
#archived.bin        <- NOT a comment: a file literally named "#archived.bin"
# /data/skip-me.bin  <- a comment (disabled entry)
```

Download URLs in the list. Any entry beginning with `http://` or `https://` is handed to Core via the `downloadfrom` header — Core fetches and scans it itself, so nothing is stored on the client. Such entries skip the local hash lookup (there are no local bytes to hash) and are reported with `source: download`. A single list may freely mix local paths and URLs. This is convenient for testing malware/early-termination handling (e.g. the EICAR test file) without writing a malicious file to local disk:

Code

```
/data/incoming/report.pdf
https://secure.eicar.org/eicar.com.txt
/data/incoming/installer.msi
```

Path prefix (--prefix). When the paths in your list are *logical* source paths (e.g. from a backup catalog) but the files are actually reachable under a different mount on the scan host, `--prefix` bridges the two. It is prepended to each entry **only to read the file locally**; the original path is what appears in `cli.input_path` and is sent to Core as the filename (so Console reports and your catalog match). Example — catalog path `/foo/bar`, NFS mount `/mnt/malware/3`:

Code

```
mdcli --from-file catalog.txt --prefix /mnt/malware/3
#   list entry:   /foo/bar
#   file read:    /mnt/malware/3/foo/bar
#   reported as:  /foo/bar   (cli.input_path and Core display_name)
```

`--prefix` does not apply to `http(s)://` download-URL entries.

Files already accessible to Core

Use `--core-local` to ask Core to read a file from its own filesystem instead of uploading its contents. Supply an absolute path as seen by Core:

Code · sh

```
mdcli --file /srv/backup/report.pdf --core-local
mdcli --from-file core-paths.txt --core-local --batch-name backup-job
mdcli --directory /srv/backup --core-local --batch-name backup-job
```

The selected Core workflow must enable **Allow scan files on server**, allow the supplied directory, and have permission to read the files. A shared drive accessible to Core also works. Windows drive paths and UNC file paths are accepted even when the CLI runs on Linux; POSIX absolute paths are accepted when the CLI runs on Windows. Relative Core paths and download URLs are rejected.

For `--file` and `--from-file`, mdcli neither opens nor stats the target file: Core checks its existence, readability, size, and scan eligibility. The list file itself is still read by mdcli. Empty server files are handled by Core rather than the CLI's local zero-byte skip. Per-file byte timings are zero because the CLI does not inspect the file; Core's result contains its file size when available.

For `--directory`, mdcli must be able to enumerate the tree locally, and the resulting absolute paths must refer to the same files on Core. This works when both run on the same host or use an identical shared mount. Core history names remain relative to the directory root. For different mount paths, supply a list of Core-visible absolute paths instead. With file/list input, `--prefix` is prepended before sending the Core path, using the CLI host's path-joining rules; it is not included in the reported name.

Core-local mode always disables client hashing and cache lookup, even when `--hash-lookup`, configuration, or environment settings enable it. It never falls back to uploading. Results keep `source: scan`; the JSON schema is unchanged. Core's own hashing and workflow behavior are unaffected.

Client hash lookup is opt-in

By default mdcli submits files without calculating a client-side hash. Add `--hash-lookup` to calculate the selected `--hash-algo` and look up an existing Core result before uploading. Configuration `"hash_lookup": true` or environment `MDCORE_HASH_LOOKUP=true` also opts in, following the usual precedence rules. `--hash-algo` alone does not enable hashing.

Existing `--no-hash-lookup` remains a disabling compatibility flag; it cannot be combined with `--hash-lookup`. Even `--no-hash-lookup=false` does not opt in. `MDCORE_NO_HASH_LOOKUP=true` disables lookup if both old and new environment variables are set. With a config file active, those environment variables are ignored. Use `--hash-lookup=false` to override an opt-in config or environment.

Connection / authentication flags

Option	Env var	Default	Description
`--url`	`MDCORE_URL`	`http://localhost:8008`	MD Core base URL (http or https)
`--apikey`	`MDCORE_APIKEY`	—	API key. Prefer the env var (see §11)
`--rule`	`MDCORE_RULE`	—	MD Core workflow/rule to scan with
`--tls-skip-verify`	`MDCORE_TLS_SKIP_VERIFY`	`false`	Disable TLS certificate verification (dev/test only)
`--client-timeout`	`MDCORE_CLIENT_TIMEOUT`	`120s`	Per-HTTP-request timeout

Scan behavior

Option	Env var	Default	Description
`-j, --concurrency`	`MDCORE_CONCURRENCY`	`8`	Max files in flight at once (primary throughput/flow-control lever)
`--poll-interval`	`MDCORE_POLL_INTERVAL`	`500ms`	How often to poll each in-progress result
`--max-poll-wait`	`MDCORE_MAX_POLL_WAIT`	`10m`	Max time to wait for a single file's result before erroring it
`--no-hash-lookup`	`MDCORE_NO_HASH_LOOKUP`	`false`	Skip the pre-scan hash lookup; always upload
`--hash-lookup`	`MDCORE_HASH_LOOKUP`	`false`	Opt in to client hashing and pre-scan cache lookup; ignored in Core-local mode
`--core-local`	`MDCORE_CORE_LOCAL`	`false`	Submit a Core-visible absolute path with no upload or client hash; JSON config key `core_local`
`--hash-algo`	`MDCORE_HASH_ALGO`	`sha256`	Hash used for lookup: `md5|sha1|sha256|sha512`
`--retries`	`MDCORE_RETRIES`	`5`	Retries per request on transient failures (network error, 5xx, 503 queue-full)
`--prefix`	`MDCORE_PREFIX`	—	Path prepended to each input path **only when reading locally**; output and Core reporting keep the original path

Run control

Option	Env var	Default	Description
`--fail-on`	`MDCORE_FAIL_ON`	`blocked`	What counts as a detection: `blocked|infected|suspicious|never`
`--stop-on-detection`	`MDCORE_STOP_ON_DETECTION`	`false`	Abort the whole run on the first detection (per `--fail-on`)
`--batch`	`MDCORE_BATCH`	`false`	Group a `--from-file` or `--directory` run into one MD Core batch (list and directory modes)
`--batch-name`	`MDCORE_BATCH_NAME`	—	Batch name, sent as MD Core batch `user_data` (implies `--batch`)

Durations accept Go duration syntax: `500ms`, `30s`, `5m`, `1h30m`.

Monitoring a batch (`batch-status` subcommand)

A read-only subcommand to check a batch's progress by id — safe to run from a separate, concurrent process while the scan is still going:

Code

```
mdcli batch-status <batch_id> [--watch] [--json] [--watch-interval 2s]
```

Flag	Default	Description
`--watch`	`false`	Refresh until the batch is closed and all files have finished
`--watch-interval`	`2s`	Refresh cadence for `--watch`
`--json`	`false`	Emit one JSON object per refresh instead of a human line

Flag	Default	Description
(connection flags)		`--url`, `--apikey`, `--tls-skip-verify`, `--client-timeout` as above

Human output:

```
batch 4725a285…  finished 12 / 15 submitted  result=Processing  [open]
```

`[open]` vs `[closed]` reflects Core's `is_closed` (whether more files may still be submitted); `finished / submitted` is files completed over files added so far. See §9 for how to obtain the `batch_id` and the note on the denominator.

---

6. Configuration & precedence

Settings can come from a **config file**, an **environment variable**, or a **command-line flag**.

6.1 Config file

A JSON file holding the settings you want to reuse from run to run. Everything except *input selection* (`--file`/`--from-file`/`--directory`) and *command intent* (scan vs `batch-status`) can live in it — those stay on the command line.

Option	Description
`--config <path>`	Load this file **and disable auto-discovery**
`--no-config`	Ignore any auto-discovered file; use flags/env only
`--print-config`	Print the effective configuration (API key redacted) and exit

Code · json

```
{
  "url": "https://mdcore.internal:8008",
  "apikey_env": "MDCORE_APIKEY",
  "concurrency": 24,
  "retries": 5,
  "client_timeout": "120s",
  "fail_on": "infected",
  "prefix": "/mnt/malware/3",
  "run_dir": "/var/lib/mdcli/runs/{runid}",
```

```
  "log_level": "info"
}
```

Keys mirror the flag names with underscores (`client_timeout`, `hash_algo`, `stop_on_detection`, `output_blocked`, `skipped_file`, `stats_file`, `timing_file`, `log_file`, `log_dir`, `log_level`, `log_facility`, `log_success_limit`, `run_id`, `run_dir`, `manifest`, `force`). Durations are strings (`"120s"`). **Unknown keys are rejected**, so a typo fails loudly instead of being silently ignored.

6.2 Ignition file (auto-discovery)

If `--config` is not given, mdcli looks for a file named **mdcli.conf** and uses the **first** one it finds — files are never merged:

Order	Location	Linux	Windows
1	working directory	`./mdcli.conf`	`.\mdcli.conf`
2	next to the binary	`<exe dir>/mdcli.conf`	`<exe dir>\mdcli.conf`
3	user config directory	`~/.config/mdcli/mdcli.conf`	`%APPDATA%\mdcli\mdcli.conf`

`--config` overrides discovery; `--no-config` disables it. `--print-config` reports which file was actually loaded in its `config_source` field.

6.3 Precedence

1. **Command-line flag** — always wins.
2. **Config file** (explicit or auto-discovered), if one is active.
3. **Environment variable** — *only when no config file is active*.
4. **Built-in default.**

> **Note**
>
> **Environment variables and config files do not mix.** Once a config file is in play, ambient `MDCORE_*` variables are ignored, so a configured run is fully determined by the file plus the command line — no surprises from a stray variable in the service environment. The one exception is explicit opt-in:

Code · json

```
{ "apikey_env": "MDCORE_APIKEY" }
```

That tells mdcli to read the API key from the named variable, so the secret never has to be stored in the file. (mdcli errors out if the variable is unset.) With no config file at all, `MDCORE_*` behaves exactly as it always has.

The `MDCORE_*` variables describe the **MD Core server** the CLI connects to (its URL, key, rule, timeouts) — not the CLI itself.

Code · sh

```
mdcli --config /etc/mdcli/nightly.conf --print-config   # dry-run: what will be used, and where
output lands
```

7. Output format

Each result is a JSON object with two parts — an additive `cli` envelope and the **verbatim, lossless** MD Core `AnalysisResult`:

Code · json

```
{
  "cli": {
    "input_path": "/data/incoming/installer.msi",
    "source": "scan",
    "verdict": "Allowed",
    "scan_result": "No Threat Detected",
    "data_id": "…",
    "hash": "…",
    "hash_algo": "sha256"
  },
  "result": { "...": "full MD Core AnalysisResult, unmodified" }
}
```

- `source`: `scan` (uploaded), `hash-cache` (resolved via hash lookup, no upload), or `download` (Core fetched it from an `http(s)://` URL via `downloadfrom`).
- `verdict`: MD Core `process_info.result` — `Allowed`, `Blocked`, …
- On a per-file failure the object has `cli.error` set and omits `result`.

Single-file mode (`--file`) prints one indented object. **Multi-file mode** (`--from-file` or `--directory`) prints **NDJSON**: one compact object per line, flushed as each file finishes, so downstream tools can consume results live. Results arrive **out of order** (concurrency) — always key off `cli.input_path` or `cli.data_id`, never line position.

Consume with `jq` (one object per line):

Code · sh

```
mdcli --from-file paths.txt | jq -r 'select(.cli.verdict=="Blocked") | .cli.input_path'
```

8. Exit codes

Code	Meaning
`0`	Completed; nothing tripped `--fail-on`
`1`	Completed; ≥1 file tripped `--fail-on` (a detection)
`2`	Usage / argument error
`3`	Authentication or connection failure
`4`	Completed, but ≥1 file errored (e.g. upload/poll failure)
`5`	Internal CLI error

Detection is **decided by MD Core and parsed by the CLI** from `process_info.result` and `scan_results.scan_all_result_i`; see §14.

Transient failures & retries

Every request to Core (hash lookup, submit, poll) is retried on **transient** failures — network errors (including Core temporarily unreachable), any HTTP **5xx**, and **503** (queue full) — up to `--retries` times (default 5) with jittered exponential backoff (250 ms, doubling to a 10 s cap; 503 honors Core's `Retry-After`). Auth failures (401/403), 4xx, and local errors (missing file) are **not** retried.

If retries are exhausted, the last error is recorded on that file and the run continues to the next one (per-file failures never abort a batch). The failed file appears in the output with `cli.error` set (and no `result`), and it maps to an exit code:

- a **connection/network** failure or auth rejection → exit **3**;
- any other per-file error (e.g. a 5xx that kept failing, a per-file `--max-poll-wait` timeout) → counts toward exit **4** (completed with errors).

Note `--client-timeout` bounds each individual HTTP attempt, and `--max-poll-wait` bounds total time waiting for one file's result; a truly unreachable Core will therefore fail a file in roughly `retries × client-timeout` at worst — tune both for your environment.

Error reference

Per-file failures appear in the JSON as `cli.error` (with no `result` block) and are also echoed to stderr as `error: <input_path> - <message>`. Messages are stage-prefixed so clients can branch on them. **Local I/O is checked up front, so an unreadable file fails immediately — before any upload — and is never retried.**

cli.error	Cause	Retried?	Exit
`read: no such file or directory (<path>)`	Path missing, or broken symlink; with `--prefix` the *resolved* path is shown	no	4
`read: permission denied (<path>)`	File not readable by the mdcli user	no	4
`read: is a directory, not a file`	Entry points at a directory	no	4
`hash lookup: …`	Hash query to Core failed	transient only	3 / 4
`submit: …` / `submit url: …`	Upload or `downloadfrom` failed	transient only	3 / 4
`submit: authentication failed (HTTP 400|401|403): …`	Bad/missing API key	no	3
`poll: …`	Result retrieval failed	transient only	3 / 4
`poll: timed out waiting for scan to complete`	Exceeded `--max-poll-wait`	no	4

Run-level failures (no JSON emitted) go to stderr only:

stderr message	Exit
`error: cannot open --from-file: …`	2
`error: one of --file, --from-file or --directory is required`	2
`error: invalid --hash-algo "…" (want md5|sha1|sha256|sha512)`	2
`error: --file, --from-file and --directory are mutually exclusive`	2

stderr message	Exit

Files Core cannot examine. A password-protected/encrypted archive is returned by Core as `Blocked` with `scan_result: "Encrypted Archive"` (`scan_all_result_i` 12) — *not* as Infected, and with no engine detection. By default that still counts as a detection, because an encrypted archive is a recognised malware-delivery vector and suppressing it would weaken an existing gate. Pass `--unscannable-as-skipped` to instead record such files in the skipped report (reasons `encrypted`, `failed-to-scan`, `not-scanned`, `scan-timeout`) and keep them out of `--output-blocked`. A genuine Infected or Suspicious result is never re-routed. Extra `scan_all_result_i` values can be added via the `unscannable_codes` config key.

Notes:

- **Zero-byte files are skipped, not errors.** An empty file has no content to analyse, so mdcli skips it silently: nothing is uploaded, **no JSON record is emitted**, and it does not affect the exit code. Skipped entries are reported only as a count in the stderr summary (`… , N skipped`), so coverage still reconciles against the input list.
- **A bad file never aborts the run.** In list mode, mdcli reports the failure and continues; the stderr summary (`done: N scanned, N detected, N errored, N skipped`) gives the totals, and the process exits 4 if anything errored.
- **A failed download URL is reported by Core, not as a client error:** the record comes back `verdict: Blocked`, `scan_result: Failed`, with `result.download_info.status = "Download Failed"`. Under the default `--fail-on blocked` that counts as a detection (exit 1); use `--fail-on infected` if only real threats should gate the exit code.

---

9. Running large volumes in production

`mdcli` is built for lists up to ~1,000,000 paths. The design keeps memory flat: the input list is read lazily line-by-line, files are streamed (never buffered whole), and results are written and flushed one at a time.

9.0 Run artifacts (unattended callers)

A single scan can produce several output files. Rather than hard-coding a path for each — which a reused config would overwrite, and concurrent runs would corrupt — give the run an identity and let `mdcli` place the artifacts:

Code · sh

```
mdcli --from-file list.txt --run-dir /var/lib/mdcli/runs/{runid} --run-id job-12345
```

Code

```
/var/lib/mdcli/runs/job-12345/
    results.ndjson     scan results
    skipped.ndjson     entries with no verdict (errors + skips)
    stats.json         end-of-run summary
    timings.ndjson     per-file timings   (only with --timing-file)
    mdcli.log          diagnostics        (only when logging is enabled)
    manifest.json      run record + absolute path of every artifact
```

Determinism contract. When you pass `--run-id`, you already know every path: it is `<run-dir>/<fixed name>`. A server process needs no callback and no parsing — it can open `…/job-12345/results.ndjson` directly. If you omit `--run-id`, `mdcli` generates `<UTC-timestamp>-<pid>` and prints it on **stderr** (`run: <id> dir: <path>`); stdout stays pure NDJSON.

Option	Purpose
`--run-id <id>`	Run identity. Letters, digits, `.`, `_`, `-` only. Generated if omitted.
`--run-dir <path>`	Directory for this run's artifacts. Supports `{runid}` `{timestamp}` `{pid}` `{host}`.
`--manifest <path>`	Manifest location when not using a run directory.
`--force`	Overwrite existing artifacts instead of failing.

Notes:

- **--run-dir sends results to results.ndjson instead of stdout.** Pass `--output -` if you want to keep piping to `jq`.
- **An explicit path always wins**: `--skipped-file /elsewhere/s.ndjson` overrides the run-dir default, so artifacts can live on different volumes.
- **Diagnostic artifacts stay opt-in.** A run directory does not silently enable per-file timings or the diagnostic log; those still require their own flag.
- **Collisions fail loudly.** A non-empty run directory, or an existing artifact, is a usage error (exit 2) — a new session can never silently destroy the previous one. Files are created with `O_EXCL`, which is atomic, so two concurrent runs cannot race onto the same path either.
- Placeholders also work in `--output`, `--skipped-file`, `--timing-file`, `--stats-file`, `--log-file` and `--log-dir` if you prefer a flat layout, and all of them expand with the *same* timestamp within a run.

Manifest (`status` is `running` until the run ends, then `completed` or `failed`; artifacts that were never written are pruned):

Code · json

```
{"run_id":"job-12345","mdcli_version":"1.0.0","host":"scanhost","pid":4821,
 "started_at":"2026-08-07T16:57:35Z","finished_at":"2026-08-07T17:41:10Z",
 "status":"completed","exit_code":1,
 "run_dir":"/var/lib/mdcli/runs/job-12345",
 "artifacts":{"results":"…/results.ndjson","skipped":"…/skipped.ndjson",
              "manifest":"…/manifest.json"},
 "summary":{"scanned":812304,"detected":12,"errored":3,"skipped":940}}
```

Exit 4 (finished with per-file errors) still counts as `completed` — the run reached the end of its work. Only usage, auth/connection and internal failures (exit 2/3/5) are `failed`.

Retention. `mdcli` never deletes anything at runtime; each run is a self-contained directory, so cleanup is a maintenance job. Use age as the criterion and `status` as a guard against removing a run still in flight:

Code · sh

```
find /var/lib/mdcli/runs -mindepth 1 -maxdepth 1 -type d -mtime +14 -exec rm -rf {} +
```

Tip: `--print-config` reports the resolved artifact paths *before* any scanning starts — a cheap dry-run check for an unattended job.

9.0.2 Diagnostic log

`mdcli` can write an operational log to a file, in **RFC 5424 syslog format** by default, so an existing pipeline (rsyslog `imfile`, Filebeat, Fluentd) can tail and forward it. The log is **independent of stderr**: stderr serves the interactive user or the calling process; the log file serves offline diagnostics.

Option	Default	Description
`--log-file <path>`	—	Explicit log path
`--log-dir <folder>`	—	Write `mdcli.log` in this folder (or the run dir)
`--log-level`	`info`	`debug|info|warn|error`
`--log-facility`	`local0`	Syslog facility
`--log-success-limit`	`100`	Log the first N clean scans individually, then one rollup per N (`0` = log all)

Code

```
<134>1 2026-08-07T19:56:24.551Z scanhost mdcli 295059 run-start [mdcli@0 run_id="job-L"]
version=1.0.0 url=… concurrency=4 fail_on=blocked batch=false
<134>1 2026-08-07T19:56:25.357Z scanhost mdcli 295059 file-clean [mdcli@0 run_id="job-L"]
path="/data/f1.txt" source=scan verdict=Allowed total_ms=804
<131>1 2026-08-07T19:56:26.783Z scanhost mdcli 295059 file-error [mdcli@0 run_id="job-L"]
path="/data/gone.txt" reason=missing error="read: no such file or directory (…)"
<134>1 2026-08-07T19:56:26.783Z scanhost mdcli 295059 progress [mdcli@0 run_id="job-L"] 5 files
processed cleanly, avg 718ms
<132>1 2026-08-07T19:56:28.887Z scanhost mdcli 295059 file-detected [mdcli@0 run_id="job-L"]
path="…/eicar.com.txt" verdict=Blocked scan_result="Infected" data_id=c37b82e7… total_ms=2104
<134>1 2026-08-07T19:56:28.887Z scanhost mdcli 295059 run-summary [mdcli@0 run_id="job-L"]
scanned=14 detected=1 errored=1 skipped=1 wall_ms=4335 files_per_sec=3.23
```

Event IDs (the RFC 5424 MSGID field) are stable and greppable: `run-start`, `run-summary`, `run-failed`, `batch-start`, `batch-close`, `batch-close-failed`, `file-clean`, `file-detected`, `file-error`, `file-skipped`, `progress`.

Volume control. Errors, detections, skips and the batch/run lifecycle are **always** logged. Only clean results are sampled: the first `--log-success-limit` are logged individually, and thereafter one `progress` line summarises each group of that many (a partial group is flushed at the end, so the counts always reconcile). At a million files with the default of 100 that is ~10,000 progress lines instead of a million.

No rotation, no appending. Each invocation writes a *new* file — mdcli is a CLI, not a daemon, so rotation is unnecessary. Re-using an existing log path is an error (exit 2); use `{runid}`/`{timestamp}` placeholders, `--log-dir` inside a per-run directory, or `--force`.

The `run_id` travels in every line's structured data, so forwarded lines stay correlatable with the run's other artifacts.

9.0.1 Performance measurement

Timings are **client-observed wall clock** — they include network and queue wait, which Core's own `processing_time` cannot show. Two independent outputs:

Option	Cost	Content
`--stats-file <path>`	one small object	End-of-run aggregate. Cheap enough for production.
`--timing-file <path>`	one line per file	Per-file phase breakdown. A tuning aid; leave off for production scans.

A run summary is **always** produced: it goes to `--stats-file` as JSON when given, otherwise to stderr as three compact lines.

Code

```
stats: 8 scanned (0 cache, 8 upload, 0 download), 0 detected, 0 errored, 1 skipped
stats: 3.6s wall, 2.20 files/s, 0.42 MB/s, 0 retries, concurrency 4
stats: latency p50 1300ms  p90 1348ms  p95 1348ms  p99 1348ms  max 1983ms
```

Per-file record (`--timing-file`):

Code · json

```
{"input_path":"/data/f3.bin","data_id":"3fc795a6…","source":"scan","bytes":200000,
 "submit_ms":1203,"wait_ms":724,"total_ms":1928,"started_at":"2026-08-07T18:29:52Z"}
```

`hash_ms` local hashing · `lookup_ms` hash-lookup round trip · `submit_ms` upload or `downloadfrom` · `wait_ms` submit verdict `total_ms` end-to-end · `retries` transient retries. Phases that did not occur are omitted, so a `hash-cache` hit shows no `submit_ms`/`wait_ms` — making the dedup saving directly visible. The aggregate's `sources` block does the same at run level.

Notes:

- **The result schema is unchanged.** Timings never appear in the results NDJSON, so existing consumers are unaffected whether or not you enable them.
- Durations are whole **milliseconds**; an operation faster than 1 ms reads `0`.
- Percentiles come from a fixed-bucket histogram, so memory is constant whether the run covers ten files or a million. Values carry two significant digits (≤10% overstatement) and are clamped to the observed maximum; `--timing-file` has the exact per-file numbers when you need them.

9.1 Recipe

Code · sh

```
export MDCORE_URL="https://mdcore.internal:8008"
export MDCORE_APIKEY="…"

# Build the work list (or maintain it however you like).
find /data/to-scan -type f > /var/tmp/scan-list.txt

# Scan, tuned for your Core's capacity, capturing NDJSON + a log.
mdcli --from-file /var/tmp/scan-list.txt \
      --concurrency 24 \
      --fail-on infected \
      --output /var/tmp/scan-results.ndjson \
      2> /var/tmp/scan.log

echo "exit=$?"   # 0 clean, 1 detection, 3 infra, 4 partial errors
```

9.2 Streaming from a generator (no intermediate list file)

Code · sh

```
find /data/to-scan -type f | mdcli --from-file - -j 24 > results.ndjson
```

9.3 Consuming results while the run is in progress

Because NDJSON is flushed per file, you can tail and process live:

Code · sh

```
tail -f results.ndjson | jq -c '{path: .cli.input_path, verdict: .cli.verdict}'
```

9.4 Flow control against Core

- **Concurrency (-j) is the main lever.** It caps how many files are in flight.
- If Core's scan queue fills, it returns **HTTP 503**; `mdcli` automatically backs off per file — exponential backoff with

full jitter, starting at 250 ms, doubling to a 10 s cap, up to 8 retries (honoring `Retry-After` when present). Workers block during backoff, which naturally reduces pressure.

- Start conservative (`-j 8`–`16`), watch throughput and the `errored` count, and raise gradually. Too high just produces more 503s and backoff, not more speed — the ceiling is Core's engine/queue capacity, not the client.

9.5 Batch mode

With `--batch` (list and directory modes), `mdcli` opens a single MD Core **batch** before submitting the first file, associates every submission with it, and closes the batch when the run finishes — so the job is one logical unit on Core (for batch-level reporting/auditing). Each submitted file gets its own `data_id`; the batch groups them. (Files resolved by dedup aren't submitted — see below.)

Code · sh

```
mdcli --from-file /var/tmp/scan-list.txt \
      --batch --batch-name "nightly-$(date +%F)" \
      --output results.ndjson
# stderr:  batch started: <batch_id> (name "nightly-2026-07-16")
#          batch closed: <batch_id>
#          done: N scanned, N detected, N errored, N skipped
```

- `--batch-name` is sent as the batch's `user_data` and implies `--batch`.
- The opened `batch_id` is announced on **stderr at the start** (`batch started: …`) and included in the output record of every **submitted** file (`cli.batch_id`). stdout stays pure JSON, so a downstream `jq` consumer is never confused by the id — capture it from stderr.
- **Dedup interacts with batches.** Hash lookup (dedup) still applies in batch mode: a cache hit is served from Core's hash store — its result is retrieved and reported (as `source: hash-cache`) — but it is **not re-submitted**, so it is not a member of the batch and carries no `batch_id`. In practice a real batch is rarely fully cached, so most files are submitted and appear in the batch. To force **every** file into the batch (e.g. a test where all inputs are already cached), add `--no-hash-lookup`.
- The batch is closed with a fresh 30 s timeout even if the run is cancelled (e.g. `--stop-on-detection`), so batches are not left open on Core.
- Without `--batch`, files are submitted independently (no batch) — the default.

Monitoring progress concurrently. Grab the `batch_id` from the start line on stderr and poll it from another process with the `batch-status` subcommand:

Code · sh

```
mdcli batch-status <batch_id> --watch --watch-interval 2s
# batch <id>  finished 12 / 15 submitted  result=Processing  [open]
# batch <id>  finished 15 / 15 submitted  result=Allowed      [closed]
```

Denominator note: a standalone monitor knows only what Core reports — **files finished over files added so far** — because the eventual total isn't fixed until every file is submitted (e.g. when the list itself is streamed). The count is

firm once the batch shows `[closed]`.

9.6 Automation patterns

- **CI/pipeline gate:** run to completion, let the exit code fail the job (`--fail-on infected` to gate on real threats only; default `blocked` to gate on any Core block, including DLP/CDR/type policy).
- **Fast fail:** `--stop-on-detection` aborts the run on the first detection. Works with `--batch`: the batch is still closed on Core (best-effort, on a fresh timeout), and files that were in flight when the run aborted are dropped from the output rather than reported as errors — the client gets the first detection and stops. A few of those in-flight files may already have reached Core and remain in the batch there; that's expected and bounded by concurrency (and Core's queue).
- **Report-only:** `--fail-on never` always exits 0 (still emits full results) so a reporting job never fails the pipeline.

---

10. Performance & tuning

Concern	Behavior / guidance
Memory	O(1) in list size. Roughly `concurrency × (response size + buffers)`. Per-response body is capped at 64 MB.
CPU	Low; the workload is network- and Core-bound. Local hashing (for the lookup) reads each file once.
Upload	File bodies are streamed, not buffered — large files do not inflate memory.
Hash lookup	Off by default. Opt in with `--hash-lookup` to calculate a client hash and reuse existing Core results. Core-local mode never hashes on the client.
In-progress hits	If a hash lookup finds an analysis still running, `mdcli` polls that existing analysis rather than re-uploading.
`--client-timeout`	Raise for very large uploads over slow links.
`--max-poll-wait`	Raise if Core legitimately takes long (large archives, deep CDR); lower to fail slow files faster.
File descriptors (Linux)	High `-j` opens more sockets/files at once. Ensure `ulimit -n` is adequate (e.g. `ulimit -n 4096`).

11. Security

- **API key:** prefer `MDCORE_APIKEY` over `--apikey`. A flag value is visible in shell history and in process listings (`ps`, Task Manager); the env var is not. The key is never written to stdout/stderr by `mdcli`.
- **TLS:** verification is **on by default**. `--tls-skip-verify` disables it and should be limited to development/testing — it exposes the connection to MITM.
- **Least privilege:** `mdcli` only needs a scan-scoped API key and read access to the target files. It uses no administrative MD Core endpoints.
- **Untrusted content:** the CLI streams bytes to Core and never executes or parses file contents locally, keeping local attack surface minimal.

12. Troubleshooting

Symptom	Likely cause / fix
Exit 3, "connection refused"	Wrong `--url`/port, Core down, or firewall. Verify `curl <url>/version`.
Exit 3, HTTP 401/403	Bad or missing API key. Check `MDCORE_APIKEY`.
Exit 1 but no malware	A non-threat block (e.g. `Sanitization Timed Out`, DLP, blocked type). Use `--fail-on infected` to gate on threats only; inspect `result.process_info.blocked_reason`.
Many `errored`, HTTP 503	Core queue saturated. Lower `-j`; backoff is already automatic.
Files stuck / slow	Core-side scanning slow (e.g. sandbox/CDR). Check Core module health; adjust `--max-poll-wait`.
TLS "certificate signed by unknown authority" (Linux)	Install `ca-certificates`, or add your private CA to the trust store. `--tls-skip-verify` for test only.
`too many open files` (Linux)	Raise `ulimit -n`, or lower `-j`.
Result shows `source: hash-cache` but you expected a fresh scan	Core returned a prior result for that hash. Use `--no-hash-lookup` to force re-upload. Note Core may flag stale results with `outdated_data`.

`mdcli` writes diagnostics and a `done: N scanned, N detected, N errored, N skipped` summary to stderr; redirect it (`2> run.log`) to keep an audit trail.

---

13. Known limitations

Current version. Planned items are on the project roadmap:

- **Operational conditions count as blocks.** With the default `--fail-on blocked`, Core results like *Sanitization Timed Out* or *Failed to Scan* trip exit 1 even though no threat was found. Use `--fail-on infected`/`suspicious` to gate on real verdicts. A dedicated result-code exit-condition map is planned.
- **No resume/checkpoint.** An interrupted large run restarts from the beginning of the list (the hash cache makes re-runs cheap for already-known files).
- **No archive child/blocked-leaves reporting** yet (the full `result` still contains Core's data; there is no CLI-level per-leaf summary).
- **Output formats:** JSON/NDJSON only (no table/CSV yet).
- **Transport:** HTTP/2 is negotiated automatically over HTTPS; plaintext HTTP is HTTP/1.1 (h2c is not used).

---

14. Appendix: verdicts & result codes

`--fail-on` decides what counts as a "detection" for the exit code and for `--stop-on-detection`:

`--fail-on`	Trips when
`blocked` (default)	`process_info.result == "Blocked"` — any reason (malware, DLP, CDR, blocked type, timeouts)
`infected`	`scan_results.scan_all_result_i == 1` (Infected) — a real AV/threat detection
`suspicious`	`scan_all_result_i` is `1` (Infected) or `2` (Suspicious)
`never`	Nothing trips; always exit 0 (report-only)

Key MD Core `scan_all_result_i` values you may see in `cli.scan_result` / `result.scan_results.scan_all_result_a`:

Code	Meaning
0	No Threat Detected
1	Infected

Code	Meaning
2	Suspicious
3	Failed to Scan
10	Not Scanned / engine unavailable
…	Other codes exist (e.g. timeouts); the full `result` block preserves them verbatim

Note

The result-code list above covers the common cases; MD Core defines additional codes. The complete, unmodified result is always available under `result` for downstream consumers.

Release Notes

1.0.0

First public release. `mdcli` is available as a free utility from My OPSWAT, for Linux x64, Linux ARM64 and Windows x64.

- Validated with MetaDefender Core 5.20.0 and later versions.

See the user guide for installation and the full command reference.

API Reference

MetaDefender Core API

API Version: v5.24.0

Developer Guide

This is the API documentation for *MetaDefender Core Public API*. If you would like to evaluate or have any questions about this documentation, please contact us via our Contact Us form.

How to Interact with MetaDefender Core using REST

Beginning with MetaDefender Core 4.x, OPSWAT recommends using the JSON-based REST API. The available methods are documented below.

Note: MetaDefender API doesn't support chunk upload, however is recommended to stream the files to MetaDefender Core as part of the upload process.

File Analysis Process

MetaDefender's main functionality is to analyze large volumes with a high throughput. Depending on the configuration and licensed technologies, the analysis times can vary. Below is a brief description of the API integration flow:

1. Upload a file for analysis (POST /file), which returns the data_id or batch_id: File Analysis).

Note: The performance depends on:

- number of engines
- type of file to be scanned
- MetaDefender Core's hardware

Batch Upload with multipart/form-data:

- When the request uses Content-Type: multipart/form-data, MetaDefender Core treats the entire submission as a batch.
- A new batch is automatically created for this media type.
- Each part in the request body is processed as an individual file within the batch.
- The API returns a batch_id instead of a data_id.
- All individual file scans are linked to the created batch.

2. You have 2 ways to retrieve the analysis report:

- **Polling**: Fetch the result with previously received data_id or batch_id (GET /file/{data_id} or GET /file/{batch_id} resource) until scan result belonging to data_id/batch_id doesn't reach the 100 percent progress_percentage: (Fetch analysis result)

Note: Too many data_id/batch_id requests can reduce performance. It is enough to just check every few hundred milliseconds.

- **Callbackurl**: Specify a callbackurl that will be called once the analysis is complete. Up to 10 callbackheader-name / callbackheader-value pairs (unsuffixed, then 1–10) are copied onto that webhook (POST /file binary upload, and POST /file/batch/{batchId}/close/ callback).

3. Retrieve the analysis results anytime after the analysis is completed with hash for files (md5, sha1, sha256, sha512) by calling Fetch analysis result by hash.

   - The hash can be found in the scan results

4. List data_id values for processed objects without fetching full scan payloads (GET /file/ids).

   - IDs are returned regardless of status, result, or verdict, subject to processing-history permission.
   - Use the listed IDs with GET /file/{data_id} or GET /file/batch/{data_id} as appropriate.

5. Retrieve processed file (sanitized, redacted, watermarked, etc.) after the analysis is complete.

   Note: Based on the configured retention policy, the files might be available for retrieval at a later time.

OPSWAT provides some sample codes on GitHub to make it easier to understand how the

MetaDefender REST API works.

CONTACT

NAME: API Support
EMAIL: feedback@opswat.com
URL: https://github.com/OPSWAT/metadefender-core-openapi3
Terms of service: https://onlinehelp.opswat.com/policies/

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
apikey	apiKey	Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

API

1. USER MANAGEMENT

User management APIs

The APIs for manage users and user directories.

`1.1 GET /admin/user`

List all users

Returns a list of all users in the server.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of users retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
api_key	string	Associated apikey with this user
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address
id	integer	User's unique identifier

NAME	TYPE	DESCRIPTION
name	`string`	User's full name
description	`string`	User's description, 256 characters maximum
roles	`array`	
ui_settings	`object`	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.2 POST /admin/user

Create user

Creates a new user in the specified user directory.

For **Local** directories, set password_change_on_first_login to true to require the new user to change their password on the next successful sign-in. Omit the field or set it to false to skip that requirement.

To have MetaDefender Core generate the user's API key, send the generate_apikey: true request header and omit api_key from the body. The generated key is returned once in the response and can authenticate subsequent API calls; it is also returned in plaintext this one time when API key hashing is enabled, while only its hash is stored.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
api_key	string	Associated apikey with this user > _**Note**:_ Omit this field (or send an empty value) when the request carries the `generate_apikey: true` header.
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address

NAME	TYPE	DESCRIPTION
name	`string`	User's full name
description	`string`	User's description, 256 characters maximum
roles	`array`	
ui_settings	`object`	
password	`string`	The user's password
password_change_on_first_login	`boolean`	Optional. When `true`, the new user must change their password on the next successful sign-in. Applies only when `directory_id` refers to a **Local** user directory; omit the field or set it to `false` to create the user without this requirement (default behavior). > _**Note**:_ If `true` is sent for a non-local directory (for example Active Directory or LDAP), the request is rejected with HTTP 400.

EXAMPLE 1:

```
"localUserRequirePasswordChange"
```

EXAMPLE 2:

```
"localUserNoPasswordChange"
```

EXAMPLE 3:

```
"localUserGeneratedApiKey"
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
`generate_apikey`	`string`		(Optional) Set to `true` to let MetaDefender Core generate the new user's API key. Applies to users in a **Local** user directory. Accepted values are `true` and `false`, case-insensitive.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
api_key	`string`	Associated apikey with this user

NAME	TYPE	DESCRIPTION
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address
name	string	User's full name
description	string	User's description, 256 characters maximum
roles	array	
ui_settings	object	

EXAMPLE 1:

```
"userCreated"
```

EXAMPLE 2:

```
"userCreatedWithGeneratedApiKey"
```

STATUS CODE - 400: Bad Request (for example invalid header, missing fields, or an unsupported `password_change_on_first_login` value for the selected directory).

Condition	`err`
`password_change_on_first_login` is `true` for a non-local directory	`Requiring a password change on first login is only supported for local users`
`generate_apikey` is present but is neither `true` nor `false` (any casing), including an empty value	`Invalid value for the generate_apikey header; expected 'true' or 'false'`
`generate_apikey` is `true` and a non-empty `api_key` is also sent in the body. An empty `api_key` counts as not provided, so the key is still generated	`Cannot use the generate_apikey header together with an explicit api_key`

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE 1:

```
"passwordChangeNotSupportedForDirectory"
```

EXAMPLE 2:

```
"invalidGenerateApiKeyHeader"
```

EXAMPLE 3:

```
"generateApiKeyConflictsWithApiKey"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server, for example MetaDefender Core could not generate a valid API key for a `generate_apikey: true` request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE 1:

```
"apiKeyGenerationFailed"
```

1.3 DELETE /admin/user/{user_id}

Delete a user

Delete a user by id from the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "result": "Success"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Item does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not allowed"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.4 PUT /admin/user/{user_id}/apikey

Change user's API key

Updates the API key for the specified user. This operation is only available to local users. Only admin users can change the API key of other users.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*user_id	string		The ID of the user whose API key is to be changed

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
api_key*	string	New API key for the user

EXAMPLE:

```
{
 "api_key": "1c717a2f9cb73b62b15bc200c647bd64cc8a"
}
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: API key changed successfully

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "result": "Successfully"
}
```

STATUS CODE - 400: Bad request (e.g invalid format of api_key, header apikey is missing or invalid, duplicated api_key)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error message

EXAMPLE:

```
{
```

```
 "err": "Field is invalid"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: User not found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error message indicating the user doesn't exist

EXAMPLE:

```
{
 "err": "User does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

1.5 GET /admin/exportusers

Export all users

Export all users in the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of users retrieved successfully in CSV format with columns for Directory Name, User, Display Name, Role, Email, Last Session (UTC Time), and IP.

RESPONSE MODEL - text/csv

EXAMPLE:

```
"\"Directory Name\",\"User\",\"Display Name\",\"Role\",\"Email\",\"Last Session (UTC Time)\",
\"IP\"\n\"LOCAL\",\"admin\",\"admin\",\"Administrators\",\"admin@local\",
\"2025-03-24T08:22:06.942Z\",\"::1\"\n\"LOCAL\",\"1\",\"1\",\"Security administrators\",
\"1@1\",\"\",\"\"\n"
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

`1.6 PUT /admin/userdirectory/changepassword`

Change Password for AD/LDAP User Directory

Change the current password for AD/LDAP user directory.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
name	`string`	The user directory name in plain text
password	`string`	The new password in plain text

EXAMPLE:

```
{
 "name": "AD Department X",
 "password": "newpassword123"
}
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	boolean	

STATUS CODE - 400: Bad request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: AD/LDAP directory does not exist, Connection towards remote user directory has failed, Failed to modify password, Missing data, Too many elements in list, Field is not allowed err: * AD/LDAP directory does not exist * Connection towards remote user directory has failed * Failed to modify password * Missing data * Too many elements in list * Field is not allowed

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

1.7 POST /user/changepassword

Change Password for local user

Modify the current password for the user identified by apikey

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
old_password	`string`	The current password in plain text
new_password	`string`	The new password in plain text

EXAMPLE:

```
{
 "old_password": "admin",
 "new_password": "123456"
}
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	boolean	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2. YARA

YARA engine configuration and source management APIs.

2.1 GET /admin/config/yara/sources

Get YARA sources

Retrieve a list of all sources for the YARA Engine.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of YARA Engine sources.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
local_sources	array	
source	string	The location (file path) of the files.
state	enum	ALLOWED: disabled, enabled Defining if this source is being used or not
http_sources	array	
source	string	The location (URL) of the package.
state	enum	ALLOWED: disabled, enabled Defining if this source is being used or not

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2.2 PUT /admin/config/yara/sources

Modify YARA sources

Modify one (or more) of the sources for the YARA Engine.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
local_sources	array	
source	string	The location (file path) of the files.
state	enum	ALLOWED: disabled, enabled Defining if this source is being used or not
http_sources	array	
source	string	The location (URL) of the package.
state	enum	ALLOWED: disabled, enabled

NAME	TYPE	DESCRIPTION
		Defining if this source is being used or not

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of YARA Engine sources.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
local_sources	array	
source	string	The location (file path) of the files.
state	enum	ALLOWED: disabled, enabled Defining if this source is being used or not
http_sources	array	
source	string	The location (URL) of the package.
state	enum	ALLOWED: disabled, enabled Defining if this source is being used or not

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2.3 GET /yara/package

YARA Generation Status

Get status of YARA package generation.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
status	enum	**ALLOWED:** idle, error, inprogress Current status of generating the package.
start_time	string	Used only when status is inprogress, otherwise its empty
issues	object	
general	array	
severity	string	
message	string	
source	array	

NAME	TYPE	DESCRIPTION
severity	`string`	
message	`string`	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

2.4 POST /yara/generate

Generate YARA package

Generate a new package based on the new added sources.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
status	enum	ALLOWED: `idle, error, inprogress` Current status of generating the package.
start_time	string	Used only when status is inprogress, otherwise its empty
issues	object	
general	array	
severity	string	
message	string	
source	array	
severity	string	
message	string	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

---

3. ADMIN

Admin specific API requests.

3.1 GET /admin/export

Export configuration

Export configuration from file.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Failed to export configuration

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Failed to export configuration"
}
```

3.2 GET /admin/export/v2

Export configuration (new version)

Export configuration from file (new version)

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
*password	string		Password to encrypt all the settings after exporting from MetaDefender Core. Only applicable when importing ZIP file.
setting	object		JSON object to control which settings to export. If not provided, all settings will be exported by default.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/octet-stream

STATUS CODE - 400: Bad request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Unable to export new configuration"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.3 POST /admin/import

Import configuration

Import configuration from file.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	`string`	

STATUS CODE - 304: The configuration has not changed

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
```

```
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.4 POST /admin/import/v2

Import configuration (new version)

Import configuration from file (new version). Please note that if the imported configuration has a different log file path compared to the current one, the **MetaDefender Core will be restarted** after importing to take effect.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
`password`	`string`		Password to decrypt all the settings before importing. Only applicable when importing ZIP file.
`import_force_update_yara`	`boolean`		To force updating and overriding YARA settings.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 304: The configuration has not changed

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad request

RESPONSE MODEL - application/json

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
```

```
}
```

3.5 GET /admin/role

List all roles

Retrieve a list of all user roles configured in the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
agents	array	
cert	array	
configlog	array	
engines	array	
external	array	
license	array	
logger	array	
quarantine	array	
retention	array	
rule	array	

NAME	TYPE	DESCRIPTION
scan	array	
scanlog	array	
skip	array	
update	array	
updatelog	array	
users	array	
workflow	array	
zone	array	
editable*	boolean	If the role can be altered or not.
id*	integer	Internal used identifier

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.6 POST /admin/role

Create new role

Add a new user role to the system.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
agents	array	
cert	array	
configlog	array	
engines	array	
external	array	
license	array	
logger	array	
quarantine	array	
retention	array	
rule	array	
scan	array	
scanlog	array	
skip	array	
update	array	
updatelog	array	
users	array	
workflow	array	

NAME	TYPE	DESCRIPTION
zone	array	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
agents	array	
cert	array	
configlog	array	
engines	array	
external	array	
license	array	
logger	array	
quarantine	array	
retention	array	
rule	array	
scan	array	
scanlog	array	
skip	array	
update	array	
updatelog	array	
users	array	
workflow	array	
zone	array	

NAME	TYPE	DESCRIPTION
editable*	`boolean`	If the role can be altered or not.
id*	`integer`	Internal used identifier

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.7 DELETE /admin/role/{role_id}

Delete a role

Delete a role by id from the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "result": "Success"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Item does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not allowed"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.8 GET /admin/userdirectory

List all user directories

Retrieve a list of all user directories configured in the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of user directories retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
id	integer	Internal used identifier
name	string	Name of the user directory
type	string	Type of the user directory (e.g., LDAP, Local, etc.)
enabled	boolean	If the user directory is enabled or not
lockout_attempts	integer	Number of failed login attempts before the user is locked out
lockout_timeout	integer	Time in seconds before the user can try to log in again after being locked out

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.9 GET /admin/license/backup

Retrieve hot-backup instance

This API to set a MetaDefender Core as a backup instance.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
is_backup	boolean	The instance is a backup or not.

EXAMPLE:

```
{
 "is_backup": true
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
```

```
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.10 POST /admin/license/backup

Setup hot-backup instance

This API to set a MetaDefender Core as a backup instance.

- Online environment: you need to input activation-key and active-deployment-id headers
- Offline environment
  :
  - Obtain license.yml file that supposed to use to active your MetaDefender Core active / production instance before.
  - Use that yaml file content in body content of the REST API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
active-deployment-id	string		Deployment ID of the main MetaDefender Core which is currently working, and you want to backup it.
activation-key	string		Activation key

RESPONSE

STATUS CODE - 200: Request successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	string	Successful message

EXAMPLE:

```
{
 "success": "The backup MetaDefender Core instance is now registered as a backup one."
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.11 DELETE /admin/license/backup

Remove a backup instance

This API to remove a backup instance.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	string	Successful message

EXAMPLE:

```
{
 "success": "This MetaDefender Core instance is no longer registered as a backup one."
}
```

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Failed to remove backup license"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

3.12 POST /admin/license/backup/nomination

Nominate a backup instance to a production instance

This API to nominate a backup instance to a production instance.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	string	Successful message

EXAMPLE:

```
{
 "success": "This MetaDefender Core instance is now fully activated and allowed to process files."
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4. ANALYSIS

File analysis APIs

Submit each file to MetaDefender Core individually or group them in batches. Each file submission will return a data_id which will be the unique identifier used to retrieve the analysis results. When Content-Type: multipart/form-data is used, the entire request is treated as a batch. A batch_id is created automatically, and each part in the multipart body is processed as an individual file within that batch. **Important**: Even though one file is being submitted, if MetaDefender Core is configured to extract the files, all compound file types (archives, Office documents, etc.) will be extracted and each file within will be analyzed as a separate entry.

- This means that if you submit an archive with 100 files in it, MetaDefender Core will process 101 files: original file as it is and each of the 100 child files
  - Note that by opening the files, detection ratio can increase even by 30%.

Note*:* MetaDefender API doesn't support chunk upload. You shouldn't load the file in memory, is recommended to stream the files to MetaDefender Core as part of the upload process.

`4.1 POST /file`

Analyze File (Asynchronous mode)

Scanning a file using a specified workflow: Scan is done asynchronously and each scan request is tracked by data id or batch id, of which result can be retrieved by API Fetch Scan Result. **Supported Content Types:** Only application/octet-stream and multipart/form-data content types are supported. Requests using multipart/form-data must conform to RFC 7578; otherwise, a 400 HTTP(S) error will be returned. If the Content-Type header is not multipart/form-data, the request body will be treated as application/octet-stream. **Batch processing with multipart/form-data:** When Content-Type: multipart/form-data is used, the request is treated as a batch, each part in the request body is processed as an individual file. The response returns a batch_id instead of a data_id. If the filename is missing from a part, it defaults to

part_<part_index>_<batch_id> (where part_index starts from 1).

Note: Chunked transfer encoding (applying header Transfer-Encoding: Chunked) is **not supported** on /file API. **Callback custom headers:** With callbackurl and a valid apikey, up to 10 callbackheader-name / callbackheader-value pairs are copied onto the webhook POST (first pair unsuffixed, then 1–10). Binary upload only (not multipart/form-data). Not supported in cluster mode. Not included if the webhook is resent after a Core restart.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
filename	string		The name of the submitted file
filepath	string		Not supported with `multipart/form-data` content type. If local file scan is enabled the path to the file (see Security rule configuration).
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used. Since MetaDefender Core 5.6.0, when user agent is matched the predefined setting in workflow rule, the product will return X-Core-Id containing identity of MetaDefender Core server as a header of response.
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
workflow	string		Name of the selected workflow (deprecated, use "rule" header parameter instead)
batch	string		Not supported with `multipart/form-data` content type. Batch id to scan with, coming from `Initiate Batch` (If it is not given, it will be a single file scan).

NAME	TYPE	EXAMPLE	DESCRIPTION
archivepwd	string		Password for archive (URL encoded UTF-8 string) Multiple passwords is also supported, format: archivepwdX * X: Could be empty * When having value, X must be a number >= 1 For example: * archivepwd1: "fox" * archivepwd2: "cow" * archivepwd3: "bear"
content-encoding	string		Content encoding of the file. This header is used to specify the encoding of the file content. The value should be a valid content encoding type, such as "base64". This header is optional and can be omitted if the encoding is not applicable.
metadata	json		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters. The `source_system` key is reserved for the system that submitted the file; it can be used to filter `Active performance`. When present, its value must be a string of 1 to 128 letters, digits, dots, underscores, colons or hyphens, otherwise the request is rejected with `400`.

NAME	TYPE	EXAMPLE	DESCRIPTION
`engines-metadata`	`json`		Since MetaDefender Core 5.0.0, preferred context / verbose information can be sent to the engines. Please see the below pages for the details: * [File Type engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata) (supported since Core 5.2.1) * [Archive engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata-header) (supported since Core 5.4.1) * [Deep CDR](https://docs.opswat.com/mdcore/deep-cdr/supported-engines-metadata-json) * [Proactive DLP](https://docs.opswat.com/mdcore/proactive-dlp/supported-engines-metadata-json)
`callbackurl`	`uri`		Client's URL where MetaDefender Core will notify scan result back to whenever scan is finished (webhooks model). * Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> * Example: http://10.0.1.100:8081/listenback * Supported protocol: HTTP / HTTPS * Method: POST
`callbackheader-name`	`string` max:128 chars		Header name Core attaches to the webhook POST (with `callbackurl`). Pair with `callbackheader-value`. Extra pairs: `callbackheader-name1` … `callbackheader-name10` with matching `callbackheader-valueN`. At most **10** pairs. Requires a valid `apikey`. Name: RFC 7230 token, max **128** characters. Duplicate names: highest index wins. Suffixes outside `1`–`10` (for example `01`, `11`) are ignored. Reserved (case-insensitive): `host`, `content-length`, `content-type`, `transfer-encoding`, `connection`, `user-agent`, `accept-charset`, `accept-language`, `digital_signature`, `x-core-id`.
`callbackheader-value`	`string` max:4096 chars		Value for `callbackheader-name` (or `callbackheader-valueN` for numbered pairs). Max **4096** characters. Empty is allowed. `CR`, `LF`, and `NUL` are rejected.

NAME	TYPE	EXAMPLE	DESCRIPTION
sanitizedurl	uri		Not supported with `multipart/form-data` content type. Client's URL where MetaDefender Core will send the sanitized file as the body of a POST request back to whenever scan is finished (webhooks model). * Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> * Example: http://10.0.1.100:8081/listensanitizedfile * Supported protocol: HTTP / HTTPS * Method: POST * Mandatory headers: * `user-agent`: "OPSWAT MetaDefender Core V4" * `dataid`: "<data id of original file submission>" * `sanitization-result`: "Sanitized Successfully" or "Sanitized Partially" or "Sanitized File Removed" * `sanitized-file-hash`: "<SHA256 hash value of sanitized file>" * `Content-Disposition`: attachment; filename="<name of sanitized file>" * `Content-Length`: <actual size of sanitized file>

NAME	TYPE	EXAMPLE	DESCRIPTION
downloadfrom	uri		Not supported with `multipart/form-data` content type. Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> Since MetaDefender Core 4.19.1, the product started supporting users to process file by just specifying a direct download link. When this header is specified, users must not have body content in HTTP(S) request, otherwise expecting to hit 400 HTTP(S) error. Check response sample of HTTP 400 for details. Specify download link where MetaDefender Core could download the entire payload for processing. * Supported protocol: HTTP / HTTPS * Support both * Individual file scan (both webhook & non-webhook) * File processing in batch via link * Simply use `downloadfrom` header for each file scan request in a batch, no further requirement on batch request itself to utilize scan via link feature. * `Note`: Cancelling a batch will result in auto cancelling all registered individual scan requests which are still in downloading status * Configurable setting under workflow rule (under "SCAN" tab): * Download timeout * Max file size to download * Support to return status of download progress back to client in HTTP(S) response. * Pre-check supported to refuse downloading if the file size does not meet configured conditions, and/or MetaDefender Core's temp folder does not have sufficient disk space to save the downloaded file Limitation: * Redirect link is not supported * Retry to download is not supported
skip-head-request	boolean		Skip HEAD request before downloading file when using `downloadfrom` header. Set this to `true` to skip the HEAD request and download the file directly. * Default: `false` (HEAD request is performed) * `true`: Skip HEAD request, download file directly * `false`: Perform HEAD request before download (default behavior)
global-timeout	integer		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

NAME	TYPE	EXAMPLE	DESCRIPTION

client-identity	json		The `client-identity` helps track and understand the workflow that a file passes through clients, offering greater visibility into the processing stages and the clients involved.
			This header is provided as a JSON object containing an array of client identity objects. Each object in the array should include the following fields:
			* `deployment_id`: *REQUIRED* | Unique identifier for the deployment.
			* `user_name`: *REQUIRED* | Name of the user or system that initiated the scan.
			* `product_type`: *REQUIRED* | Type of product or system that is performing the scan.
			* `version`: *OPTIONAL* | Version of the product or system.
			* `host`: *OPTIONAL* | Hostname or IP address of the client machine.
			* `timestamp`: *REQUIRED* | Timestamp of when the scan was initiated, in milliseconds since epoch.
			Example:
			``json
			{
			"clients":
			[
			{
			"deployment_id": "abcdef1234567890",
			"user_name": "Gemini",
			"product_type": "MetaDefender Kiosk",
			"version": "1.0.0",
			"host": "client.example.com",
			"timestamp": 1744949754890
			}
			]
			}
			``
			It should be performed the *URL-Encoding* before sending to MetaDefender Core to prevent unexpected issues related to encoding errors or unsafe characters.
			This header is optional and can be omitted if not needed.

RESPONSE

STATUS CODE - 200: Successful file submission

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
ONE:OF	object	
OPTION:1	object	
data_id*	string	Unique submission identifier for single file upload. Use this value to reference or query the status/results of the submission.
OPTION:2	object	
batch_id*	string	Unique identifier for a batch upload (multipart/form-data). Use this value to reference and query the status or results of the submission.

EXAMPLE 1:

```
"singleFile"
```

EXAMPLE 2:

```
"formDataUpload"
```

STATUS CODE - 400: Invalid `callbackurl`, `apikey`, download link, callback header pair or `metadata` header.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Both body data and download link were given.
OPTION:2	object	
err	string	Callback url is invalid.
OPTION:3	object	
err	string	File upload rejected due to insufficient disk space
OPTION:4	object	
err	string	Callback header validation failed. `<N>` is the pair suffix (empty for the first pair, `1`–`10` for numbered pairs). * `callbackheader-name and callbackheader-value can only be used with a valid apikey` * `Too many callbackheader-name / callbackheader-value pairs, at most 10 are allowed` * `Both callbackheader-name<N> and callbackheader-value<N> have to be sent together` * `Header name in callbackheader-name<N> is not a valid HTTP header name` * `Header name in callbackheader-name<N> is reserved and cannot be overridden` * `Header value in callbackheader-value<N> contains an invalid character` * `Header name or value in callbackheader-name<N> / callbackheader-value<N> is too long`
OPTION:5	object	
err	string	The `source_system` value in the `metadata` header is invalid.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 411: Content-Length header is missing from the request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Missing Content-Length header."
}
```

STATUS CODE - 422: Body input is empty.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "File is empty."
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

STATUS CODE - 503: Server is too busy, scan queue is full. Try again later.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Server is too busy. Try again later."
}
```

4.2 POST /file/sync

Analyze File (Synchronous mode)

Note: Only available starting with MetaDefender Core 4.21.0

Synchronous (sync) scan mode does not support:

- Non-persistent scan mode (dbmode = 3 described in MetaDefender Core Configuration)
- Individual scan with sync mode linked to a batch closed with a callbackurl header (webhook)

Scanning a file using a specified workflow.

Scan is done **synchronously (blocking mode)**, the connection of file submission request must be held by client side until MetaDefender Core returns processing result of individual scan when finished (response code = 200), or return data ID associated to that scan request if timed out (response code = 408).

Note: In case of receiving data ID, expecting the client side to perform query polling for scan result / progress then, see more at Fetch Analysis Result

The timeout value could be configurable, see more at Synchronous scan connection timeout - Modification

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
filename	string		Specify name of file captured and displayed on corresponding MetaDefender Core scan result
filepath	string		If local file scan is enabled the path to the file (see Security rule configuration)
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used. Since MetaDefender Core 5.6.0, when user agent is matched the predefined setting in workflow rule, the product will return X-Core-Id containing identity of MetaDefender Core server as a header of response.
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
workflow	string		Name of the selected workflow (deprecated, use "rule" header parameter instead)
batch	string		Batch id to scan with, coming from `Initiate Batch` (If it is not given, it will be a single file scan.)
archivepwd	string		Password for archive (URL encoded UTF-8 string) Multiple passwords is also supported, format: archivepwd<X> * X: Could be empty * When having value, X must be a number >= 1 For example: * archivepwd1: "fox" * archivepwd2: "cow" * archivepwd3: "bear"
content-encoding	string		Content encoding of the file. This header is used to specify the encoding of the file content. The value should be a valid content encoding type, such as "base64". This header is optional and can be omitted if the encoding is not applicable.

NAME	TYPE	EXAMPLE	DESCRIPTION
`metadata`	`json`		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters. The `source_system` key is reserved for the system that submitted the file; it can be used to filter `Active performance`. When present, its value must be a string of 1 to 128 letters, digits, dots, underscores, colons or hyphens, otherwise the request is rejected with `400`.
`engines-metadata`	`json`		Since MetaDefender Core 5.0.0, preferred context / verbose information can be sent to the engines. Please see the below pages for the details: * [File Type engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata) (supported since Core 5.2.1) * [Archive engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata-header) (supported since Core 5.4.1) * [Deep CDR](https://docs.opswat.com/mdcore/deep-cdr/supported-engines-metadata-json) * [Proactive DLP](https://docs.opswat.com/mdcore/proactive-dlp/supported-engines-metadata-json)

NAME	TYPE	EXAMPLE	DESCRIPTION
downloadfrom	uri		Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> When this header is specified, users must not have body content in HTTP(S) request, otherwise expecting to hit 400 HTTP(S) error. Check response sample of HTTP 400 for details. Specify download link where MetaDefender Core could download the entire payload for processing. * Supported protocol: HTTP / HTTPS * Support both individual file scan (both webhook & non-webhook) and file processing in batch via link * Configurable setting under workflow rule (under "SCAN" tab): * Download timeout * Max file size to download * Support to return status of download progress back to client in HTTP(S) response * Pre-check supported to refuse downloading if the file size does not meet configured conditions, and/or MetaDefender Core's temp folder does not have sufficient disk space to save the downloaded file Limitation: * Redirect link is not supported * Retry to download is not supported

NAME	TYPE	EXAMPLE	DESCRIPTION

client-identity	json		The `client-identity` helps track and understand the workflow that a file passes through clients, offering greater visibility into the processing stages and the clients involved.
			This header is provided as a JSON object containing an array of client identity objects. Each object in the array should include the following fields:
			* `deployment_id`: *REQUIRED* | Unique identifier for the deployment.
			* `user_name`: *REQUIRED* | Name of the user or system that initiated the scan.
			* `product_type`: *REQUIRED* | Type of product or system that is performing the scan.
			* `version`: *OPTIONAL* | Version of the product or system.
			* `host`: *OPTIONAL* | Hostname or IP address of the client machine.
			* `timestamp`: *REQUIRED* | Timestamp of when the scan was initiated, in milliseconds since epoch.
			Example:
			``json
			{
			"clients":
			[
			{
			"deployment_id": "abcdef1234567890",
			"user_name": "Gemini",
			"product_type": "MetaDefender Kiosk",
			"version": "1.0.0",
			"host": "client.example.com",
			"timestamp": 1744949754890
			}
			]
			}
			``
			It should be performed the *URL-Encoding* before sending to MetaDefender Core to prevent unexpected issues related to encoding errors or unsafe characters.
			This header is optional and can be omitted if not needed.

RESPONSE

STATUS CODE - 200: The scan is completed before the connection timeout occurs

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
aicontentinspector_info	object	
final_verdict*	object	

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `Not Detected, Detected, Uncertain` AI Content Inspector engine verdict. Values: - Detected: AI-generated or manipulated content was detected. - Not Detected: No AI-generated or manipulated content was found. - Uncertain: Content could not be confidently classified.
verdict_explanation	array	
output_images	array	
image_alias	string	Alias identifier for the output image, used to retrieve the image via the API.
result_description*	object	
result_template_hash*	string	Hash of the result template used internally for AI Content Inspector rendering and coordination.
coo_info	object	
final_verdict*	object	
verdict	enum	**ALLOWED:** `blocked, allowed, unknown` Country of Origin engine verdict. Values: - blocked: The file is considered blocked based on origin evaluation. - allowed: The file origin passed all policy checks. - unknown: Origin could not be confidently determined.
verdict_explanation	array	
full_report	object	
overview_report*	object	
company_name	string	Name of the company that created or signed the file, extracted from file metadata or digital signatures.
country_code	string	**PATTERN:**`^[A-Z]{2}$` ISO 3166-1 alpha-2 country code.
country_name	string	Full country name where the file originated, based on company information and digital signatures.
signer	string	Name of the entity that digitally signed the file, extracted from the digital signature certificate.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
deflection_info	object	
final_verdict	object	
verdict	enum	**ALLOWED:** `clean, malicious, unknown, Cancelled` Deflection engine verdict. Values: - clean: the file is considered clean. - malicious: the file is considered malicious. - unknown: the engine could not confidently classify the file. - Cancelled: processing was cancelled before a verdict was produced.
confidence	integer	`between 0 and 100` Confidence of the verdict, as a percentage (0-100).
detail	object	
data_id	string	data identifier of the analyzed file.

NAME	TYPE	DESCRIPTION
file_type	`string`	Detected file type of the analyzed file.
result_template_hash	`string`	Hash of the result template used internally for Deflection rendering and coordination.
data_id	`string`	data identifier of the requested file
original_data_id	`string`	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
dlp_info	`object`	
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
anonymization	`object`	
result	`string`	Result of the anonymization process.
crop_embedded_images	`object`	
result	`string`	Result of the crop embedded images process.
current_file_certainty	`string`	Describes how certain the current file hit is.
current_file_verdict	`integer`	The result for the current scanned file.
detection_policy	`object`	
result	`string`	Status of detection policy.
document_identification	`object`	
result	`string`	Result of the document identification process.
errors	`object`	
filename	`string`	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	`object`	
ccn	`object`	
display_name	`string`	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	`array`	
after	`string`	The context after the matched data.
before	`string`	The context before the matched data.

NAME	TYPE	DESCRIPTION
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
intentional_data_leak	object	
result	string	Result of the intentional data leak detection process.
metadata_removal	object	
result	enum	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
processing_info	object	
encoding	string	Character encoding used for processing the file.
recursive_processing	object	
result	string	Result of the recursive processing.
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
scan	object	
result	string	Overall result of scanning process.
types	object	
ai	object	
display_name	string	Display name for AI scanning.
result	string	Result of AI scanning.
regex	object	

NAME	TYPE	DESCRIPTION
display_name	`string`	Display name for Regex scanning.
result	`string`	Result of Regex scanning.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	`object`	
error_detail	`string`	Revealed detailed reason why the download failed.
progress	`number`	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [1, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`) if the download stopped unexpectedly.

NAME	TYPE	DESCRIPTION

status	`string`	Indicates download status, which could be either
		- `Downloading`
		- Check `progress` key value for actual download percentage
		``json
		"download_info": {
		"progress": 7,
		"status": "Downloading",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``
		- `Download Success`
		``json
		"download_info": {
		"status": "Download Success",
		"url": "https://secure.eicar.org/eicar.com"
		}
		``
		- `Download Failed`
		- Check `error_detail` key value for an error explanation
		``json
		"download_info": {
		"error_detail": "Connection error",
		"status": "Download Failed",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Timeout`
		- Expecting to occur when the download progress takes longer than what time window allowed in MetaDefender Core's pre-configured setting under workflow rule (under "SCAN" tab)
		``json
		"download_info": {
		"status": "Download Timeout",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Cancelled`
		- Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to
		``json
		"download_info": {
		"status": "Download Cancelled",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``

url	`string`	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	`object`	
decrypted_status	`enum`	**ALLOWED:** `Success, Failed` Indicate that decryption phase is successful or not.
err_category	`string`	Error category
err_code	`integer`	Error code
err_description	`string`	Error description
err_details	`string`	Error message

NAME	TYPE	DESCRIPTION
is_encrypted_file	boolean	Indicate if file is password-protected or not.
stop_processing_reason	string	The verdict of the extracted file that triggered the archive stop-processing policy of the workflow rule, e.g. `Infected`. Only present when `stopped_by_policy` is.
stopped_by_policy	boolean	Present (always `true`) when processing of this archive, or of an archive nested in it, was stopped early by the archive stop-processing policy of the workflow rule. Extracted files that were stopped before they finished get the `Early Exit` (81) result.
total_extraction_error_count	integer	Total number of extraction errors encountered.
total_reused_extraction_errors	integer	Indicates how many extraction errors were reused from a previous analysis instead of being detected during the current analysis. This can include errors from the archive itself or from nested archives. A value of 0 means no extraction errors were reused. Available starting with MetaDefender Core 5.22.0.
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
file_type_id	string	File type identifier code.
is_skip_hash	boolean	Whether hash calculation was skipped for this file.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)

NAME	TYPE	DESCRIPTION
filestructurevalidation_info	`object`	
final_verdict	`object`	
verdict	`enum`	ALLOWED: `allowed, blocked, cancelled` * `allowed` - the file structure was accepted * `blocked` - the file structure was rejected * `cancelled` - processing was aborted before a verdict was produced
verdict_explanation	`array`	
err_code	`integer`	Error or limit code, returned on a failure or an output-object limit and only then - a determinate result carries `final_verdict` instead. Codes not listed below can be passed through from the engine. * `1` - File Structure Validation failed (engine error) * `5` - File Structure Validation timed out * `9` - Global timeout exceeded * `10` - Exceeded structural depth (output-object nesting limit) * `11` - Exceeded structural object count (output-object count limit) * `12` - Exceeded structural object size (output-object size limit)
err_category	`enum`	ALLOWED: `Engine error, Timeout, Global Timeout Exceeded, Output object limit exceeded` Category of the reported `err_code`. * `Engine error` - code 1 and any unrecognised code * `Timeout` - code 5 * `Global Timeout Exceeded` - code 9 * `Output object limit exceeded` - codes 10, 11 and 12
err_details	`string`	Human readable description of the error or limit. Defaults to the action name matching `err_code` when the engine does not supply a more specific message.
process_details	`object`	
description	`string`	Summary of what the engine did with the file.
skipped_reason	`string`	Reason why validation actions were not applied to the detected objects. `analysis_mode` means FSV only reported the file structure without acting on it.
failure_category	`string`	Engine-supplied failure category, for example `invalid file structure` or `failed to sanitize`.
details	`array`	
action	`string`	Action FSV applied to the objects in this group, for example `none` when it only reported them, or `extracted` when it pulled them out into their own files.
count	`integer`	Number of objects of this kind covered by the entry. It is a multiplicity and is not always `1`, so the number of detected objects is the sum of `count` across the array, not the length of the array.
object_name	`string`	Kind of the detected object.
file_name	`string`	Path of the internal part the object was found in. Present for objects that map to a named part of the container.
result_of_action	`string`	Outcome of `action` for this group.
object_sha256	`array`	
object_details	`array`	
ONE:OF	`object`	

NAME	TYPE	DESCRIPTION
prop0	string	The detected object as the engine found it in the container.
OPTION:1	object	
object_detail	string	Identifier of the object - a target path, a URL, or similar.
structural_data	array	
target_mode	string	Whether the object points inside the container or outside of it. The engine is not consistent about capitalisation - both `internal` and `External` occur - so compare case-insensitively.
file_type	string	Detected type of the embedded object.
mime_type	string	MIME type of the embedded object.
image_hash	string	SHA256 of the embedded image.
binary_object_details	array	
binary_object_encoding	string	Encoding applied to every entry of `binary_object_details`.
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	ALLOWED: `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	array	
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	
description*	string	File type description or identifier from the detector.

NAME	TYPE	DESCRIPTION
detector*	string	Name of the detection engine that provided this result.
score*	integer	Confidence score of the detection result.
model_version	string	Version of the model used by the detector (if applicable).
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
fsv_output_files	object	
first_index	integer	Index of the first listed object, i.e. the offset the page starts at (driven by the `first` query parameter, default `0`).
page_size	integer	Number of objects the response lists information about (driven by the `size` query parameter, default `50`).
fsv_output_files_count	integer	Number of output objects dispatched directly out of this file. It counts what was analyzed, not what was detected - an output-object limit caps it without marking the list as truncated.
total_fsv_output_files	integer	Number of objects in the whole FSV output subtree of this file, i.e. including the objects extracted out of the extracted objects.
files_in_fsv_output	array	
data_id	string	data identifier of the extracted object. Its own analysis result carries `parent_type: fsv_output` and `parent_data_id` pointing back to this file.
object_type	string	Kind of the embedded object FSV extracted, as reported by the engine (for example `image`, `hyperlink`, `macro`).
object_name	string	Name the engine gave to the embedded object.
display_name	string	Name the extracted object is analyzed under.
file_size	integer	Size of the extracted object in bytes.
file_type	string	MIME type of the extracted object.
file_type_description	string	File type description of the extracted object.
file_type_id	string	File type ID of the extracted object.
type_category	array	
detected_by	integer	Number of engines that reported a threat in the extracted object.
scanned_with	integer	Number of engines the extracted object was scanned with.
progress_percentage	integer	Percentage of processing completed for the extracted object (from 1-100).

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall analysis result of the extracted object as string.
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall analysis result of the extracted object as index in the Processing Results table.
files_extracted_count	integer	Number of files extracted out of this object, when the object is itself a container. Omitted otherwise.
original_data_id	string	data identifier of the original analysis this result was reused from. Only present when the result was served from an earlier analysis.
total_reused_files	integer	Number of this object's children that were served from an earlier analysis. Omitted when nothing was reused.
instance_name	string	Name of the instance that processed the object. Only present when MetaDefender Core runs in shared or instance database mode.
process_info	object	

NAME	TYPE	DESCRIPTION
blocked_reason	`string`	Provides the reason why the extracted object is blocked. Only present when `result` is `Blocked`.
blocked_reasons	`array`	
rejected_reason	`string`	Why the request was rejected. Only present when `is_rejected_request` is true.
allowlisted_reason	`string`	What allowlisted the object, when it was allowlisted.
blocklisted_reason	`string`	What blocklisted the object, when it was blocklisted.
is_rejected_request	`boolean`	Indicates if the request was rejected due to configuration such as file size limits. Only present once processing completed.
progress_percentage	`integer`	Percentage of processing completed (from 1-100).
result	`string`	The final result of processing the extracted object (Allowed / Blocked / Processing).
verdicts	`array`	
post_processing	`object`	
actions_failed	`string`	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	`string`	List of successful actions, separated by "|". Empty string if otherwise.
insightsti_info	`object`	
opswatfilescan_info	`object`	
final_verdict	`object`	
blocked	`boolean`	Indicates if the file is blocked based on Adaptive Sandbox analysis
confidence	`number`	`between 0 and 1` Confidence level of the verdict (0.0 to 1.0)
threat_level	`number`	`between 0 and 1` Threat level assessment (0.0 to 1.0, where higher values indicate higher threat)
verdict	`enum`	**ALLOWED:** `UNKNOWN, BENIGN, NO THREAT DETECTED, SUSPICIOUS, LIKELY_MALICIOUS, MALICIOUS` The verdict classification from Adaptive Sandbox
overview_report	`object`	
report_url	`string`	URL to access the detailed analysis report
result_template_hash	`string`	Hash identifier for the result template used
parent_type	`enum`	**ALLOWED:** `archive, batch, fsv_output` How this file was produced from `parent_data_id`. * `archive` - the file was extracted from an archive or another compound file * `batch` - the file was submitted as part of a batch * `fsv_output` - the file is an embedded object the File Structure Validation engine extracted from its parent, and it is listed in the parent's `fsv_output_files`
post_action	`object`	
details*	`array`	
duration*	`integer`	Execution duration of the post-action in milliseconds.

NAME	TYPE	DESCRIPTION
name*	string	Name or identifier of the post-action.
return_value*	integer	Return value or exit code from the post-action execution.
verdict*	integer	Verdict code representing the outcome of the post-action.
total_blocked*	integer	Total number of files that were indicated to be blocked by Post Action execution.
total_executed*	integer	Total number of post-actions that were executed.
total_failed*	integer	Total number of post-actions that failed during execution.
total_post_action*	integer	Total number of post-actions configured or available.
total_success*	integer	Total number of post-actions that executed successfully.
process_info	object	
archive_handling_details	object	
description	string	Describes what occurred during archive extraction.
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
allowlisted_reason	string	What allowlisted the file. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Blocklisted`.
blocklisted_reason	string	What blocklisted the file, one of `By File Type`, `By File Size`, `By File Name`, `By MIME Type`, `Skiplist - Hash`, `By Hash`, `Unsupported Sanitization File Type`, `Prevent Data Leak Unsupported File Type`, `Password Protection Required`. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Allowlisted`.
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
is_rejected_request	boolean	Indicates if the request was rejected due to configuration such as file size limits.
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
aigcd_time	integer	AI Content Inspector engine's processing time.
aigcd_wait_time	integer	AI Content Inspector engine's wait time.
av_scan_time	integer	AV engines' processing time.
av_wait_time	integer	AV engines' wait time.
cdr_time	integer	Deep CDR engine's sanitization time.
cdr_storage_time	integer	Deep CDR engine's storage time.
cdr_wait_time	integer	Deep CDR engine's wait time.
child_files_hash_time	integer	Time elapsed for hashing child files.
cloud_lookup_time	integer	Cloud lookup processing time.
coo_time	integer	Country of Origin engine's processing time.

NAME	TYPE	DESCRIPTION
decrypted_time	integer	File decryption processing time.
dlp_time	integer	Proactive DLP engine's processing time.
dlp_storage_time	integer	Proactive DLP engine's storage time.
extraction_time	integer	Archive extraction engine's processing time.
extraction_wait_time	integer	Archive extraction engine's wait time.
filetype_time	integer	FileType engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
fsv_time	integer	File Structure Validation engine's processing time.
fsv_wait_time	integer	File Structure Validation engine's wait time.
opswatfilescan_time	integer	Adaptive Sandbox engine's processing time.
opswatfilescan_wait_time	integer	Adaptive Sandbox engine's wait time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
post_action_time	integer	Post action processing time.
quarantine_storage_time	integer	Quarantine storage processing time.
reputation_time	integer	Reputation engine's processing time.
sbom_time	integer	SBOM (Software Bill of Materials) processing time.
vul_time	integer	Vulnerability engine's lookup time.
wait_child_files_time	integer	Wait time for child files processing.
yara_time	integer	YARA engine's processing time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.

NAME	TYPE	DESCRIPTION
converted_destination	`string`	Contains the name of the sanitized file.
converted_to	`string`	Contains target type name of sanitization.
copy_move_destination	`string`	Contains target type name of sanitization.
sanitization_details	`object`	
cdr_wait_time	`integer`	The time in milliseconds that the CDR process took to complete.
description	`string`	Action was successful or not.
details	`array`	
action*	`enum`	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	`integer`	The number of objects that were sanitized/removed.
details	`object`	
action	`enum`	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	`integer`	The number of objects that were sanitized/removed.
object_details	`array`	
object_name	`string`	The object type that was sanitized/removed.
description	`string`	Action was successful or not.
file_name	`string`	If an embedded file was sanitized.
object_details	`array`	
object_name*	`string`	The object type that was sanitized/removed.
failure_category	`string`	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	`enum`	**ALLOWED:** `Sanitized, Sanitized failed, Sanitized skipped` The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	`string`	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MD Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	`object`	
file_size	`integer`	Size of sanitized file in bytes.
sha256	`string`	SHA256 hash of sanitized file.

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `blocked, allowed` The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
status	enum	**ALLOWED:** `success, failure, skipped` The status of the Deep CDR process (This field is only available for Deep CDR processing and does not apply to archive sanitization). - **Queued**: indicate the file that is currently waiting in the MD Core's queue before being sent to the Deep CDR engine. - **Processing**: indicate the file that has been sent to the Deep CDR engine and is currently being processed. - **Finished**: the Deep CDR process was finished. At this point, there are several additional states you can further check: - Check whether the file was sanitized successfully. Indicator: `post_processing.actions_ran` contains `Sanitized`. - Check whether the file failed to be sanitized. Indicator: `post_processing.actions_failed` contains `Sanitization Failed`. - Check whether the file is marked as blocked by the Deep CDR engine. Indicator: `post_processing.sanitization_details.verdict` = `blocked`.
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
engine_results	object	
sanitization_result	enum	**ALLOWED:** `Success, Failed, Not Run` The overall sanitization result across all engines.
reputation_info	object	
final_verdict*	object	
verdict*	enum	**ALLOWED:** `Known Good, Known Bad, Unknown` Reputation engine verdict. Values: - Known Good: The file has a positive reputation. - Known Bad: The file has a negative reputation. - Unknown: Reputation could not be determined.
verdict_explanation*	array	
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
sbom_info	object	
final_verdict*	object	

NAME	TYPE	DESCRIPTION
blocked*	boolean	Whether the container/application is blocked based on security policies.
licensing_package_counts*	object	
allowed*	integer	Number of packages with allowed licenses.
blocked*	integer	Number of packages with blocked licenses.
unknown*	integer	Number of packages with unknown licensing status.
unsupported*	integer	Number of packages with unsupported license detection.
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Overall security severity level of the SBOM.
total_package_count*	integer	Total number of packages analyzed in the SBOM.
verdict*	string	Human-readable verdict summary.
verdict_explanation*	array	
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities.
overview_report*	object	
has_pe_info*	boolean	Indicates whether PE (Portable Executable) information is available.
is_partial_report*	boolean	Indicates if this is a partial report with incomplete data.
package_count*	integer	Total number of packages analyzed in the overview.
packages*	array	
archive*	boolean	Indicates if the package is an archive.
authors*	string	Package author(s).
ecosystem*	string	Package ecosystem or format.
group*	string	Package group classification.
hashes*	array	
file_name*	string	Name of the file.
sha256*	string	SHA-256 hash of the file.
latest*	object	
release_date*	string	Release date of the latest version.
version*	string	Latest version number.
licenses*	object	

NAME	TYPE	DESCRIPTION
allowed	array	
category*	string	License category classification.
id*	string	SPDX license identifier.
url*	string	URL to license details.
usage_restriction*	object	
conditions*	array	
limitations*	array	
permissions*	array	
package_name*	string	Name of the package.
package_type*	string	Type classification of the package.
package_version*	string	Version of the package.
release_date*	string	Package release date.
target*	string	Target file or manifest where the package was found.
uid*	string	Unique package identifier (Package URL format).
vulnerabilities*	array	
cwes*	array	
fixed_versions*	string	Versions where the vulnerability is fixed.
id*	string	Vulnerability identifier (CVE, GHSA, etc.).
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Vulnerability severity level.
source*	string	Vulnerability database source.
type*	string	Type of security issue.
vulnerability_counts*	object	
critical*	integer	
high*	integer	
low*	integer	
medium*	integer	
unknown*	integer	
scan_mode*	enum	**ALLOWED:** `container, filesystem, archive, source_code` Type of scan performed for SBOM analysis.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the overview report.
timestamp*	string	Timestamp when the overview report was generated (ISO 8601 format).
vulnerability_count*	integer	Total number of vulnerabilities found in the overview.
vulnerability_counts*	object	

NAME	TYPE	DESCRIPTION
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities in the overview.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
bom_info	object	
final_verdict	object	
verdict	string	
result_subtrees	array	
cbom_info	object	
aibom_info	object	
scan_results	object	
current_av_result_a	string	The AV scan result for the current file itself (e.g., archive container) as string, excluding extracted contents. Uses the same enumeration values as scan_all_result_a
current_av_result_i	string	The AV scan result for the current file itself (e.g., archive container) as index, excluding extracted contents. Uses the same enumeration values as scan_all_result_i
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table

NAME	TYPE	DESCRIPTION
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
threatintelligence_info	object	
final_verdict*	object	
blocked	boolean	Whether the file should be blocked based on Threat Intelligence evaluation.
ioc_reputation	integer	Aggregate reputation score derived from matched indicators of compromise.
most_similar_file	number	Similarity score of the most similar known file (0 to 1).
result*	object	
result_template_hash*	string	Hash of the result template used internally for Threat Intelligence rendering and coordination.
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained
severity	enum	ALLOWED: `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`

NAME	TYPE	DESCRIPTION
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time
published-epoch	string	An epoch timestamp indicating source publishing time
references	array	

NAME	TYPE	DESCRIPTION
severity	enum	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara_info	object	
hits	object	
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |

STATUS CODE - 400: Invalid request, e.g. the API key or the `metadata` header is invalid.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Invalid apikey given
OPTION:2	object	
err	string	Both body data and download link were given.
OPTION:3	object	
err	string	Batch closed during file upload.
OPTION:4	object	
err	string	Can not scan in given batch.
OPTION:5	object	

NAME	TYPE	DESCRIPTION
err	string	Cannot scan batch in another instance.
OPTION:6	object	
err	string	Both body data and local file path were given.
OPTION:7	object	
err	string	File upload rejected due to insufficient disk space
OPTION:8	object	
err	string	Redirect link not supported.
OPTION:9	object	
err	string	Invalid download url.
OPTION:10	object	
err	string	Download url must start with http:// or https://.
OPTION:11	object	
err	string	The `source_system` value in the `metadata` header is invalid.

STATUS CODE - 403: Callback URL is not supported.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	callbackurl is not supported

STATUS CODE - 408: Timed out response. The scan is still in progress, but the connection is timed out.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	

EXAMPLE:

```
{
 "data_id": "61dffeaa728844adbf49eb090e4ece0e"
}
```

STATUS CODE - 411: Content-Length header is missing from the request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Missing Content-Length header."
}
```

STATUS CODE - 422: Body input is empty.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "File is empty."
}
```

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	License is expired
OPTION:2	object	
err	string	No available rule is present for scanning.
OPTION:3	object	
err	string	Failed to request scan. Try again later.
OPTION:4	object	
err	string	File not found, invalid path or access.
OPTION:5	object	
err	string	Failed to request scan. File size exceeded the maximum size permitted by your configuration.

STATUS CODE - 503: Server is too busy, scan queue is full. Try again later.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Server is too busy. Try again later."
}
```

4.3 GET /file/last

Get last processed file

Retrieve the information about the most recent file that has completed processing. The most recent file is determined based on the receive_data_timestamp of the scan request.

Notes: This API is accessible only with "Processing result fetching" permission set to Anyone.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
view	string		In shared database mode, specifies which instance's data is used to determine the last processed file. Provide an instance name to return the last processed file from that specific instance, or use `all` to return the most recent file across all instances. If omitted, only the last processed file from the current instance is returned.
rule	string		Filter by workflow name; returns the last processed file for the specified workflow.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Last processed file retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
data_id	string	Unique identifier of the scan request.
file_info	object	
display_name	string	Display name of the file.
file_size	integer	Size of the file in bytes.
file_type_description	string	Description of the detected file type.
md5	string	MD5 hash of the file.
receive_data_timestamp	string	Timestamp when the file was received for processing.
sha1	string	SHA1 hash of the file.
sha256	string	SHA256 hash of the file.
sha512	string	SHA512 hash of the file.
instance_name	string	Name of the MetaDefender Core instance that processed the file.
process_info	object	
result	string	Overall processing result.
scan_info	object	
rule	string	Workflow rule used for processing.
source	string	Source IP address of the scan request.
username	string	Username that submitted the scan request.
scan_results	object	
scan_all_result_a	string	Overall scan result in human-readable format.
start_time	string	Timestamp when the scan started.
total_time	integer	Total scan time in milliseconds.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: No file found that matches the specified conditions.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason.

EXAMPLE:

```
{
 "err": "No scan request found"
}
```

STATUS CODE - 405: User does not have "Processing result fetching" permission set to "Anyone".

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason.

EXAMPLE:

```
{
 "err": "Access denied"
}
```

4.4 GET /file/ids

List data IDs of processed objects

Return a paged list of data_id values for processed objects on the **local instance**. IDs are returned **regardless of status, result, or verdict**. The response contains only data_id strings — no filename, verdict, engines, or tags.

Use this endpoint when you need to discover IDs and then call a detail API such as GET /file/{data_id} or GET /file/batch/{batchId}. The response carries no scan payload, and there are no filename or verdict filters.

Authentication and permission: Header apikey is required. The caller must have the scanlog:read right (Processing History). With **fetch:anyone**, all local-instance IDs are returned. With **fetch:selfonly**, only IDs belonging to the caller are returned.

Request types: Each processed object is listed separately. request_type is a comma-separated,

case-sensitive list of rest, batch, batched, and extracted. Omit the parameter to include all four types. Duplicates are de-duplicated.

The default list mixes all four kinds and does **not** label them. An ID that belongs to a batch container is passed as batchId to GET /file/batch/{batchId}, not as data_id to GET /file/{data_id}. Clients that must route IDs should issue one call per request_type.

Pagination and sorting:

- Results are always ordered by receive (RC) timestamp (receive_data_timestamp). The sort column is not selectable.
- order_type sets the direction. The default is DESC (newest first).
- size defaults to 50 and must be **1–10000**. size=0 is rejected; omit the parameter to take the default.
- first is a 0-based offset into the filtered, sorted list.
- has_more is true when at least one more ID exists after this page.
- total_count equals first + items.length and is returned **only** when the page ends the list (has_more is false and either items is non-empty or first is 0). It is omitted when has_more is true, and when paging past the end of the list.
- A parameter that is present but blank (?size=, ?order_type=, …) is a **400**; only omitting the parameter selects the default.
- Unrecognized query parameters are **ignored**.

Time window: starttime_from and starttime_to are inclusive bounds on receive_data_timestamp in **epoch milliseconds**. Passing epoch **seconds** is not an error; the values are treated as dates far in the past and typically return an empty page.

Paging guidance:

- On a DESC sweep, new scans land at the head of the list and can duplicate or skip IDs across pages. Pin starttime_to at sweep start to freeze the window. An ASC sweep is drift-free because new IDs append at the tail.
- Response time degrades at very deep offsets. Prefer narrow time slices with first restarted at 0 for each slice. To reach the oldest IDs, use order_type=ASC&first=0 rather than a large DESC offset.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
size	integer between 1 and 10000	50	Page size. Must be an integer from **1** to **10000**. If omitted, defaults to `50`. `0`, a negative value, or a value greater than 10000 returns 400.
first	integer >=0		0-based offset into the filtered, sorted list. Defaults to `0`. A negative or non-numeric value returns 400.
starttime_from	int64	1755000000000	Inclusive lower bound on `receive_data_timestamp`, as **epoch milliseconds**.
starttime_to	int64	1755086400000	Inclusive upper bound on `receive_data_timestamp`, as **epoch milliseconds**. Pin this value at the start of a `DESC` sweep to avoid page drift while new scans arrive.
request_type	string	rest,batch	Comma-separated list of request kinds to include. Matching is **case-sensitive**. Allowed tokens: `rest` (top-level submitted file), `batch` (batch container), `batched` (file submitted as part of a batch), `extracted` (extracted child file). Omit the parameter to include all four types. Duplicates are de-duplicated. An unknown token, wrong case (for example `REST`), or empty token (trailing comma) returns 400.
order_type	string	DESC	Sort direction for receive (RC) timestamp (`receive_data_timestamp`). `ASC` (oldest first) or `DESC` (newest first). Case-insensitive. Defaults to `DESC` whether or not a time range is set.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Page of `data_id` values retrieved successfully.

Example paging over 137 matching IDs with `size=50`:
- `GET /file/ids?size=50` → 50 items, `has_more: true` (no `total_count`)
- `GET /file/ids?size=50&first=50` → 50 items, `has_more: true`
- `GET /file/ids?size=50&first=100` → 37 items, `has_more: false`, `total_count: 137`
- `GET /file/ids?size=50&first=137` → empty `items`, `has_more: false`, no `total_count`

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
items*	`array`	
has_more*	`boolean`	`true` if at least one further ID exists after this page under the same filter.
total_count	`integer`	Present only when the list has been walked to its end. Equals `first + items.length`. Absent whenever `has_more` is `true`, and absent when paging past the end of the list.

EXAMPLE 1:

```
"hasMore"
```

EXAMPLE 2:

```
"lastPage"
```

EXAMPLE 3:

```
"pastEnd"
```

EXAMPLE 4:

```
"emptyHistory"
```

STATUS CODE - 400: Invalid query parameter. The body is `{ "err": "<message>" }`.

Condition	`err`
Unknown, wrong-case, or empty `request_type` token	`Value of request_type can only be a comma separated list of 'rest', 'batch', 'batched' and 'extracted'`
`size` is `0`, negative, or greater than 10000	`The 'size' parameter must be between 1 and 10000`
`order_type` is not `ASC` or `DESC`	`Invalid value for param 'order_type'`
Negative `first`, a blank value, or a non-numeric value in any numeric parameter	`Invalid value for param '<name>'`

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE 1:

```
"invalidRequestType"
```

EXAMPLE 2:

```
"invalidSize"
```

EXAMPLE 3:

```
"invalidOrderType"
```

EXAMPLE 4:

```
"invalidParam"
```

STATUS CODE - 403: Missing or invalid `apikey`, or a valid key without the processing-history fetch right (`scanlog:read` / fetch permission).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: The server could not retrieve the list. The reason is logged and is not returned in the response.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Internal server error"
}
```

4.5 GET /file/{data_id}

Fetch Analysis Result

Retrieve scan results.

Scan is done asynchronously and each scan request is tracked by a data ID.

Initiating file scans and retrieving the results need to be done using two separate API calls. This

request needs to be made multiple times until the scan is complete. Scan completion can be traced using scan_results.progress_percentage value from the response.

Note*:* The REST API also supports pagination for archive file result. A completed response description with archive detection:

- extracted_files: information about extracted files
  - files_extracted_count: the number of extracted files
  - files_in_archive: array of files in archive
    - detected_by: number of engines reported threat
    - scanned_with: number of engines used for scanning the file
    - original_data_id: the data ID of the previous analysis whose result was reused for this file; present only when the result was reused. Available starting with MetaDefender Core 5.22.0.

  - first_index: it tells that from which file (index of the file, 0 is the first) the result JSON contains information about extracted files. (default=0)
  - page_size: it tells how many files the result JSON contains information about (default=50). So by default, the result JSON contains information about the first 50 extracted files.
  - total_extracted_files: total number of extracted files in the archive
  - total_extracted_size: total size of all extracted files in bytes
  - total_reused_files: indicates how many extracted files reused results from a previous analysis instead of being processed again. A value of 0 means no extracted files were reused. If the entire archive is reused, original_data_id is set and files_in_archive is empty, while the archive totals are still included in the response. Available starting with MetaDefender Core 5.22.0.
  - worst_data_id: data id of the file that has the worst result in the archive

- fsv_output_files: information about the objects the File Structure Validation (FSV) engine extracted out of the file. Absent when FSV dispatched no object.
  - files_in_fsv_output: array of extracted objects, each with its own data_id, plus object_type and object_name describing the embedded object it came from
  - fsv_output_files_count: number of objects dispatched directly out of this file
  - total_fsv_output_files: number of objects in the whole FSV output subtree of this file
  - first_index / page_size: paged by the same first and size parameters as extracted_files - the two lists share one window

- scan_results
  - last_file_scanned (stored only in memory, not in database): If available, the name of the most recent processed file

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer		The first item order in the list child files of archive file. Also applies to `fsv_output_files.files_in_fsv_output`, which shares this window.
size	integer	1	The number of child files in the archive to return, starting from the item order specified by the `first` query parameter. If child file information is not needed, set this to `1` to optimize query time. The same window applies to `fsv_output_files.files_in_fsv_output`, so lowering it truncates that list too.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used. Since MetaDefender Core 5.6.0, when user agent is matched the predefined setting in workflow rule, the product will return X-Core-Id containing identity of MetaDefender Core server as a header of response.

RESPONSE

STATUS CODE - 200: Entire analysis report generated by MetaDefender Core

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
aicontentinspector_info	`object`	
final_verdict*	`object`	
verdict	`enum`	**ALLOWED:** `Not Detected, Detected, Uncertain` AI Content Inspector engine verdict. Values: - Detected: AI-generated or manipulated content was detected. - Not Detected: No AI-generated or manipulated content was found. - Uncertain: Content could not be confidently classified.
verdict_explanation	`array`	
output_images	`array`	
image_alias	`string`	Alias identifier for the output image, used to retrieve the image via the API.
result_description*	`object`	
result_template_hash*	`string`	Hash of the result template used internally for AI Content Inspector rendering and coordination.
coo_info	`object`	
final_verdict*	`object`	
verdict	`enum`	**ALLOWED:** `blocked, allowed, unknown` Country of Origin engine verdict. Values: - blocked: The file is considered blocked based on origin evaluation. - allowed: The file origin passed all policy checks. - unknown: Origin could not be confidently determined.
verdict_explanation	`array`	
full_report	`object`	
overview_report*	`object`	
company_name	`string`	Name of the company that created or signed the file, extracted from file metadata or digital signatures.
country_code	`string`	**PATTERN:**`^[A-Z]{2}$` ISO 3166-1 alpha-2 country code.
country_name	`string`	Full country name where the file originated, based on company information and digital signatures.
signer	`string`	Name of the entity that digitally signed the file, extracted from the digital signature certificate.
result_template_hash*	`string`	Hash of the result template used internally for rendering and engine/Core coordination.
deflection_info	`object`	
final_verdict	`object`	
verdict	`enum`	**ALLOWED:** `clean, malicious, unknown, Cancelled` Deflection engine verdict. Values: - clean: the file is considered clean. - malicious: the file is considered malicious. - unknown: the engine could not confidently classify the file. - Cancelled: processing was cancelled before a verdict was produced.

NAME	TYPE	DESCRIPTION
confidence	integer	between 0 and 100 Confidence of the verdict, as a percentage (0-100).
detail	object	
data_id	string	data identifier of the analyzed file.
file_type	string	Detected file type of the analyzed file.
result_template_hash	string	Hash of the result template used internally for Deflection rendering and coordination.
data_id	string	data identifier of the requested file
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
dlp_info	object	
certainty	enum	ALLOWED: Very Low, Low, Medium, High, Very High Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
anonymization	object	
result	string	Result of the anonymization process.
crop_embedded_images	object	
result	string	Result of the crop embedded images process.
current_file_certainty	string	Describes how certain the current file hit is.
current_file_verdict	integer	The result for the current scanned file.
detection_policy	object	
result	string	Status of detection policy.
document_identification	object	
result	string	Result of the document identification process.
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	object	
ccn	object	
display_name	string	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	array	
after	string	The context after the matched data.
before	string	The context before the matched data.

NAME	TYPE	DESCRIPTION
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
intentional_data_leak	object	
result	string	Result of the intentional data leak detection process.
metadata_removal	object	
result	enum	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
processing_info	object	
encoding	string	Character encoding used for processing the file.
recursive_processing	object	
result	string	Result of the recursive processing.
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
scan	object	
result	string	Overall result of scanning process.
types	object	
ai	object	
display_name	string	Display name for AI scanning.
result	string	Result of AI scanning.
regex	object	

NAME	TYPE	DESCRIPTION
display_name	`string`	Display name for Regex scanning.
result	`string`	Result of Regex scanning.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	`object`	
error_detail	`string`	Revealed detailed reason why the download failed.
progress	`number`	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [1, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`) if the download stopped unexpectedly.

NAME	TYPE	DESCRIPTION

status	`string`	Indicates download status, which could be either
		- `Downloading`
		- Check `progress` key value for actual download percentage
		``json
		"download_info": {
		"progress": 7,
		"status": "Downloading",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``
		- `Download Success`
		``json
		"download_info": {
		"status": "Download Success",
		"url": "https://secure.eicar.org/eicar.com"
		}
		``
		- `Download Failed`
		- Check `error_detail` key value for an error explanation
		``json
		"download_info": {
		"error_detail": "Connection error",
		"status": "Download Failed",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Timeout`
		- Expecting to occur when the download progress takes longer than what time window allowed in MetaDefender Core's pre-configured setting under workflow rule (under "SCAN" tab)
		``json
		"download_info": {
		"status": "Download Timeout",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Cancelled`
		- Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to
		``json
		"download_info": {
		"status": "Download Cancelled",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``

url	`string`	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	`object`	
decrypted_status	`enum`	**ALLOWED:** `Success, Failed` Indicate that decryption phase is successful or not.
err_category	`string`	Error category
err_code	`integer`	Error code
err_description	`string`	Error description
err_details	`string`	Error message

NAME	TYPE	DESCRIPTION
is_encrypted_file	boolean	Indicate if file is password-protected or not.
stop_processing_reason	string	The verdict of the extracted file that triggered the archive stop-processing policy of the workflow rule, e.g. `Infected`. Only present when `stopped_by_policy` is.
stopped_by_policy	boolean	Present (always `true`) when processing of this archive, or of an archive nested in it, was stopped early by the archive stop-processing policy of the workflow rule. Extracted files that were stopped before they finished get the `Early Exit` (81) result.
total_extraction_error_count	integer	Total number of extraction errors encountered.
total_reused_extraction_errors	integer	Indicates how many extraction errors were reused from a previous analysis instead of being detected during the current analysis. This can include errors from the archive itself or from nested archives. A value of 0 means no extraction errors were reused. Available starting with MetaDefender Core 5.22.0.
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
file_type_id	string	File type identifier code.
is_skip_hash	boolean	Whether hash calculation was skipped for this file.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)

NAME	TYPE	DESCRIPTION
filestructurevalidation_info	`object`	
final_verdict	`object`	
verdict	`enum`	ALLOWED: `allowed, blocked, cancelled` * `allowed` - the file structure was accepted * `blocked` - the file structure was rejected * `cancelled` - processing was aborted before a verdict was produced
verdict_explanation	`array`	
err_code	`integer`	Error or limit code, returned on a failure or an output-object limit and only then - a determinate result carries `final_verdict` instead. Codes not listed below can be passed through from the engine. * `1` - File Structure Validation failed (engine error) * `5` - File Structure Validation timed out * `9` - Global timeout exceeded * `10` - Exceeded structural depth (output-object nesting limit) * `11` - Exceeded structural object count (output-object count limit) * `12` - Exceeded structural object size (output-object size limit)
err_category	`enum`	ALLOWED: `Engine error, Timeout, Global Timeout Exceeded, Output object limit exceeded` Category of the reported `err_code`. * `Engine error` - code 1 and any unrecognised code * `Timeout` - code 5 * `Global Timeout Exceeded` - code 9 * `Output object limit exceeded` - codes 10, 11 and 12
err_details	`string`	Human readable description of the error or limit. Defaults to the action name matching `err_code` when the engine does not supply a more specific message.
process_details	`object`	
description	`string`	Summary of what the engine did with the file.
skipped_reason	`string`	Reason why validation actions were not applied to the detected objects. `analysis_mode` means FSV only reported the file structure without acting on it.
failure_category	`string`	Engine-supplied failure category, for example `invalid file structure` or `failed to sanitize`.
details	`array`	
action	`string`	Action FSV applied to the objects in this group, for example `none` when it only reported them, or `extracted` when it pulled them out into their own files.
count	`integer`	Number of objects of this kind covered by the entry. It is a multiplicity and is not always `1`, so the number of detected objects is the sum of `count` across the array, not the length of the array.
object_name	`string`	Kind of the detected object.
file_name	`string`	Path of the internal part the object was found in. Present for objects that map to a named part of the container.
result_of_action	`string`	Outcome of `action` for this group.
object_sha256	`array`	
object_details	`array`	
ONE:OF	`object`	

NAME	TYPE	DESCRIPTION
prop0	string	The detected object as the engine found it in the container.
OPTION:1	object	
object_detail	string	Identifier of the object - a target path, a URL, or similar.
structural_data	array	
target_mode	string	Whether the object points inside the container or outside of it. The engine is not consistent about capitalisation - both `internal` and `External` occur - so compare case-insensitively.
file_type	string	Detected type of the embedded object.
mime_type	string	MIME type of the embedded object.
image_hash	string	SHA256 of the embedded image.
binary_object_details	array	
binary_object_encoding	string	Encoding applied to every entry of `binary_object_details`.
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	ALLOWED: `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	array	
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	
description*	string	File type description or identifier from the detector.

NAME	TYPE	DESCRIPTION
detector*	string	Name of the detection engine that provided this result.
score*	integer	Confidence score of the detection result.
model_version	string	Version of the model used by the detector (if applicable).
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
fsv_output_files	object	
first_index	integer	Index of the first listed object, i.e. the offset the page starts at (driven by the `first` query parameter, default `0`).
page_size	integer	Number of objects the response lists information about (driven by the `size` query parameter, default `50`).
fsv_output_files_count	integer	Number of output objects dispatched directly out of this file. It counts what was analyzed, not what was detected - an output-object limit caps it without marking the list as truncated.
total_fsv_output_files	integer	Number of objects in the whole FSV output subtree of this file, i.e. including the objects extracted out of the extracted objects.
files_in_fsv_output	array	
data_id	string	data identifier of the extracted object. Its own analysis result carries `parent_type: fsv_output` and `parent_data_id` pointing back to this file.
object_type	string	Kind of the embedded object FSV extracted, as reported by the engine (for example `image`, `hyperlink`, `macro`).
object_name	string	Name the engine gave to the embedded object.
display_name	string	Name the extracted object is analyzed under.
file_size	integer	Size of the extracted object in bytes.
file_type	string	MIME type of the extracted object.
file_type_description	string	File type description of the extracted object.
file_type_id	string	File type ID of the extracted object.
type_category	array	
detected_by	integer	Number of engines that reported a threat in the extracted object.
scanned_with	integer	Number of engines the extracted object was scanned with.
progress_percentage	integer	Percentage of processing completed for the extracted object (from 1-100).

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall analysis result of the extracted object as string.
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall analysis result of the extracted object as index in the Processing Results table.
files_extracted_count	integer	Number of files extracted out of this object, when the object is itself a container. Omitted otherwise.
original_data_id	string	data identifier of the original analysis this result was reused from. Only present when the result was served from an earlier analysis.
total_reused_files	integer	Number of this object's children that were served from an earlier analysis. Omitted when nothing was reused.
instance_name	string	Name of the instance that processed the object. Only present when MetaDefender Core runs in shared or instance database mode.
process_info	object	

NAME	TYPE	DESCRIPTION
blocked_reason	`string`	Provides the reason why the extracted object is blocked. Only present when `result` is `Blocked`.
blocked_reasons	`array`	
rejected_reason	`string`	Why the request was rejected. Only present when `is_rejected_request` is true.
allowlisted_reason	`string`	What allowlisted the object, when it was allowlisted.
blocklisted_reason	`string`	What blocklisted the object, when it was blocklisted.
is_rejected_request	`boolean`	Indicates if the request was rejected due to configuration such as file size limits. Only present once processing completed.
progress_percentage	`integer`	Percentage of processing completed (from 1-100).
result	`string`	The final result of processing the extracted object (Allowed / Blocked / Processing).
verdicts	`array`	
post_processing	`object`	
actions_failed	`string`	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	`string`	List of successful actions, separated by "|". Empty string if otherwise.
insightsti_info	`object`	
opswatfilescan_info	`object`	
final_verdict	`object`	
blocked	`boolean`	Indicates if the file is blocked based on Adaptive Sandbox analysis
confidence	`number`	`between 0 and 1` Confidence level of the verdict (0.0 to 1.0)
threat_level	`number`	`between 0 and 1` Threat level assessment (0.0 to 1.0, where higher values indicate higher threat)
verdict	`enum`	**ALLOWED:** `UNKNOWN, BENIGN, NO THREAT DETECTED, SUSPICIOUS, LIKELY_MALICIOUS, MALICIOUS` The verdict classification from Adaptive Sandbox
overview_report	`object`	
report_url	`string`	URL to access the detailed analysis report
result_template_hash	`string`	Hash identifier for the result template used
parent_type	`enum`	**ALLOWED:** `archive, batch, fsv_output` How this file was produced from `parent_data_id`. * `archive` - the file was extracted from an archive or another compound file * `batch` - the file was submitted as part of a batch * `fsv_output` - the file is an embedded object the File Structure Validation engine extracted from its parent, and it is listed in the parent's `fsv_output_files`
post_action	`object`	
details*	`array`	
duration*	`integer`	Execution duration of the post-action in milliseconds.

NAME	TYPE	DESCRIPTION
name*	string	Name or identifier of the post-action.
return_value*	integer	Return value or exit code from the post-action execution.
verdict*	integer	Verdict code representing the outcome of the post-action.
total_blocked*	integer	Total number of files that were indicated to be blocked by Post Action execution.
total_executed*	integer	Total number of post-actions that were executed.
total_failed*	integer	Total number of post-actions that failed during execution.
total_post_action*	integer	Total number of post-actions configured or available.
total_success*	integer	Total number of post-actions that executed successfully.
process_info	object	
archive_handling_details	object	
description	string	Describes what occurred during archive extraction.
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
allowlisted_reason	string	What allowlisted the file. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Blocklisted`.
blocklisted_reason	string	What blocklisted the file, one of `By File Type`, `By File Size`, `By File Name`, `By MIME Type`, `Skiplist - Hash`, `By Hash`, `Unsupported Sanitization File Type`, `Prevent Data Leak Unsupported File Type`, `Password Protection Required`. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Allowlisted`.
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
is_rejected_request	boolean	Indicates if the request was rejected due to configuration such as file size limits.
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
aigcd_time	integer	AI Content Inspector engine's processing time.
aigcd_wait_time	integer	AI Content Inspector engine's wait time.
av_scan_time	integer	AV engines' processing time.
av_wait_time	integer	AV engines' wait time.
cdr_time	integer	Deep CDR engine's sanitization time.
cdr_storage_time	integer	Deep CDR engine's storage time.
cdr_wait_time	integer	Deep CDR engine's wait time.
child_files_hash_time	integer	Time elapsed for hashing child files.
cloud_lookup_time	integer	Cloud lookup processing time.
coo_time	integer	Country of Origin engine's processing time.

NAME	TYPE	DESCRIPTION
decrypted_time	integer	File decryption processing time.
dlp_time	integer	Proactive DLP engine's processing time.
dlp_storage_time	integer	Proactive DLP engine's storage time.
extraction_time	integer	Archive extraction engine's processing time.
extraction_wait_time	integer	Archive extraction engine's wait time.
filetype_time	integer	FileType engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
fsv_time	integer	File Structure Validation engine's processing time.
fsv_wait_time	integer	File Structure Validation engine's wait time.
opswatfilescan_time	integer	Adaptive Sandbox engine's processing time.
opswatfilescan_wait_time	integer	Adaptive Sandbox engine's wait time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
post_action_time	integer	Post action processing time.
quarantine_storage_time	integer	Quarantine storage processing time.
reputation_time	integer	Reputation engine's processing time.
sbom_time	integer	SBOM (Software Bill of Materials) processing time.
vul_time	integer	Vulnerability engine's lookup time.
wait_child_files_time	integer	Wait time for child files processing.
yara_time	integer	YARA engine's processing time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.

NAME	TYPE	DESCRIPTION
converted_destination	`string`	Contains the name of the sanitized file.
converted_to	`string`	Contains target type name of sanitization.
copy_move_destination	`string`	Contains target type name of sanitization.
sanitization_details	`object`	
cdr_wait_time	`integer`	The time in milliseconds that the CDR process took to complete.
description	`string`	Action was successful or not.
details	`array`	
action*	`enum`	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	`integer`	The number of objects that were sanitized/removed.
details	`object`	
action	`enum`	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	`integer`	The number of objects that were sanitized/removed.
object_details	`array`	
object_name	`string`	The object type that was sanitized/removed.
description	`string`	Action was successful or not.
file_name	`string`	If an embedded file was sanitized.
object_details	`array`	
object_name*	`string`	The object type that was sanitized/removed.
failure_category	`string`	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	`enum`	**ALLOWED:** `Sanitized, Sanitized failed, Sanitized skipped` The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	`string`	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MD Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	`object`	
file_size	`integer`	Size of sanitized file in bytes.
sha256	`string`	SHA256 hash of sanitized file.

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `blocked, allowed` The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
status	enum	**ALLOWED:** `success, failure, skipped` The status of the Deep CDR process (This field is only available for Deep CDR processing and does not apply to archive sanitization). - **Queued**: indicate the file that is currently waiting in the MD Core's queue before being sent to the Deep CDR engine. - **Processing**: indicate the file that has been sent to the Deep CDR engine and is currently being processed. - **Finished**: the Deep CDR process was finished. At this point, there are several additional states you can further check: - Check whether the file was sanitized successfully. Indicator: `post_processing.actions_ran` contains `Sanitized`. - Check whether the file failed to be sanitized. Indicator: `post_processing.actions_failed` contains `Sanitization Failed`. - Check whether the file is marked as blocked by the Deep CDR engine. Indicator: `post_processing.sanitization_details.verdict` = `blocked`.
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
engine_results	object	
sanitization_result	enum	**ALLOWED:** `Success, Failed, Not Run` The overall sanitization result across all engines.
reputation_info	object	
final_verdict*	object	
verdict*	enum	**ALLOWED:** `Known Good, Known Bad, Unknown` Reputation engine verdict. Values: - Known Good: The file has a positive reputation. - Known Bad: The file has a negative reputation. - Unknown: Reputation could not be determined.
verdict_explanation*	array	
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
sbom_info	object	
final_verdict*	object	

NAME	TYPE	DESCRIPTION
blocked*	boolean	Whether the container/application is blocked based on security policies.
licensing_package_counts*	object	
allowed*	integer	Number of packages with allowed licenses.
blocked*	integer	Number of packages with blocked licenses.
unknown*	integer	Number of packages with unknown licensing status.
unsupported*	integer	Number of packages with unsupported license detection.
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Overall security severity level of the SBOM.
total_package_count*	integer	Total number of packages analyzed in the SBOM.
verdict*	string	Human-readable verdict summary.
verdict_explanation*	array	
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities.
overview_report*	object	
has_pe_info*	boolean	Indicates whether PE (Portable Executable) information is available.
is_partial_report*	boolean	Indicates if this is a partial report with incomplete data.
package_count*	integer	Total number of packages analyzed in the overview.
packages*	array	
archive*	boolean	Indicates if the package is an archive.
authors*	string	Package author(s).
ecosystem*	string	Package ecosystem or format.
group*	string	Package group classification.
hashes*	array	
file_name*	string	Name of the file.
sha256*	string	SHA-256 hash of the file.
latest*	object	
release_date*	string	Release date of the latest version.
version*	string	Latest version number.
licenses*	object	

NAME	TYPE	DESCRIPTION
allowed	array	
category*	string	License category classification.
id*	string	SPDX license identifier.
url*	string	URL to license details.
usage_restriction*	object	
conditions*	array	
limitations*	array	
permissions*	array	
package_name*	string	Name of the package.
package_type*	string	Type classification of the package.
package_version*	string	Version of the package.
release_date*	string	Package release date.
target*	string	Target file or manifest where the package was found.
uid*	string	Unique package identifier (Package URL format).
vulnerabilities*	array	
cwes*	array	
fixed_versions*	string	Versions where the vulnerability is fixed.
id*	string	Vulnerability identifier (CVE, GHSA, etc.).
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Vulnerability severity level.
source*	string	Vulnerability database source.
type*	string	Type of security issue.
vulnerability_counts*	object	
critical*	integer	
high*	integer	
low*	integer	
medium*	integer	
unknown*	integer	
scan_mode*	enum	**ALLOWED:** `container, filesystem, archive, source_code` Type of scan performed for SBOM analysis.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the overview report.
timestamp*	string	Timestamp when the overview report was generated (ISO 8601 format).
vulnerability_count*	integer	Total number of vulnerabilities found in the overview.
vulnerability_counts*	object	

NAME	TYPE	DESCRIPTION
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities in the overview.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
bom_info	object	
final_verdict	object	
verdict	string	
result_subtrees	array	
cbom_info	object	
aibom_info	object	
scan_results	object	
current_av_result_a	string	The AV scan result for the current file itself (e.g., archive container) as string, excluding extracted contents. Uses the same enumeration values as scan_all_result_a
current_av_result_i	string	The AV scan result for the current file itself (e.g., archive container) as index, excluding extracted contents. Uses the same enumeration values as scan_all_result_i
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table

NAME	TYPE	DESCRIPTION
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
threatintelligence_info	object	
final_verdict*	object	
blocked	boolean	Whether the file should be blocked based on Threat Intelligence evaluation.
ioc_reputation	integer	Aggregate reputation score derived from matched indicators of compromise.
most_similar_file	number	Similarity score of the most similar known file (0 to 1).
result*	object	
result_template_hash*	string	Hash of the result template used internally for Threat Intelligence rendering and coordination.
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained
severity	enum	ALLOWED: `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`

NAME	TYPE	DESCRIPTION
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time
published-epoch	string	An epoch timestamp indicating source publishing time
references	array	

NAME	TYPE	DESCRIPTION
severity	enum	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara_info	object	
hits	object	
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.6 GET /file/{data_id}/blocked-leaves

Fetch a list of blocked leaf files inside archive

Retrieve a list of the deepest blocked files inside an input original archive file based on the data_id of that archive file. Leaf file is the one that not having any successful extracted file inside. In case there's no blocked file, user will see an empty array. This list only returns 100 items at most.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the list of blocked leaf files.

RESPONSE MODEL - application/octet-stream

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
details	array	

NAME	TYPE	DESCRIPTION
blocked_reasons	array	
engines	array	
engine	string	Engine's name
threat_name	string	Threat's name
data_id	string	Data identifier of this blocked leaf file.
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
display_name	string	The name of this blocked leaf file.
file_type	string	The filetype using mimetype.
parent_id	string	Data identifer of the original archive file.
path	string	Full path to this blocked leaf file.
process_info	object	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
verdicts	array	
yara_info	object	
hits	array	
limit_reached	boolean	Indicates if there're indeed over 100 blocked leaf files.
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
total	integer	Total number of files in the above array.

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.7 GET /file/{data_id}/extraction-errors

Fetch top 100 of extraction errors inside an archive

Retrieve extraction errors detected while processing the archive (and any nested archives) identified by data_id. The response separates the error information for the root archive from errors found in nested archives. Field meanings:

- total: Total number of archives (root + nested) that reported extraction errors.
- root_archive: Error information about the top-level (root) archive. Omitted if no root-level error.
- nested_archives: List of nested archives that encountered extraction errors.

If no extraction errors were recorded, total will be 0 and nested_archives may be an empty array; root_archive may be absent. If the archive result was reused from a previous analysis, the API returns the extraction errors from that analysis. You can identify this case when original_data_id is set in the GET /file/{data_id} response. The first segment of each archive_path is updated to

use the name of the requested file. Available starting with MetaDefender Core 5.22.0.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	Data identifier of the original (root) archive file.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Extraction error information for the specified archive.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
total	integer	Total count of archives (root + nested) that have extraction errors.
root_archive	object	
err_category	string	High-level error category.
err_details	string	Detailed diagnostic message.
err_description	string	Human-readable description of the error.
nested_archives	array	
data_id	string	Data identifier of the nested archive.
archive_path	array	
err_category	string	High-level error category.
err_details	string	Detailed diagnostic message.
err_description	string	Human-readable description of the nested archive issue.

EXAMPLE 1:

```
"sample"
```

STATUS CODE - 400: Invalid request (e.g. malformed data_id or bad query parameters).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Description of the error.

EXAMPLE:

```
{
 "err": "Invalid query parameters"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 423: The file is still being processed; extraction errors not finalized yet.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Description of the error.

EXAMPLE:

```
{
 "err": "File is on processing. Abort"
```

```
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.8 GET /archive/{data_id}

Fetch analysis reports containing all files in archive

Retrieving Analysis Reports Using Data ID containing all files in archive.

Under extracted_files key, all the info about extracted files will be listed recursively.

Note: Using this method the following fields will not be shown compared to /file request containing extracted files:

- files_extracted_count
- files_in_archive
- first_index
- page_size
- worst_data_id

Also the "outdated_data" field will only be shown in the root archive.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Entire analysis report containing all extracted files generated by MetaDefender Core

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
coo_info	object	
final_verdict*	object	
verdict	enum	**ALLOWED:** blocked, allowed, unknown Country of Origin engine verdict. Values: - blocked: The file is considered blocked based on origin evaluation. - allowed: The file origin passed all policy checks. - unknown: Origin could not be confidently determined.
verdict_explanation	array	
full_report	object	
overview_report*	object	
company_name	string	Name of the company that created or signed the file, extracted from file metadata or digital signatures.
country_code	string	**PATTERN:** ^[A-Z]{2}$ ISO 3166-1 alpha-2 country code.
country_name	string	Full country name where the file originated, based on company information and digital signatures.
signer	string	Name of the entity that digitally signed the file, extracted from the digital signature certificate.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
data_id	string	data identifier of the requested file

NAME	TYPE	DESCRIPTION
original_data_id	`string`	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
dlp_info	`object`	
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
anonymization	`object`	
result	`string`	Result of the anonymization process.
crop_embedded_images	`object`	
result	`string`	Result of the crop embedded images process.
current_file_certainty	`string`	Describes how certain the current file hit is.
current_file_verdict	`integer`	The result for the current scanned file.
detection_policy	`object`	
result	`string`	Status of detection policy.
document_identification	`object`	
result	`string`	Result of the document identification process.
errors	`object`	
filename	`string`	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	`object`	
ccn	`object`	
display_name	`string`	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	`array`	
after	`string`	The context after the matched data.
before	`string`	The context before the matched data.
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	`integer`	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]

NAME	TYPE	DESCRIPTION
hit	`string`	The matched data.
location	`string`	The location of the hit that is found in a file.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	`boolean`	If file was redacted or not.
intentional_data_leak	`object`	
result	`string`	Result of the intentional data leak detection process.
metadata_removal	`object`	
result	`enum`	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
processing_info	`object`	
encoding	`string`	Character encoding used for processing the file.
recursive_processing	`object`	
result	`string`	Result of the recursive processing.
redact	`object`	
result	`enum`	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
result_template_hash	`string`	Hash of the result template used internally for rendering and engine/Core coordination.
scan	`object`	
result	`string`	Overall result of scanning process.
types	`object`	
ai	`object`	
display_name	`string`	Display name for AI scanning.
result	`string`	Result of AI scanning.
regex	`object`	
display_name	`string`	Display name for Regex scanning.
result	`string`	Result of Regex scanning.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss

NAME	TYPE	DESCRIPTION
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	`object`	
error_detail	`string`	Revealed detailed reason why the download failed.
progress	`number`	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [1, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`) if the download stopped unexpectedly.

NAME	TYPE	DESCRIPTION

status	`string`	Indicates download status, which could be either
		- `Downloading`
		- Check `progress` key value for actual download percentage
		``json
		"download_info": {
		"progress": 7,
		"status": "Downloading",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``
		- `Download Success`
		``json
		"download_info": {
		"status": "Download Success",
		"url": "https://secure.eicar.org/eicar.com"
		}
		``
		- `Download Failed`
		- Check `error_detail` key value for an error explanation
		``json
		"download_info": {
		"error_detail": "Connection error",
		"status": "Download Failed",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Timeout`
		- Expecting to occur when the download progress takes longer than what time window allowed in MetaDefender Core's pre-configured setting under workflow rule (under "SCAN" tab)
		``json
		"download_info": {
		"status": "Download Timeout",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Cancelled`
		- Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to
		``json
		"download_info": {
		"status": "Download Cancelled",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``

url	`string`	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extracted_files	`array`	
coo_info	`object`	
final_verdict*	`object`	
verdict	`enum`	**ALLOWED:** `blocked, allowed, unknown` Country of Origin engine verdict. Values: - blocked: The file is considered blocked based on origin evaluation. - allowed: The file origin passed all policy checks. - unknown: Origin could not be confidently determined.

NAME	TYPE	DESCRIPTION
verdict_explanation	`array`	
full_report	`object`	
overview_report*	`object`	
company_name	`string`	Name of the company that created or signed the file, extracted from file metadata or digital signatures.
country_code	`string`	**PATTERN:** `^[A-Z]{2}$` ISO 3166-1 alpha-2 country code.
country_name	`string`	Full country name where the file originated, based on company information and digital signatures.
signer	`string`	Name of the entity that digitally signed the file, extracted from the digital signature certificate.
result_template_hash*	`string`	Hash of the result template used internally for rendering and engine/Core coordination.
data_id	`string`	data identifier of the requested file
original_data_id	`string`	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
dlp_info	`object`	
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
anonymization	`object`	
result	`string`	Result of the anonymization process.
crop_embedded_images	`object`	
result	`string`	Result of the crop embedded images process.
current_file_certainty	`string`	Describes how certain the current file hit is.
current_file_verdict	`integer`	The result for the current scanned file.
detection_policy	`object`	
result	`string`	Status of detection policy.
document_identification	`object`	
result	`string`	Result of the document identification process.
errors	`object`	
filename	`string`	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	`object`	
ccn	`object`	

NAME	TYPE	DESCRIPTION
display_name	`string`	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	`array`	
after	`string`	The context after the matched data.
before	`string`	The context before the matched data.
certainty	`enum`	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	`integer`	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	`string`	The matched data.
location	`string`	The location of the hit that is found in a file.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	`boolean`	If file was redacted or not.
intentional_data_leak	`object`	
result	`string`	Result of the intentional data leak detection process.
metadata_removal	`object`	
result	`enum`	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
processing_info	`object`	
encoding	`string`	Character encoding used for processing the file.
recursive_processing	`object`	
result	`string`	Result of the recursive processing.
redact	`object`	
result	`enum`	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
result_template_hash	`string`	Hash of the result template used internally for rendering and engine/Core coordination.
scan	`object`	
result	`string`	Overall result of scanning process.

NAME	TYPE	DESCRIPTION
types	`object`	
ai	`object`	
display_name	`string`	Display name for AI scanning.
result	`string`	Result of AI scanning.
regex	`object`	
display_name	`string`	Display name for Regex scanning.
result	`string`	Result of Regex scanning.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
extracted_files	`array`	
coo_info	`object`	
final_verdict*	`object`	
verdict	`enum`	**ALLOWED:** `blocked, allowed, unknown` Country of Origin engine verdict. Values: - blocked: The file is considered blocked based on origin evaluation. - allowed: The file origin passed all policy checks. - unknown: Origin could not be confidently determined.
verdict_explanation	`array`	
full_report	`object`	
overview_report*	`object`	
company_name	`string`	Name of the company that created or signed the file, extracted from file metadata or digital signatures.
country_code	`string`	**PATTERN:**`^[A-Z]{2}$` ISO 3166-1 alpha-2 country code.
country_name	`string`	Full country name where the file originated, based on company information and digital signatures.

NAME	TYPE	DESCRIPTION
signer	string	Name of the entity that digitally signed the file, extracted from the digital signature certificate.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
data_id	string	data identifier of the requested file
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
dlp_info	object	
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
anonymization	object	
result	string	Result of the anonymization process.
crop_embedded_images	object	
result	string	Result of the crop embedded images process.
current_file_certainty	string	Describes how certain the current file hit is.
current_file_verdict	integer	The result for the current scanned file.
detection_policy	object	
result	string	Status of detection policy.
document_identification	object	
result	string	Result of the document identification process.
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	object	
ccn	object	
display_name	string	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	array	
after	string	The context after the matched data.
before	string	The context before the matched data.

NAME	TYPE	DESCRIPTION
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
intentional_data_leak	object	
result	string	Result of the intentional data leak detection process.
metadata_removal	object	
result	enum	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
processing_info	object	
encoding	string	Character encoding used for processing the file.
recursive_processing	object	
result	string	Result of the recursive processing.
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
scan	object	
result	string	Overall result of scanning process.
types	object	
ai	object	
display_name	string	Display name for AI scanning.
result	string	Result of AI scanning.
regex	object	

NAME	TYPE	DESCRIPTION
display_name	`string`	Display name for Regex scanning.
result	`string`	Result of Regex scanning.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
extraction_info	`object`	
decrypted_status	`enum`	**ALLOWED:** `Success, Failed` Indicate that decryption phase is successful or not.
err_category	`string`	Error category
err_code	`integer`	Error code
err_description	`string`	Error description
err_details	`string`	Error message
is_encrypted_file	`boolean`	Indicate if file is password-protected or not.
stop_processing_reason	`string`	The verdict of the extracted file that triggered the archive stop-processing policy of the workflow rule, e.g. `Infected`. Only present when `stopped_by_policy` is.
stopped_by_policy	`boolean`	Present (always `true`) when processing of this archive, or of an archive nested in it, was stopped early by the archive stop-processing policy of the workflow rule. Extracted files that were stopped before they finished get the `Early Exit` (81) result.
total_extraction_error_count	`integer`	Total number of extraction errors encountered.
total_reused_extraction_errors	`integer`	Indicates how many extraction errors were reused from a previous analysis instead of being detected during the current analysis. This can include errors from the archive itself or from nested archives. A value of 0 means no extraction errors were reused. Available starting with MetaDefender Core 5.22.0.
file_info	`object`	
display_name	`string`	The filename reported via `filename` header.
file_size	`integer`	Total file size in bytes.

NAME	TYPE	DESCRIPTION
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
file_type_id	string	File type identifier code.
is_skip_hash	boolean	Whether hash calculation was skipped for this file.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	

NAME	TYPE	DESCRIPTION
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	**ALLOWED:** allowed, blocked Final verdict of the file type analysis.
verdict_explanation*	array	
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	
description*	string	File type description or identifier from the detector.
detector*	string	Name of the detection engine that provided this result.
score*	integer	Confidence score of the detection result.
model_version	string	Version of the model used by the detector (if applicable).
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
insightsti_info	object	
opswatfilescan_info	object	
final_verdict	object	
blocked	boolean	Indicates if the file is blocked based on Adaptive Sandbox analysis
confidence	number	between 0 and 1 Confidence level of the verdict (0.0 to 1.0)
threat_level	number	between 0 and 1 Threat level assessment (0.0 to 1.0, where higher values indicate higher threat)
verdict	enum	**ALLOWED:** UNKNOWN, BENIGN, NO THREAT DETECTED, SUSPICIOUS, LIKELY_MALICIOUS, MALICIOUS The verdict classification from Adaptive Sandbox
overview_report	object	
report_url	string	URL to access the detailed analysis report
result_template_hash	string	Hash identifier for the result template used
post_action	object	

NAME	TYPE	DESCRIPTION
details*	array	
duration*	integer	Execution duration of the post-action in milliseconds.
name*	string	Name or identifier of the post-action.
return_value*	integer	Return value or exit code from the post-action execution.
verdict*	integer	Verdict code representing the outcome of the post-action.
total_blocked*	integer	Total number of files that were indicated to be blocked by Post Action execution.
total_executed*	integer	Total number of post-actions that were executed.
total_failed*	integer	Total number of post-actions that failed during execution.
total_post_action*	integer	Total number of post-actions configured or available.
total_success*	integer	Total number of post-actions that executed successfully.
process_info	object	
archive_handling_details	object	
extracted_files_count	integer	Number of files extracted from archive.
max_depth_reached	integer	Maximum extraction depth reached.
extraction_method	string	Method used for extraction.
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
outdated_data	array	
hash_time	integer	Time taken to calculate file hash (in milliseconds).
is_rejected_request	boolean	Indicates if the request was rejected due to policy or limits.
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
av_scan_time	integer	AV engines' processing time.
av_wait_time	integer	Time spent waiting for AV engines to become available.
cdr_storage_time	integer	Time spent storing CDR sanitized files.
cdr_time	integer	Deep CDR engine's sanitization time.
cdr_wait_time	integer	Time spent waiting for CDR engine to become available.
child_files_hash_time	integer	Time spent calculating hashes for extracted child files.
cloud_lookup_time	integer	Time spent on cloud reputation lookups.
coo_time	integer	Chain of Ownership (COO) processing time.
decrypted_time	integer	Time spent on file decryption processes.
dlp_storage_time	integer	Time spent storing DLP processing results.

NAME	TYPE	DESCRIPTION
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
extraction_wait_time	integer	Time spent waiting for extraction engine to become available.
filetype_time	integer	FileType engine's processing time.
filetype_wait_time	integer	Time spent waiting for FileType engine to become available.
fsv_time	integer	File Structure Validation engine's processing time.
fsv_wait_time	integer	Time spent waiting for File Structure Validation engine to become available.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
opswatfilescan_wait_time	integer	Time spent waiting for OPSWAT Filescan engine to become available.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * File processing's hash calculation time * External scanner (if configured) * Post action (if configured) * Other internal procssing time among components in the product
parse_dgsg_time	integer	Time spent parsing DGSG (Data Governance and Security Gateway) information.
post_action_time	integer	Time spent executing post-processing actions.
quarantine_storage_time	integer	Time spent storing files in quarantine.
reputation_time	integer	Time spent on reputation analysis and lookups.
sbom_time	integer	Time spent on Software Bill of Materials (SBOM) analysis.
vul_time	integer	Vulnerability engine's lookup time.
yara_time	integer	YARA engine's processing time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.

NAME	TYPE	DESCRIPTION
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
details	object	
action	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	**ALLOWED:** `Sanitized, Sanitized failed, Sanitized skipped` The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MD Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `blocked, allowed` The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
status	enum	**ALLOWED:** `success, failure, skipped` The status of the Deep CDR process (This field is only available for Deep CDR processing and does not apply to archive sanitization). - **Queued**: indicate the file that is currently waiting in the MD Core's queue before being sent to the Deep CDR engine. - **Processing**: indicate the file that has been sent to the Deep CDR engine and is currently being processed. - **Finished**: the Deep CDR process was finished. At this point, there are several additional states you can further check: - Check whether the file was sanitized successfully. Indicator: `post_processing.actions_ran` contains `Sanitized`. - Check whether the file failed to be sanitized. Indicator: `post_processing.actions_failed` contains `Sanitization Failed`. - Check whether the file is marked as blocked by the Deep CDR engine. Indicator: `post_processing.sanitization_details.verdict` = `blocked`.
reputation_info	object	
final_verdict*	object	
verdict*	enum	**ALLOWED:** `Known Good, Known Bad, Unknown` Reputation engine verdict. Values: - Known Good: The file has a positive reputation. - Known Bad: The file has a negative reputation. - Unknown: Reputation could not be determined.
verdict_explanation*	array	
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
sbom_info	object	
final_verdict*	object	
blocked*	boolean	Whether the container/application is blocked based on security policies.
licensing_package_counts*	object	
allowed*	integer	Number of packages with allowed licenses.
blocked*	integer	Number of packages with blocked licenses.
unknown*	integer	Number of packages with unknown licensing status.
unsupported*	integer	Number of packages with unsupported license detection.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the SBOM.
total_package_count*	integer	Total number of packages analyzed in the SBOM.
verdict*	string	Human-readable verdict summary.
verdict_explanation*	array	

NAME	TYPE	DESCRIPTION
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities.
overview_report*	object	
has_pe_info*	boolean	Indicates whether PE (Portable Executable) information is available.
is_partial_report*	boolean	Indicates if this is a partial report with incomplete data.
package_count*	integer	Total number of packages analyzed in the overview.
packages*	array	
archive*	boolean	Indicates if the package is an archive.
authors*	string	Package author(s).
ecosystem*	string	Package ecosystem or format.
group*	string	Package group classification.
hashes*	array	
file_name*	string	Name of the file.
sha256*	string	SHA-256 hash of the file.
latest*	object	
release_date*	string	Release date of the latest version.
version*	string	Latest version number.
licenses*	object	
allowed	array	
category*	string	License category classification.
id*	string	SPDX license identifier.
url*	string	URL to license details.
usage_restriction*	object	
conditions*	array	
limitations*	array	
permissions*	array	
package_name*	string	Name of the package.
package_type*	string	Type classification of the package.
package_version*	string	Version of the package.

NAME	TYPE	DESCRIPTION
release_date*	string	Package release date.
target*	string	Target file or manifest where the package was found.
uid*	string	Unique package identifier (Package URL format).
vulnerabilities*	array	
cwes*	array	
fixed_versions*	string	Versions where the vulnerability is fixed.
id*	string	Vulnerability identifier (CVE, GHSA, etc.).
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Vulnerability severity level.
source*	string	Vulnerability database source.
type*	string	Type of security issue.
vulnerability_counts*	object	
critical*	integer	
high*	integer	
low*	integer	
medium*	integer	
unknown*	integer	
scan_mode*	enum	**ALLOWED:** `container, filesystem, archive, source_code` Type of scan performed for SBOM analysis.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the overview report.
timestamp*	string	Timestamp when the overview report was generated (ISO 8601 format).
vulnerability_count*	integer	Total number of vulnerabilities found in the overview.
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities in the overview.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
bom_info	object	
final_verdict	object	
verdict	string	
result_subtrees	array	

NAME	TYPE	DESCRIPTION
cbom_info	`object`	
aibom_info	`object`	
scan_results	`object`	
current_av_result_a	`string`	The AV scan result for the current file itself (e.g., archive container) as string, excluding extracted contents. Uses the same enumeration values as scan_all_result_a
current_av_result_i	`string`	The AV scan result for the current file itself (e.g., archive container) as index, excluding extracted contents. Uses the same enumeration values as scan_all_result_i
data_id	`string`	Data ID of the requested file
progress_percentage	`integer`	Track analysis progress until reaches 100.
scan_all_result_a	`enum`	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall scan result as string

NAME	TYPE	DESCRIPTION
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	**ALLOWED:** `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained

NAME	TYPE	DESCRIPTION
severity	enum	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time

NAME	TYPE	DESCRIPTION
published-epoch	`string`	An epoch timestamp indicating source publishing time
references	`array`	
severity	`enum`	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	`integer`	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	`integer`	An OPSWAT identifier for the vulnerability
verdict	`integer`	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara_info	`object`	
hits	`object`	
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |
file_info	`object`	
display_name	`string`	The filename reported via `filename` header.
file_size	`integer`	Total file size in bytes.
file_type	`string`	The filetype using mimetype.
file_type_description	`string`	The filetype in human readable format.
file_type_id	`string`	File type identifier code.
is_skip_hash	`boolean`	Whether hash calculation was skipped for this file.
md5	`string`	File's MD5 hash.
sha1	`string`	File's SHA1 hash.
sha256	`string`	File's SHA256 Hash.
sha512	`string`	File's SHA512 Hash.
signer_infos	`array`	

NAME	TYPE	DESCRIPTION
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	**ALLOWED:** `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	array	
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.

NAME	TYPE	DESCRIPTION
other_detections	array	
description*	string	File type description or identifier from the detector.
detector*	string	Name of the detection engine that provided this result.
score*	integer	Confidence score of the detection result.
model_version	string	Version of the model used by the detector (if applicable).
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
insightsti_info	object	
opswatfilescan_info	object	
final_verdict	object	
blocked	boolean	Indicates if the file is blocked based on Adaptive Sandbox analysis
confidence	number	`between 0 and 1` Confidence level of the verdict (0.0 to 1.0)
threat_level	number	`between 0 and 1` Threat level assessment (0.0 to 1.0, where higher values indicate higher threat)
verdict	enum	**ALLOWED:** `UNKNOWN, BENIGN, NO THREAT DETECTED, SUSPICIOUS, LIKELY_MALICIOUS, MALICIOUS` The verdict classification from Adaptive Sandbox
overview_report	object	
report_url	string	URL to access the detailed analysis report
result_template_hash	string	Hash identifier for the result template used
post_action	object	
details*	array	
duration*	integer	Execution duration of the post-action in milliseconds.
name*	string	Name or identifier of the post-action.
return_value*	integer	Return value or exit code from the post-action execution.
verdict*	integer	Verdict code representing the outcome of the post-action.
total_blocked*	integer	Total number of files that were indicated to be blocked by Post Action execution.
total_executed*	integer	Total number of post-actions that were executed.
total_failed*	integer	Total number of post-actions that failed during execution.
total_post_action*	integer	Total number of post-actions configured or available.

NAME	TYPE	DESCRIPTION
total_success*	integer	Total number of post-actions that executed successfully.
process_info	object	
archive_handling_details	object	
description	string	Describes what occurred during archive extraction.
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
is_rejected_request	boolean	Indicates if the request was rejected due to configuration such as file size limits.
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
av_scan_time	integer	AV engines' processing time.
av_wait_time	integer	AV engines' wait time.
cdr_time	integer	Deep CDR engine's sanitization time.
cdr_storage_time	integer	Deep CDR engine's storage time.
cdr_wait_time	integer	Deep CDR engine's wait time.
child_files_hash_time	integer	Time elapsed for hashing child files.
cloud_lookup_time	integer	Cloud lookup processing time.
coo_time	integer	Country of Origin engine's processing time.
decrypted_time	integer	File decryption processing time.
dlp_time	integer	Proactive DLP engine's processing time.
dlp_storage_time	integer	Proactive DLP engine's storage time.
extraction_time	integer	Archive extraction engine's processing time.
extraction_wait_time	integer	Archive extraction engine's wait time.
filetype_time	integer	FileType engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
fsv_time	integer	File Structure Validation engine's processing time.
fsv_wait_time	integer	File Structure Validation engine's wait time.
opswatfilescan_time	integer	Adaptive Sandbox engine's processing time.
opswatfilescan_wait_time	integer	Adaptive Sandbox engine's wait time.

NAME	TYPE	DESCRIPTION
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * File processing's hash calculation time * External scanner (if configured) * Post action (if configured) * Other internal procssing time among components in the product
vul_time	integer	Vulnerability engine's lookup time.
wait_child_files_time	integer	Wait time for child files processing.
yara_time	integer	YARA engine's processing time.
parse_dgsg_time	integer	Digital signature analyzing time.
post_action_time	integer	Post action processing time.
quarantine_storage_time	integer	Quarantine storage processing time.
reputation_time	integer	Reputation engine's processing time.
sbom_time	integer	SBOM (Software Bill of Materials) processing time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
details	object	

NAME	TYPE	DESCRIPTION
action	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	**ALLOWED:** `Sanitized, Sanitized failed, Sanitized skipped` The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MD Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.
verdict	enum	**ALLOWED:** `blocked, allowed` The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	

NAME	TYPE	DESCRIPTION
status	enum	**ALLOWED:** `success, failure, skipped` The status of the Deep CDR process (This field is only available for Deep CDR processing and does not apply to archive sanitization). - **Queued**: indicate the file that is currently waiting in the MD Core's queue before being sent to the Deep CDR engine. - **Processing**: indicate the file that has been sent to the Deep CDR engine and is currently being processed. - **Finished**: the Deep CDR process was finished. At this point, there are several additional states you can further check: - Check whether the file was sanitized successfully. Indicator: `post_processing.actions_ran` contains `Sanitized`. - Check whether the file failed to be sanitized. Indicator: `post_processing.actions_failed` contains `Sanitization Failed`. - Check whether the file is marked as blocked by the Deep CDR engine. Indicator: `post_processing.sanitization_details.verdict` = `blocked`.
reputation_info	object	
final_verdict*	object	
verdict*	enum	**ALLOWED:** `Known Good, Known Bad, Unknown` Reputation engine verdict. Values: - Known Good: The file has a positive reputation. - Known Bad: The file has a negative reputation. - Unknown: Reputation could not be determined.
verdict_explanation*	array	
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
sbom_info	object	
final_verdict*	object	
blocked*	boolean	Whether the container/application is blocked based on security policies.
licensing_package_counts*	object	
allowed*	integer	Number of packages with allowed licenses.
blocked*	integer	Number of packages with blocked licenses.
unknown*	integer	Number of packages with unknown licensing status.
unsupported*	integer	Number of packages with unsupported license detection.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the SBOM.
total_package_count*	integer	Total number of packages analyzed in the SBOM.
verdict*	string	Human-readable verdict summary.
verdict_explanation*	array	
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.

NAME	TYPE	DESCRIPTION
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities.
overview_report*	object	
has_pe_info*	boolean	Indicates whether PE (Portable Executable) information is available.
is_partial_report*	boolean	Indicates if this is a partial report with incomplete data.
package_count*	integer	Total number of packages analyzed in the overview.
packages*	array	
archive*	boolean	Indicates if the package is an archive.
authors*	string	Package author(s).
ecosystem*	string	Package ecosystem or format.
group*	string	Package group classification.
hashes*	array	
file_name*	string	Name of the file.
sha256*	string	SHA-256 hash of the file.
latest*	object	
release_date*	string	Release date of the latest version.
version*	string	Latest version number.
licenses*	object	
allowed	array	
category*	string	License category classification.
id*	string	SPDX license identifier.
url*	string	URL to license details.
usage_restriction*	object	
conditions*	array	
limitations*	array	
permissions*	array	
package_name*	string	Name of the package.
package_type*	string	Type classification of the package.
package_version*	string	Version of the package.
release_date*	string	Package release date.
target*	string	Target file or manifest where the package was found.
uid*	string	Unique package identifier (Package URL format).
vulnerabilities*	array	
cwes*	array	

NAME	TYPE	DESCRIPTION
fixed_versions*	string	Versions where the vulnerability is fixed.
id*	string	Vulnerability identifier (CVE, GHSA, etc.).
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Vulnerability severity level.
source*	string	Vulnerability database source.
type*	string	Type of security issue.
vulnerability_counts*	object	
critical*	integer	
high*	integer	
low*	integer	
medium*	integer	
unknown*	integer	
scan_mode*	enum	**ALLOWED:** container, filesystem, archive, source_code Type of scan performed for SBOM analysis.
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Overall security severity level of the overview report.
timestamp*	string	Timestamp when the overview report was generated (ISO 8601 format).
vulnerability_count*	integer	Total number of vulnerabilities found in the overview.
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities in the overview.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
bom_info	object	
final_verdict	object	
verdict	string	
result_subtrees	array	
cbom_info	object	
aibom_info	object	
scan_results	object	
current_av_result_a	string	The AV scan result for the current file itself (e.g., archive container) as string, excluding extracted contents. Uses the same enumeration values as scan_all_result_a

NAME	TYPE	DESCRIPTION
current_av_result_i	string	The AV scan result for the current file itself (e.g., archive container) as index, excluding extracted contents. Uses the same enumeration values as scan_all_result_i
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).

NAME	TYPE	DESCRIPTION
scan_details	`object`	
ClamAV	`object`	
def_time	`string`	The database definition time for this engine
eng_id	`string`	The unique identification string for the engine
location	`string`	Where this engine is deployed (local/remote).
scan_result_i	`integer`	Scan result as index in the Processing Results table
scan_time	`integer`	The time elapsed during scan with this engine (in milliseconds).
threat_found	`string`	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	`integer`	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	`object`	
result	`object`	
code	`integer`	The result code for vulnerability check, 0 means a successful check
hash	`string`	The file's SHA1 hash value
method	`enum`	**ALLOWED:** `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	`string`	Timestamp of the request issued
timing	`integer`	The vulnerability check's duration in milliseconds
detected_product	`object`	
has_kb	`boolean`	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	`boolean`	Indicates whether any vulnerabilities have been associated with the particular product
is_current	`boolean`	True if this product's patch level is current, defaults to true
product	`object`	
id	`integer`	The OPSWAT product id
name	`string`	The product name
remediation_link	`string`	A link where product updates or patches can be obtained
severity	`enum`	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	`string`	Product signature descriptor
signature	`integer`	OPSWAT signature id

NAME	TYPE	DESCRIPTION
vendor	`object`	
id	`integer`	The OPSWAT vendor id
name	`string`	The vendor name
version	`string`	The installed product version
version_data	`object`	
count_behind	`integer`	The number of patches behind of the installed product
feed_id	`integer`	The remote feed ID used to determine patch level
version	`string`	The current version of the product in the remote feed
vulnerabilites	`array`	
description	`string`	A text description of the specific vulnerability
details	`object`	
cpe	`string`	A CPE product reference
cve	`string`	A CVE identification string
cvss	`object`	
access-complexity	`string`	A CVSS access-complexity descriptor
access-vector	`string`	A CVSS access-vector descriptor
authentication	`string`	A CVSS authentication descriptor
availability-impact	`string`	A CVSS availability impact descriptor
confidentiality-impact	`string`	A CVSS confidentiality impact descriptor
generated-on-epoch	`string`	An epoch timestamp indicating CVSS generation time
integrity-impact	`string`	A CVSS integrity impact descriptor
score	`string`	A CVSS 10-point severity score
source	`string`	A CVSS source descriptor
cwe	`string`	A CWE group identification string
last_modified_epoch	`string`	An epoch timestamp indicating source last update time
published-epoch	`string`	An epoch timestamp indicating source publishing time
references	`array`	
severity	`enum`	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`

NAME	TYPE	DESCRIPTION
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara_info	object	
hits	object	
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
file_type_id	string	File type identifier code.
is_skip_hash	boolean	Whether hash calculation was skipped for this file.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.

NAME	TYPE	DESCRIPTION
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	**ALLOWED:** `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	array	
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	
description*	string	File type description or identifier from the detector.
detector*	string	Name of the detection engine that provided this result.
score*	integer	Confidence score of the detection result.
model_version	string	Version of the model used by the detector (if applicable).

NAME	TYPE	DESCRIPTION
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
insightsti_info	object	
opswatfilescan_info	object	
final_verdict	object	
blocked	boolean	Indicates if the file is blocked based on Adaptive Sandbox analysis
confidence	number	`between 0 and 1` Confidence level of the verdict (0.0 to 1.0)
threat_level	number	`between 0 and 1` Threat level assessment (0.0 to 1.0, where higher values indicate higher threat)
verdict	enum	**ALLOWED:** `UNKNOWN, BENIGN, NO THREAT DETECTED, SUSPICIOUS, LIKELY_MALICIOUS, MALICIOUS` The verdict classification from Adaptive Sandbox
overview_report	object	
report_url	string	URL to access the detailed analysis report
result_template_hash	string	Hash identifier for the result template used
post_action	object	
details*	array	
duration*	integer	Execution duration of the post-action in milliseconds.
name*	string	Name or identifier of the post-action.
return_value*	integer	Return value or exit code from the post-action execution.
verdict*	integer	Verdict code representing the outcome of the post-action.
total_blocked*	integer	Total number of files that were indicated to be blocked by Post Action execution.
total_executed*	integer	Total number of post-actions that were executed.
total_failed*	integer	Total number of post-actions that failed during execution.
total_post_action*	integer	Total number of post-actions configured or available.
total_success*	integer	Total number of post-actions that executed successfully.
process_info	object	
archive_handling_details	object	
description	string	Describes what occurred during archive extraction.
blocked_reason	string	Provides the reason why the file is blocked (if so).

NAME	TYPE	DESCRIPTION
blocked_reasons	array	
allowlisted_reason	string	What allowlisted the file. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Blocklisted`.
blocklisted_reason	string	What blocklisted the file, one of `By File Type`, `By File Size`, `By File Name`, `By MIME Type`, `Skiplist - Hash`, `By Hash`, `Unsupported Sanitization File Type`, `Prevent Data Leak Unsupported File Type`, `Password Protection Required`. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Allowlisted`.
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
is_rejected_request	boolean	Indicates if the request was rejected due to configuration such as file size limits.
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
av_scan_time	integer	AV engines' processing time.
av_wait_time	integer	AV engines' wait time.
cdr_time	integer	Deep CDR engine's sanitization time.
cdr_storage_time	integer	Deep CDR engine's storage time.
cdr_wait_time	integer	Deep CDR engine's wait time.
child_files_hash_time	integer	Time elapsed for hashing child files.
cloud_lookup_time	integer	Cloud lookup processing time.
coo_time	integer	Country of Origin engine's processing time.
decrypted_time	integer	File decryption processing time.
dlp_time	integer	Proactive DLP engine's processing time.
dlp_storage_time	integer	Proactive DLP engine's storage time.
extraction_time	integer	Archive extraction engine's processing time.
extraction_wait_time	integer	Archive extraction engine's wait time.
filetype_time	integer	FileType engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
fsv_time	integer	File Structure Validation engine's processing time.
fsv_wait_time	integer	File Structure Validation engine's wait time.
opswatfilescan_time	integer	Adaptive Sandbox engine's processing time.
opswatfilescan_wait_time	integer	Adaptive Sandbox engine's wait time.

NAME	TYPE	DESCRIPTION
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * File processing's hash calculation time * External scanner (if configured) * Post action (if configured) * Other internal procssing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
post_action_time	integer	Post action processing time.
quarantine_storage_time	integer	Quarantine storage processing time.
reputation_time	integer	Reputation engine's processing time.
sbom_time	integer	SBOM (Software Bill of Materials) processing time.
vul_time	integer	Vulnerability engine's lookup time.
wait_child_files_time	integer	Wait time for child files processing.
yara_time	integer	YARA engine's processing time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	ALLOWED: `sanitized, removed` The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
details	object	

NAME	TYPE	DESCRIPTION
action	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	**ALLOWED:** `Sanitized, Sanitized failed, Sanitized skipped` The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MD Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.
verdict	enum	**ALLOWED:** `blocked, allowed` The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	

NAME	TYPE	DESCRIPTION
status	enum	**ALLOWED:** `success, failure, skipped` The status of the Deep CDR process (This field is only available for Deep CDR processing and does not apply to archive sanitization). - **Queued**: indicate the file that is currently waiting in the MD Core's queue before being sent to the Deep CDR engine. - **Processing**: indicate the file that has been sent to the Deep CDR engine and is currently being processed. - **Finished**: the Deep CDR process was finished. At this point, there are several additional states you can further check: - Check whether the file was sanitized successfully. Indicator: `post_processing.actions_ran` contains `Sanitized`. - Check whether the file failed to be sanitized. Indicator: `post_processing.actions_failed` contains `Sanitization Failed`. - Check whether the file is marked as blocked by the Deep CDR engine. Indicator: `post_processing.sanitization_details.verdict` = `blocked`.
reputation_info	object	
final_verdict*	object	
verdict*	enum	**ALLOWED:** `Known Good, Known Bad, Unknown` Reputation engine verdict. Values: - Known Good: The file has a positive reputation. - Known Bad: The file has a negative reputation. - Unknown: Reputation could not be determined.
verdict_explanation*	array	
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
sbom_info	object	
final_verdict*	object	
blocked*	boolean	Whether the container/application is blocked based on security policies.
licensing_package_counts*	object	
allowed*	integer	Number of packages with allowed licenses.
blocked*	integer	Number of packages with blocked licenses.
unknown*	integer	Number of packages with unknown licensing status.
unsupported*	integer	Number of packages with unsupported license detection.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the SBOM.
total_package_count*	integer	Total number of packages analyzed in the SBOM.
verdict*	string	Human-readable verdict summary.
verdict_explanation*	array	
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.

NAME	TYPE	DESCRIPTION
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities.
overview_report*	object	
has_pe_info*	boolean	Indicates whether PE (Portable Executable) information is available.
is_partial_report*	boolean	Indicates if this is a partial report with incomplete data.
package_count*	integer	Total number of packages analyzed in the overview.
packages*	array	
archive*	boolean	Indicates if the package is an archive.
authors*	string	Package author(s).
ecosystem*	string	Package ecosystem or format.
group*	string	Package group classification.
hashes*	array	
file_name*	string	Name of the file.
sha256*	string	SHA-256 hash of the file.
latest*	object	
release_date*	string	Release date of the latest version.
version*	string	Latest version number.
licenses*	object	
allowed	array	
category*	string	License category classification.
id*	string	SPDX license identifier.
url*	string	URL to license details.
usage_restriction*	object	
conditions*	array	
limitations*	array	
permissions*	array	
package_name*	string	Name of the package.
package_type*	string	Type classification of the package.
package_version*	string	Version of the package.
release_date*	string	Package release date.
target*	string	Target file or manifest where the package was found.
uid*	string	Unique package identifier (Package URL format).
vulnerabilities*	array	
cwes*	array	

NAME	TYPE	DESCRIPTION
fixed_versions*	string	Versions where the vulnerability is fixed.
id*	string	Vulnerability identifier (CVE, GHSA, etc.).
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Vulnerability severity level.
source*	string	Vulnerability database source.
type*	string	Type of security issue.
vulnerability_counts*	object	
critical*	integer	
high*	integer	
low*	integer	
medium*	integer	
unknown*	integer	
scan_mode*	enum	**ALLOWED:** container, filesystem, archive, source_code Type of scan performed for SBOM analysis.
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Overall security severity level of the overview report.
timestamp*	string	Timestamp when the overview report was generated (ISO 8601 format).
vulnerability_count*	integer	Total number of vulnerabilities found in the overview.
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities in the overview.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
bom_info	object	
final_verdict	object	
verdict	string	
result_subtrees	array	
cbom_info	object	
aibom_info	object	
scan_results	object	
current_av_result_a	string	The AV scan result for the current file itself (e.g., archive container) as string, excluding extracted contents. Uses the same enumeration values as scan_all_result_a

NAME	TYPE	DESCRIPTION
current_av_result_i	string	The AV scan result for the current file itself (e.g., archive container) as index, excluding extracted contents. Uses the same enumeration values as scan_all_result_i
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).

NAME	TYPE	DESCRIPTION
scan_details	`object`	
ClamAV	`object`	
def_time	`string`	The database definition time for this engine
eng_id	`string`	The unique identification string for the engine
location	`string`	Where this engine is deployed (local/remote).
scan_result_i	`integer`	Scan result as index in the Processing Results table
scan_time	`integer`	The time elapsed during scan with this engine (in milliseconds).
threat_found	`string`	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	`integer`	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	`object`	
result	`object`	
code	`integer`	The result code for vulnerability check, 0 means a successful check
hash	`string`	The file's SHA1 hash value
method	`enum`	**ALLOWED:** `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	`string`	Timestamp of the request issued
timing	`integer`	The vulnerability check's duration in milliseconds
detected_product	`object`	
has_kb	`boolean`	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	`boolean`	Indicates whether any vulnerabilities have been associated with the particular product
is_current	`boolean`	True if this product's patch level is current, defaults to true
product	`object`	
id	`integer`	The OPSWAT product id
name	`string`	The product name
remediation_link	`string`	A link where product updates or patches can be obtained
severity	`enum`	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	`string`	Product signature descriptor
signature	`integer`	OPSWAT signature id

NAME	TYPE	DESCRIPTION
vendor	`object`	
id	`integer`	The OPSWAT vendor id
name	`string`	The vendor name
version	`string`	The installed product version
version_data	`object`	
count_behind	`integer`	The number of patches behind of the installed product
feed_id	`integer`	The remote feed ID used to determine patch level
version	`string`	The current version of the product in the remote feed
vulnerabilites	`array`	
description	`string`	A text description of the specific vulnerability
details	`object`	
cpe	`string`	A CPE product reference
cve	`string`	A CVE identification string
cvss	`object`	
access-complexity	`string`	A CVSS access-complexity descriptor
access-vector	`string`	A CVSS access-vector descriptor
authentication	`string`	A CVSS authentication descriptor
availability-impact	`string`	A CVSS availability impact descriptor
confidentiality-impact	`string`	A CVSS confidentiality impact descriptor
generated-on-epoch	`string`	An epoch timestamp indicating CVSS generation time
integrity-impact	`string`	A CVSS integrity impact descriptor
score	`string`	A CVSS 10-point severity score
source	`string`	A CVSS source descriptor
cwe	`string`	A CWE group identification string
last_modified_epoch	`string`	An epoch timestamp indicating source last update time
published-epoch	`string`	An epoch timestamp indicating source publishing time
references	`array`	
severity	`enum`	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	`integer`	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown

NAME	TYPE	DESCRIPTION
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara_info	object	
hits	object	
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.9 GET /hash/{md5|sha1|sha256|sha512}

Fetch Analysis Result By Hash

Retrieve analysis result by hash

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*md5|sha1|sha256|sha512	string		Hash value to search. This can be md5, sha1, sha256, sha512

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer		The first item order in the list child files of archive file
size	integer		The number of items to be fetched next, counting from the item order indicated in first header

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
selfonly	boolean		Useful to archive hash lookup. Allow specifying to only perform hash lookup against the original archive file self only, and skip searching all child files result within the original archive. Default value is false.
timerange	integer		Optional. Scoping down the recent number of hours that hash lookup task should start from till now, instead of searching the entire scan history in MetaDefender Core database. Default value is 0. That means no time scope.
include-inprogress	boolean		False (default): API will return "Not Found" if the verdict is in progress. True: If the queried hash has a completed processing result before, API will return the completed processing result. If this hash doesn't have any completed processing result, API will return this In-progress result.

RESPONSE

STATUS CODE - 200: Get information of file

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
aicontentinspector_info	`object`	
final_verdict*	`object`	
verdict	`enum`	**ALLOWED:** `Not Detected, Detected, Uncertain` AI Content Inspector engine verdict. Values: - Detected: AI-generated or manipulated content was detected. - Not Detected: No AI-generated or manipulated content was found. - Uncertain: Content could not be confidently classified.
verdict_explanation	`array`	
output_images	`array`	
image_alias	`string`	Alias identifier for the output image, used to retrieve the image via the API.
result_description*	`object`	
result_template_hash*	`string`	Hash of the result template used internally for AI Content Inspector rendering and coordination.
coo_info	`object`	
final_verdict*	`object`	
verdict	`enum`	**ALLOWED:** `blocked, allowed, unknown` Country of Origin engine verdict. Values: - blocked: The file is considered blocked based on origin evaluation. - allowed: The file origin passed all policy checks. - unknown: Origin could not be confidently determined.
verdict_explanation	`array`	
full_report	`object`	
overview_report*	`object`	
company_name	`string`	Name of the company that created or signed the file, extracted from file metadata or digital signatures.
country_code	`string`	**PATTERN:**`^[A-Z]{2}$` ISO 3166-1 alpha-2 country code.
country_name	`string`	Full country name where the file originated, based on company information and digital signatures.
signer	`string`	Name of the entity that digitally signed the file, extracted from the digital signature certificate.
result_template_hash*	`string`	Hash of the result template used internally for rendering and engine/Core coordination.
deflection_info	`object`	
final_verdict	`object`	
verdict	`enum`	**ALLOWED:** `clean, malicious, unknown, Cancelled` Deflection engine verdict. Values: - clean: the file is considered clean. - malicious: the file is considered malicious. - unknown: the engine could not confidently classify the file. - Cancelled: processing was cancelled before a verdict was produced.

NAME	TYPE	DESCRIPTION
confidence	integer	between 0 and 100 Confidence of the verdict, as a percentage (0-100).
detail	object	
data_id	string	data identifier of the analyzed file.
file_type	string	Detected file type of the analyzed file.
result_template_hash	string	Hash of the result template used internally for Deflection rendering and coordination.
data_id	string	data identifier of the requested file
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
dlp_info	object	
certainty	enum	ALLOWED: Very Low, Low, Medium, High, Very High Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
anonymization	object	
result	string	Result of the anonymization process.
crop_embedded_images	object	
result	string	Result of the crop embedded images process.
current_file_certainty	string	Describes how certain the current file hit is.
current_file_verdict	integer	The result for the current scanned file.
detection_policy	object	
result	string	Status of detection policy.
document_identification	object	
result	string	Result of the document identification process.
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's worflow rule)
hits	object	
ccn	object	
display_name	string	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	array	
after	string	The context after the matched data.
before	string	The context before the matched data.

NAME	TYPE	DESCRIPTION
certainty	enum	**ALLOWED:** `Very Low, Low, Medium, High, Very High` The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
intentional_data_leak	object	
result	string	Result of the intentional data leak detection process.
metadata_removal	object	
result	enum	**ALLOWED:** `removed, not removed, failed to remove` Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
processing_info	object	
encoding	string	Character encoding used for processing the file.
recursive_processing	object	
result	string	Result of the recursive processing.
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
scan	object	
result	string	Overall result of scanning process.
types	object	
ai	object	
display_name	string	Display name for AI scanning.
result	string	Result of AI scanning.
regex	object	

NAME	TYPE	DESCRIPTION
display_name	`string`	Display name for Regex scanning.
result	`string`	Result of Regex scanning.
severity	`enum`	**ALLOWED:** `0, 1` (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	`enum`	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	`object`	
result	`enum`	**ALLOWED:** `added, not added, failed to add` Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	`object`	
error_detail	`string`	Revealed detailed reason why the download failed.
progress	`number`	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [1, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`) if the download stopped unexpectedly.

NAME	TYPE	DESCRIPTION

status	`string`	Indicates download status, which could be either
		- `Downloading`
		- Check `progress` key value for actual download percentage
		``json
		"download_info": {
		"progress": 7,
		"status": "Downloading",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``
		- `Download Success`
		``json
		"download_info": {
		"status": "Download Success",
		"url": "https://secure.eicar.org/eicar.com"
		}
		``
		- `Download Failed`
		- Check `error_detail` key value for an error explanation
		``json
		"download_info": {
		"error_detail": "Connection error",
		"status": "Download Failed",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Timeout`
		- Expecting to occur when the download progress takes longer than what time window allowed in MetaDefender Core's pre-configured setting under workflow rule (under "SCAN" tab)
		``json
		"download_info": {
		"status": "Download Timeout",
		"url": "http://192.168.200.97:8080/2gb.zip"
		}
		``
		- `Download Cancelled`
		- Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to
		``json
		"download_info": {
		"status": "Download Cancelled",
		"url": "http://192.168.200.97:8080/5gb.zip"
		}
		``

url	`string`	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	`object`	
decrypted_status	`enum`	**ALLOWED:** `Success, Failed` Indicate that decryption phase is successful or not.
err_category	`string`	Error category
err_code	`integer`	Error code
err_description	`string`	Error description
err_details	`string`	Error message

NAME	TYPE	DESCRIPTION
is_encrypted_file	boolean	Indicate if file is password-protected or not.
stop_processing_reason	string	The verdict of the extracted file that triggered the archive stop-processing policy of the workflow rule, e.g. `Infected`. Only present when `stopped_by_policy` is.
stopped_by_policy	boolean	Present (always `true`) when processing of this archive, or of an archive nested in it, was stopped early by the archive stop-processing policy of the workflow rule. Extracted files that were stopped before they finished get the `Early Exit` (81) result.
total_extraction_error_count	integer	Total number of extraction errors encountered.
total_reused_extraction_errors	integer	Indicates how many extraction errors were reused from a previous analysis instead of being detected during the current analysis. This can include errors from the archive itself or from nested archives. A value of 0 means no extraction errors were reused. Available starting with MetaDefender Core 5.22.0.
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
file_type_id	string	File type identifier code.
is_skip_hash	boolean	Whether hash calculation was skipped for this file.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)

NAME	TYPE	DESCRIPTION
filestructurevalidation_info	`object`	
final_verdict	`object`	
verdict	`enum`	ALLOWED: `allowed, blocked, cancelled` * `allowed` - the file structure was accepted * `blocked` - the file structure was rejected * `cancelled` - processing was aborted before a verdict was produced
verdict_explanation	`array`	
err_code	`integer`	Error or limit code, returned on a failure or an output-object limit and only then - a determinate result carries `final_verdict` instead. Codes not listed below can be passed through from the engine. * `1` - File Structure Validation failed (engine error) * `5` - File Structure Validation timed out * `9` - Global timeout exceeded * `10` - Exceeded structural depth (output-object nesting limit) * `11` - Exceeded structural object count (output-object count limit) * `12` - Exceeded structural object size (output-object size limit)
err_category	`enum`	ALLOWED: `Engine error, Timeout, Global Timeout Exceeded, Output object limit exceeded` Category of the reported `err_code`. * `Engine error` - code 1 and any unrecognised code * `Timeout` - code 5 * `Global Timeout Exceeded` - code 9 * `Output object limit exceeded` - codes 10, 11 and 12
err_details	`string`	Human readable description of the error or limit. Defaults to the action name matching `err_code` when the engine does not supply a more specific message.
process_details	`object`	
description	`string`	Summary of what the engine did with the file.
skipped_reason	`string`	Reason why validation actions were not applied to the detected objects. `analysis_mode` means FSV only reported the file structure without acting on it.
failure_category	`string`	Engine-supplied failure category, for example `invalid file structure` or `failed to sanitize`.
details	`array`	
action	`string`	Action FSV applied to the objects in this group, for example `none` when it only reported them, or `extracted` when it pulled them out into their own files.
count	`integer`	Number of objects of this kind covered by the entry. It is a multiplicity and is not always `1`, so the number of detected objects is the sum of `count` across the array, not the length of the array.
object_name	`string`	Kind of the detected object.
file_name	`string`	Path of the internal part the object was found in. Present for objects that map to a named part of the container.
result_of_action	`string`	Outcome of `action` for this group.
object_sha256	`array`	
object_details	`array`	
ONE:OF	`object`	

NAME	TYPE	DESCRIPTION
prop0	string	The detected object as the engine found it in the container.
OPTION:1	object	
object_detail	string	Identifier of the object - a target path, a URL, or similar.
structural_data	array	
target_mode	string	Whether the object points inside the container or outside of it. The engine is not consistent about capitalisation - both `internal` and `External` occur - so compare case-insensitively.
file_type	string	Detected type of the embedded object.
mime_type	string	MIME type of the embedded object.
image_hash	string	SHA256 of the embedded image.
binary_object_details	array	
binary_object_encoding	string	Encoding applied to every entry of `binary_object_details`.
result_template_hash	string	Hash of the result template used internally for rendering and engine/Core coordination.
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	ALLOWED: `allowed, blocked` Final verdict of the file type analysis.
verdict_explanation*	array	
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	
description*	string	File type description or identifier from the detector.

NAME	TYPE	DESCRIPTION
detector*	string	Name of the detection engine that provided this result.
score*	integer	Confidence score of the detection result.
model_version	string	Version of the model used by the detector (if applicable).
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
fsv_output_files	object	
first_index	integer	Index of the first listed object, i.e. the offset the page starts at (driven by the `first` query parameter, default `0`).
page_size	integer	Number of objects the response lists information about (driven by the `size` query parameter, default `50`).
fsv_output_files_count	integer	Number of output objects dispatched directly out of this file. It counts what was analyzed, not what was detected - an output-object limit caps it without marking the list as truncated.
total_fsv_output_files	integer	Number of objects in the whole FSV output subtree of this file, i.e. including the objects extracted out of the extracted objects.
files_in_fsv_output	array	
data_id	string	data identifier of the extracted object. Its own analysis result carries `parent_type: fsv_output` and `parent_data_id` pointing back to this file.
object_type	string	Kind of the embedded object FSV extracted, as reported by the engine (for example `image`, `hyperlink`, `macro`).
object_name	string	Name the engine gave to the embedded object.
display_name	string	Name the extracted object is analyzed under.
file_size	integer	Size of the extracted object in bytes.
file_type	string	MIME type of the extracted object.
file_type_description	string	File type description of the extracted object.
file_type_id	string	File type ID of the extracted object.
type_category	array	
detected_by	integer	Number of engines that reported a threat in the extracted object.
scanned_with	integer	Number of engines the extracted object was scanned with.
progress_percentage	integer	Percentage of processing completed for the extracted object (from 1-100).

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall analysis result of the extracted object as string.
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall analysis result of the extracted object as index in the Processing Results table.
files_extracted_count	integer	Number of files extracted out of this object, when the object is itself a container. Omitted otherwise.
original_data_id	string	data identifier of the original analysis this result was reused from. Only present when the result was served from an earlier analysis.
total_reused_files	integer	Number of this object's children that were served from an earlier analysis. Omitted when nothing was reused.
instance_name	string	Name of the instance that processed the object. Only present when MetaDefender Core runs in shared or instance database mode.
process_info	object	

NAME	TYPE	DESCRIPTION
blocked_reason	`string`	Provides the reason why the extracted object is blocked. Only present when `result` is `Blocked`.
blocked_reasons	`array`	
rejected_reason	`string`	Why the request was rejected. Only present when `is_rejected_request` is true.
allowlisted_reason	`string`	What allowlisted the object, when it was allowlisted.
blocklisted_reason	`string`	What blocklisted the object, when it was blocklisted.
is_rejected_request	`boolean`	Indicates if the request was rejected due to configuration such as file size limits. Only present once processing completed.
progress_percentage	`integer`	Percentage of processing completed (from 1-100).
result	`string`	The final result of processing the extracted object (Allowed / Blocked / Processing).
verdicts	`array`	
post_processing	`object`	
actions_failed	`string`	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	`string`	List of successful actions, separated by "|". Empty string if otherwise.
insightsti_info	`object`	
opswatfilescan_info	`object`	
final_verdict	`object`	
blocked	`boolean`	Indicates if the file is blocked based on Adaptive Sandbox analysis
confidence	`number`	`between 0 and 1` Confidence level of the verdict (0.0 to 1.0)
threat_level	`number`	`between 0 and 1` Threat level assessment (0.0 to 1.0, where higher values indicate higher threat)
verdict	`enum`	**ALLOWED:** `UNKNOWN, BENIGN, NO THREAT DETECTED, SUSPICIOUS, LIKELY_MALICIOUS, MALICIOUS` The verdict classification from Adaptive Sandbox
overview_report	`object`	
report_url	`string`	URL to access the detailed analysis report
result_template_hash	`string`	Hash identifier for the result template used
parent_type	`enum`	**ALLOWED:** `archive, batch, fsv_output` How this file was produced from `parent_data_id`. * `archive` - the file was extracted from an archive or another compound file * `batch` - the file was submitted as part of a batch * `fsv_output` - the file is an embedded object the File Structure Validation engine extracted from its parent, and it is listed in the parent's `fsv_output_files`
post_action	`object`	
details*	`array`	
duration*	`integer`	Execution duration of the post-action in milliseconds.

NAME	TYPE	DESCRIPTION
name*	string	Name or identifier of the post-action.
return_value*	integer	Return value or exit code from the post-action execution.
verdict*	integer	Verdict code representing the outcome of the post-action.
total_blocked*	integer	Total number of files that were indicated to be blocked by Post Action execution.
total_executed*	integer	Total number of post-actions that were executed.
total_failed*	integer	Total number of post-actions that failed during execution.
total_post_action*	integer	Total number of post-actions configured or available.
total_success*	integer	Total number of post-actions that executed successfully.
process_info	object	
archive_handling_details	object	
description	string	Describes what occurred during archive extraction.
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
allowlisted_reason	string	What allowlisted the file. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Blocklisted`.
blocklisted_reason	string	What blocklisted the file, one of `By File Type`, `By File Size`, `By File Name`, `By MIME Type`, `Skiplist - Hash`, `By Hash`, `Unsupported Sanitization File Type`, `Prevent Data Leak Unsupported File Type`, `Password Protection Required`. Only present when the result is `Allowlisted` (7) or `Blocklisted` (8); empty for `Allowlisted`.
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
is_rejected_request	boolean	Indicates if the request was rejected due to configuration such as file size limits.
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
aigcd_time	integer	AI Content Inspector engine's processing time.
aigcd_wait_time	integer	AI Content Inspector engine's wait time.
av_scan_time	integer	AV engines' processing time.
av_wait_time	integer	AV engines' wait time.
cdr_time	integer	Deep CDR engine's sanitization time.
cdr_storage_time	integer	Deep CDR engine's storage time.
cdr_wait_time	integer	Deep CDR engine's wait time.
child_files_hash_time	integer	Time elapsed for hashing child files.
cloud_lookup_time	integer	Cloud lookup processing time.
coo_time	integer	Country of Origin engine's processing time.

NAME	TYPE	DESCRIPTION
decrypted_time	integer	File decryption processing time.
dlp_time	integer	Proactive DLP engine's processing time.
dlp_storage_time	integer	Proactive DLP engine's storage time.
extraction_time	integer	Archive extraction engine's processing time.
extraction_wait_time	integer	Archive extraction engine's wait time.
filetype_time	integer	FileType engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
fsv_time	integer	File Structure Validation engine's processing time.
fsv_wait_time	integer	File Structure Validation engine's wait time.
opswatfilescan_time	integer	Adaptive Sandbox engine's processing time.
opswatfilescan_wait_time	integer	Adaptive Sandbox engine's wait time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
post_action_time	integer	Post action processing time.
quarantine_storage_time	integer	Quarantine storage processing time.
reputation_time	integer	Reputation engine's processing time.
sbom_time	integer	SBOM (Software Bill of Materials) processing time.
vul_time	integer	Vulnerability engine's lookup time.
wait_child_files_time	integer	Wait time for child files processing.
yara_time	integer	YARA engine's processing time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.

NAME	TYPE	DESCRIPTION
converted_destination	`string`	Contains the name of the sanitized file.
converted_to	`string`	Contains target type name of sanitization.
copy_move_destination	`string`	Contains target type name of sanitization.
sanitization_details	`object`	
cdr_wait_time	`integer`	The time in milliseconds that the CDR process took to complete.
description	`string`	Action was successful or not.
details	`array`	
action*	`enum`	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	`integer`	The number of objects that were sanitized/removed.
details	`object`	
action	`enum`	**ALLOWED:** `sanitized, removed` The type of action that was performed
count	`integer`	The number of objects that were sanitized/removed.
object_details	`array`	
object_name	`string`	The object type that was sanitized/removed.
description	`string`	Action was successful or not.
file_name	`string`	If an embedded file was sanitized.
object_details	`array`	
object_name*	`string`	The object type that was sanitized/removed.
failure_category	`string`	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	`enum`	**ALLOWED:** `Sanitized, Sanitized failed, Sanitized skipped` The result of the CDR process. - **Sanitized**: the file was successfully sanitized. - **Sanitized failed**: the file could not be sanitized due to an error during the process. - **Sanitized skipped**: the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	`string`	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MD Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	`object`	
file_size	`integer`	Size of sanitized file in bytes.
sha256	`string`	SHA256 hash of sanitized file.

NAME	TYPE	DESCRIPTION
verdict	enum	**ALLOWED:** `blocked, allowed` The verdict of the CDR process. - **blocked**: the file is recommended for blocking by Deep CDR. - **allowed**: the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
status	enum	**ALLOWED:** `success, failure, skipped` The status of the Deep CDR process (This field is only available for Deep CDR processing and does not apply to archive sanitization). - **Queued**: indicate the file that is currently waiting in the MD Core's queue before being sent to the Deep CDR engine. - **Processing**: indicate the file that has been sent to the Deep CDR engine and is currently being processed. - **Finished**: the Deep CDR process was finished. At this point, there are several additional states you can further check: - Check whether the file was sanitized successfully. Indicator: `post_processing.actions_ran` contains `Sanitized`. - Check whether the file failed to be sanitized. Indicator: `post_processing.actions_failed` contains `Sanitization Failed`. - Check whether the file is marked as blocked by the Deep CDR engine. Indicator: `post_processing.sanitization_details.verdict` = `blocked`.
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
engine_results	object	
sanitization_result	enum	**ALLOWED:** `Success, Failed, Not Run` The overall sanitization result across all engines.
reputation_info	object	
final_verdict*	object	
verdict*	enum	**ALLOWED:** `Known Good, Known Bad, Unknown` Reputation engine verdict. Values: - Known Good: The file has a positive reputation. - Known Bad: The file has a negative reputation. - Unknown: Reputation could not be determined.
verdict_explanation*	array	
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
sbom_info	object	
final_verdict*	object	

NAME	TYPE	DESCRIPTION
blocked*	boolean	Whether the container/application is blocked based on security policies.
licensing_package_counts*	object	
allowed*	integer	Number of packages with allowed licenses.
blocked*	integer	Number of packages with blocked licenses.
unknown*	integer	Number of packages with unknown licensing status.
unsupported*	integer	Number of packages with unsupported license detection.
severity*	enum	**ALLOWED:** UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL Overall security severity level of the SBOM.
total_package_count*	integer	Total number of packages analyzed in the SBOM.
verdict*	string	Human-readable verdict summary.
verdict_explanation*	array	
vulnerability_counts*	object	
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities.
overview_report*	object	
has_pe_info*	boolean	Indicates whether PE (Portable Executable) information is available.
is_partial_report*	boolean	Indicates if this is a partial report with incomplete data.
package_count*	integer	Total number of packages analyzed in the overview.
packages*	array	
archive*	boolean	Indicates if the package is an archive.
authors*	string	Package author(s).
ecosystem*	string	Package ecosystem or format.
group*	string	Package group classification.
hashes*	array	
file_name*	string	Name of the file.
sha256*	string	SHA-256 hash of the file.
latest*	object	
release_date*	string	Release date of the latest version.
version*	string	Latest version number.
licenses*	object	

NAME	TYPE	DESCRIPTION
allowed	array	
category*	string	License category classification.
id*	string	SPDX license identifier.
url*	string	URL to license details.
usage_restriction*	object	
conditions*	array	
limitations*	array	
permissions*	array	
package_name*	string	Name of the package.
package_type*	string	Type classification of the package.
package_version*	string	Version of the package.
release_date*	string	Package release date.
target*	string	Target file or manifest where the package was found.
uid*	string	Unique package identifier (Package URL format).
vulnerabilities*	array	
cwes*	array	
fixed_versions*	string	Versions where the vulnerability is fixed.
id*	string	Vulnerability identifier (CVE, GHSA, etc.).
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Vulnerability severity level.
source*	string	Vulnerability database source.
type*	string	Type of security issue.
vulnerability_counts*	object	
critical*	integer	
high*	integer	
low*	integer	
medium*	integer	
unknown*	integer	
scan_mode*	enum	**ALLOWED:** `container, filesystem, archive, source_code` Type of scan performed for SBOM analysis.
severity*	enum	**ALLOWED:** `UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL` Overall security severity level of the overview report.
timestamp*	string	Timestamp when the overview report was generated (ISO 8601 format).
vulnerability_count*	integer	Total number of vulnerabilities found in the overview.
vulnerability_counts*	object	

NAME	TYPE	DESCRIPTION
critical*	integer	Number of critical vulnerabilities.
high*	integer	Number of high severity vulnerabilities.
low*	integer	Number of low severity vulnerabilities.
medium*	integer	Number of medium severity vulnerabilities.
unknown*	integer	Number of vulnerabilities with unknown severity.
vulnerable_package_count*	integer	Total number of packages with known vulnerabilities in the overview.
result_template_hash*	string	Hash of the result template used internally for rendering and engine/Core coordination.
bom_info	object	
final_verdict	object	
verdict	string	
result_subtrees	array	
cbom_info	object	
aibom_info	object	
scan_results	object	
current_av_result_a	string	The AV scan result for the current file itself (e.g., archive container) as string, excluding extracted contents. Uses the same enumeration values as scan_all_result_a
current_av_result_i	string	The AV scan result for the current file itself (e.g., archive container) as index, excluding extracted contents. Uses the same enumeration values as scan_all_result_i
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table

NAME	TYPE	DESCRIPTION
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
threatintelligence_info	object	
final_verdict*	object	
blocked	boolean	Whether the file should be blocked based on Threat Intelligence evaluation.
ioc_reputation	integer	Aggregate reputation score derived from matched indicators of compromise.
most_similar_file	number	Similarity score of the most similar known file (0 to 1).
result*	object	
result_template_hash*	string	Hash of the result template used internally for Threat Intelligence rendering and coordination.
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: `50700` The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained
severity	enum	ALLOWED: `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`

NAME	TYPE	DESCRIPTION
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time
published-epoch	string	An epoch timestamp indicating source publishing time
references	array	

NAME	TYPE	DESCRIPTION
severity	enum	**ALLOWED:** `LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN` String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown
static_id	integer	An OPSWAT identifier for the vulnerability
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out
yara_info	object	
hits	object	
verdict	enum	**ALLOWED:** `0, 1, 2, 3, 4` The overall result for the analyzed file. Value will be one of the following: | index | status | |---|---| | 0 | Clean | | 1 | Found matched data | | 2 | Suspicious | | 3 | Failed | | 4 | Not scanned |

STATUS CODE - 404: Invalid hash format

4.10 GET /hash/{md5|sha1|sha256|sha512}/blocked-leaves

Fetch a list of blocked leaf files inside archive by hash

Retrieve a list of the deepest blocked files inside an input original archive file based on the hash of that archive file. Leaf file is the one that not having any successful extracted file inside. In case there's no blocked file, user will see an empty array. This list only returns 100 items at most.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*md5|sha1|sha256|sha512	string	8101abae27be4d63859c55d9e0ed0135	Hash value to search. This can be md5, sha1, sha256, sha512

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
timerange	integer		Optional. Scoping down the recent number of hours that hash lookup task should start from till now, instead of searching the entire scan history in MetaDefender Core database. Default value is 0. That means no time scope.

RESPONSE

STATUS CODE - 200: Returns the list of blocked leaf files.

RESPONSE MODEL - application/octet-stream

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
details	array	
blocked_reasons	array	
engines	array	
engine	string	Engine's name
threat_name	string	Threat's name
data_id	string	Data identifier of this blocked leaf file.
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
display_name	string	The name of this blocked leaf file.
file_type	string	The filetype using mimetype.
parent_id	string	Data identifer of the original archive file.
path	string	Full path to this blocked leaf file.
process_info	object	

NAME	TYPE	DESCRIPTION
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by "|".
actions_ran	string	List of successful actions, separated by "|". Empty string if otherwise.
verdicts	array	
yara_info	object	
hits	array	
limit_reached	boolean	Indicates if there're indeed over 100 blocked leaf files.
original_data_id	string	The data ID of the previous analysis whose result was reused. This field is present only when the result was returned from an earlier analysis through the workflow rule's 'Reuse Processing Result' option instead of being processed again. Available starting with MetaDefender Core 5.22.0.
total	integer	Total number of files in the above array.

STATUS CODE - 400: Invalid timerange header value

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 404: Hash not found or file is in progress

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.11 GET /file/webhook/{data_id}

Query webhook status

Prior to being notified by Core when webhooks mode is set, client can anytime ask Core for file / batch processing webhooks status.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Either `data_id` or `batch_id`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Webhooks status is fetched successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	The file submission identifier

NAME	TYPE	DESCRIPTION
request_time	`string`	A timestamp when the request has been made.

NAME	TYPE	DESCRIPTION
url	string	What was the called URL (should match the `callbackurl` header).

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.12 GET /file/rules

Fetching Available Analysis Rules

Retrieve all available rules with their custom configurations. Fetching available processing rules.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication. Only those rules are returned, that: * Match the apikey's role sent using the apikey header, or * Are not restricted to a specific role.
user_agent	string		The user agent string value sent in the header (specified by the client). Only those rules are returned, that: * Match the client's user agent sent using the user_agent header, or * Are not restricted to a specific user agent. For details see KB article [What are Security Policies and how do I use them?](https://onlinehelp.opswat.com/corev4/What_are_Security_Policies_and_how_do_I_use_them_.html).

RESPONSE

STATUS CODE - 200: Returns the list of available rules.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
max_file_size	integer	The maximum allowed file size (in bytes) for this rule.
name	string	A unique identifier for identify in the used rule for a scan..

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.13 GET /file/converted/{data_id}

Download a sanitized file

Retrieve sanitized file based on the data_id.

Recommendation: Use /file/download/{data_id} instead, which also supports DLP processed files in addition to sanitized files.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the sanitized content.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.14 HEAD /file/converted/{data_id}

Verify if a sanitized file exists

Verify whether the sanitized file associated with the data_id still exists or has been removed due to data retention policies before attempting to download.

Recommendation: Clients are encouraged to use /file/download/{data_id} instead, which also supports DLP processed files in addition to sanitized files.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Return http 200 if the sanitized file exists.

STATUS CODE - 404: Return HTTP 404 if the sanitized file does not exist or has been removed in accordance with data retention policies. This indicates that the file is no longer available for download.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "File could not be found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 410: The sanitized file was discarded because the option `Discard processed files when the original file is blocked` is enabled.

RESPONSE MODEL - application/json

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.15 GET /file/download/{data_id}

Download either sanitized files or DLP processed files

Retrieve sanitized file based on the data_id. In case there's no sanitized file, and DLP processed file is available, user will retrieve DLP processed file.

Recommendation: This is the recommended endpoint for downloading processed files. Use this instead of /file/converted/{data_id}, as it supports both sanitized and DLP processed files.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the sanitized or DLP processed content.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 404: File could not be found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "File could not be found"
```

```
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.16 GET /file/{data_id}/export

Export processing result as a PDF file

Retrieve a PDF processing result file. The content of this PDF file can be changed over MD Core version.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the exported result PDF file

RESPONSE MODEL - application/pdf

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.17 GET /quarantine/{sha256}/download

Download Quarantined File

Retrieve a quarantined file based on the sha256 of the original file

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*sha256	string	4c4c10348935210f1b5f1a0b6cbb46b56f941570c5aac8ce934c9b594324268c	The SHA256 of the original file. This information can be retrieved from the result of `Analyze a file`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the quarantined file.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.18 GET /file/{dataid}/sbom/report

Download SBOM Report

Retrieve an SBOM report in JSON format or a specific JSON format, such as CycloneDX or SPDX.

The path parameter accepts either a single-file data_id or a batch_id. When a batch_id is supplied, the report aggregates the SBOM of every file in the batch — non-archive files, archive files, and all child and nested-archive files.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`. For **batch** scans, supply the `batch_id` returned by `Initiate Batch` (or by a `multipart/form-data` submission) to retrieve the aggregated SBOM report for all files in the batch.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
type	string		The specific value can be `cyclonedx` or `spdx`, with the default being JSON format as specified by SBOM.
modules	string	sbom,cbom	Comma-separated list of SBOM engine analysis modules to return, e.g. `sbom`, `cbom`, `aibom` (case-insensitive, the `_info` suffix is optional). The response then has one `<module>_info` object per requested module (`{}` when the file has no data for it), plus `bom_info` when the SBOM engine reported an overall result. Ignored when `type` is set. An invalid list returns `400`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the SBOM report in the specified format.

RESPONSE MODEL - application/json

EXAMPLE 1:

```
"default"
```

EXAMPLE 2:

```
"modules"
```

STATUS CODE - 400: The `modules` list is invalid.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid input"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.19 POST /file/{data_id}/cancel

Cancel File Analysis

When cancelling a file analysis, the connected analysis (e.g. files in an archive) that are still in progress will be cancelled also.

The cancelled analysis will be automatically closed.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Analysis was sucessfully cancelled.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "<<data_id>>": "cancelled"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Data ID not found (invalid id) or Requests resource was not found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

5. AUTH

Authentication APIs

User authentication is done via username & password. Additional integrations are available within the product:

- **LDAP** integration
- **Active Directory** integration
- **SAML** integration (starting with v4.18.0)

`5.1 POST /login`

Login

Initiate a new session. Required for using protected REST APIs.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
user*	`string`	Username
password*	`string`	User's password

EXAMPLE:

```
{
 "user": "admin",
 "password": "admin"
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
oms-csrf-token*	string	The randomly generated token used to prevent CSRF attacks
session_id*	string	The apikey used to make API calls which requires authentication

STATUS CODE - 403: Invalid credentials

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	<error message> will describe the incident. More details would be logged in MetaDefender Core logs

EXAMPLE:

```
{
 "err": "Failed to login"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

5.2 POST /logout

Logout

Destroy session for not using protected REST APIs.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
response*	string	

STATUS CODE - 400: Bad Request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6. BATCH

Group the analysis requests in batches.

6.1 POST /file/batch

Initiate Batch

Create a new batch and retrieve the batch_id

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used. Since MetaDefender Core 5.6.0, when user agent is matched the predefined setting in workflow rule, the product will return X-Core-Id containing identity of MetaDefender Core server as a header of response.
user-data	string		Name of the batch (max 1024 bytes, URL encoded UTF-8 string).

NAME	TYPE	EXAMPLE	DESCRIPTION
`metadata`	`json`		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending metadata to MetaDefender Core to prevent unexpected issues related to encoding errors or unsafe characters. The `source_system` key is reserved for the system that submitted the file; it can be used to filter `Active performance`. When present, its value must be a string of 1 to 128 letters, digits, dots, underscores, colons or hyphens, otherwise the request is rejected with `400`.

NAME	TYPE	EXAMPLE	DESCRIPTION

client-identity	json		The `client-identity` helps track and understand the workflow that a file passes through clients, offering greater visibility into the processing stages and the clients involved.
			This header is provided as a JSON object containing an array of client identity objects. Each object in the array should include the following fields:
			* `deployment_id`: *REQUIRED* | Unique identifier for the deployment.
			* `user_name`: *REQUIRED* | Name of the user or system that initiated the scan.
			* `product_type`: *REQUIRED* | Type of product or system that is performing the scan.
			* `version`: *OPTIONAL* | Version of the product or system.
			* `host`: *OPTIONAL* | Hostname or IP address of the client machine.
			* `timestamp`: *REQUIRED* | Timestamp of when the scan was initiated, in milliseconds since epoch.
			Example:
			``json
			{
			"clients":
			[
			{
			"deployment_id": "abcdef1234567890",
			"user_name": "Gemini",
			"product_type": "MetaDefender Kiosk",
			"version": "1.0.0",
			"host": "client.example.com",
			"timestamp": 1744949754890
			}
			]
			}
			``
			It should be performed the *URL-Encoding* before sending to MetaDefender Core to prevent unexpected issues related to encoding errors or unsafe characters.
			This header is optional and can be omitted if not needed.

RESPONSE

STATUS CODE - 200: Batch created successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_id*	string	The batch identifier used to submit files in the batch and to close the batch.

EXAMPLE:

```
{
 "batch_id": "74c85f475147439bac4d33b181853923"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.2 POST /file/batch/{batchId}/close

Close Batch

The batch will be closed and files can no longer be added to the current batch.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch successfully closed.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	object	
batch_count	integer	The total number of files/entries in the batch.
current_finished_files	integer	The number of files/entries that have been completely processed.
files_in_batch	array	
data_id	string	Unique identifer for the file.
detected_by	integer	Total number of engines that detected this file.
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
progress_percentage	integer	Percentage of processing completed (from 1-100).
result	string	The final result of processing the file (Allowed / Blocked / Processing).

NAME	TYPE	DESCRIPTION
verdicts	array	
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
scanned_with	integer	The total number of engines used to analyze this file.
first_index	integer	The starting index in the batch. Used for pagination.
page_size	integer	The number of entries per page.
batch_id	string	The batch unique identifer
is_closed	boolean	The batch status (open/close).
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.

NAME	TYPE	DESCRIPTION
profile	`string`	The used rule name.
result	`string`	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	`string`	Identifier for the REST Client that calls the API.
username	`string`	User identifier who submitted scan request earlier.
scan_results	`object`	
batch_id	`string`	The batch unique identifer
scan_all_result_a	`enum`	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall scan result as string
scan_all_result_i	`enum`	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall scan result as index in the Processing Results table.
start_time	`string`	Timestamp when the scanning process starts.
total_avs	`integer`	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.
total_time	`integer`	Total time elapsed during scan (in milliseconds).
user_data	`string`	Metadata submitted at batch creation.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.3 POST /file/batch/{batchId}/close/callback

Close Batch with webhook

The batch will be closed and files can no longer be added to the current batch.

With a valid apikey, up to 10 callbackheader-name / callbackheader-value pairs are copied onto the webhook POST (first pair unsuffixed, then 1–10). Not supported in cluster mode. Not included if the webhook is resent after a Core restart.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
*callbackurl	uri		Client's URL where MetaDefender Core will notify scan result back to whenever scan is finished (webhooks model). * Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> * Example: http://10.0.1.100:8081/listenback * Supported protocol: HTTP / HTTPS * Method: POST

NAME	TYPE	EXAMPLE	DESCRIPTION
`callbackheader-name`	`string` max:128 chars		Header name Core attaches to the webhook POST (with `callbackurl`). Pair with `callbackheader-value`. Extra pairs: `callbackheader-name1` … `callbackheader-name10` with matching `callbackheader-valueN`. At most **10** pairs. Requires a valid `apikey`. Name: RFC 7230 token, max **128** characters. Duplicate names: highest index wins. Suffixes outside `1`–`10` (for example `01`, `11`) are ignored. Reserved (case-insensitive): `host`, `content-length`, `content-type`, `transfer-encoding`, `connection`, `user-agent`, `accept-charset`, `accept-language`, `digital_signature`, `x-core-id`.
`callbackheader-value`	`string` max:4096 chars		Value for `callbackheader-name` (or `callbackheader-valueN` for numbered pairs). Max **4096** characters. Empty is allowed. `CR`, `LF`, and `NUL` are rejected.

RESPONSE

STATUS CODE - 200: Batch successfully closed.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	`object`	
batch_count	`integer`	The total number of files/entries in the batch.
current_finished_files	`integer`	The number of files/entries that have been completely processed.
files_in_batch	`array`	
data_id	`string`	Unique identifer for the file.
detected_by	`integer`	Total number of engines that detected this file.
display_name	`string`	The filename reported via `filename` header.
file_size	`integer`	Total file size in bytes.
file_type	`string`	The filetype using mimetype.
file_type_description	`string`	The filetype in human readable format.
process_info	`object`	
blocked_reason	`string`	Provides the reason why the file is blocked (if so).
progress_percentage	`integer`	Percentage of processing completed (from 1-100).

NAME	TYPE	DESCRIPTION
result	string	The final result of processing the file (Allowed / Blocked / Processing).
verdicts	array	
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
scanned_with	integer	The total number of engines used to analyze this file.
first_index	integer	The starting index in the batch. Used for pagination.
page_size	integer	The number of entries per page.
batch_id	string	The batch unique identifer
is_closed	boolean	The batch status (open/close).
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).

NAME	TYPE	DESCRIPTION
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
profile	string	The used rule name.
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
scan_results	object	
batch_id	string	The batch unique identifer
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.
total_time	integer	Total time elapsed during scan (in milliseconds).

NAME	TYPE	DESCRIPTION
user_data	`string`	Metadata submitted at batch creation.

STATUS CODE - 400: Invalid header, apikey, or callback header pair.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Callback header validation failed. `<N>` is the pair suffix (empty for the first pair, `1`–`10` for numbered pairs). * `callbackheader-name and callbackheader-value can only be used with a valid apikey` * `Too many callbackheader-name / callbackheader-value pairs, at most 10 are allowed` * `Both callbackheader-name<N> and callbackheader-value<N> have to be sent together` * `Header name in callbackheader-name<N> is not a valid HTTP header name` * `Header name in callbackheader-name<N> is reserved and cannot be overridden` * `Header value in callbackheader-value<N> contains an invalid character` * `Header name or value in callbackheader-name<N> / callbackheader-value<N> is too long`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.4 GET /file/batch/{batchId}

Status of Batch Analysis

Retrieve status report for the entire batch

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer		The first item order in the list of files in this batch
size	integer		The number of items to be fetched next, counting from the item order indicated in first header

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch progress paginated report (50 entries/page).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	object	
batch_count	integer	The total number of files/entries in the batch.
current_finished_files	integer	The number of files/entries that have been completely processed.
files_in_batch	array	
data_id	string	Unique identifer for the file.
detected_by	integer	Total number of engines that detected this file.
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
progress_percentage	integer	Percentage of processing completed (from 1-100).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
verdicts	array	
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading` The overall scan result as string
scan_all_result_i	enum	**ALLOWED:** `0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257` The overall scan result as index in the Processing Results table.
scanned_with	integer	The total number of engines used to analyze this file.
first_index	integer	The starting index in the batch. Used for pagination.
page_size	integer	The number of entries per page.
batch_id	string	The batch unique identifer
is_closed	boolean	The batch status (open/close).
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
profile	string	The used rule name.
result	string	The final result of processing the file (Allowed / Blocked / Processing).

NAME	TYPE	DESCRIPTION
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
client_identity	object	
clients	array	
deployment_id*	string	Unique identifier for the deployment.
user_name*	string	Name of the user or system that initiated the scan.
product_type*	string	Type of product or system that is performing the scan.
version	string	Version of the product or system.
host	string	Hostname or IP address of the client machine.
timestamp*	integer	Timestamp of when the scan was initiated, in milliseconds since epoch.
scan_results	object	
batch_id	string	The batch unique identifer
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Allowlisted, Blocklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, License Risk Verdict by SBOM, Blocked Verdict by File Type, Multipart Upload Timed Out, Sanitized, Exceeded Limited Sandbox File Size, Blocked Verdict by Archive Extraction, Blocked Verdict by Archive Compression, Sandbox Execution Limit Reached, Rejected, Empty Split Archive, Split Archive Upload Timed Out, Unsupported Split Archive Type, Split Archive Not Supported, Exceeded Archive Expansion Factor, Blocked Verdict by Threat Intelligence, Allowed Verdict by Threat Intelligence, AI Generated, Not AI Generated, AI Detection Uncertain, RMS Protected Document, OPSWAT AI Confidently Clean, OPSWAT AI Confidently Malicious, OPSWAT AI Undetermined, Download Timeout, Download Failed, Allowed Verdict by File Structure Validation, Blocked Verdict by File Structure Validation, Exceeded Structural Depth, Exceeded Structural Object Count, Exceeded Structural Object Size, Component Risk Found, No Component Risk Found, Processed, Not Processed, Early Exit, In Progress, Multipart Uploading, Split Archive Uploading The overall scan result as string

NAME	TYPE	DESCRIPTION
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 40, 41, 42, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 255, 256, 257 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.
total_time	integer	Total time elapsed during scan (in milliseconds).
user_data	string	Metadata submitted at batch creation.
multipart_form_batch	boolean	Indicates whether the batch was created using a multipart/form-data upload.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.5 GET /file/batch/{batchId}/certificate

Download Signed Batch Result

Download digitally signed status report for the entire batch

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

NAME	TYPE	EXAMPLE	DESCRIPTION
metadata	json		In JSON format, this can be used to: Include additional information in the response YML. Currently, one supported field in the metadata is `include_vul_info`, which can be set to `true` or `false` to indicate whether vulnerability processing information should be included. It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters.

RESPONSE

STATUS CODE - 200: Signed batch result and certificate are sent back in response body (YAML format).

RESPONSE MODEL - application/x-yaml

EXAMPLE:


```
"--- batch_id: 092876200fb54cfb80b6e3332c410ae9 user_data: the user data from the header from
batch creation cert_sha1_fingerprint: <some cert serial value> batch_files:\n  batch_count: 1\n
files_in_batch:\n  - data_id: 9112b225f0634f189a2bb46ec1a7826f\n    display_name:
New%20Text%20Document.txt\n    file_size: 5\n    scan_all_result_i: 0\n    process_info:\n
blocked_reason:\n      result: Allowed\n    md5: 42b130c3ce46e058f30712838cebf420\n    sha1:
ed94baf55ca851055fb76045f6949bca2f865605\n    sha256:
f4191b3ec6ce93aaf712919a38e52815c5da9c91d2b141df920bc8bcb5cbb8e3\n    sha512: \"\"\n
vulnerabilities:\n      - cve: CVE-2021-45463\n        cvss:\n          score: 6.8\n
cvss_3_0:\n          base_score: 7.8\n      - cve: CVE-2018-12713\n        cvss:\n
score: 6.4\n        cvss_3_0:\n          base_score: 9.1\nprocess_info:\n  blocked_reason:\n
file_type_skipped_scan: false\n  profile: File scan\n  result: Allowed\n  user_agent:
webscan\nscan_results:\n  scan_all_result_a: No Threat Detected\n  scan_all_result_i: 0\n
start_time: 2017-05-23T11:22:03.010Z\n  total_avs: 14\n  total_time: 995\n...\n--- signature:
881d22220c4ca0557d7c7d5c5794d53a8a2780997cd65b27b6e7f1c099a15de03dbcb5edbeaea7aafa6099fab37be07
017b39e3e3a7d66c550f44eb59a096c54d5b9555cb28198546fbec57c33b717751d333a09733d95dd876e2798d044c8
caef828f4352b91f9a6d057253bb1a9461e0e0e0bf4313a80895998d645bebc81841ff3499589c80ffc4e8a190d1ec9
b3e4126d86659d303b0e1f22d9289c9c4671d35532b55ad4620e048a78bb405b573897da63efdd5f036692c934a82d9
bdc9b9862e7fea5e8abeeb1444be0689d50373c5c0632484950c0fe0337ed5f91bdf26986f7cff8aa3431bf4bc948fc
127c16ba13ec679fe9f67e7586075c1f467454fa8cf40e9cd501291c95d862eb16f4477c17d1711294f0ff2b3a1140b
d53dbd1fbb0846af6062e9e4e2e1a09af3448503ed11e342164e535fc268bf7d8fbc28ed946cd2bb8ea075f2295d2fa
8392076d41608c3b5decf8fab3a5ec7de190f07583331e0517e5f361735cd59326622dc8b07b10a464028de781a063e
408f918c1d5534329140f4e4dc1a717d808d6784410410b00d36cb9a345f5bbc11fa1c58ee28f8e7b863f3ea2c923ec
5fb2ac29eaa4ddc0d6d9dfd3f16a97f207dc2858410a577c7f4a92ff01bad3229f5fcdb08e21df9869a113272aa9d96
bfdfe8bfb3a50414c174e16a3504e5780c2718779b0757298546f287ef7ea86e67510d48a8 certificate: |\n
-----BEGIN CERTIFICATE-----\n
MIIGJzCCBA+gAwIBAgIBATANBgkqhkiG9w0BAQUFADCBsjELMAkGA1UEBhMCRlIx\n
DzANBgNVBAgMBkFsc2FjZTETMBEGA1UEBwwKU3RyYXNib3VyZzEYMBYGA1UECgwP\n
d3d3LmZyZWVsYW4ub3JnMRAwDgYDVQQLDAdmcmVlbGFuMS0wKwYDVQQDDCRGcmVl\n
bGFuIFNhbXBsZSBDZXJ0aWZpY2F0ZSBBdXRob3JpdHkxIjAgBgkqhkiG9w0BCQEW\n
E2NvbnRhY3RAZnJlZWxhbi5vcmcwHhcNMTIwNDI3MTAzMTE4WhcNMjIwNDI1MTAz\n
MTE4WjB+MQswCQYDVQQGEwJGUjEPMA0GA1UECAwGQWxzYWNlMRgwFgYDVQQKDA93\n
d3cuZnJlZWxhbi5vcmcxEDAOBgNVBAsMB2ZyZWVsYW4xDjAMBgNVBAMMBWFsaWNl\n
MSIwIAYJKoZIhvcNAQkBFhNjb250YWN0QGZyZWVsYW4ub3JnMIICIjANBgkqhkiG\n
```

```
9w0BAQEFAAOCAg8AMIICCgKCAgEA3W29+ID6194bH6ejLrIC4hb2Ugo8v6ZC+Mrc\n
k2dNYMNPjcOKABvxxEtBamnSaeU/IY7FC/giN622LEtV/3oDcrua0+yWuVafyxmZ\n  yTKUb4/GUgafRQPf/
eiX9urWurtIK7XgNGFNUjYPq4dSJQPPhwCHE/LKAykWnZBX\n
RrX0Dq4XyApNku0IpjIjEXH+8ixE12wH8wt7DEvdO7T3N3CfUbaITl1qBX+Nm2Z6\n  q4Ag/
u5rl8NJfXg71ZmXA3XOj7zFvpyapRIZcPmkvZYn7SMCp8dXyXHPdpSiIWL2\n  uB3KiO4JrUYvt2GzLBUThp+lNSZaZ/
Q3yOaAAUkOx+1h08285Pi+P8lO+H2Xic4S\n
vMq1xtLg2bNoPC5KnbRfuFPuUD2/3dSiiragJ6uYDLOyWJDivKGt/72OVTEPAL9o\n
6T2pGZrwbQuiFGrGTMZOvWMSpQtNl+tCCXlT4mWqJDRwuMGrI4DnnGzt3IKqNwS4\n
Qyo9KqjMIPwnXZAmWPm3FOKe4sFwc5fpawKO01JZewDsYTDxVj+cwXwFxbE2yBiF\n  z2FAHwfopwaH35p3C6lkcgP2k/
zgAlnBluzACUI+MKJ/G0gv/uAhj1OHJQ3L6kn1\n  SpvQ41/
ueBjlunExqQSYD7GtZ1Kg8uOcq2r+WISE3Qc9MpQFFkUVllmgWGwYDuN3\n
Zsez95kCAwEAAaN7MHkwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNT\n
TCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFFlfyRO6G8y5qEFKikl5\n
ajb2fT7XMB8GA1UdIwQYMBaAFCNsLT0+KV14uGw+quK7Lh5sh/JTMA0GCSqGSIb3\n
DQEBBQUAA4ICAQAT5wJFPqervbja5+90iKxi1d0QVtVGB+z6aoAMuWK+qgi0vgvr\n
mu9ot2lvTSCSnRhjeiP0SIdqFMORmBtOCFk/kYDp9M/91b+vS+S9eAlxrNCB5VOf\n  PqxEPp/wv1rBcE4GBO/
c6HcFon3F+oBYCsUQbZDKSSZxhDm3mj7pb67FNbZbJIzJ\n
70HDsRe2O04oiTx+h6g6pW3cOQMgIAvFgKN5Ex727K4230B0NIdGkzuj4KSML0NM\n
slSAcXZ41OoSKNjy44BVEZv0ZdxTDrRM4EwJtNyggFzmtTuV02nkUj1bYYYC5f0L\n
ADr6s0XMyaNk8twlWYlYDZ5uKDpVRVBfiGcq0uJIzIvemhuTrofh8pBQQNkPRDFT\n  Rq1iTo1Ihhl3/
Fl1kXk1WR3jTjNb4jHX7lIoXwpwp767HAPKGhjQ9cFbnHMEtkro\n
RlJYdtRq5mccDtwT0GFyoJLLBZdHHMHJz0F9H7FNk2tTQQMhK5MVYwg+LIaee586\n
CQVqfbscp7evlgjLW98H+5zylRHAgoH2G79aHljNKMp9BOuq6SnEglEsiWGVtu2l\n  hnx8SB3sVJZHeer8f/
UQQwqbAO+Kdy70NmbSaqaVtp8jOxLiidWkwSyRTsuU6D8i\n
DiH5uEqBXExjrj0FslxcVKdVj5glVcSmkLwZKbEU1OKwleT/iXFhvooWhQ==\n  -----END CERTIFICATE-----\n...
\n"
```


STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

6.6 POST /file/batch/{batchId}/cancel

Cancel Batch

When cancelling a batch, the connected analysis that are still in progress will be cancelled also.

The cancelled batch will be closed.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch cancelled.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "<<batch_id>>": "cancelled"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Batch not found (invalid id)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7. CONFIG

Configure the product through APIs (especially the Settings). Will require admin apikey.

7.1 PUT /admin/config/auditlog

Audit clean up

Setting audit records (update history) clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.2 PUT /admin/config/quarantine

Quarantine clean up

Setting quarantine clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.3 PUT /admin/config/sanitize

Sanitized files clean up

Setting sanitized files clean up time (clean up records older than).

Note:The clean up range is defined in minutes.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
maxage	integer	The number of minutes of data retention. Anything older than this number will be deleted.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
maxage	integer	The number of minutes of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.4 PUT /admin/config/scanhistory

Processing history clean up

Setting processing history clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.5 PUT /admin/config/session

Session settings

Configure settings for session generated upon a successful login See more at Login

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
absoluteSessionTimeout	integer	The interval (in milliseconds) for overall session length timeout (regardless of activity).
allowCrossIpSessions	boolean	Allow requests from the same user to come from different IPs.
allowDuplicateSession	boolean	Allow same user to have multiple active sessions.
sessionTimeout	integer	The interval (in milliseconds) for the user's session timeout, based on last activity. Timer starts after the last activity for the apikey.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
absoluteSessionTimeout	integer	The interval (in milliseconds) for overall session length timeout (regardless of activity).
allowCrossIpSessions	boolean	Allow requests from the same user to come from different IPs.
allowDuplicateSession	boolean	Allow same user to have multiple active sessions.
sessionTimeout	integer	The interval (in milliseconds) for the user's session timeout, based on last activity. Timer starts after the last activity for the apikey.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.6 PUT /admin/config/update

Modules Update Source and Frequency

Setting update frequency for all modules and engines.

Note:The clean up range is defined in minutes.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
autoupdateperiod	integer	The interval (in minutes) for checking for new updates (Must be non-negative).
deleteafterimport	boolean	If you want to clean the pickup folder after the updates have been applied.
disabledupdate	array	
days	string	A particular day (index of the week) or a defined interval (Fri-Sun -> 5-7)
from	integer	When to start the lockdown, expressed in minutes (8 AM=480)
to	integer	When to end the lockdown, expressed in minutes (4 PM=960)
pickupfolder	string	The folder where MetaDefender will look for the new engine files.
source	enum	**ALLOWED:** internet, folder, manual Define where the updates will be loaded from. <p> This can be either: * `internet` -> if selected, will check for new updates every `autoupdateperiod` minutes * `folder` -> make sure that MetaDefender has access/permission to that folder * `manual` -> requires manually uploading the packages in Inventory > Modules > Upload package.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
autoupdateperiod	integer	The interval (in minutes) for checking for new updates (Must be non-negative).
deleteafterimport	boolean	If you want to clean the pickup folder after the updates have been applied.
disabledupdate	array	
days	string	A particular day (index of the week) or a defined interval (Fri-Sun -> 5-7)
from	integer	When to start the lockdown, expressed in minutes (8 AM=480)
to	integer	When to end the lockdown, expressed in minutes (4 PM=960)
pickupfolder	string	The folder where MetaDefender will look for the new engine files.
source	enum	**ALLOWED:** internet, folder, manual Define where the updates will be loaded from. <p> This can be either: * `internet` -> if selected, will check for new updates every `autoupdateperiod` minutes * `folder` -> make sure that MetaDefender has access/permission to that folder * `manual` -> requires manually uploading the packages in Inventory > Modules > Upload package.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.7 GET /config/skip/{skip_type}/template

Download CSV template for 'skip by hash' CSV upload.

Download CSV template for 'skip by hash' CSV upload.

Note: Only available since MetaDefender Core 5.12.1.

Note: Since MetaDefender Core 5.22.0, the header includes a trailing expire_at column. Sample rows are included below the header; each is prefixed with # so the file can be uploaded unchanged. Remove the leading # from a row to import it.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*skip_type	string		The type of 'skip by hash' template to download, can be `blocklist`, `allowlist` or `enginelist`.

RESPONSE

STATUS CODE - 200: CSV template of the requested 'skip by hash' type.

RESPONSE MODEL - text/csv

EXAMPLE 1:

```
"allowlist_or_blocklist"
```

EXAMPLE 2:

```
"enginelist"
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.8 POST /admin/config/skip/{skip_type}

Upload csv file of 'skip by hash' list.

Add hashes by uploading csv file of 'skip by hash' list.

Note: Only available since MetaDefender Core 5.12.1. Skip hashes will be overwritten if it already exists.

File Size Limit: Since MetaDefender Core 5.14.2, CSV files should not exceed 500 MB.

Note: Since MetaDefender Core 5.21.0:

- Each entry of the responded hash list includes source and readonly, indicating whether the entry is owned by My OPSWAT Central Management.
- When this MetaDefender Core instance is enrolled to My OPSWAT Central Management, the rows whose hashes are owned by Central Management are skipped instead of being overwritten. Unlike POST and PUT /admin/config/skip, the upload applies partial success: the remaining rows are still stored and the number of skipped rows is reported in skipped_managed.

Note: Since MetaDefender Core 5.22.0, an optional trailing expire_at column is accepted (same values as JSON expire_at). Files without that column still import.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*skip_type	string		The type of 'skip by hash' list to upload, can be `blocklist`, `allowlist` or `enginelist`.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
summary	boolean		The response body includes summaries on the number of hashes added and modified for each type of 'skip by hash'. The default value is `false`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of all skip/allow/block-listed hashes or the summary of new/modfied hashes per type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
blacklist*	object	
skip*	object	
whitelist*	object	
total_count*	integer	
OPTION:2	object	
number_blocklist_accepted*	integer	Total number of new or modified hashes of blacklist. _**Note**_: Since MetaDefender Core 5.21.0, the rows owned by My OPSWAT Central Management are excluded from this value and reported in `skipped_managed` instead.
number_enginelist_accepted*	integer	Total number of new or modified hashes of skip engine list. _**Note**_: Since MetaDefender Core 5.21.0, the rows owned by My OPSWAT Central Management are excluded from this value and reported in `skipped_managed` instead.
number_allowlist_accepted*	integer	Total number of new or modified hashes of whitelist. _**Note**_: Since MetaDefender Core 5.21.0, the rows owned by My OPSWAT Central Management are excluded from this value and reported in `skipped_managed` instead.

NAME	TYPE	DESCRIPTION
skipped_managed*	integer	_**Note**_: Only available since MetaDefender Core 5.21.0. Total number of rows of the uploaded csv file that were skipped because the hashes are owned by My OPSWAT Central Management. Unlike `POST` and `PUT /admin/config/skip`, uploading a csv file applies partial success: the rows that are not owned by My OPSWAT Central Management are still stored. This field is always returned, including when its value is `0`.

STATUS CODE - 400: Bad Request(e.g. invalid headers, empty body content, file size exceeding the limit).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 422: Unprocessable Content (e.g. invalid CSV row). The whole upload is rolled back.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	Error reason

EXAMPLE 1:

```
"invalid_csv"
```

EXAMPLE 2:

```
"invalid_expire_at"
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.9 GET /admin/config/skip

Get 'skip by hash' list

Get list of hashes that match the search conditions.

Note: Since MetaDefender Core 5.12.1:

- Support 4 query strings: first, size, search and types.
- If no query strings are defined, all hash data will be returned.
- The hash list responded includes an additional value for the total search counts and a timestamp for each hash indicating when it was added or modified.

Note: Since MetaDefender Core 5.21.0, each entry of the responded hash list includes source and readonly, indicating whether the entry is owned by My OPSWAT Central Management. Both

fields are returned regardless of whether the query strings above are used.

Note: Since MetaDefender Core 5.22.0, each entry includes expire_at as epoch milliseconds (0 = never expires, never null). Expired hashes are omitted from the list and from total_count.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
`first`	`integer`		The first item order in the list of hashes. Default is `0`.
`size`	`integer`		The number of items to be fetched, counting from the item order indicated in `first` parameter. Default is `20`.
`search`	`string`		Text to search. The can be a part of (MD5, SHA-1, SHA-256, SHA-512) hashes, or a part of the description and must be URL-encoded. Default is empty (any value).
`types`	`string`		The types of hashes to search. This option can include a single type, a couple of types, or all three types in the list (blocklist, allowlist, enginelist) separated by `|`. Default value is `blocklist|enginelist|allowlist`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Search results are a list of skip/allow/block-listed hashes.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
blacklist*	`object`	
skip*	`object`	
whitelist*	`object`	
total_count*	`integer`	

STATUS CODE - 400: Bad Request(e.g. invalid parameters).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.10 PUT /admin/config/skip

Modify 'skip by hash' list

Modify/Add hashes or overwrite all hashes in 'skip by hash' list.

Note: Since MetaDefender Core 5.12.1:

- Support 2 query strings: overwrite and summary.
- The hash list responded includes an additional value for the total hash counts and a timestamp for each hash indicating when it was added or modified.

Note: Since MetaDefender Core 5.21.0:

- Each entry of the responded hash list includes source and readonly, indicating whether the entry is owned by My OPSWAT Central Management.
- When this MetaDefender Core instance is enrolled to My OPSWAT Central Management, the entries synchronized from Central Management are read-only:
  - With overwrite=false, the whole request is rejected with 400 if any of the submitted hashes is owned by Central Management.
  - With overwrite=true, the hashes owned by Central Management are preserved instead of being overwritten, and their number is reported in skipped_managed.

Note: Since MetaDefender Core 5.22.0, each hash object may include optional expire_at (same rules as POST /admin/config/skip).

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
`overwrite`	`boolean`		Overwrite all hashes in 'skip by hash' list. The default value is `true`.
`summary`	`boolean`		The response body includes summaries on the number of hashes added and/or modified for each type of 'skip by hash'. The default value is `false`.

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
blacklist	object	
skip	object	
whitelist	object	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of all skip/allow/block-listed hashes or the summary of new/modfied hashes per type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
blacklist*	object	
skip*	object	
whitelist*	object	
total_count*	integer	
OPTION:2	object	
number_blocklist_accepted*	integer	Total number of new or modified hashes of blacklist. _**Note**_: Since MetaDefender Core 5.21.0, this value counts the hashes that were actually stored instead of the hashes that were submitted. The submitted hashes owned by My OPSWAT Central Management are excluded and reported in `skipped_managed` instead.
number_enginelist_accepted*	integer	Total number of new or modified hashes of skip engine list. _**Note**_: Since MetaDefender Core 5.21.0, this value counts the hashes that were actually stored instead of the hashes that were submitted. The submitted hashes owned by My OPSWAT Central Management are excluded and reported in `skipped_managed` instead.

NAME	TYPE	DESCRIPTION
number_allowlist_accepted*	integer	Total number of new or modified hashes of whitelist. _**Note**_: Since MetaDefender Core 5.21.0, this value counts the hashes that were actually stored instead of the hashes that were submitted. The submitted hashes owned by My OPSWAT Central Management are excluded and reported in `skipped_managed` instead.
skipped_managed	integer	_**Note**_: Only available since MetaDefender Core 5.21.0. Total number of submitted hashes that were preserved instead of being overwritten, because they are owned by My OPSWAT Central Management. This field is only returned when `overwrite` is `true` **and** this MetaDefender Core instance is enrolled to My OPSWAT Central Management. It is omitted when its value would be `0`.

STATUS CODE - 400: Bad Request(e.g. invalid headers, invalid request body, submitted hashes owned by My OPSWAT Central Management, invalid `expire_at`).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	Error reason

EXAMPLE 1:

```
"invalid_content_type"
```

EXAMPLE 2:

```
"managed_by_central_management"
```

EXAMPLE 3:

```
"ownership_lookup_failed"
```

EXAMPLE 4:

```
"invalid_expire_at"
```

EXAMPLE 5:

```
"expire_at_null"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.11 POST /admin/config/skip

Add hashes to 'skip by hash' list.

Add hashes to 'skip by hash' list.

Note: Since MetaDefender Core 5.12.1:

- Support query string: summary.
- Skip hashes will be overwritten if it already exists.
- The hash list responded includes an additional value for the total hash counts and a timestamp for each hash indicating when it was added or modified.

Note: Since MetaDefender Core 5.21.0:

- Each entry of the responded hash list includes source and readonly, indicating whether the entry is owned by My OPSWAT Central Management.
- When this MetaDefender Core instance is enrolled to My OPSWAT Central Management, the entries synchronized from Central Management are read-only. The whole request is rejected with 400 if any of the submitted hashes is owned by Central Management.

Note: Since MetaDefender Core 5.22.0:

- Each hash object may include optional expire_at (epoch milliseconds, or an ISO-8601 date / datetime). Omit the field, send 0, or send an empty string for never expires. JSON null is rejected.
- A Central Management push of the same hash replaces a local expiry with 0 (no expiry).

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
summary	boolean		The response body includes summaries on the number of hashes added and modified for each type of 'skip by hash'. The default value is `false`.

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
blacklist	object	
skip	object	
whitelist	object	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of all skip/allow/block-listed hashes or the summary of new/modfied hashes per type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
blacklist*	object	
skip*	object	
whitelist*	object	
total_count*	integer	
OPTION:2	object	
number_blocklist_accepted*	integer	Total number of new or modified hashes of blacklist.
number_enginelist_accepted*	integer	Total number of new or modified hashes of skip engine list.
number_allowlist_accepted*	integer	Total number of new or modified hashes of whitelist.

STATUS CODE - 400: Bad Request(e.g. invalid header, invalid request body, submitted hashes owned by My OPSWAT Central Management, invalid `expire_at`).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	Error reason

EXAMPLE 1:

```
"invalid_content_type"
```

EXAMPLE 2:

```
"managed_by_central_management"
```

EXAMPLE 3:

```
"ownership_lookup_failed"
```

EXAMPLE 4:

```
"invalid_expire_at"
```

EXAMPLE 5:

```
"expire_at_null"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 422: Unprocessable Content (e.g. invalid content of file).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid CSV format"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.12 DELETE /admin/config/skip

Delete hashes from 'skip by hash' list

Delete multiple hashes from "skip by hash" list.

Note: Since MetaDefender Core 5.12.1:

- Support query string: summary.

Note: Since MetaDefender Core 5.21.0:

- Each entry of the responded hash list includes source and readonly, indicating whether the entry is owned by My OPSWAT Central Management.
- When this MetaDefender Core instance is enrolled to My OPSWAT Central Management, the entries synchronized from Central Management are read-only. The whole request is rejected with 400 if any of the submitted hashes is owned by Central Management.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
summary	boolean		The response body includes summaries on the number of hashes deleted. The default value is `false`.

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of all skip/allow/block-listed hashes or the summary of deleted hashes per type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
blacklist*	object	
skip*	object	
whitelist*	object	
total_count*	integer	
OPTION:2	object	
number_hashes_removed*	integer	Total number of hashes deleted.

STATUS CODE - 400: Bad Request(e.g. invalid header, invalid request body, submitted hashes owned by My OPSWAT Central Management).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	Error reason

EXAMPLE 1:

```
"invalid_summary"
```

EXAMPLE 2:

```
"managed_by_central_management"
```

EXAMPLE 3:

```
"ownership_lookup_failed"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.13 GET /admin/config/webhook

Webhook get configuration

Getting settings supported for webhook mode

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
maxretrytime	number	Number of allowed retries sending callback to client when failed. > _**For example**_: maxretrytime= 3 Sending callback failed for some reasons * 1st retry triggered after delayduration (ms) * 2nd retry triggered after delayduration+ delayprogression (ms) * 3rd retry triggered after delayduration+ 2*delayprogression (ms)
delayduration	number	in milliseconds
delayprogression	number	in milliseconds
requestqueue	number	Total requests can be sent concurrently for each worker.
transfertimeout	number	After this transfer timeout (in milliseconds), if MetaDefender Core still doesn't receive webhook server's response, it will try to send webhook again.
workercount	number	Total number of workers for sending the result back to Webhook client. Default is 1. Min: 1, Max: 100. The recommended value is from 1 to 4 depending on your system resources. > _**Service restart is required for changes to take effect.**_

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.14 PUT /admin/config/webhook

Webhook set configuration

Modifying settings supported for webhook mode

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
maxretrytime	`number`	Number of allowed retries sending callback to client when failed. > _**For example**_: maxretrytime= 3 Sending callback failed for some reasons * 1st retry triggered after delayduration (ms) * 2nd retry triggered after delayduration+ delayprogression (ms) * 3rd retry triggered after delayduration+ 2*delayprogression (ms)

NAME	TYPE	DESCRIPTION
delayduration	number	in milliseconds
delayprogression	number	in milliseconds
requestqueue	number	Total requests can be sent concurrently for each worker.
transfertimeout	number	After this transfer timeout (in milliseconds), if MetaDefender Core still doesn't receive webhook server's response, it will try to send webhook again.
workercount	number	Total number of workers for sending the result back to Webhook client. Default is 1. Min: 1, Max: 100. The recommended value is from 1 to 4 depending on your system resources. > _**Service restart is required for changes to take effect.**_

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
maxretrytime	number	Number of allowed retries sending callback to client when failed. > _**For example**_: maxretrytime= 3 Sending callback failed for some reasons * 1st retry triggered after delayduration (ms) * 2nd retry triggered after delayduration+ delayprogression (ms) * 3rd retry triggered after delayduration+ 2*delayprogression (ms)
delayduration	number	in milliseconds
delayprogression	number	in milliseconds
requestqueue	number	Total requests can be sent concurrently for each worker.
transfertimeout	number	After this transfer timeout (in milliseconds), if MetaDefender Core still doesn't receive webhook server's response, it will try to send webhook again.
workercount	number	Total number of workers for sending the result back to Webhook client. Default is 1. Min: 1, Max: 100. The recommended value is from 1 to 4 depending on your system resources. > _**Service restart is required for changes to take effect.**_

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.15 GET /admin/config/proxy

Get proxy

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of all proxy.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
enabled	boolean	
feature_proxy	array	
exclusion	string	
id	enum	**ALLOWED:** callbackurl, downloadfrom, license, mdcloud, updateengine, sanitizedurl, ssologin, yara Feature ID.
port	string	
server	string	
use_proxy	enum	**ALLOWED:** 0, 1, 2 0 - this feature does not use proxy. 1 - this feature use master proxy setting. 2 - this feature use a custom proxy.
username	string	
exclusion	string	
port	string	
server	string	
username	string	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

7.16 PUT /admin/config/proxy

Modify proxy

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
enabled*	boolean	When true, use a proxy server for MetaDefender Core
feature_proxy	array	
exclusion	string	
id	enum	ALLOWED: callbackurl, downloadfrom, license, mdcloud, updateengine, sanitizedurl, ssologin, yara Feature ID.
port	string	
server	string	
use_proxy	enum	ALLOWED: 0, 1, 2 0 - this feature does not use proxy. 1 - this feature use master proxy setting. 2 - this feature use a custom proxy.
username	string	
exclusion	string	Not use the proxy server for the addresses starting with the following entries
port*	string	Proxy server port
server*	string	Proxy server host address
username	string	Username for proxy authentication
password	string	Password for proxy authentication

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of all proxy.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
enabled	`boolean`	
feature_proxy	`array`	
exclusion	`string`	
id	`enum`	**ALLOWED:** `callbackurl, downloadfrom, license, mdcloud, updateengine, sanitizedurl, ssologin, yara` Feature ID.
port	`string`	
server	`string`	
use_proxy	`enum`	**ALLOWED:** `0, 1, 2` 0 - this feature does not use proxy. 1 - this feature use master proxy setting. 2 - this feature use a custom proxy.
username	`string`	
exclusion	`string`	
port	`string`	
server	`string`	
username	`string`	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	`object`	
OPTION:1	`object`	
err	`string`	Missing enabled field
OPTION:2	`object`	
err	`string`	Error while parsing input json
OPTION:3	`object`	
err	`string`	Missing port
OPTION:4	`object`	
err	`string`	Missing server address
OPTION:5	`object`	
err	`string`	Test proxy connection failed. Error: Proxy requires authentication

7.17 POST /admin/config/proxy/testconnection

Check connection to proxy server

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
enabled*	boolean	This redundant attribute will be removed soon.
exclusion	string	Not use the proxy server for the addresses starting with the following entries
port*	string	Proxy server port
server*	string	Proxy server host address
username	string	Username for proxy authentication
password	string	Password for proxy authentication

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Missing port
OPTION:2	object	
err	string	Missing server address
OPTION:3	object	
err	string	Test proxy connection failed. Error: Proxy requires authentication

7.18 GET /admin/config/file/sync

Synchronous scan connection timeout - Retrieval

Retrieving sync scan related connection timeout value (in minutes)

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
timeout*	number	Sync scan related connection timeout value (in minutes).

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Error while requesting configuration.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Error while requesting configuration
OPTION:2	object	
err	string	Internal server error

7.19 PUT /admin/config/file/sync

Synchronous scan connection timeout - Modification

Modifying sync scan related connection timeout value (in minutes)

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
timeout*	number	Sync scan related connection timeout value (in minutes). The value is limited in range [1, 30] minutes.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
timeout*	`number`	Sync scan related connection timeout value (in minutes).

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Error while modifying configuration.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	`object`	
OPTION:1	`object`	
err	`string`	Error while modifying configuration Setting timeout value out of the limit range [1, 30] minutes, or the request body is not in JSON format.
OPTION:2	`object`	
err	`string`	Internal server error

7.20 GET /admin/config/sessioncookie

Get session cookie attributes

Getting session cookie attributes

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
samesite	string	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.21 PUT /admin/config/sessioncookie

Update session cookie attributes

Modifying session cookie attributes

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
samesite	`string`	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
samesite	string	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.22 GET /admin/config/cacheuser

Get user caching setting

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
enabled	`boolean`	If you want to enable user caching to improve performance, set this key to true.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
enabled	`boolean`	If you want to enable user caching to improve performance, set this key to true.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.23 PUT /admin/config/cacheuser

Enable user caching

If you want to enable user caching to improve performance, set this key to true.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
enabled	boolean	If you want to enable user caching to improve performance, set this key to true.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
enabled	boolean	If you want to enable user caching to improve performance, set this key to true.

STATUS CODE - 400: Bad request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: `Missing key enabled, Invalid value: enabled, Invalid number of keys. There should be 1 key in total` err: * Missing key enabled * Invalid value: enabled * Invalid number of keys. There should be 1 key in total

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.24 GET /admin/config/customresponseheader

Get a list of custom response headers

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
include_address	`object`	
enabled	`boolean`	Include data from address_str.
address_str	`string`	IP address.
include_deployment_id	`boolean`	Include generated deployment ID.
specify_users_agent	`object`	
enabled	`boolean`	True: required matched header user_agent to given list in allowed_users_agent to return new header X-Core-Id in response.
allowed_users_agent	`array`	
authorized_only	`boolean`	Only return new X-Core-Id header for authorized user.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.25 PUT /admin/config/customresponseheader

Update list of custom response headers

Modifying list of custom response headers

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
include_address	object	
enabled	boolean	Include data from address_str.
address_str	string	IP address.
include_deployment_id	boolean	Include generated deployment ID.
specify_users_agent	object	
enabled	boolean	True: required matched header user_agent to given list in allowed_users_agent to return new header X-Core-Id in response.
allowed_users_agent	array	
authorized_only	boolean	Only return new X-Core-Id header for authorized user.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
include_address	`object`	
enabled	`boolean`	Include data from address_str.
address_str	`string`	IP address.
include_deployment_id	`boolean`	Include generated deployment ID.
specify_users_agent	`object`	
enabled	`boolean`	True: required matched header user_agent to given list in allowed_users_agent to return new header X-Core-Id in response.
allowed_users_agent	`array`	
authorized_only	`boolean`	Only return new X-Core-Id header for authorized user.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.26 GET /admin/config/healthcheck

Get health check

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Health check criterias to determine whether MetaDefender Core server status is healthy or not.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
database	boolean	
enable_feature	boolean	
number_active_av_engines	integer	
engines	array	
id	string	
display_name	string	
is_selected	boolean	Whether this engine is in required_engines
enable_outdate_engine	boolean	Whether out-of-date checking is enabled for this engine
threshold	integer	Threshold in days for out-of-date checking
license	boolean	
available_disk_space_threshold	integer	Minimum free disk space required as a percentage of total disk. 0 = feature disabled (default).
scan_queue	integer	unit is percentage "%". To disable health check on scan queue status, set 0.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

7.27 POST /admin/config/healthcheck

Modify health check

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
enable_feature	`boolean`	
scan_queue	`integer`	
number_active_av_engines	`integer`	
required_engines	`array`	
available_disk_space_threshold	`integer`	`between 0 and 100` DEFAULT:`0` Minimum free disk space required as a percentage of total disk. 0 = feature disabled (default). Valid range [0, 100].
outdated_engines	`array`	

NAME	TYPE	DESCRIPTION
id*	`string`	Engine ID (must exist in required_engines)
enable	`boolean`	Enable out-of-date checking for this engine
threshold	`integer`	`between 1 and 10000` DEFAULT:7 Threshold in days for this engine

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Health check criterias to determine whether MetaDefender Core server status is healthy or not.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
database	`boolean`	
enable_feature	`boolean`	
number_active_av_engines	`integer`	
engines	`array`	
id	`string`	
display_name	`string`	
is_selected	`boolean`	Whether this engine is in required_engines
enable_outdate_engine	`boolean`	Whether out-of-date checking is enabled for this engine
threshold	`integer`	Threshold in days for out-of-date checking
license	`boolean`	
available_disk_space_threshold	`integer`	Minimum free disk space required as a percentage of total disk. 0 = feature disabled (default).
scan_queue	`integer`	unit is percentage "%". To disable health check on scan queue status, set 0.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
ONE:OF	object	
OPTION:1	object	
err	string	Failed to set required_engines when license is not valid
OPTION:2	object	
err	string	Failed to set required_engines. Required_engines must be a licensed engine
OPTION:3	object	
err	string	Engine in outdated_engines is not in required_engines
OPTION:4	object	
err	string	Invalid value for available_disk_space_threshold

STATUS CODE - 500: Error while modifying configuration.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error while modifying configuration Setting timeout value out of the limit range [1, 30] minutes, or the request body is not in JSON format.

7.28 GET /admin/config/logger/logfile

Retrieve the log file configuration details

This API is for retrieving the current details of the log file configuration.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
logfile	`string`	The path of the log file.
loglevel	`enum`	ALLOWED: `debug, info, warning, error` The log file output level.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error while `<action>` configuration for local log configuration: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for local log configuration: <reason>"
}
```

7.29 PUT /admin/config/logger/logfile

Update the log file configuration

This API is used to update the log file configuration, including the logfile path and loglevel.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
logfile	string	The path of the log file. If not specified, the current value will be used. On Windows, the default value is `MetaDefender Core installation directory\data\logs\core.log`. On Linux, the default value is `/var/log/ometascan/ometascan.log`. The MD Core service will automatically restart to apply the change.
loglevel	enum	**ALLOWED:** debug, info, warning, error The log file output level. The default value is `info`

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
logfile	string	The path of the log file.
loglevel	enum	**ALLOWED:** debug, info, warning, error The log file output level.
is_restarting	boolean	Indicates whether the MD Core service is restarting to apply the change.
logfile_change_status	enum	**ALLOWED:** unchanged, success, failed The status of the log file path change.
message	string	The message indicating the change status.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflicting request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Logger configuration setting is processing, please wait for it to finish"
```

```
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error while `<action>` configuration for syslog: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for syslog: <reason>"
}
```

STATUS CODE - 501: Unsuported operation

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	This error indicates that you are trying to do an unsupported operation.

EXAMPLE:

```
{
 "err": "Log file path configuration is not supported in containerized deployments"
}
```

7.30 GET /admin/config/logger/syslog

Retrieve the syslog configuration details

This API is for retrieving the current details of the syslog configuration.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
syslog_level	`enum`	**ALLOWED:** `debug, info, warning, error` The syslog output level.
allow_self_signed_cert	`boolean`	Whether self-signed certificates are accepted for syslog servers using TLS connections
sources	`array`	
syslog	`string`	The address of the syslog server: `local"|"<protocol>://<address>:<port>`
status	`enum`	**ALLOWED:** `connecting, retrying, connected, ready` The status of the connection to the syslog server. connecting - attempting to establish a connection to the syslog server. retrying - connection to the syslog server failed. Retrying to establish the connection.... connected - successfully connected to the syslog server and ready to send log messages. ready - successfully connected to the local syslog server (for requests to add a 'local' syslog on Linux OS).
message	`string`	The message to describe information or error.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error while `<action>` configuration for syslog: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for syslog: <reason>"
}
```

7.31 PUT /admin/config/logger/syslog

Add new syslog servers or change syslog level

This API is used to add new syslog servers or to request a change in the syslog level. This will not impact any existing syslog servers. If a syslog server already exists, it will be ignored. If no syslog_level is specified in the API request header, the existing one will be retained. The default value for syslog_level is info.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
syslog_level	`enum`	**ALLOWED:** `debug, info, warning, error` The syslog output level. The default value is `info`
allow_self_signed_cert	`boolean`	This is optional - it defines whether to accept self-signed certificates for syslog servers using TLS connections. Default is false
sources	`array`	
syslog	`string`	The address of the syslog server: `local"|"<protocol>://<address>:<port>`. Supported protocols are TCP/UDP/TCPS (TLS over TCP)

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
syslog_level	`enum`	**ALLOWED:** `debug, info, warning, error` The syslog output level.
sources	`array`	
syslog	`string`	The address of the syslog server: `local"|"<protocol>://<address>:<port>`
result	`enum`	**ALLOWED:** `success, initializing_failed, existed, failed` Result of the request to add syslog servers. success - successfully added the syslog server. initializing_failed - failed to establish a connection to the syslog server and retrying... existed - successfully connected to the syslog server and ready to send log messages. failed - failed to add the syslog server while creating a local syslog on Windows OS or due to other errors (further investigation is needed to determine the cause).
message	`string`	The message to describe information or error.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflicting request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Logger configuration setting is processing, please wait for it to finish"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error while `<action>` configuration for syslog: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for syslog: <reason>"
}
```

7.32 POST /admin/config/logger/syslog

Forcefully apply the new syslog server configurations

This API is used to forcefully apply new syslog server configurations, replacing any existing configurations or creating a completely new syslog connection instance. If no syslog_level is

specified in the API request header, the current value will be retained. The default value for syslog_level is info.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
syslog_level	`enum`	**ALLOWED:** `debug, info, warning, error` The syslog output level. The default value is `info`
allow_self_signed_cert	`boolean`	This is optional - it defines whether to accept self-signed certificates for syslog servers using TLS connections. Default is false
sources	`array`	
syslog	`string`	The address of the syslog server: `local"|"<protocol>://<address>:<port>`. Supported protocols are TCP/UDP/TCPS (TLS over TCP)

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
syslog_level	`enum`	**ALLOWED:** `debug, info, warning, error` The syslog output level.
sources	`array`	
syslog	`string`	The address of the syslog server: `local"|"<protocol>://<address>:<port>`
result	`enum`	**ALLOWED:** `success, initializing_failed, existed, failed` Result of the request to add syslog servers. success - successfully added the syslog server. initializing_failed - failed to establish a connection to the syslog server and retrying... existed - successfully connected to the syslog server and ready to send log messages. failed - failed to add the syslog server while creating a local syslog on Windows OS or due to other errors (further investigation is needed to determine the cause).
message	`string`	The message to describe information or error.

STATUS CODE - 400: Invalid request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Request JSON body of logger configuration API is invalid"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflicting request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Logger configuration setting is processing, please wait for it to finish"
```

```
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error while `<action>` configuration for syslog: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for syslog: <reason>"
}
```

7.33 DELETE /admin/config/logger/syslog

Remove syslog servers or clear all syslog configurations

This API is used to request the removal of syslog servers or to clear all syslog configurations from the MD Core.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
ONE:OF	object	
OPTION:1	object	
sources	array	
syslog	string	The address of the syslog server: `local"|"<protocol>://<address>:<port>`
OPTION:2	object	
delete_all	boolean	Set this to true if you wish to completely remove the syslog configuration.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
syslog_level	`enum`	**ALLOWED:** `debug, info, warning, error` The current syslog output level.
sources	`array`	
syslog	`string`	The address of the syslog server: `local"|"<protocol>://<address>:<port>`
result	`enum`	**ALLOWED:** `success, initializing_failed, existed, failed` Result of the request to remove syslog servers. success - successfully removed the syslog server. not_existed - the specified address does not exist and cannot be removed failed - indicates that the syslog server could not be removed. This is a rare occurrence, potentially caused by an unhandled error. It should be reported for further investigation.
message	`string`	The message to describe the error.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflicting request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Logger configuration setting is processing, please wait for it to finish"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error while `<action>` configuration for syslog: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for syslog: <reason>"
}
```

7.34 GET /admin/config/logger/general

Retrieve the general logger configuration details

This API is for retrieving the current details of the general logger configuration.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
local_timezone	boolean	Enable local time zone for log timestamps. Default is `false`. **Note:** This field cannot be set to `true` when `log_format.format` is `rfc3164`. RFC-3164 format does not include timezone offset information, so local timezone is not supported to avoid timestamp confusion. The default timezone for RFC-3164 is UTC +00:00.
enable_cef	boolean	DEPRECATED **[DEPRECATED]** Enable CEF (Common Event Format) for log output. Default is `false`. This field is deprecated. Please use `log_format` instead for better flexibility and future format support. **Note:** Cannot be used together with `log_format` field.
log_format	object	
format	enum	**ALLOWED:** `cef, default, rfc3164, rfc5424` The log format to use. - `cef`: Common Event Format - `default`: Standard MetaDefender Core log format - `rfc3164`: BSD syslog format (RFC-3164). Note: `local_timezone` must be `false` when using this format as RFC-3164 does not include timezone offset information. Default timezone is UTC +00:00. - `rfc5424`: IETF syslog format (RFC-5424)
cef_ms_ts	boolean	Enable milliseconds in log timestamps. When enabled, timestamps will include milliseconds precision. Examples: - CEF format: `Sep 30 16:28:26.003 SE Asia Standard Time` - Syslog format: `2025-09-30T09:40:46.639Z` Default is `false`.
enable_override_log_id	boolean	Enable override of log ID for log entries. Default is `false`.
override_log_ids	string	Override log ID for log entries.

EXAMPLE:

```
{
 "local_timezone": false,
 "log_format": {
  "format": "cef",
  "cef_ms_ts": true
 },
 "enable_override_log_id": true,
 "override_log_ids": [
  "1723",
  "663:info"
 ]
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error while `<action>` configuration for local log configuration: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for local log configuration: <reason>"
}
```

7.35 PUT /admin/config/logger/general

Update the general logger configuration

This API is used to update the general logger configuration.

Validation Rules:

- enable_cef and log_format cannot be used together in the same request.
- When log_format.format is rfc3164, local_timezone cannot be set to true (RFC-3164 format does not include timezone offset information, so local timezone is not supported to avoid timestamp confusion. Default timezone is UTC +00:00).

Notes:

- The local_timezone field is automatically set to false upon first install.
- Configuration changes are applied dynamically without service restart.
- This feature is backward compatible - any old format and configuration will still work when upgrading.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
local_timezone	boolean	Enable local time zone for log timestamps. Default is `false`. **Note:** This field cannot be set to `true` when `log_format.format` is `rfc3164`. RFC-3164 format does not include timezone offset information, so local timezone is not supported to avoid timestamp confusion. The default timezone for RFC-3164 is UTC +00:00. MD Core will automatically set this value to `false` upon first install and change it when an API request is received to update it.
enable_cef	boolean	DEPRECATED **[DEPRECATED]** Enable CEF (Common Event Format) for log output. Default is `false`.

NAME	TYPE	DESCRIPTION
		This field is deprecated. Please use `log_format` instead for better flexibility and future format support. **Note:** Cannot be used together with `log_format` field.
log_format	`object`	
format	`enum`	**ALLOWED:** `cef, default, rfc3164, rfc5424` The log format to use. - `cef`: Common Event Format - `default`: Standard MetaDefender Core log format - `rfc3164`: BSD syslog format (RFC-3164). Note: `local_timezone` must be `false` when using this format as RFC-3164 does not include timezone offset information. Default timezone is UTC +00:00. - `rfc5424`: IETF syslog format (RFC-5424)
cef_ms_ts	`boolean`	Enable milliseconds in log timestamps. When enabled, timestamps will include milliseconds precision. Examples: - CEF format: `Sep 30 16:28:26.003 SE Asia Standard Time` - Syslog format: `2025-09-30T09:40:46.639Z` Default is `false`.
enable_override_log_id	`boolean`	Enable override of log ID for log entries. Default is `false`.
override_log_ids	`string`	Override specific log ids to display them on another level, e.g.: "1723:error,663:info". If there is no level set for an id, it will be displayed on every occasion. e.g.: "1723,663:info" means id 1723 dump message will be displayed every time and id 663 warning message is reduced to info level.

EXAMPLE 1:

```
"usingLogFormatDefault"
```

EXAMPLE 2:

```
"usingLogFormat"
```

EXAMPLE 3:

```
"usingRfc3164Format"
```

EXAMPLE 4:

```
"usingRfc5424Format"
```

EXAMPLE 5:

```
"legacyEnableCef"
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
local_timezone	boolean	Enable local time zone for log timestamps. **Note:** This field will be `false` when `log_format.format` is `rfc3164`.
enable_cef	boolean	DEPRECATED **[DEPRECATED]** Enable CEF (Common Event Format) for log output. This field is deprecated. Please use `log_format` instead for better flexibility and future format support. **Note:** Cannot be used together with `log_format` field.
log_format	object	
format	enum	ALLOWED: `cef, default, rfc3164, rfc5424` The log format to use. - `cef`: Common Event Format - `default`: Standard MetaDefender Core log format - `rfc3164`: BSD syslog format (RFC-3164). Note: `local_timezone` will be `false` when using this format. - `rfc5424`: IETF syslog format (RFC-5424)
cef_ms_ts	boolean	Enable milliseconds in log timestamps. When enabled, timestamps will include milliseconds precision. Examples: - CEF format: `Sep 30 16:28:26.003 SE Asia Standard Time` - Syslog format: `2025-09-30T09:40:46.639Z` Default is `false`.
enable_override_log_id	boolean	Enable override of log ID for log entries.
override_log_ids	string	Override specific log ids to display them on another level e.g.: "1723:error,663:info".

STATUS CODE - 400: Bad Request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason. Common errors include: - `Cannot set both enable_cef and log_format` - when both `enable_cef` and `log_format` fields are provided in the same request - `Cannot enable local timezone feature for RFC-3164 log format` - when `log_format.format` is `rfc3164` and `local_timezone` is set to `true` - Invalid header or apikey issues

EXAMPLE 1:

```
"mutualExclusivity"
```

EXAMPLE 2:

```
"rfc3164LocalTimezoneConflict"
```

EXAMPLE 3:

```
"invalidHeader"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflicting request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Logger configuration setting is processing, please wait for it to finish"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error while `<action>` configuration for syslog: `<reason>` action - can be `getting`, `setting`, `adding`, `deleting`, `verifying` reason - is the unhandled error reason

EXAMPLE:

```
{
 "err": "Error while <action> configuration for syslog: <reason>"
}
```

8. DATABASE

8.1 POST /dbconfig/changepasswdpgconnection

Change PostgreSQL Connection Password

Changes the password for the PostgreSQL database connection. Requires an API key and pin code authentication.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
new_password*	string	The new password for PostgreSQL connection
pin_code*	string	Pin code for authentication
is_changed_pg_password*	boolean	Flag indicating if the password should be changed in the PostgreSQL database also

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Password changed successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

9. ENGINES

Enable/disable or pin/unpin the engines through API.

9.1 POST /admin/engine/{engineId}/pin

Pin engine to prevent auto-updates

Pin engines to prevent applying automatic updates on them. Manual updates still can be applied.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	string		The unique engine identifier

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
type	enum ALLOWED: engine, database		Pin engine or database to prevent applying automatic updates on it. (If the type is not defined both engine and database will be pinned.)

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	enum	ALLOWED: Engine is pinned, Database is pinned, Both engine and database are pinned

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

9.2 POST /admin/engine/{engineId}/unpin

Unpin engine to prevent auto-updates

Unpin engines so automatic updates will be applied on them.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	string		The unique engine identifier

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
type	enum ALLOWED: engine, database		Unpin engine or database to applying automatic updates on it. (If it is not defined both engine and database will be unpinned.)

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	enum	ALLOWED: Engine is unpinned, Database is unpinned, Both engine and database are unpinned

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

9.3 POST /admin/engine/{engineId}/enable

Enable engines

Enable to use the selected engine on the nodes.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	string		The unique engine identifier

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

9.4 POST /admin/engine/{engineId}/disable

Disable engines

Disable to use the selected engines on the nodes.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	string		The unique engine identifier

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

`9.5 GET /admin/engine/{engineId}/config`

Retrieve engine configuration

Retrieve a list of advanced configuration of an engine.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	`string`		The unique engine identifier

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

9.6 PUT /admin/engine/{engineId}/config

Update engine configuration

Modifying advanced configurations of an engine.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	string		The unique engine identifier

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

9.7 DELETE /admin/engine/{engineId}

Delete an engine or database package

Delete an engine or database package based on engine ID. Examples:

- For Windows OS: filetype_*_windows
- For Linux OS: filetype_*_linux

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*engineId	string		ID of the engine to delete

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		API key for authentication
keep_advanced_config	number		Optional - whether to keep advanced configuration of engine (0 or 1)
*type	string		Type of package to delete - '*' : Delete both engine and database packages - 'engine' : Delete engine package - 'database' : Delete database package

RESPONSE

STATUS CODE - 200: Engine deleted successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
database	object	
err	string	
succeed	boolean	
engine	object	
err	string	
succeed	boolean	

EXAMPLE:

```
{
 "database": {
  "err": "",
  "succeed": true
 },
 "engine": {
  "err": "",
  "succeed": true
 }
}
```

STATUS CODE - 400: Invalid request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error message describing the issue

EXAMPLE 1:

```
"invalidHeaders"
```

EXAMPLE 2:

```
"engineRemovalError"
```

STATUS CODE - 403: Access denied

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Access denied"
}
```

10. LICENSE

Activate the product or get licensing information. Will require admin apikey.

10.1 POST /admin/license/activation

Activate license online

This API initiates an online activation of the deployment. In case of error, check the application logs for more details.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
activationKey*	string	The license key provided by OPSWAT (either eval or production license).
quantity*	integer	The number of MetaDefender nodes attached to this instance (must be equal or greater than 1).
comment	string	Metadata field allowing you to provide relevant identification information for this instance.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	boolean	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

10.2 GET /admin/license

Get current license information

Fetch all details about the licensing status of the product.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Information about the licensed product (product type, number of activations, deploymentId, expiration date and days left)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
days_left	integer	Number of days left before expiration
deployment	string	Unique identifier which is maps the current deployment to the activation
expiration	string	Expiration date in MM/DD/YYYY format.
licensed_engines	array	
max_node_count	string	Total number of deployed MetaDefender Nodes attached to this MetaDefender Core instance.
online_activated	boolean	Track online/offline activation mode
product_id	string	Official MetaDefender base SKU licensed.
product_name	string	Official MetaDefender base product name licensed.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

10.3 POST /admin/license

Upload license key file

Uploading a license file to the Metadefender Core.

There are two ways two obtain a license key file:

- via https://portal.opswat.com/activation portal
- via activation server REST API: https://activation.dl.opswat.com/activation?key=%activation_key%&deployment=%deployment_unique_ID%&quantity=%quantity%

Deployment unique ID can be fetched via Get Current License Information API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	boolean	

STATUS CODE - 400: Invalid license file

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "err": "Invalid license"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

10.4 POST /admin/license/deactivation

Deactivate license

This API is used for deactivating the license of a deployment. In case of error, please check the application logs for more detailed information.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
success	boolean	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

11. MANAGEMENT

11.1 POST /admin/update/folder

Force update folder trigger

Triggers an update folder operation even during pause timeframe when update source is configured as 'Local folder'

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		API key for authentication

RESPONSE

STATUS CODE - 200: Successful request

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Update source is not 'folder'

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

11.2 PUT /pincode

Create new pincode or update pincode

Creates a new PIN code (when no PIN code exists) or updates an existing PIN code. For creating a new PIN code, only provide new_pin_code. For updating an existing PIN code, provide both new_pin_code and old_pin_code.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
new_pin_code*	`string`	The new pin code to set
old_pin_code	`string`	The current pin code for verification (required only when updating an existing PIN code)

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Pin code updated successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
new_pin_code	`string`	The new pin code that was set
old_pin_code	`string`	The previous pin code (if an update was performed)

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
```

```
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server - or wrong pincode

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Error while setting pin code"
}
```

11.3 HEAD /pincode

Check if a pincode exists

Checks for the existence of a pincode without returning the resource itself

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Pincode exists

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 404: Pincode does not exist

11.4 POST /admin/pincode/validate

Validate PIN code

Validates a PIN code with authentication required. This API requires admin role and API key authentication. Rate limited to 3 requests per minute.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
pin_code*	`string`	PIN code to check. 6 digits long.

EXAMPLE:

```
{
 "pin_code": 111111
}
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: PIN code is valid and matches

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
pin_code	string	

EXAMPLE:

```
{
 "pin_code": 111111
}
```

STATUS CODE - 400: Bad request - various validation errors

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error message describing the issue

EXAMPLE 1:

```
"parsing_error"
```

EXAMPLE 2:

```
"incorrect_pin"
```

EXAMPLE 3:

```
"pin_not_set"
```

EXAMPLE 4:

```
"wrong_format"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 503: Service unavailable. API call exceeded rate limit.

12. MULTIPART

Upload a file for scan by multiple parts.

12.1 POST /file/multipart

Create multipart uploading session

Note: Only available starting with MetaDefender Core 5.11.0. The timeout value could be configurable

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*total-length	integer		Size of the whole submitted file in bytes
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
filename	string		Specify name of file captured and displayed on corresponding MetaDefender Core scan result
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
batch	string		Batch id to scan with, coming from `Initiate Batch` (If it is not given, it will be a single file scan.)

NAME	TYPE	EXAMPLE	DESCRIPTION
archivepwd	string		Password for archive (URL encoded UTF-8 string) Multiple passwords is also supported, format: archivepwdX * X: Could be empty * When having value, X must be a number >= 1 For example: * archivepwd1: "fox" * archivepwd2: "cow" * archivepwd3: "bear"
metadata	json		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters. The `source_system` key is reserved for the system that submitted the file; it can be used to filter `Active performance`. When present, its value must be a string of 1 to 128 letters, digits, dots, underscores, colons or hyphens, otherwise the request is rejected with `400`.
engines-metadata	json		Since MetaDefender Core 5.0.0, preferred context / verbose information can be sent to the engines. Please see the below pages for the details: * [File Type engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata) (supported since Core 5.2.1) * [Archive engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata-header) (supported since Core 5.4.1) * [Deep CDR](https://docs.opswat.com/mdcore/deep-cdr/supported-engines-metadata-json) * [Proactive DLP](https://docs.opswat.com/mdcore/proactive-dlp/supported-engines-metadata-json)

NAME	TYPE	EXAMPLE	DESCRIPTION
`callbackurl`	`uri`		Client's URL where MetaDefender Core will notify scan result back to whenever scan is finished (webhooks model). * Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> * Example: http://10.0.1.100:8081/listenback * Supported protocol: HTTP / HTTPS * Method: POST
`sanitizedurl`	`uri`		Client's URL where MetaDefender Core will send the sanitized file as the body of a POST request back to whenever scan is finished (webhooks model). * Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> * Example: http://10.0.1.100:8081/listensanitizedfile * Supported protocol: HTTP / HTTPS * Method: POST * Mandatory headers: * `user-agent`: "OPSWAT MetaDefender Core V4" * `dataid`: "<data id of original file submission>" * `sanitization-result`: "Sanitized Successfully" or "Sanitized Partially" or "Sanitized File Removed" * `sanitized-file-hash`: "<SHA256 hash value of sanitized file>" * `Content-Disposition`: attachment; filename="<name of sanitized file>" * `Content-Length`: <actual size of sanitized file>
`global-timeout`	`integer`		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

NAME	TYPE	EXAMPLE	DESCRIPTION

client-identity	json		The `client-identity` helps track and understand the workflow that a file passes through clients, offering greater visibility into the processing stages and the clients involved.
			This header is provided as a JSON object containing an array of client identity objects. Each object in the array should include the following fields:
			* `deployment_id`: *REQUIRED* | Unique identifier for the deployment.
			* `user_name`: *REQUIRED* | Name of the user or system that initiated the scan.
			* `product_type`: *REQUIRED* | Type of product or system that is performing the scan.
			* `version`: *OPTIONAL* | Version of the product or system.
			* `host`: *OPTIONAL* | Hostname or IP address of the client machine.
			* `timestamp`: *REQUIRED* | Timestamp of when the scan was initiated, in milliseconds since epoch.
			Example:
			``json
			{
			"clients":
			[
			{
			"deployment_id": "abcdef1234567890",
			"user_name": "Gemini",
			"product_type": "MetaDefender Kiosk",
			"version": "1.0.0",
			"host": "client.example.com",
			"timestamp": 1744949754890
			}
			]
			}
			``
			It should be performed the *URL-Encoding* before sending to MetaDefender Core to prevent unexpected issues related to encoding errors or unsafe characters.
			This header is optional and can be omitted if not needed.

RESPONSE

STATUS CODE - 200: Successful creation of multipart uploading session

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id*	string	Unique submission identifier. Use this value to reference the upload and submission.

EXAMPLE:

```
{
 "data_id": "62dffeaa728944adbf49eb090e4ece0f"
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Invalid apikey given
OPTION:2	object	
err	string	Batch closed during file upload.
OPTION:3	object	
err	string	Can not scan in given batch.
OPTION:4	object	
err	string	No available multipart session is present for uploading.
OPTION:5	object	
err	string	Invalid engine metadata.
OPTION:6	object	
err	string	Invalid callback URL.
OPTION:7	object	
err	string	Null metadata is not allowed.
OPTION:8	object	
err	string	Invalid metadata format.
OPTION:9	object	
err	string	The `source_system` value in the `metadata` header is invalid.

STATUS CODE - 403: Unavailable workflow rule.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Request is refused because of unavailable workflow rule.

STATUS CODE - 415: API not supported.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	

EXAMPLE:

```
{
 "err": "API is not supported"
}
```

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	License is expired
OPTION:2	object	
err	string	No available rule is present for scanning.
OPTION:3	object	
err	string	Failed to request scan. Try again later.
OPTION:4	object	
err	string	Failed to request scan. File size exceeded the maximum size permitted by your configuration.

STATUS CODE - 503: Server is too busy, scan queue is full. Try again later.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
 "err": "Server is too busy. Try again later."
}
```

12.2 GET /file/multipart/{data_id}

Fetch uploading status

Fetch status of uploading a file by multiple parts

Note:

- Sticky session must be provided if this API calls to a Load Balancer that is in front of multiple MetaDefender Core instances. The session is responded by Load Balancer from POST /file/multipart API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Uploading status fetched successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	Data identifier of the requested file.
status	string	Uploading status, can be one of `completed`, `uploading`, `timed out`, or `canceled`.
total_length	number	The size of whole file in bytes.
ranges_received	string	A list of parts that have been received successfully so far, in format of `<start_1>-<end_1>, <start_2>-<end_2>`.

EXAMPLE:

```
{
 "data_id": "62dffeaa728944adbf49eb090e4ece0f",
 "status": "uploading",
 "total_length": 10485760,
 "ranges_received": "0-2000000, 4000000-10485759"
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Cannot get multipart status from another instance.

STATUS CODE - 404: Not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Multipart uploading session not found.

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Failed to request scan. Try again later.

12.3 POST /file/multipart/{data_id}

Upload individual part of file

Upload an individual part of file to the created multipart uploading session

Note:

- Invidual parts can be uploaded simultaneously.
- When all parts are fully received, MetaDefender Core will automatically combine them into a single file and start processing it.
- Clients are free to upload broken parts again via this API if the network or service is interrupted.
- Sticky session must be provided if the part is uploaded to a Load Balancer that is in front of multiple MetaDefender Core instances. The session is responded by Load Balancer from POST /file/multipart API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*content-range	string		Offset and size of the uploaded part, in format of `offset-size/total_size`, in which * `offset`: the offset of uploaded part in whole file. * `size`: the size of uploaded part in bytes. * `total_size`: the size of whole file in bytes. For example: the submitted file is of 10,485,760 bytes (10MBs) and divided into 3 parts. Parts are uploaded with ranges as following: * Part 01: 0-2000000/10485760 * Part 02: 2000000-2000000/10485760 * Part 03: 4000000-6485760/10485760 _**Note**_: An uploaded part can overlap its next and previous.
*content-length	integer		Size of request body in bytes
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Part uploaded successfuly

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
ranges_received*	string	A list of parts that have been received successfully so far, in format of `<start_1>-<end_1>, <start_2>-<end_2>`.

EXAMPLE:

```
{
 "ranges_received": "0-2000000, 4000000-10485759"
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Invalid apikey given
OPTION:2	object	
err	string	Invalid content type.
OPTION:3	object	
err	string	Cannot upload part from another instance.

STATUS CODE - 404: Not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Multipart uploading session not found.

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflict.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Multipart uploading session timed out.
OPTION:2	object	
err	string	Multipart uploading session has already finished.

STATUS CODE - 416: Range not satisfiable.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Invalid content range.
OPTION:2	object	
err	string	The part is duplicated.

STATUS CODE - 422: Cancelled.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Multipart uploading session cancelled.

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Failed to request scan. Try again later.

12.4 DELETE /file/multipart/{data_id}

Abort multipart uploading

Abort uploading a file by multiple parts

Note:

- Sticky session must be provided if this API calls to a Load Balancer that is in front of multiple MetaDefender Core instances. The session is responded by Load Balancer from POST /file/multipart API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Uploading session aborted successfully

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "62dffeaa728944adbf49eb090e4ece0f": "cancelled"
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Invalid apikey given
OPTION:2	object	
err	string	Cannot abort from another instance.

STATUS CODE - 404: Not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Multipart uploading session not found.

STATUS CODE - 409: Conflict.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Multipart uploading session timed out.
OPTION:2	object	
err	string	Multipart uploading session has already finished.
OPTION:3	object	
err	string	Multipart uploading session already cancelled.

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Failed to request scan. Try again later.

13. SPLITARCHIVE

Processes a split archive uploaded in multiple parts. Only the extracted files from the reassembled split archive are processed; the combined archive file itself is not supported. Therefore, Quarantine and PostAction do not apply.

13.1 POST /file/splitarchive

Create split archive upload session

Creates a new split archive upload session and returns a unique data_id that serves as the bearer token for subsequent part uploads.

Note:

- Only available starting with MetaDefender Core 5.17.0.
- Requires database mode: STANDALONE_MODE (1) or SHARED_MODE (4).
- Session has absolute timeout deadline (default: 60 minutes from initialization, configurable).
- Requires Archive Extraction engine with split archive capability enabled.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*total-parts	integer >=2		Total number of parts that will be uploaded (positive integer, minimum 2)
total-length	integer >=1		Total size of the file in bytes (for validation purposes)
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

NAME	TYPE	EXAMPLE	DESCRIPTION
filename	string		Specify name of file captured and displayed on corresponding MetaDefender Core scan result
user_agent	string		User agent string used to identify the client application
rule	string		Optional. Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
batch	string		Batch id to scan with, coming from `Initiate Batch` (If it is not given, it will be a single file scan.)
archivepwd	string		Password for archive (URL encoded UTF-8 string) Multiple passwords is also supported, format: archivepwdX * X: Could be empty * When having value, X must be a number >= 1 For example: * archivepwd1: "fox" * archivepwd2: "cow" * archivepwd3: "bear"
metadata	json		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters. The `source_system` key is reserved for the system that submitted the file; it can be used to filter `Active performance`. When present, its value must be a string of 1 to 128 letters, digits, dots, underscores, colons or hyphens, otherwise the request is rejected with `400`.

NAME	TYPE	EXAMPLE	DESCRIPTION
`engines-metadata`	`json`		Since MetaDefender Core 5.0.0, preferred context / verbose information can be sent to the engines. Please see the below pages for the details: * [File Type engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata) (supported since Core 5.2.1) * [Archive engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata-header) (supported since Core 5.4.1) * [Deep CDR](https://docs.opswat.com/mdcore/deep-cdr/supported-engines-metadata-json) * [Proactive DLP](https://docs.opswat.com/mdcore/proactive-dlp/supported-engines-metadata-json)
`callbackurl`	`uri`		Client's URL where MetaDefender Core will notify scan result back to whenever scan is finished (webhooks model). * Format: \<protocol://\>\<ip | domain\>:\<port\>\</path\> * Example: http://10.0.1.100:8081/listenback * Supported protocol: HTTP / HTTPS * Method: POST
`global-timeout`	`integer`		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

NAME	TYPE	EXAMPLE	DESCRIPTION

client-identity	json		The `client-identity` helps track and understand the workflow that a file passes through clients, offering greater visibility into the processing stages and the clients involved.
			This header is provided as a JSON object containing an array of client identity objects. Each object in the array should include the following fields:
			* `deployment_id`: *REQUIRED* | Unique identifier for the deployment.
			* `user_name`: *REQUIRED* | Name of the user or system that initiated the scan.
			* `product_type`: *REQUIRED* | Type of product or system that is performing the scan.
			* `version`: *OPTIONAL* | Version of the product or system.
			* `host`: *OPTIONAL* | Hostname or IP address of the client machine.
			* `timestamp`: *REQUIRED* | Timestamp of when the scan was initiated, in milliseconds since epoch.
			Example:
			``json
			{
			"clients":
			[
			{
			"deployment_id": "abcdef1234567890",
			"user_name": "Gemini",
			"product_type": "MetaDefender Kiosk",
			"version": "1.0.0",
			"host": "client.example.com",
			"timestamp": 1744949754890
			}
			]
			}
			``
			It should be performed the *URL-Encoding* before sending to MetaDefender Core to prevent unexpected issues related to encoding errors or unsafe characters.
			This header is optional and can be omitted if not needed.

RESPONSE

STATUS CODE - 200: Successful creation of split archive upload session

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id*	string	Unique session identifier (32 hexadecimal characters). Use this value to reference the upload and submission.

EXAMPLE:

```
{
 "data_id": "a1b2c3d4e5f67890abcdef0123456789"
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Invalid apikey given
OPTION:2	object	
err	string	Batch closed during file upload.
OPTION:3	object	
err	string	Can not scan in given batch.
OPTION:4	object	
err	string	Invalid engine metadata.
OPTION:5	object	
err	string	Invalid callback URL.
OPTION:6	object	
err	string	Null metadata is not allowed.
OPTION:7	object	
err	string	Invalid metadata format.
OPTION:8	object	
err	string	The `source_system` value in the `metadata` header is invalid.
OPTION:9	object	
err	enum	**ALLOWED:** `Invalid client identity, Failed to request scan. Total parts exceeded the maximum number permitted by your configuration, Failed to request scan. Total parts is less than the minimum number 2, Failed to request scan. Total parts is missing or invalid`

STATUS CODE - 403: Unavailable workflow rule.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Request is refused because of unavailable workflow rule.

STATUS CODE - 415: API not supported.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: API is not supported, Split archive upload session cancelled: Archive Extraction engine does not support split archive feature

EXAMPLE:

```
{
 "err": "API is not supported"
}
```

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	License is expired
OPTION:2	object	
err	string	No available rule is present for scanning.
OPTION:3	object	
err	string	Failed to request scan. Try again later.
OPTION:4	object	
err	enum	ALLOWED: Split archive upload session cancelled: file size exceeded the maximum size permitted by your configuration, Not enough disk space

STATUS CODE - 503: Server is too busy or extraction engine unavailable.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Server is too busy. Try again later., Archive Extraction engine is not active. Unable to process, FileType engine is not active. Unable to process

EXAMPLE:

```
{
 "err": "Server is too busy. Try again later."
}
```

13.2 GET /file/splitarchive/{data_id}

Fetch split archive upload status

Fetch status of uploading a split archive file by multiple parts

Note:

- Sticky session must be provided if this API calls to a Load Balancer in front of multiple MetaDefender Core instances.
- Use this endpoint to implement resume logic after crashes or network failures.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string PATTERN: ^[a-f0-9]{32}$		Session identifier (32 hexadecimal characters)

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Uploading status fetched successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id*	string	Session identifier (echoed from request).
parts_received	array	

NAME	TYPE	DESCRIPTION
status*	`enum`	**ALLOWED:** `uploading, completed, timed out, cancelled` Session state: - `uploading`: Session is active and accepting parts - `completed`: All parts received and successfully combined - `timed out`: Session exceeded timeout deadline - `cancelled`: Session aborted via DELETE API
submit_deadline	`string`	**Only present when status is "uploading".** ISO 8601 timestamp when session will timeout.
total_parts*	`integer`	Total number of parts expected (from Initialize API).

EXAMPLE 1:

```
"uploading"
```

EXAMPLE 2:

```
"completed"
```

EXAMPLE 3:

```
"timed_out"
```

EXAMPLE 4:

```
"cancelled"
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`enum`	**ALLOWED:** `Cannot get split archive upload session status from another instance`

STATUS CODE - 404: Not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`enum`	**ALLOWED:** `Split archive upload session not found`

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 415: Unsupported Media Type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: API is not supported

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Failed to request scan. Try again later.

13.3 POST /file/splitarchive/{data_id}

Upload individual part of split archive

Upload an individual part of split archive to the created upload session

Note:

- Parts can be uploaded in any order (1-based numbering).
- When all parts are fully received, MetaDefender Core will automatically start processing it.
- Clients can re-upload parts if network or service is interrupted (duplicate part upload returns HTTP 416).
- Sticky session must be provided if uploading to a Load Balancer in front of multiple MetaDefender Core instances.
- Session has absolute timeout deadline that is never extended by part uploads.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string PATTERN: ^[a-f0-9]{32}$		Session identifier returned from Initialize API (32 hexadecimal characters)

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*part-number	integer >=1		Part number being uploaded (1-based positive integer, must be ≤ total-parts declared during initialization)
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Part uploaded successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
parts_received*	array	

EXAMPLE:

```
{
 "parts_received": [
  1,
  2,
  3,
  5
 ]
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Header 'part-number' is required, Invalid apikey given, Cannot upload part from another instance

STATUS CODE - 404: Not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Split archive upload session not found

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflict - Session in terminal state.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Split archive upload session already timed out, Split archive upload session already finished, Split archive upload session already cancelled

STATUS CODE - 415: Unsupported Media Type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: API is not supported, Archive Extraction engine does not support split archive feature

STATUS CODE - 416: Range not satisfiable - Invalid or duplicate part number.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Invalid part number: must be a positive integer, Invalid part number: out of range 1 to total parts, Part number already received

STATUS CODE - 422: Unprocessable Entity - Invalid archive format.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	enum	ALLOWED: Split archive upload session cancelled: first part is not a supported split archive format

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ONE:OF	object	
OPTION:1	object	
err	string	Failed to request scan. Try again later.
OPTION:2	object	
err	enum	ALLOWED: Failed to request scan. File size exceeded the maximum size permitted by your configuration., Not enough disk space

STATUS CODE - 503: Service Unavailable - Extraction engine not active.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Archive Extraction engine is not active. Unable to process, FileType engine is not active. Unable to process

13.4 DELETE /file/splitarchive/{data_id}

Abort split archive upload

Abort uploading a split archive file by multiple parts and delete all uploaded parts

Note:

- Sticky session must be provided if this API calls to a Load Balancer in front of multiple MetaDefender Core instances.
- This operation is irreversible. The session transitions to "cancelled" state and cannot accept new parts.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string PATTERN: ^[a-f0-9]{32}$		Session identifier (32 hexadecimal characters)

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Uploading session aborted successfully

RESPONSE MODEL - application/json

EXAMPLE:

```
{
 "a1b2c3d4e5f67890abcdef0123456789": "cancelled"
}
```

STATUS CODE - 400: Bad request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Invalid apikey given, Cannot abort from another instance

STATUS CODE - 404: Not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	ALLOWED: Split archive upload session not found

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 409: Conflict - Cannot abort terminal state session.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	**ALLOWED:** `Split archive upload session already finished, Split archive upload session already timed out, Split archive upload session already cancelled`

STATUS CODE - 415: Unsupported Media Type.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	enum	**ALLOWED:** `API is not supported`

STATUS CODE - 500: Internal Server Error.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Failed to request scan. Try again later.

14. STATS

Health check and statistics about MetaDefender Core usage.

14.1 GET /admin/stat/server

Get server statistics

Retrieve server statistics including CPU load, disk space information, and memory usage.

Units:

- cpu_load: percentage
- disk_space.free: bytes
- disk_space.total: bytes
- memory.available_memory: bytes
- memory.total_memory: bytes

The disk_space array will contain information for each drive on the system.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
info	enum ALLOWED: memory, cpu_load, disk_space	memory	Optional. Show specific information categories. Available options: - memory: show memory information only - cpu_load: show CPU load information only - disk_space: show disk space information only If not provided, all information will be returned.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Server statistics retrieved successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cpu_load	integer	CPU load percentage
disk_space	array	
dirs	array	
free	integer	Free disk space in bytes
location	string	Drive letter or location
total	integer	Total disk space in bytes
memory	object	
available_memory	integer	Available memory in bytes
total_memory	integer	Total memory in bytes

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

14.2 GET /version

Get Product Version

Fetch details about the product version.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Provides productId and version information.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
product_id	`string`	Product identifier.
version	`string`	The product version currently used.

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

14.3 GET /stat/engines

Engines Status

The response is an array of engines with database information.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: An array with all the engines and their details.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
active	boolean	If used by at least one engine
def_time	string	The database definition time for this engine
download_progress	integer	The percentage progress of download
download_time	string	When this engine downloaded from the update server.
eng_id	string	Engine internal ID
eng_name	string	Engine name
eng_type	string	Engine type in human readable form

NAME	TYPE	DESCRIPTION
eng_ver	`string`	Engine's version (format differs from one engine to another).
engine_type	`enum`	**ALLOWED:** `av, archive, filetype` Engine's type: * av * archive * filetype
last_update_triggered	`string`	When the most recent engine update cycle was started on this Core instance, either manually (Update Now / Update All) or by the automatic update schedule. This is a single instance-wide value, so every engine entry carries the same timestamp.
state	`enum`	**ALLOWED:** `downloading, downloaded, staging, production, removed, temporary failed, permanently failed, content invalid, download failed` Status of the engine: * downloading * downloaded * staging * production * removed * temporary failed * permanently failed * content invalid * download failed
supported	`boolean`	Indicates whether the engine package is compatible with the current version of MetaDefender Core: * `true` - the engine package is compatible with the current Core * `false` - the engine package is not compatible with the current Core

14.4 GET /stat/nodes

Core Status

Get a list of connected Core and status.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	`string`		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: An array with all the engines and their details.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
external_nodes_allowed	`boolean`	Configuration option if additional external nodes could be connected.
max_node_count	`integer`	Remaining available slots to connect to this instance.
statuses	`array`	
address	`string`	The location of the Core. If local, is empty string.
available_mem	`integer`	The number of available RAM in this system.
cpu_cores	`integer`	The number of CPU Cores allocated to this Core instance.
engines	`array`	
active	`boolean`	If used by at least one engine
db_ver	`string`	The database version for this engine
def_time	`string`	The database definition time for this engine
eng_name	`string`	Engine name
eng_ver	`string`	Engine's version (format differs from one engine to another).
engine_type	`enum`	**ALLOWED:** `av, archive, filetype` Engine's type: * av * archive * filetype
id	`string`	Engine internal ID
free_disk_space	`integer`	Reported available disk on Core instance (in bytes).
id	`string`	Core identfier
info_disk_space	`array`	
dirs	`array`	
disk_stat_info	`object`	
disk_utilization	`number`	Disk utilization percentage
read_speed	`number`	Disk read speed (MB/s)
write_speed	`number`	Disk write speed (MB/s)
free	`integer`	Free disk space in bytes
location	`string`	Disk location identifier
total	`integer`	Total disk space in bytes
issues	`array`	
description	`string`	Text detailing the issue.
severity	`string`	How important is the reported issue.
load	`integer`	Current CPU utilization on Core instance (in percentage).

NAME	TYPE	DESCRIPTION
os	string	Current used OS
scan_queue	integer	Current load on the Core instance, how many files are in the queue
scan_queue_details	object	
total_scan_queue	integer	Total scan queue
file_slots	integer	The number of slots is taken by files scan from REST API
extracted_file_slots	integer	The number of slots is taken by extracted files
available_slots	integer	The number of slots is available
total_disk_space	integer	The amount of disk space is allocated on Core instance (in Byte).
total_mem	integer	How much memory is allocated on Core instance (in MB).
total_scan_queue	integer	The maximum queue size is allowed.
uptime	integer	How long this Core is in operation (in second).
version	string	Product version

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

14.5 GET /stat/activeperformance

Active performance

Get active performance statistics.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
hours_range	integer		Collect data from last hours

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule_name	string		The name of workflow rule to filter.
metadata_key	enum ALLOWED: source_system	source_system	Limit the statistics to files submitted with a given `metadata` value. The only supported key is `source_system`, matched against the `source_system` field of the `metadata` header sent with `Analyze a file`. Must be sent together with `metadata_value`.
metadata_value	string PATTERN: ^[A-Za-z0-9._:-]{1,128}$	erp-system	Value of `metadata_key` to filter on: 1 to 128 letters, digits, dots, underscores, colons or hyphens. Must be sent together with `metadata_key`. When the filter is applied, `success_results` and `failed_results` are not returned.

RESPONSE

STATUS CODE - 200: Active performance report on MetaDefender Core

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
active_performance	object	
failed_results	integer	Number of failed processing results.
hash_time	object	

NAME	TYPE	DESCRIPTION
avg	integer	Average hash calculating time.
max	integer	Maximum hash calculating time.
min	integer	Minimum hash calculating time.
processing_time	object	
avg	integer	Average overall processing time.
max	integer	Maximum overall processing time.
min	integer	Minimum overall processing time.
queue_time	object	
avg	integer	Average queue time.
max	integer	Maximum queue time.
min	integer	Minimum queue time.
success_results	integer	Total number of successfully-processed objects (excluding failed or not-scanned results)
total_filesize	integer	Total processed file size in bytes.
total_requests	integer	Total number of file submissions.
wait_time	object	
avg	integer	Average waiting time.
max	integer	Maximum waiting time.
min	integer	Minimum waiting time.
from_time	integer	The data is gathered in this recent milliseconds.

STATUS CODE - 400: Invalid metadata filter:
* `Both headers metadata_key and metadata_value are required`
* `Header metadata_key is not supported. Must be source_system`
* `Value of metadata_value is invalid. Must be 1 to 128 characters of letters, digits, dot, underscore, colon or hyphen`

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Both headers metadata_key and metadata_value are required"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

14.6 GET /stat/user/totalprocess

Retrieve file scan history per user

Retrieve the complete scan history of files or objects for each user within a specified time range.

Time range parameters:

- Use hours_range to collect data from the last N hours (0–17520).
- Use
  from
  and
  to
  to specify an explicit time range. Supported formats:
  - **Epoch timestamp (hour):** hours since Unix epoch (e.g. 470418)
  - **ISO 8601 (hour precision):** YYYY-MM-DDThh for local time, or YYYY-MM-DDThhZ for UTC (e.g. 2023-09-01T01 or 2023-09-01T01Z)

- If none of hours_range, from, or to are provided, the last 24 hours are returned.
- Maximum supported time window is **2 years**.
- hours_range and from/to cannot be used simultaneously.

Permission:

- **Executive report (Read-Only|Full) + User management (Read-Only|Full):** Allows viewing details for all users who have submitted file scan requests.
- **Executive report only (Read-Only|Full):** Allows viewing details for the current user (the requester) only.

Notes:

- Users can have one of the following states: Active, Deleted, or Remote.
- Users authenticated via AD/LDAP groups or SAML/OIDC directories are assigned the Remote state.
- Only users who have submitted file scan requests are included in the results.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
`hours_range`	`integer` between 0 and 17520		Collect data from the last N hours. Valid range: 0–17520 (up to 2 years). Cannot be used together with `from`/`to`.

NAME	TYPE	EXAMPLE	DESCRIPTION
request_type	enum ALLOWED: object, file		Choose the report type. Valid values: `object` or `file` (case-insensitive). Defaults to `object`.
view	enum ALLOWED: all		In shared database mode, specifies whether to query data from all instances or only the current instance. If not provided, only the current instance is queried. Specify `all` to aggregate across all MD Core instances.
from	string		Start of the time range. Accepts epoch timestamp in hours (e.g. `470418`) or ISO 8601 hour-precision format: `YYYY-MM-DDThh` (local time) or `YYYY-MM-DDThhZ` (UTC). Cannot be used together with `hours_range`.
to	string		End of the time range. Accepts epoch timestamp in hours (e.g. `471869`) or ISO 8601 hour-precision format: `YYYY-MM-DDThh` (local time) or `YYYY-MM-DDThhZ` (UTC). Cannot be used together with `hours_range`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: File scan history per user retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
from_time	integer	Start of the reporting period as a Unix timestamp in milliseconds.
to_time	integer	End of the reporting period as a Unix timestamp in milliseconds.
user_total_scan	array	
id	string	Internal user ID.
name	string	Username in `Directory/Name` format. Empty string for anonymous users.
status	enum	ALLOWED: Active, Deleted, Remote Current status of the user account.
total_amount	integer	Total number of files/objects scanned by this user in the period.
total_size	integer	Total size (in bytes) of all files/objects scanned by this user.

NAME	TYPE	DESCRIPTION
max_size	integer	Size (in bytes) of the largest file/object scanned by this user.

STATUS CODE - 400: Bad request. Possible causes: - Time window exceeds the maximum of 2 years. - Both `hours_range` and `from`/`to` parameters are used simultaneously. - Invalid value for `from` or `to` parameter.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason.

EXAMPLE 1:

```
"TimeWindowExceeded"
```

EXAMPLE 2:

```
"ConflictingParams"
```

EXAMPLE 3:

```
"InvalidFromTo"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

14.7 GET /readyz

Get health check status

Fetch current health check status of MetaDefender Core server.

Parameter priorities:

- If both verbose and info are provided → verbose will have higher priority → ignore info
- If both additional_info and info are provided → info will have higher priority → ignore additional_info

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
verbose	boolean		Optional. Show detailed result of health check (has highest priority)

NAME	TYPE	EXAMPLE	DESCRIPTION
info	enum ALLOWED: all, database, license, avg_duration_time, cpu_load, scan_queue, number_active_av_engines, engines, disk_space	license,avg_duration_time,cpu_load	Optional. Show specific information categories. Available options: - all: return all the information (equivalent to verbose=true + cpu_load + avg_duration_time) - database: show DB status - license: show license status - avg_duration_time: show average processing time - cpu_load: show CPU load at the time of calling the API - scan_queue: show scan queue stats - number_active_av_engines: show number of active AV engines threshold - engines: show engine status - disk_space: show disk space status for installation disk Note: Takes priority over additional_info parameter but is ignored if verbose is provided.
additional_info	string	cpu_load,avg_duration_time	Optional. Show additional information (e.g., cpu_load, avg_duration_time). - cpu_load is the current CPU load of the server. - avg_duration_time is the average duration time in milliseconds. (Does not support for non-persistent mode) Note: This parameter is ignored if info parameter is provided.

RESPONSE

STATUS CODE - 200: Service is currently healthy

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
database	`object`	
status	`boolean`	Connection status
license	`object`	
status	`enum`	**ALLOWED:** `expired, invalid, ok` License status
avg_duration_time	`object`	
avg_time	`integer`	Average time
time_interval	`integer`	Time interval
cpu_load	`integer`	CPU load
scan_queue	`object`	
max_number_in_queue	`integer`	Maximum slots in scan queue
number_in_queue	`integer`	Current queue status
threshold	`integer`	User-defined threshold
status	`boolean`	status
number_active_av_engines	`object`	
threshold	`integer`	User-defined threshold
status	`boolean`	Status
engines	`array`	
engine_id	`string`	Engine ID
required	`boolean`	This engine is mandatorily active
status	`boolean`	Status
outdated	`boolean`	Whether this engine's definition database is out of date
disk_space	`object`	
status	`boolean`	true if free space is at or above threshold (or check is disabled), false if below threshold

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	`string`	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

STATUS CODE - 503: Service is currently unhealthy

---

15. SUPPORTPACKAGE

Generate, list, download and delete support packages. Will require admin apikey.

15.1 GET /admin/supportpackage

Get support packages

Return the support packages of this MetaDefender Core instance, newest first, with their generation status.

MetaDefender Core keeps only the two newest packages. When a new generation starts, older packages are deleted.

Requires an admin apikey.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: The list of support packages. The list is empty if there is no package.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
id*	integer	The support package ID.

NAME	TYPE	DESCRIPTION
status*	enum	**ALLOWED:** INPROGRESS, FINISHED, FAILED, UPLOADING, FAILED_UPLOAD - `INPROGRESS`: the package is being generated. - `FINISHED`: the package is ready to download. - `FAILED`: the generation failed. See `log_detail`. - `UPLOADING` / `FAILED_UPLOAD`: only for a package requested by Central Management. The package is generated and is being uploaded, or the upload failed. You can download it.
start_time*	integer	The start time of the generation, in Unix **milliseconds**.
end_time*	integer	The time the generation ended, in Unix **milliseconds**. `0` while the package is being generated.
log_detail*	string	The generation or upload log when `status` is `FAILED`, `UPLOADING` or `FAILED_UPLOAD`. Empty for other statuses.
metadata*	object	
request_origin	enum	**ALLOWED:** CORE, MOCM `CORE` if the package was requested from MetaDefender Core (API or web UI). `MOCM` if it was requested by Central Management.
origin_id	string	The request ID from Central Management. Only when `request_origin` is `MOCM`.
options	object	
start_date	integer	The `startdate` header value in Unix **seconds**, also when the request used ISO 8601. `0` if no date range was set.
end_date	integer	The end of the date range in Unix **seconds**. If `enddate` was not set, this is the time of the request. `0` if no date range was set, or if `enddate` was `0` (now).
included_log	array	
max_log	integer	The `maxlog` header value. `0` if not set.
lite	boolean	
enable_redact	boolean	

EXAMPLE:


```
[
 {
  "id": 1792480533006,
  "status": "FINISHED",
  "start_time": 1792480533006,
  "end_time": 1792480601532,
  "log_detail": "",
  "metadata": {
   "request_origin": "CORE",
   "options": {
    "start_date": 1792368000,
    "end_date": 1792481400,
    "included_log": [
     "core",
     "engine"
    ],
    "max_log": 50,
    "lite": false,
    "enable_redact": true
   }
```

```
  }
 },
 {
  "id": 1792394101118,
  "status": "FAILED",
  "start_time": 1792394101118,
  "end_time": 1792394102230,
  "log_detail": "<output of the generation>",
  "metadata": {
   "request_origin": "CORE",
   "options": {
    "start_date": 0,
    "end_date": 0,
    "included_log": [],
    "max_log": 0,
    "lite": true,
    "enable_redact": false
   }
  }
 }
]
```


STATUS CODE - 400: The support packages could not be read.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:


```
{
 "err": "Unable to get support package"
}
```


STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:


```
{
 "err": "Access denied"
}
```


15.2 GET /admin/supportpackage/generate

Generate a support package

Start to generate a support package and return its id immediately. The package is generated in the background.

To get the package, call Get support packages until the status of this id is FINISHED, then call Download a support package.

Only one package can be generated at a time. If a generation is already in progress, the call does not start a new one. It returns 200 with the id and filepath of the package in progress.

Requires an admin apikey.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
startdate	string	2026-10-19	Collect logs from this time. Use Unix time in **seconds**, or ISO 8601: - a date, for example `2026-10-19` (from 00:00:00 in the server time zone) - a date and time with `Z` or an offset, for example `2026-10-19T08:30:00Z` or `2026-10-19T15:30:00+07:00` - a date and time without a zone, for example `2026-10-19T08:30:00` (server local time) Must not be later than the current time. If not set, logs are collected from the oldest log.

NAME	TYPE	EXAMPLE	DESCRIPTION
enddate	string	2026-10-19T15:30:00+07:00	Collect logs up to this time. Same formats as `startdate`. A date only means 23:59:59 of that day in the server time zone. `0` means now. Must be later than `startdate`. If not set, logs are collected up to now.
includedlog	string	core,engine	The logs to include: `all`, or a comma-separated list of `core`, `nginx`, `engine`, `database`, `system`. Use `all` alone, not in a list. Default is `all`.
maxlog	integer between 0 and 100	50	The maximum number of log files to collect for each log type, from `0` to `100`. `0` means `100`. Default is `100`.
lite	boolean		`true` generates a lightweight package with less data, which is faster to generate. Any other value means `false`. Default is `false`. When `lite` is `true`, `maxlog` is set to `5` and `includedlog` to `all`. The values you send in these two headers are ignored.
enableredact	boolean	true	`true` removes sensitive data (for example credentials, host names and absolute paths) from the collected logs. Any other value means `false`. Default is `false`.

RESPONSE

STATUS CODE - 200: The generation started, or a generation is already in progress.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
id*	integer	The support package ID. It is the start time of the generation in Unix **milliseconds**. Use it to download or delete the package.
filepath*	string	The location of the package file in the MetaDefender Core data folder.
message*	string	The generation state.

EXAMPLE 1:

```
"started"
```

EXAMPLE 2:

```
"inProgress"
```

STATUS CODE - 400: A header value is not valid, or the generation could not start.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE 1:

```
"invalidStartDate"
```

EXAMPLE 2:

```
"futureStartDate"
```

EXAMPLE 3:

```
"endBeforeStart"
```

EXAMPLE 4:

```
"invalidIncludedLog"
```

EXAMPLE 5:

```
"invalidMaxLog"
```

EXAMPLE 6:

```
"generationFailed"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

15.3 GET /admin/supportpackage/{id}

Download a support package

Download a support package file. The file is a .zip file on Windows and a .tar.gz file on Linux. The Content-Type is always application/zip. Use the file name from the Content-Disposition header to get the real format.

You can download a package when its status is FINISHED, UPLOADING or FAILED_UPLOAD.

Requires an admin apikey.

Note*:* Download and delete share one rate limit for each client IP address.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*id	int64	1792480533006	The support package ID from [Generate a support package](/docs/mdcore/metadefender-core/ref#generatesupportpackage) or [Get support packages](/docs/mdcore/metadefender-core/ref#getsupportpackages).

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: The support package file.

RESPONSE MODEL - application/zip

STATUS CODE - 400: The package cannot be downloaded.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE 1:

```
"notFound"
```

EXAMPLE 2:

```
"notFinished"
```

EXAMPLE 3:

```
"fileMissing"
```

EXAMPLE 4:

```
"fileNotOpened"
```

EXAMPLE 5:

```
"readFailed"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Access denied"
}
```

STATUS CODE - 503: Rate limit exceeded. Download and delete requests share one limit for each client IP. Wait a moment and retry.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Too many support package requests. Try again later."
}
```

15.4 DELETE /admin/supportpackage/{id}

Delete a support package

Delete a support package and its file.

You cannot delete a package while it is being generated, uploaded to Central Management, or downloaded.

Requires an admin apikey.

Note: Download and delete share one rate limit for each client IP address.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*id	int64	1792480533006	The support package ID from [Generate a support package](/docs/mdcore/metadefender-core/ref#generatesupportpackage) or [Get support packages](/docs/mdcore/metadefender-core/ref#getsupportpackages).

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-core/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: The package is deleted.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

EXAMPLE:

```
{
 "result": "Support package has been removed"
}
```

STATUS CODE - 400: The package cannot be deleted.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE 1:

```
"notFound"
```

EXAMPLE 2:

```
"running"
```

EXAMPLE 3:

```
"uploading"
```

EXAMPLE 4:

```
"downloading"
```

EXAMPLE 5:

```
"deleteFailed"
```

EXAMPLE 6:

```
"fileDeleteFailed"
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
```

```
 "err": "Access denied"
}
```

STATUS CODE - 503: Rate limit exceeded. Download and delete requests share one limit for each client IP. Wait a moment and retry.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "Too many support package requests. Try again later."
}
```

OPSWAT.

