OPSWAT.



MetaDefender™ Core

MetaDefender Core
v5.15.0

Table of Contents

Installation 1

Recommended System Configuration 1
Installation 10
- Standalone mode 11
- Shared Database Mode 12
- Deployment automation support 15
- CIS Level 1 Guidelines 27
- CIS Level 2 Guidelines 31

Wizard Setup 33
License Activation 39
- Scripted license management 44

File Processing 48
Product Upgrading 49
Using external Load Balancer 54
SSL connection for PostgreSQL communication 59
Performance and Load Estimation 64
- Standalone DB - Windows 66
- Standalone DB - Linux 79
- Shared DB - Windows 90
- Shared DB - Linux 99

MetaDefender Core Documentation 108

Configuration 109

Management Console - Login 109
- Password Recovery 110

MetaDefender Configuration 115
Policy Configuration 131
- Workflow Configuration Template 132
- Security zone configuration 147

Workflow rule configuration 148
Quarantine 154
MetaDefender Core Performance Tuning Guide 159
User management 182
Roles 186
User directories 191
Active Directory attributes 197
Change user password 201
Single Sign-On (SSO) 203
User role mapping 204
Amazon Cognito 209
Configuring OIDC Single Sign-on 210
ForgeRock 223
Configuring SAML Single Sign-on 224
Keycloak 233
Configuring SAML Single Sign-on 234
Microsoft Active Directory Federation Services (ADFS) 249
Configuring SAML Single Sign-on 250
Configuring OIDC Single Sign-on 266
Microsoft Entra ID 286
Configuring SAML Single Sign-on 287
Configuring OIDC Single Sign-on 302
Okta 314
Configuring OIDC Single Sign-on 315
Configuring SAML Single Sign-on 326
OneLogin 337
Configuring SAML Single Sign-on 338
Configurating OIDC Single Sign-on 349
PingFederate 356
Configuring OIDC Single Sign-on 357
Configuring SAML Single Sign-on 392
PingOne 415

Configuring SAML Single Sign-on 416
Configuring OIDC Single Sign-on 426
Engines Update Configuration 435
General configuration 439
Password Policy 442
Session Timeout 444
Enabling HTTPS 445
Request Rate Limit 451
Data Retention 453
Scan from link settings 454
Nginx configurations 456
API Rate Limiting 460
SSL Configurations 462
Origin Client Source Identification 464
Web Server Processing Statistics Query 466
Logging 468
Log Message Format 469
Syslog Message Format 471
Error Message Description Table 476
External Scanners And Post Actions 509
Hardening feature 518
Database Management 522
Email Configuration 526
Proxy Configuration 528
Hot Swap Backup Configuration 530
Health Check API Configuration 533
My OPSWAT Central Management 536
Enrollment and MetaDefender Core Configuration on My OPSWAT Central Management 537
Settings related to My OPSWAT Central Management 540
Email Notification 541
Schedule Report 545
Encryption for Quarantined Files 548

Operating 550

Process files with MetaDefender Core 550
Dashboard 552
Average processing time 558
Understanding Block Reason Statistics 560
Processing history 562
Inventory Management 565
Regular Maintenance 580
Database Defragmentation and Optimization 581
Database Maintenance | MetaDefender Core 4.19.0 or above 582
Database Maintenance | MetaDefender Core 4.18.0 584
Download single scan report as PDF 587
Import/Export configuration 591

Integration 605

Scan result codes and explanations 605
Request rate limiting 611
Multipart upload guideline 616

Cloud Deployment 622

Overview 622
Recommended Architectures 623
EKS Cluster 624
Single EC2 630
Autoscaling EC2 634
AWS Database Services 639
Licensing Automation on AWS 643
How to deploy using AWS services 645
EKS Cluster 650
Single EC2 655

Autoscaling EC2 664
Monitoring in AWS 666
Troubleshooting and Maintenance 668
Customize OPSWAT MetaDefender AMI from Marketplace 670
Recommended Architectures 671
AKS Cluster 672
Single Azure VM 675
VM Scale Set 677
Licensing Automation on Azure 679
Deploy using Azure services 683
AKS Cluster 688
Single Azure VM 694
VM Scale Set 700
Recommended Architectures 704
GKE Cluster 705
GCP Deployments 709
GKE Cluster 714
Provisioning OpenShift on your CSP 719
Deploy your Own OpenShift Cluster 721
MD Core installation 722

Distributed Cluster Deployment 726

MetaDefender Distributed Cluster Overview 726
Recommended System Configuration 729
Roadmap 732
Installation 737
VM Setup 739
Container-Based Setup 749
Best practices 770
Configurations 772
MetaDefender Distributed Cluster File Storage 773
MetaDefender Distributed Cluster Identity Service 779

MetaDefender Distributed Cluster Worker 786
MetaDefender Distributed Cluster Control Center 792
Performance and Load Estimation 797
Troubleshooting 811
How To Collect Logs In MetaDefender Distributed Cluster Deployments 812
License Activation 818
System Upgrade 821
Upgrade to MetaDefender Distributed Cluster v2.1.0 826

Container Deployment 828

Overview 828
Kubernetes Components 829
Deploying on K8S 833
Installing MetaDefender Core in K8S 835
Resources Requirements 842
Licensing in K8S 844
Storage Configuration on K8S 846
High Availability on K8S 848
Scaling K8S Cluster 850
Troubleshooting 853
Overview 857
Docker image published on OPSWAT Docker Hub 858
Using build tool kit with your own docker image 873

Metascan Engines 875

Anti-malware capabilities 875
Supported file type 878
MetaDefender MetaScan Engine Packages 880
Public Sector | Anti-malware capabilities 882
Configure ClamAV to use a custom database 885
Set up Windows Defender as a custom engine 887
AV scoring with post action 892

Deep CDR 894

Supported File Types 894
Supported versions 907
Max File Size 911
Configurations 914
Retain password protection on sanitized files 926
Allow Macros 928
Allow Hyperlinks 931
Hyperlink redirection 933
Image limit configurations 934
QR code sanitization 936
ASCII-based QR code 938
Base64-encoded data in JSON 939
Remove invisible characters in TXT 942
Mark Sanitization Result as Skipped 944
Analysis mode 945
Block file based on objects 949
Deep CDR Performance (Throughput) 956
Archive Sanitization 962
Tombstone file 964
Setup output file name 967
Deep CDR Details 972
Reason for Action 976
Troubleshooting sanitization failures 978
Supported engines-metadata JSON 980

Proactive DLP 981

Overview 981
Supported File Types 983
Detect sensitive information 992
Detection Policy 996

Detection Policy - Supported Entities 999
Secrets Detection 1003
Optical Character Recognition (OCR) 1007
Leverage metadata info added by data classification systems 1009
Process hidden parts in a cropped image 1010
Regular expressions 1011
Importing Regular Expressions via a TSV File 1016
Remove alternative images in PDF 1018
Supported engines-metadata JSON 1021
Supported metadata type in PDLP 1022
Redact/Substitution sensitive information 1025
Watermark files 1030
Remove metadata 1033
DICOM anonymization 1035
Document Identification 1040
Performance (Throughput) 1043

Sandbox 1045

Overview 1045
Supported File Types 1048
System requirements 1053
Upgrade to Java 17 1056
Installation 1058
Licensing 1060
Configuration 1062
Scan Profiles 1069
Release notes 1071
Knowledge Base 1072
Internet Connection 1073
Enhancing Threat Detection with Deep CDR & Adaptive Sandbox 1074
How do I obtain my OPSWAT MetaDefender Cloud Reputation API Key? 1076
What is displayed on the process result tile? 1078

Enhancing Threat Detection with YARA & Adaptive Sandbox 1080
Sandbox-Core High Availability Setup 1082

Reputation Engine 1083

Overview 1083
Licensing 1087
Reporting False Detection 1088
System requirements 1089
Installation 1090
Configuration 1092
Release notes 1095
Knowledge Center 1096

Software Bill of Materials 1097

Software Bill of Materials (SBOM) 1097
Source code 1098
Recursive check 1101
Container image 1103
License risk detection 1104
JSON response 1106

Country of Origin 1112

Country of Origin 1112
Digital Certificate Verification 1114

Knowledge Base 1120

Why are Japanese TXT files detected as "application/octet-stream" instead of "text/plain"? 1120
How to Connect a MetaDefender Core Container to the same instance in Shared Database Mode? 1122
Can local AVs interrupt ongoing scans? 1123
Can I create new workflows? 1125

Can I control access to the RAM disk in MetaDefender Core v5? 1127
Can I view configuration changes on the Core side (audit)? 1129
How can I add a mail server to the Core deployment? 1131
How do I prevent Symantec Endpoint Protection from interfering with the optimal functioning of MetaDefender Core? 1133
How can I find a sanitized file scanned with MetaDefender Core v5? 1136
What is the retry mechanism for MetaDefender Core connecting to the remote Postgres Database? 1137
How to use MetaDefender Core V5 Blocklist/Allowlist feature? 1138
How can I configure the maximum queue size in MetaDefender Core v5 ? 1140
How do I remove an engine from my MetaDefender instance? 1143
How can the Temp folder be changed? 1144
How can I increase parallel count and Max queue size? 1146
How can I change MetaDefender Core Deep CDR's timeout settings? 1148
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? 1151
Why are password protected archives blocked and how do I unblock them? 1154
How to disable WEB UI file scan without user authentication? 1158
How To Allow Only Certain Files to be Scanned with MetaDefender Core? 1160
How to generate an API key on the Core deployment? 1162
How to Modify the Hostname of Your MetaDefender Core Server and Potential Impact? 1164
What are the permissions on the shared folder for the temp directory? 1166
How to implement a numerical escape function for HTML? 1168
How to reset password for users? 1171
How to Block/Allow Custom Extensions using RegEx? 1174
How to migrate MetaDefender Core from PostgreSQL local to PostgreSQL remote? 1176
How to change Core Web Management Interface Address Port? 1179
Does MetaDefender Core store original files? 1183
Why Japanese characters are not recognized or displayed correctly in archives? 1184
How to install pre-built libgdiplus on RedHat 9? 1186
Does Deep CDR support sanitizing split archives? 1188
How to exclude OPSWAT folder from Anti-Virus Windows Defender? 1189
Where are MetaDefender Core Install Directories and Registry? 1192

Under what circumstances does the scan result not show the username?	1194
How to send MetaDefender Core logs to NAS (Network Attached Storage) Path?	1196
Why is a file blocked by Deep CDR or Proactive DLP but I can still download the sanitized version?	1206
Does Deep CDR support Adobe Illustrator (AI) files in PostScript format?	1208
How to add the source IP in scan details?	1210
How do I integrate Jenkins with MetaDefender Core?	1212
Can I apply a new license key before the current one expires?	1216
How to automate backup configuration settings for MD Core? (API)	1217
Why I can't push configuration due to password storage?	1221
How do I install MetaDefender Core in Rocky Linux 9	1222
How to properly configure Syslog Server for MetaDefender Core?	1225
Why MetaDefender Core's nginx ssl.conf changed to ssl.mdcore.conf?	1227
How to configure SMTP server with authentication under MetaDefender Core to send notification to M365?	1229
How to manage Disk Space Usage in pg_data\base directory in MetaDefender Core for Windows?	1231
What storage type should I use on MetaDefender Core in K8S?	1233
What permissions are needed for the user of the storage of Google Cloud?	1234
How do I setup Hot Swap Backup Configuration between MetaDefender Core instances?	1236
Where are MetaDefender Core Install Directories and Advanced Configurations?	1241
How to retrieve the CRL and OCSP URLs for the Server and ICA Certificates?	1245
Is Import Password Mandatory When Using Ignition File for MetaDefender Core installation?	1248
How does the recursion level work in MetaDefender Core?	1252
How to Enable Debug Mode in a MetaDefender Core Kubernetes Deployment?	1256
How to Select a Specific MetaDefender Core Instance in a PostgreSQL Database with Multiple Instances?	1258
Can you import configuration settings from the latest version to the previous version of Core?	1259
What can I do when MetaDefender Core (Windows) has an insufficient disk space error?	1261
Why is MetaDefender Core unable to connect to the database?	1264
What can I do if MetaDefender shows a high number of files that failed to scan?	1268
What should I do if an engine is in "failed" or "permanently failed" status?	1270
What are the engine clean-up instructions?	1271

Why is the scan stuck in "processing" state on WebScan UI, when the Core Processing History shows that it is already finished?	1274
Why have the ESET and Kaspersky scan engines failed to initialize on the hardened Linux OS?	1275
Why has the Symantec custom engine failed to deploy?	1276
What are common reasons for Deep CDR scan/sanitization failures and how do I fix them?	1277
What do the various error messages mean in MetaDefender Core Rest API?	1281
Why are some modules not appearing in my MetaDefender Core inventory?	1298
Why does my MetaDefender Core install fail?	1300
Why do you sometimes see a discrepancy between the File Type and MIME Type information?	1304
Why files are not sanitized?	1306
Why am I receiving a “400” error for API file submission?	1309
How to Mitigate Scan Failures in MetaDefender Core Triggered by Real-Time Protection Systems?	1310
How to Resolve MetaDefender Core Service Start-up Failures on IPv6-Disabled Linux Systems?	1312
How to Understand OPSWAT's Multiscanning AV Detection Differences Compared to Other Scanners?	1314
How to Troubleshoot Inaccessible Management Console?	1317
How do I fix module update error “Error parsing metadescriptor, message='Could not download metadescriptor'”?	1318
Why does ClamAV show "permanently failed" after enabling the daemon log file?	1322
How to troubleshoot HTTPS failure on MetaDefender Core?	1325
How to Remove Engines from MetaDefender Core using command line?	1327
How to enable debug logging for MetaDefender Core?	1329
Why did the deployment of the Threat Intelligence Module Fail?	1331
Why did the deployment of the Reputation Module Fail?	1332
Why is MetaDefender Core not starting after moving or deleting the certificates from the initial location?	1333
Why doesn’t CDR detect the JavaScript I appended to a PDF file?	1334
Why Deep CDR is not Sanitizing Files Inside a ZIP?	1335
Why does the “deleteafterimport” error appear when importing an MD Core configuration from standalone mode to shared database mode?	1336
What is the meaning of action_ran and action_failed in JSON scan result?	1339

Why can't I sanitize HEIC files? 1341
Why Am I Getting "No Rows to Show" When Using a Proxy Without Authentication in Browser Settings? 1343
How Does TLS 1.0/1.1 Deprecation Affect Compatibility Across OPSWAT Products? 1345
Why can small file changes bypass the Antivirus detection? 1348
How do I verify if MetaDefender products are sending out syslog messages to the syslog server? 1350
How do I fix files being blocked due to unexpected "File type mismatch" verdict? 1352
Why are MetaDefender Core engines permanently failed? 1356
Why Does MetaDefender Core Fail to Connect to the Database After a System Restart? 1358
What do you do if your engine fails to update? 1361
What is causing my Scans to be slow on MetaDefender Core? 1363
What do I do if my modules disappear from the Core UI? 1365
Why Is Core Receiving HTTP 403 or Timeout Errors when downloading engine updates from OCM? 1367
What do I do when licensed engines are not detected despite active licenses? 1369
Why do I receive errors when I upload the Config file in MetaDefender Core? 1372
How to access MetaDefender Core when SSO fails? 1374
Why am I seeing error message "Validation failed" after saving Sandbox module settings? 1376
What Are the Default Credentials for MetaDefender Core on Azure Marketplace VM? 1377
Why does the MetaDefender Core service not start? 1379
Why does Kiosk show "Invalid Core Setup"? 1384
Why is my file rejected even though it appears to be under the configured size limit? 1388
How Can I Solve a Database Indexing Issue? 1391
Why Does MetaDefender Core Show a 'Calculate hash error' When Scanning Malicious Files? 1393
How to Use a RAMDISK for the tempdirectory? 1395
Why does ClamAV seem to be slower than other engines? 1398
What temporary folder do Custom Engines use ? 1400
What PDF file annotations are Supported/Non-Supported by MetaDefender Core's Deep CDR? 1401
What does "Potentially Vulnerable File" result mean? 1402
What CAB files are Supported/Non-Supported by MetaDefender's Deep CDR? 1403
What are the MetaDefender Core Security Policies and how do they work? 1405

Is Metadefender Core compromised while scanning files?	1409
What is the Queue Mechanism on Metadefender Core?	1410
How does the Post Actions feature work?	1412
How does the External Scanners feature work?	1415
How to generate a Core's scan result summary in JSON format	1418
How and why are my image files altered during processing with Deep CDR and Proactive DLP?	1422
Does MetaDefender Core v5 offer real-time antivirus protection on the system where it is installed?	1424
Does MetaDefender Core v5 Detect the NotPetya Ransomware?	1426
Are there any limitations to the MetaDefender Core scan engines?	1429
Can MetaDefender Core block renamed files (such as renamed .exe files)?	1432
How can I export the processing history from the Core console?	1435
I forgot my login password, what can I do?	1437
How to Utilize YARA Rules with MetaDefender Core?	1439
What is the Design of MetaDefender Core and Its Protective Measures?	1443
What does the items sanitized/removed count in Deep CDR sanitization report mean?	1444
Why image sanitization doesn't report any object in details?	1446
What are the requirements to process XDW/XBD/XCT files with CDR?	1447
How do we utilize API uploads so the correct file type and sanitization occurs every time?	1448
What are the differences between AVs in OPSWAT and the other services?	1450
Does OPSWAT Use Security Checks on Our Code, Libraries, Credentials, etc?	1452
How to access and analyze detailed scan results for an infected file?	1453
Why MetaDefender Core blocks Sandbox Verdicts that are not marked to be blocked?	1456
How to scan multiple files at once with MetaDefender Core	1458
Is MetaDefender Core able to extract, detect and block malicious embedded macros?	1459
Does OPSWAT provide a file dataset that can be used to evaluate MetaDefender Core's capabilities?	1461
How would discontinuing Threat Intelligence affect customers?	1462
How do I Validate FileType Digital Signature on MetaDefender Core?	1464
How to add custom detection for a file type?	1468
What can I do if MetaDefender Core shows a high number of files that failed to scan?	1473
Why did the required password pop up when scanning an unencrypted file?	1476

How to Test the Syslog Server functionality from MetaDefender Core? 1479
When and how to use “PDF to Image Configuration” Deep CDR feature? 1481
How do I send specific log entries to syslog integration? 1486
How do I send the MetaDefender Core logs in Windows event log? 1488
Does installing an antivirus impact a containerized deployment? 1490
Why does the DeepCDR engine delete the content of the HTML file? 1491
What is the Document Encryption Timeout setting and how does it work? 1494
How do I manually clean up the MD Core engine's local folder? 1496
How can I handle a high CPU load on my Core server 1497
Does MetaDefender Core generate a file download link after all scan sessions? 1499
Can Workflow Settings Be Updated Globally Across Multiple Workflows? 1502
Does MetaDefender Core support Elliptic Curve (EC) Private Key and Certificate for enabling HTTPS? 1505
How to manage false positives in MetaDefender Core? 1507
How to handle split archive files? 1511
How to check the Nginx version used in MetaDefender Core? 1513
How to Sanitize Non-Compliant MDZIP Files? 1514
How do I obtain a Trial License for MetaDefender Core? 1515
How long is the support lifecycle for a specific version of MetaDefender Core? 1518
How do I get a False Detection (False Positive or False Negative) corrected? 1521
What dependencies does MetaDefender’s Deep CDR for Linux have? 1524
What links, target-services or target host-IPs need to be allowed for Metadefender Core to function accurately? 1527
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? 1529
How do I deploy MetaDefender Core to an offline Windows environment? 1531
How do I deploy MetaDefender Core to an offline Linux environment? 1538
Does the Deployment ID change at any time? 1545
How do I activate my offline deployment of MetaDefender Core? 1547
How do I activate/deactivate a MetaDefender Core Deployment via API calls? 1550
How do I update to a new license key on a Metadefender Core server? 1553
What types of licenses are available for MetaDefender Core? 1556

After license renewal does the product activation happen automatically (if the license key is the same)?	1558
How to install .NET 8 Runtime dependencies on Linux Distributions?	1559
What are the List of folders that should be removed after MetaDefender Core uninstallation on Windows?	1562
How to connect LDAP via SSL on Linux	1563
How to Backup and Restore standalone database for MetaDefender Core?	1565
How to install msttcore-fonts on Linux systems?	1571
How to upgrade Amazon Corretto JRE?	1574
Can I activate the product license before the start date?	1575
How to completely uninstall MetaDefender Core?	1576
How to Trust MetaDefender Core's Self-Signed Certificate When HTTPS Is Enabled?	1578
How are MetaDefender Core AWS AMI images secured and updated?	1580
How to handle high CPU load on MetaDefender Core server?	1581
What will happen to the MD core after the license is expired?	1583
Is there a virus test I could use to test MetaDefender Core engines?	1585
How do I check if "noexec" flag exists on a Linux OS?	1586
How can I run tests to see the different scan results on MetaDefender Core?	1588
Are scan engine updates automatic?	1590
What operating system patches should be applied to the system hosting MetaDefender Core?	1592
How do I check the update status of MetaDefender Core's licensed AV scan engines?	1593
What is the frequency of signature/definition updates on MetaDefender Core?	1596
Why does the Deep CDR engine keep failing to update?	1599
What Happens After the Discontinuation of Kaspersky Support for Windows and Linux?	1601
How to Troubleshoot an Error related to Connection to Database Failing after an Upgrade to v5.11.0?	1603
Does a CVE affect MetaDefender Core?	1607
End-of-Life Support for CentOS 7 and RedHat 7 in MetaDefender Core and Engines	1611
End of Life Support for Ubuntu 20.04 and Amazon Linux 2 in MetaDefender Core	1614
Will MetaDefender Core Experience Downtime During Engine Updates?	1616
How to Update Antivirus Engines in MetaDefender Core Manually?	1618
Are MetaDefender Core v5 upgrades free?	1624

Are MetaDefender Core version upgrades automatic? 1625
How do I manually upgrade to the latest MetaDefender Core version? 1627
What is the latest MetaDefender Core version? 1655
How to upgrade an End-of-Life [EOL] MetaDefender Core? 1656
Can I upgrade MetaDefender Core from an very old version such as version 4.x.x to the latest version 5.x.x? 1659
Can I upgrade MetaDefender Core from a very old version directly to the latest available? 1661
What is the impact of upgrading MetaDefender Core from version 5.12.0 to 5.13.3? 1662

Utilities engines 1664

File Type Verification Engine 1664
Supported file types 1665
Configuration 1666
Supported engines-metadata header 1669
File info details 1671
Text detection with AI 1674
Custom detection 1676
Archive Extraction Engine 1681
Supported file types 1682
Supported engines-metadata header 1691
Decrypt GPG files 1692
Configuration 1694
Differences between OPSWAT Archive engine and 7-zip 1697
Archive Compression Engine 1698
Configuration 1699
YARA Engine 1702
Configuration 1704

Release Notes 1705

Release notes 1705
Archived release notes 1716
Metascan release notes 1808

Deep CDR release notes 1829
Proactive DLP release notes 1854
File Type Module release notes 1866
Archive Module release notes 1876
SBOM release notes 1887
YARA engine release notes 1890
Adaptive Sandbox release notes 1892
Embedded Engine 1893
Remote Engine 1909
Reputation Engine release notes 1917
Country of Origin release notes 1919
MetaDefender Distributed Cluster release notes 1920

Troubleshooting 1922

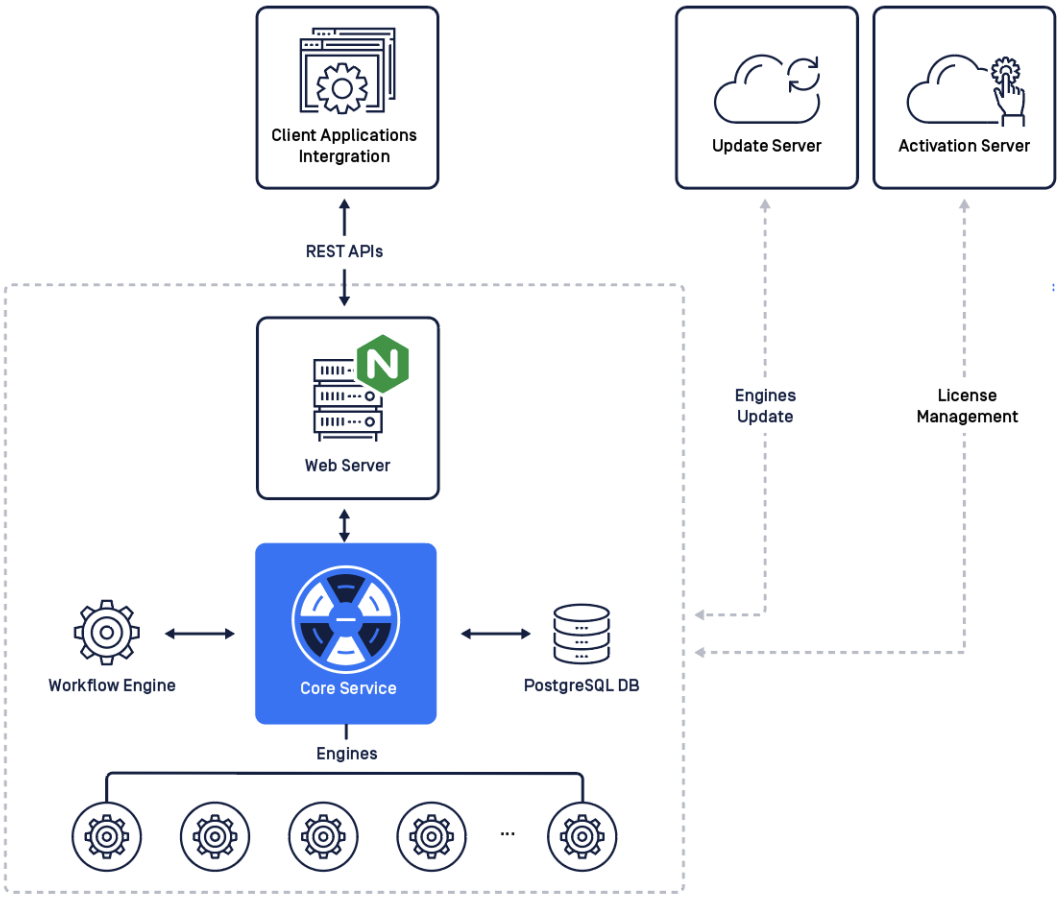
Troubleshooting MetaDefender Core 1922
How to Create Support Package With Web Console 1923
How to Create Support Package With Bundle Tools? 1925
Debug logging 1928
How to Read the Metadefender Core Log? 1929
Inaccessible Management Console 1931
Open Connection On PostgreSQL Server 1932
Too Many Sockets or Files Open 1934
Too Many TIME_WAIT Socket 1936
Engine Clean-up Tool 1938
TCP/IP Port Exhaustion 1944
Archive Compression Engine Permanently Failed on MetaDefender Core v4 1946
HTTPS Failure on MetaDefender Core 5.2.0 (or newer) 1948
HTTPS Configuration Removal Tool 1950
Customize Internal PostgreSQL User 1951
PostgreSQL User Privilege Requirements 1953
Remote PostgreSQL User Privileges Checking Tool 1958
Database Cleanup Tool in Shared DB Mode 1962

Rectify malformed FileType data in PostgreSQL database 1964
Migrate MetaDefender Core from one machine to another machine 1967
Working with self-signed certificate 1970
Authentication Failure After Upgrading From Core 5.10.1/5.11.0 1971

MetaDefender Core 1975

MetaDefender Distributed Cluster 2253

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.

> **Note**
> 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 10, Windows 11 23H2, 11 IoT (22H2) only for End-User systems (e.g. Kiosk deployments)
  - 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.
- Windows Server 2016, 2019, 2022

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

> **Plan for legacy OS support**
> OPSWAT will discontinue support for the following OS in MetaDefender Core and its associated engines starting **October 2025**:
>
> - Windows Server 2016.
> - Windows 10 versions prior to 21H2.
>
> If you need some more time to transition from legacy OS, refer to this instruction Lock Your Engine and Plan for Migration.

[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 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

- It is suggested to use SSD for system where MetaDefender Core is installed.
- See more disk space details in Storage Usage Information.
- For a better performance with Deep CDR and Proactive DLP technologies (if licensed), consider adding 8GB RAM, 4 CPU cores (at least 12 CPU cores in total) additionally.
- SBOM processing requires more disk space, please allow at least 15GB free disk for the engine to work properly.

[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 14.17
    - From 15.7
    - From 16.4

Some engines also have dependencies as described below:

Engine Name	Dependency
Deep CDR	Microsoft Visual C++ 2017 Redistributable Package (Only applicable to engine version 5.8.0 or above)
ESET	MetaDefender Core temporary directory should have more than 200MB free disk space
Adaptive Sandbox	Embedded and Remote Engine: Dependencies and system Requirements
Proactive DLP	CPU must support AVX2 and SSE4.1 instruction set to use OCR feature
Scrutiny	Microsoft Visual C++ 2015 - 2019 Redistributable x86 and x64
Systweak	.NET runtime 6.0, 7.0 or 8.0
Bkav Pro	Microsoft Visual C++ 2015 - 2022 Redistributable x64

[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 (since v9.5)
- Debian 11, 12
- Ubuntu 22.04
- 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.

For Hardended Linux OS
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 auto$ for remediation details.

Disclaimer
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
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

- It is suggested to use SSD for system where MetaDefender Core resides on.
- See more disk space details in Storage Usage Information.
- For better performance with Deep CDR and Proactive DLP technologies (if licensed), consider adding 8GB RAM, 4 CPU cores (at least 12 CPU cores in total) additionally.

[Linux] Third Party Dependencies

Dependencies list:

- openssl
- grep
- lib32stdc++6 (>= 4.5)
- libc6-i386 (>= 2.10)
- procps
- zlib1g
- libcurl3 (>= 7.19.7)
- libcurl4
- postgresql:
  - From 14.5 to 14.17

- From 15.7
- From 16.4

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, libgdiplus 6.0.5 or above For Debian 12: use libgdiplus 6.0.4 Rocky Linux 9.x/Red Hat 9.x: libnsl
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, compat-openssl11 • Ubuntu/Debian: libgomp1, libncurses5 • libgdiplus 6.0.5 or above ◦ For Debian 12: use libgdiplus 6.0.4 • .NET 8 dependencies
Adaptive Sandbox	Embedded and Remote Engine: Dependencies and system Requirements
FileType engine	libnsl Amazon Linux 2023: libxcrypt-compat
Country of Origin	libnsl Amazon Linux 2023: libxcrypt-compat
Proactive DLP	libgdiplus, mscorefonts .NET 8 dependencies CPU must support AVX2 and SSE4.1 instruction set to use OCR feature
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 4.5GB 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. • 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.

Allowlisting URLs
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

Warning
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

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 support Central Management use-case with shared DB mode.
- 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 $link[page,354687,Reuse results by hash,reuse-scan-results-for-the-same-hash] may not work smoothly.

Requirements for shared database setup:

- The central remote PostgreSQL server (for shared database) could be utilized with a built-in (native) PostgreSQL database server installed with one of the MetaDefender Core instances in the network. Or using a dedicated existing PostgreSQL database server (see detailed requirements for PostgreSQL version at $link[page,354733,auto$]).

- Before setting up, make sure to follow steps at $link[page,355041,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: $link[page,355051,auto$]
- 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) authentication settings (if using) cannot be shared among MetaDefender Core instances because SSO requires an unique login URL for every different Core.
- 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)

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:

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

Clean install scenario

1. Create ignition file if not existed

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

An example of ignition file:

2. Run installers using command line to install

Please refer to $link[page,354743,Command Line Installation,command-line-installation] for 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:
    a. Standalone Mode > Command Line Installation (applicable on version 4.19.0 or newer).
    b. 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.

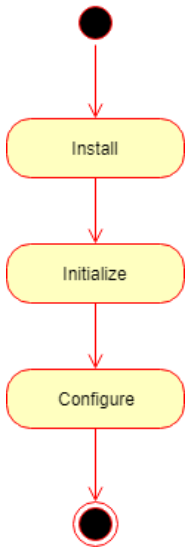
Ignition file location to create:

- **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



Installation

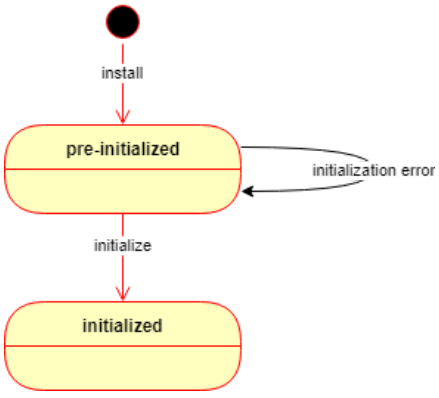
To automate the installation, install the product from the command line and provide the installation-time options as parameters to the installer. For further details see Command Line Installation

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.

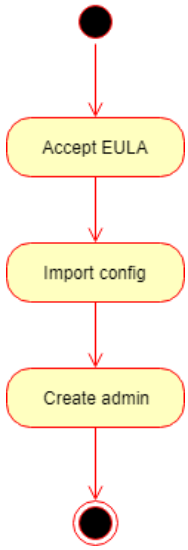


Initialization

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.



Ignition file

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

> **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:

Section	Key	Required	Accepted Values	Description
	eula	Optional (mandatory when having [user] section)	true false (default)	Whether to accept the End User License Agreement. This key must be set at the top of ignition file. This key must be set to "true" to accept the EULA. Any other value will cause the initialization to fail. When having [user] section defined, then this key must be set.
[global]		Optional		Global configuration
	dbmode	Optional	Define database mode: • 1: Standalone Core (default) • 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)	

Section	Key	Required	Accepted Values	Description
	instance_name	Optional		Define instance name. If no define, instance name will be automatically generated. Unicode and number At least 3 characters. No whitespace
	storage_path	Optional		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 folder is open access (read and write) to MetaDefender Core. For example: • C:\folder • \192.168.1.1\folder
[user]		Optional		Initial administrator user account properties. The Administrator role is granted to the account.
	name	Optional	string	User name for the initial administrator user account.
	password	Optional	string	Password for the initial administrator user account. WARNING! Clear text password The password in this configuration file must be stored in its clear-text format and as so it may be visible for unauthorized parties.
	email	Optional	string	E-mail address for the initial administrator user account.

Section	Key	Required	Accepted Values	Description
	apikey	Optional	string	API key for the initial administrator user account.
[config]		Optional		
	import	Optional	string	Path to a file in json/zip format that contains a previously exported configuration to be imported. Zip-format file contains a full configuration including user management settings, so it will overwrite [user] section.
	import_password	Optional	string	Password to decrypt .zip configuration package.
[security]		Optional		
	session_cookie_samesite	Optional	Strict Lax (default) None	Declare if session cookie should be restricted to a first-party or same-site context.
[internal]		Optional		
	skip_startup_pages	Optional	true false (default)	Skip all welcome and other startup pages (e.g. upgrade result) when installing / upgrading MetaDefender, and going straight to the product index default page.
	skip_upgrade_scan_data	Optional	true false (default)	Skip upgrading processing result data (Processing History), files sanitized by Deep CDR, files processed by Proactive DLP, quarantined files.
[dbserver]		Optional		

Section	Key	Required	Accepted Values	Description
	type	Mandatory when having [dbserver] section	local remote	Where the PostgreSQL database server should physically locate. "remote" means PostgreSQL server and MetaDefender Core are not in the same machine.
	host	Mandatory when having [dbserver] section	string	IP address / domain name of the server where PostgreSQL server locates. "localhost" can be used when applicable.
	port	Mandatory when having [dbserver] section	number	Port of PostgreSQL server is listening for connections from clients (i.e. MetaDefender Core).
	user	Mandatory when having [dbserver] section	string	PostgreSQL server's user. SUPERUSER privilege is required for MetaDefender Core to setup its database and extensions for the first time. **Only non-Unicode characters supported.**
	password	Mandatory when having [dbserver] section	string	PostgreSQL server's user credentials. **Only non-Unicode characters supported.**

Section	Key	Required	Accepted Values	Description
	private_username	Optional	string	PostgreSQL server's internal user created for MetaDefender Core own operational purpose. If not specified, then MetaDefender Core will auto generate this user. See details at Customize Internal PostgreSQL User
	private_password	Optional	string	PostgreSQL server's internal user created for MetaDefender Core own operational purpose. If not specified, then MetaDefender Core will auto generate this user. See details at Customize Internal PostgreSQL User
[ocm]		Optional		
	ocm_url	Optional	string	URL to My OPSWAT server.
	reg_code	Optional	string	A registration code to enroll to My OPSWAT server.
[max_body_size]		Optional		
	body_size	Optional	number	Defines the maximum body size that the server accepts for client uploads, independent of any specific workflow rule. The body_size can be set to 0 for unlimited size (default value) or must be at least 1,048,576 bytes (1 MB).

Section	Key	Required	Accepted Values	Description
[https]		Optional		
	cert_name	Optional	string	Name of the cert defined by users. Does not accept a "None" value or any special characters " []:|;=+*?<>@/"
	cert_path	Optional	string	Absolute path of cert file
	key_path	Optional	string	Absolute path of key file
	passphrase	Optional	string	Passphrase of the cert file (if any) - if cert does not have passphrase, this field can be omitted
	tls_version	Optional		TLS version, it could be 1.2 or 1.3 or both of them "1.2,1.3"

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.

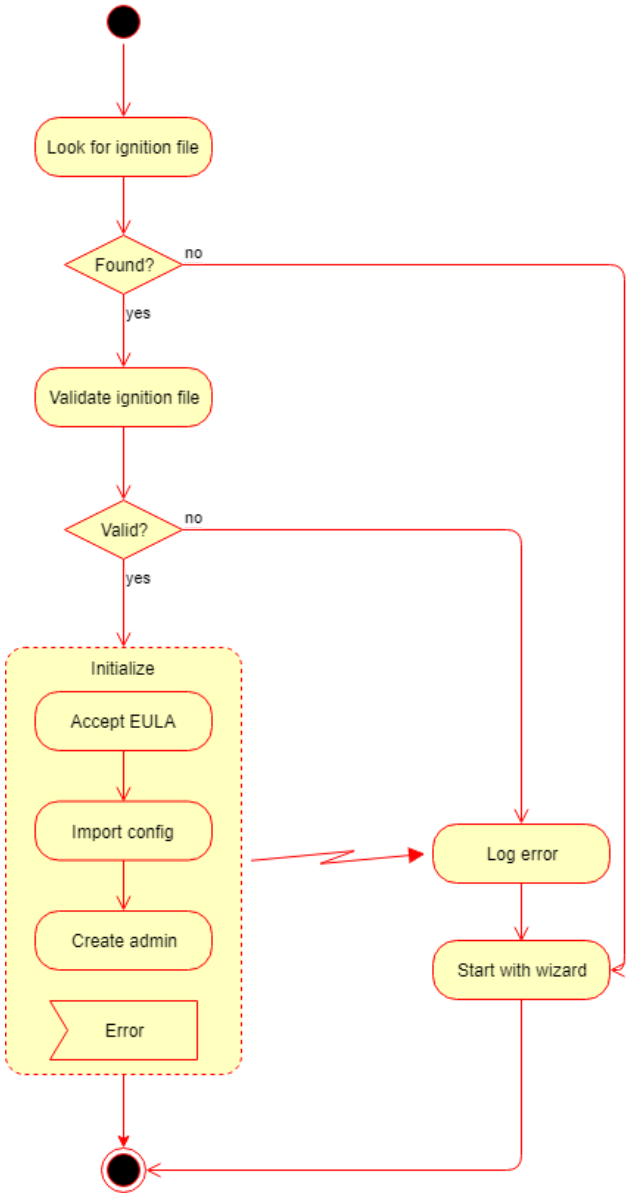
> **Info**
> In this case the product starts normally: if for example the basic configuration wizard has not been completed yet, then it must be completed first.

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.



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 the initialization is complete, the product is ready with the default and the imported configuration.

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

CIS Level 1 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

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:

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.

bash

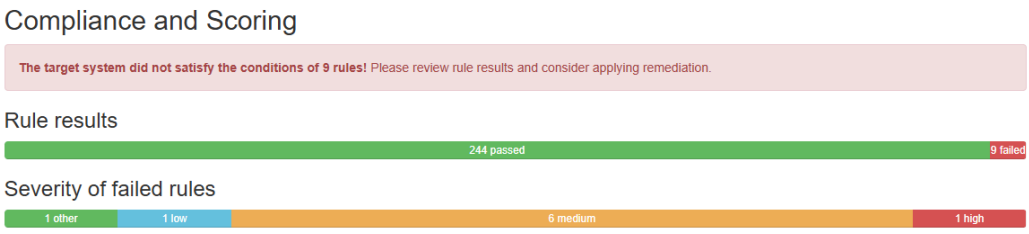
```
./remediations.sh
```

IV. Review the results after remediation

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

bash

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

Set up the Ubuntu Security Guide

bash

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

Check SCAP Content Overview (Security Content Automation Protocol)

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_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**

bash

```
sudo usg fix cis_level2_server
```

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

bash

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

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 Guidelines

Support Ubuntu Pro 22.04
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

bash

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

Check SCAP Content Overview (Security Content Automation Protocol)

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 that apply CIS rules

Method 1: directly using usg command - recommend

bash

```
sudo usg fix cis_level2_server
```

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

bash

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

Notes

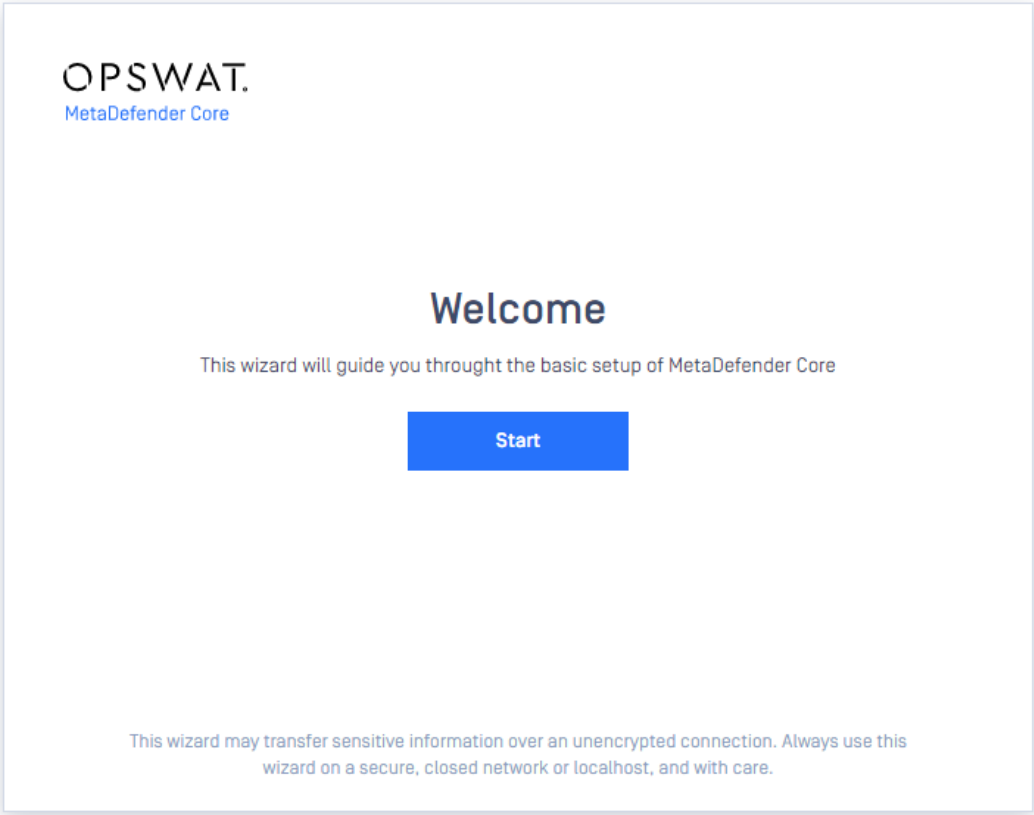
- CIS Level 2 requires `/tmp` folder to be mounted in a separate partition. Please ensure that that new partition have enough disk space for MetaDefender Core to run.

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.

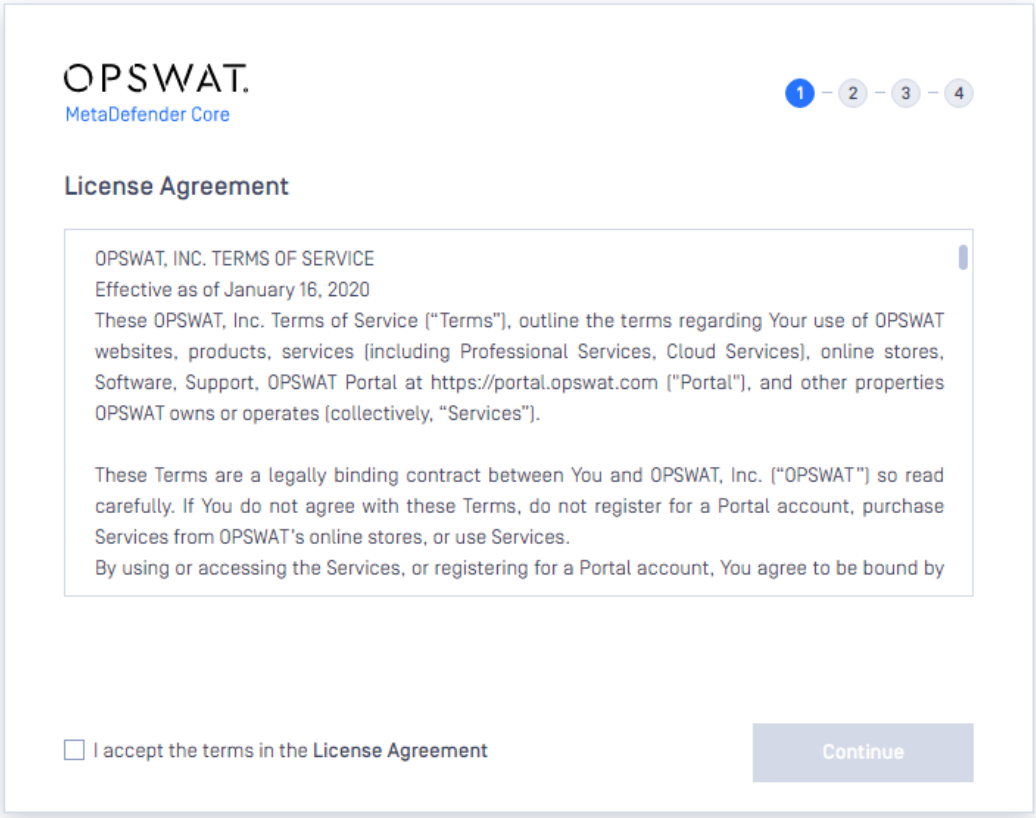
The administrative management console will be available only upon you have successfully finished this wizard stage.



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.

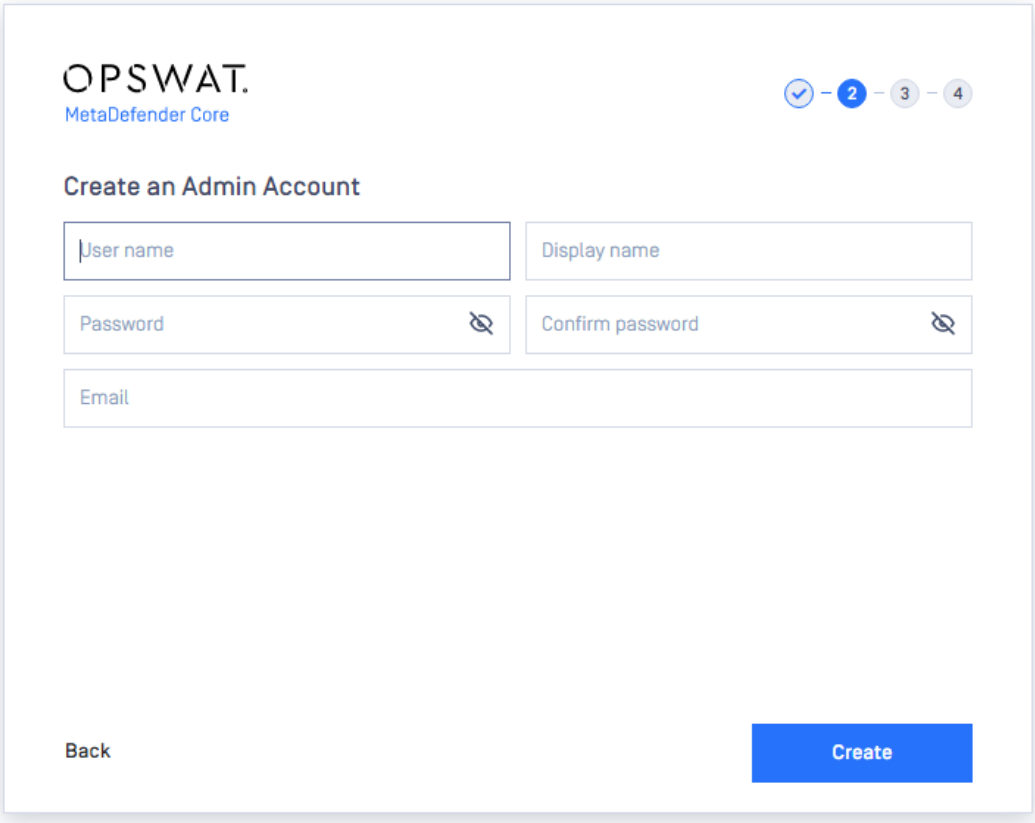
This account will be the first one being able to access the administrative management console, and to create accounts for other users. You have to fill out all fields in this page to be able to move forward. When you are done, click NEXT to continue.

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.



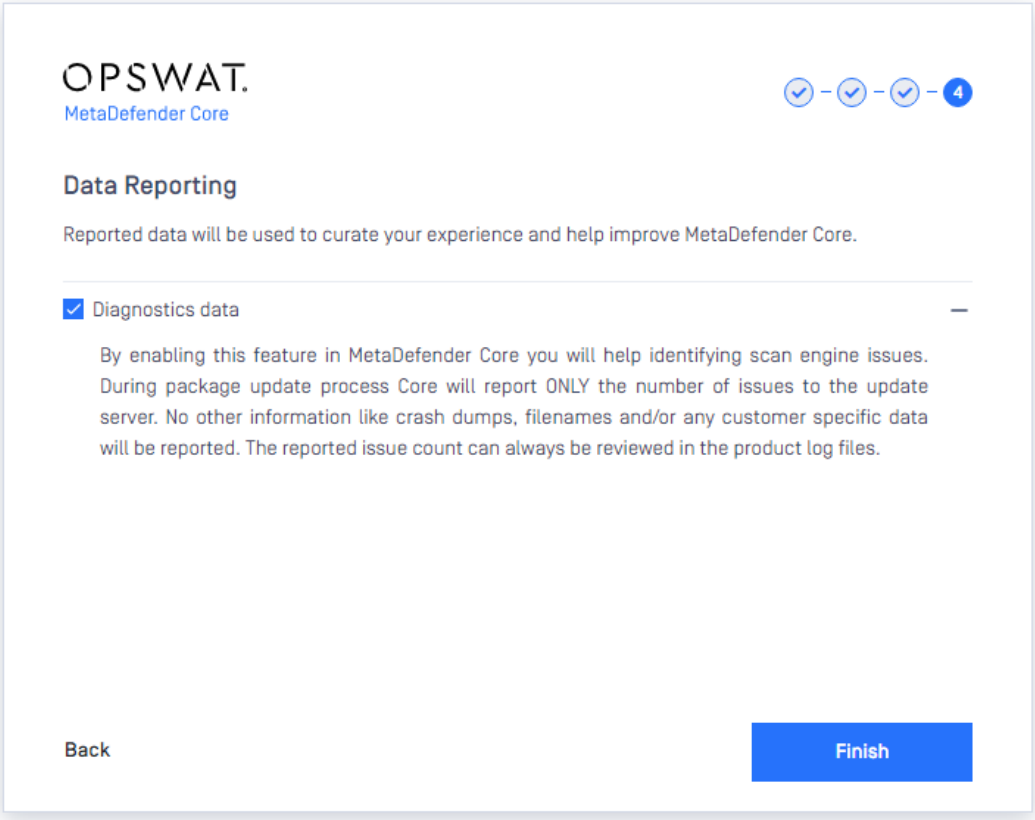
License activation

For license activation details see License Activation

Data Reporting

Enabling this option to allow MetaDefender Core collect its own engine issues and report back to OPSWAT for further analysis.

MetaDefender will not collect any other information.

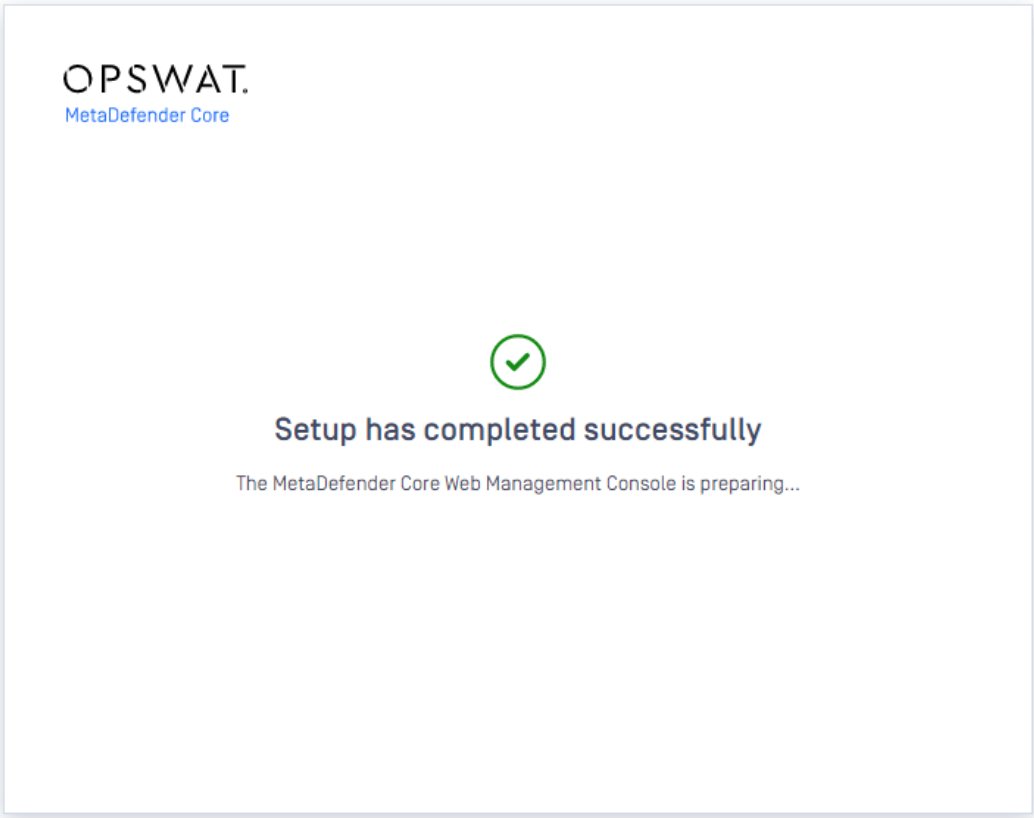


Wizard completion

After you have completed every steps you are ready to finish the wizard and start using the product. Click the `Fishish` 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 **Inventory > Modules** 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.

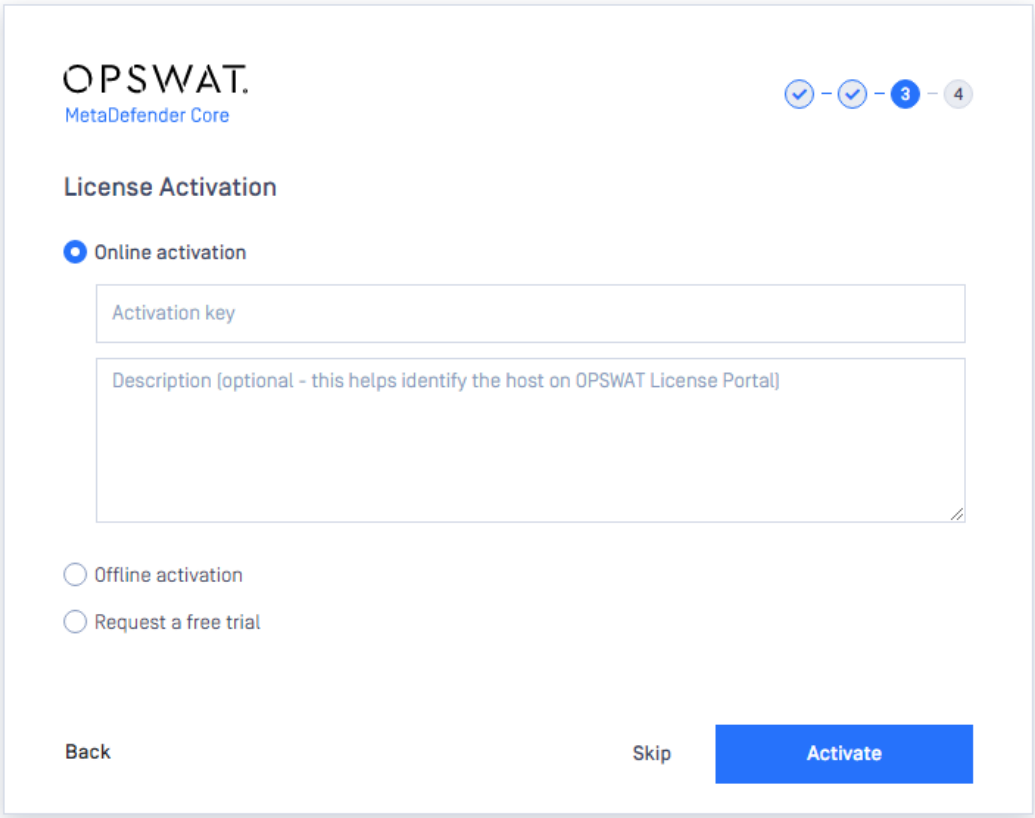
> **Notice**
> Deployment ID is auto generated by MetaDefender Core product upon installed, and it might be auto changed when you alter the system where MetaDefender Core relies on (e.g. network card, OS hostname).
>
> Deployment ID is an UUID version 4 random based string. Learn more at: https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_4_(random)

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.

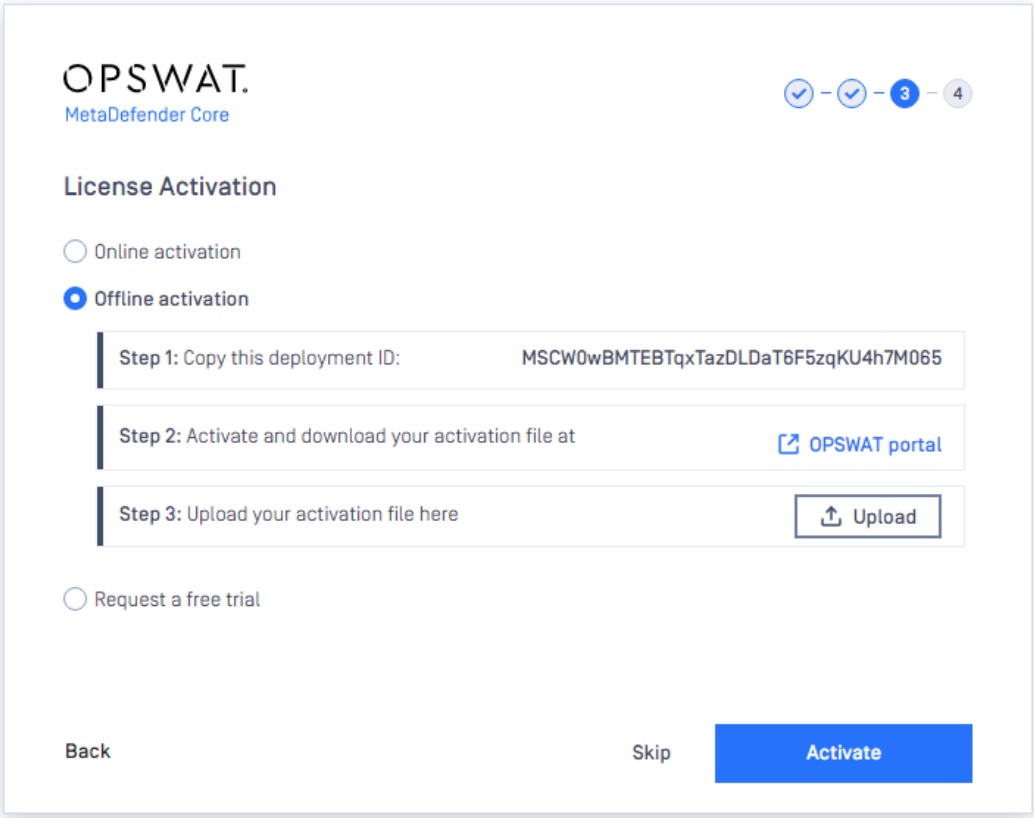
Required to input your Activation License key, and optional for description (but we recommend this for troubleshooting and auditing purpose if needed).

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.*

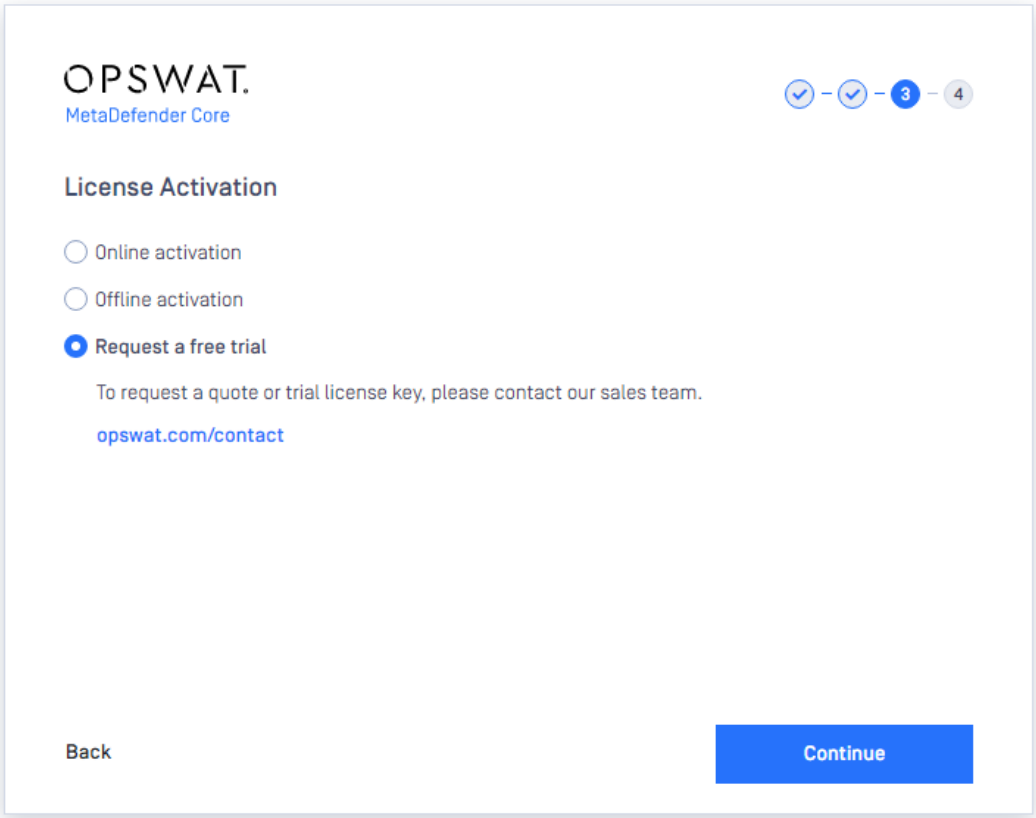
Offline activation

With no internet connection on the server the Metadefender v4 instance may be activated indirectly from a different machine, that has internet connection. The Deployment ID of the Metadefender v4 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

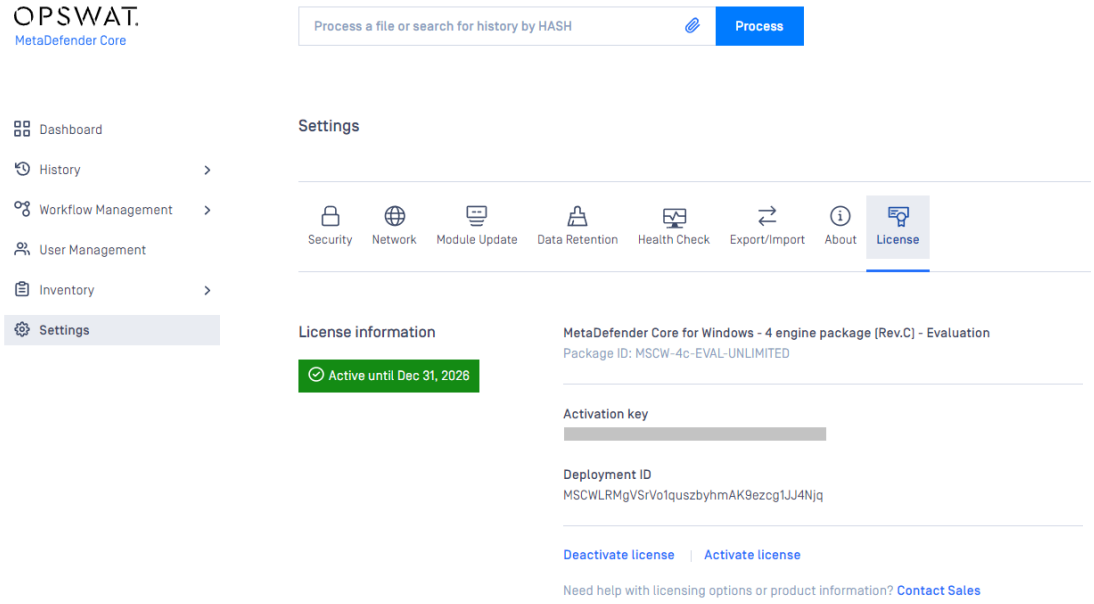
An evaluation license could be acquired by contacting our sales team.



Check Your License Details

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

- Product name: product name as on your order (e.g. MetaDefender Core for Windows - 4 engine package (Rev.C) - Evaluation)
- Package ID: product identification as on your order (e.g. MSCW-4c-EVAL-UNLIMITED)
- Expiration: last day of license validity
- Activation key: key used for activating product
- Deployment ID: identification of this installation


OPSWAT.
MetaDefender Core
Process a file or search for history by HASH
Process
Dashboard
History
Workflow Management
User Management
Inventory
Settings
Settings
Security
Network
Module Update
Data Retention
Health Check
Export/Import
About
License
License information
Active until Dec 31, 2026
MetaDefender Core for Windows - 4 engine package (Rev.C) - Evaluation
Package ID: MSCW-4c-EVAL-UNLIMITED
Activation key
Deployment ID
MSCWLRMgVSrVo1quszbyhmAK9ezcg1JJ4Njq
Deactivate license
Activate license
Need help with licensing options or product information? Contact Sales

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 Internet connection
- another computer that has 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, deployment ID and activation key are needed to generate the license file.
    a. Activation key should be purchased from OPSWAT.
    b. MetaDefender Core deployment ID can be queried by using REST API (for details see GET Get Current License Information API 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 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 successfully, a license file is downloaded. Save this file.

HTTP status codes can be:

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

Deactivation steps

1. For deactivation, following infomation is required:
    a. Activation key
    b. MetaDefender Core deployment ID: can be queried by using REST API (for details see GET Get Current License Information API 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 case of one or more major hardware or software parameter change the product might turn into deactivated status. Operating system updates, other software updates should not affecting the activation status.

If the product is online activated and has live Internet connection then in case of deployment ID change the product reactivates itself. If the product doesn't have Internet connection then the administrator is responsible to reactivate the product.

Because of the above 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

> **Info**
> For command line mode, upgrading (e.g. on **Linux**), prior to running the installer file, please make sure to **create the ignition file** with PostgreSQL server information included. Details:
>
> - Standalone DB mode: Command Line Installation (Windows and Linux)
> - Shared DB mode: Command Line Installation

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 the 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.

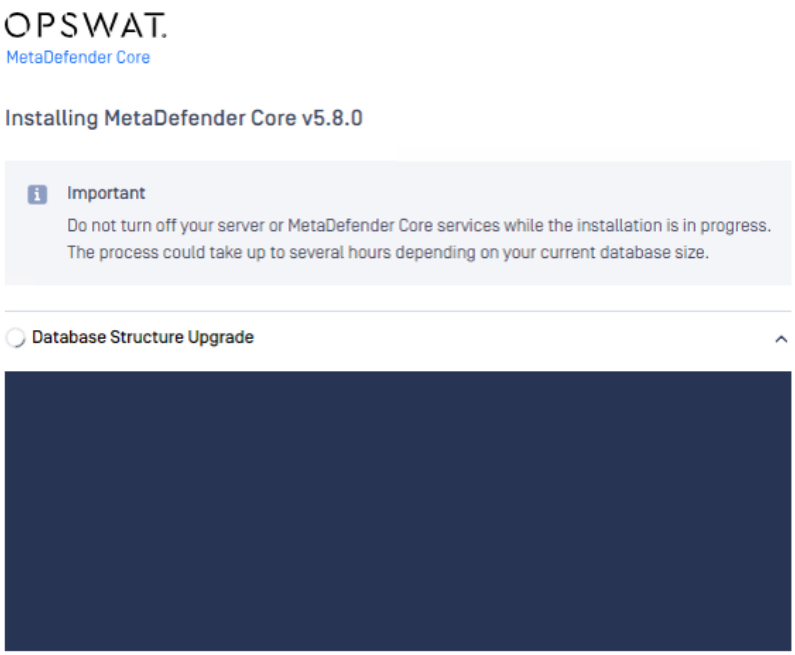
- Disk availability is greater than MetaDefender Core database size.

While doing the upgrade

Applicable for shared database deployment:

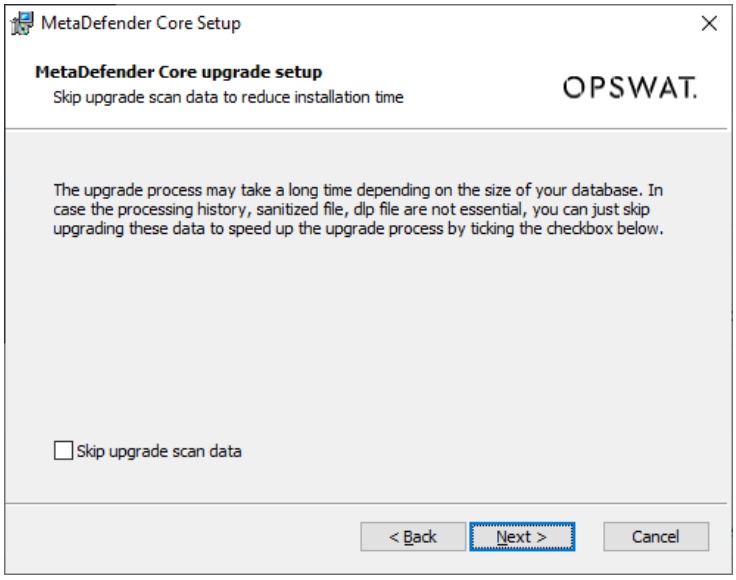
- Stop all MetaDefender Core instances' scanning service before performing the upgrade.
- Upgrade one (1) MetaDefender Core instance at a time.
- Proceed to upgrade MetaDefender Core instance where the bundle local PostgreSQL server is used for shared database setup.
- Otherwise is using remote PostgreSQL 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:

none

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

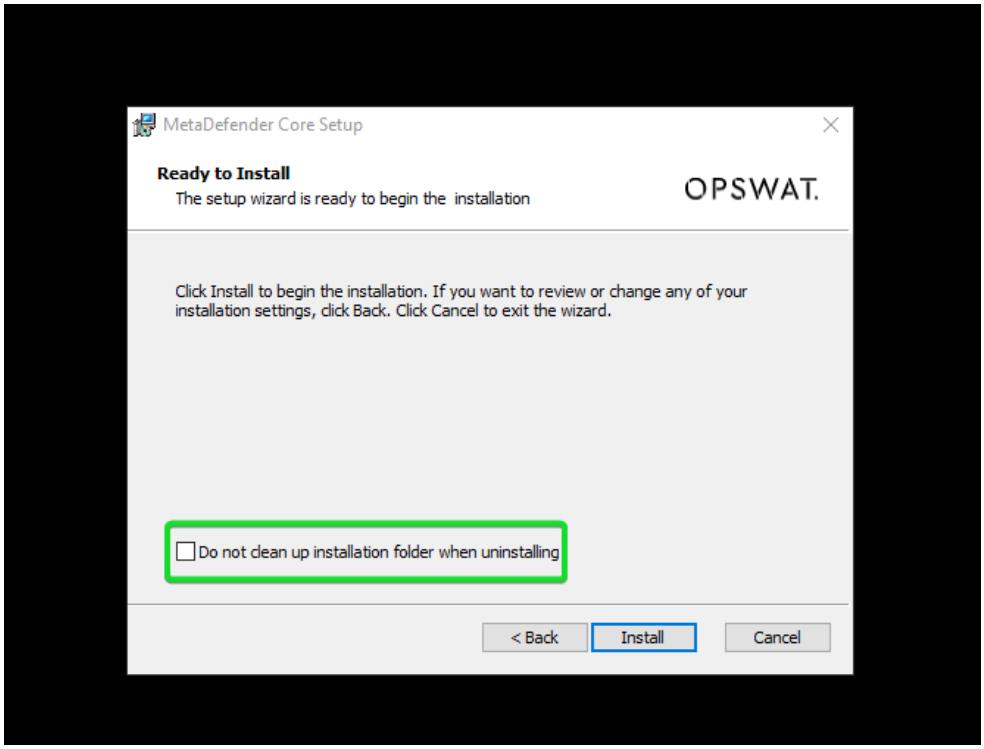
Info

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



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 API)	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 API)	Send cookie saved in step 2
4.	Getting result	If processing is in progress (See GET File Analysis API), wait a little while and repeat step 3.	Send cookie saved in step 2

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 API)	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) API)	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 API)	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 API	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 API). (This will only be successful if all the files sent to the batch have been processed already. Repeat this step until batch is closed.)	Send cookie saved in step 2
8.	Getting results	Request results related to batch ID saved in step 2. (See GET Download Batch Signed Result API	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	Further information
MetaDefender Kiosk	4.3.4	-
OPSWAT Client	Windows: 7.6.247.0 Mac: 10.4.243.0	-
MetaDefender Email Security	4.3.0	-
MetaDefender ICAP Server	4.3.0	-
MetaDefender Vault	1.3.0	-
MetaDefender for Secure Storage	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 API)	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 API)	Use IP saved in step 2
6.	Getting result	If processing is in progress (See GET File Analysis API), 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 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.	Open batch	Initiate processing file(s) in batch. Request a batch ID. (See POST Initiate Batch API)	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) API)	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 API)	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 API	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 API). (This will only be successful if all the files sent to the batch have been processed already. Repeat this step until batch is closed.)	Use IP saved in step 2
10.	Getting results	Request results related to batch ID saved in step 2. (See GET Download Batch Signed Result API).	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	Further information
MetaDefender Kiosk	does not support yet	-
MetaDefender Client	does not support yet	-
MetaDefender Email Security	does not support yet	-
MetaDefender ICAP Server	does not support yet	-
MetaDefender Vault	does not support yet	-

SSL connection for PostgreSQL communication

> **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

bash

```
openssl genrsa -aes256 -out server.key 2048
```

Sample output:

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

bash

```
openssl rsa -in server.key -out server.key
```

> **Note**
> On Linux, make sure to set permission on the server.key file

bash

```
chmod og-rwx server.key
```

3) Create a self-signed certificate

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.

none

```
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:

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:

none

```
ssl = on
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ca_file = '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)

none

```
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.

none

```
# 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:

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:

bash

```
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
```

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	c5.4xlarge	16	32	Up to 10	SSD	Amazon EC2 c5.4xlarge - Geekbench

Client tool

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

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	c5.2xlarge	8	16	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
	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 5.14.0
- 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, NANO, Comodo, VirusBlokAda, Zillya!
  - Deep CDR: 7.4.0
  - Proactive DLP: 2.23.0
  - Archive: 7.4.0
  - File type analysis: 7.4.0
  - File-based vulnerability assessment: 4.2.416.0

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update

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

- Max file size: 99999999
- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Performance test results

Summary

MetaDefender Core with single engine (technology):

Use case	Scan duration	Throughput	Avg. processing time	Avg./Max CPU usage	Avg./Max RAM usage	Avg. Network speed
	(minutes)	*(processed objects/hour)*	*(seconds/object)*	*(%)*	*(%)*	*(KB/s)*
Metascan 8	17.8	429,387	0.008	73.2 / 92	47 / 50	7,076
Metascan 12	29.6	258,131	0.014	86 / 99	51 / 56.2	4,370
Metascan 16	36.6	208,229	0.017	83.3 / 99.4	58.2 / 63	4,463
Deep CDR	19.0	401,349	0.009	80 / 95	50.1 / 54.6	5,573
Proactive DLP	14.2	536,902	0.007	55.4 / 78.4	48.4 / 50.5	8,924
Vulnerability	13.2	577,242	0.006	60 / 84	48.4 / 51	9,482

MetaDefender Core with common engine packages:

Use case	Scan duration	Throughput	Avg. processing time	Avg./Max CPU usage	Avg./Max RAM usage	Avg. Network speed
	(minutes)	*(processed objects/hour)*	*(seconds/object)*	*(%)*	*(%)*	*(KB/s)*
Metascan 8 **+ Deep CDR**	24.2	315,830	0.011	89 / 97.3	50.3 / 56.3	5,789
Metascan 8 **+ Deep CDR** **+ Proactive DLP**	26.8	284,531	0.013	92.2 / 97.3	52.4 / 58.7	5,849
Metascan 8 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	27.7	275,255	0.013	94 / 98.5	53.1 / 59.5	4,314
Metascan 12 **+ Deep CDR**	38	200,713	0.018	92 / 99.5	53.4 / 60	3,779
Metascan 12 **+ Deep CDR** **+ Proactive DLP**	41	186,722	0.019	95 / 99.8	57 / 64.5	3,504
Metascan 12 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	43	177,963	0.02	92.8 / 99.7	53.1 / 61	3,643
Metascan 16 **+ Deep CDR**	43.2	176,734	0.02	93.5 / 99.4	63.3 / 70.8	3,023.3

Use case	Scan duration	Throughput	Avg. processing time	Avg./Max CPU usage	Avg./Max RAM usage	Avg. Network speed
Metascan 16 **+ Deep CDR** **+ Proactive DLP**	46.6	163,570	0.022	92.4 / 99.5	62 / 68.5	3,548.5
Metascan 16 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	47.6	160,264	0.022	92.5 / 99.7	61 / 66.5	3,467

System resource utilization charts

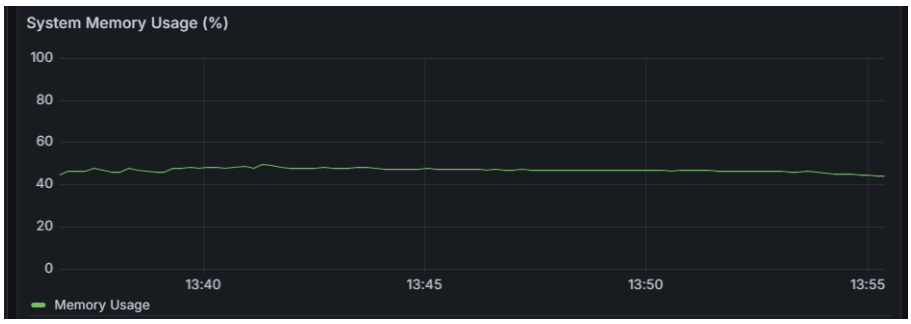
MetaDefender Core with Metascan 8:

Utilization charts

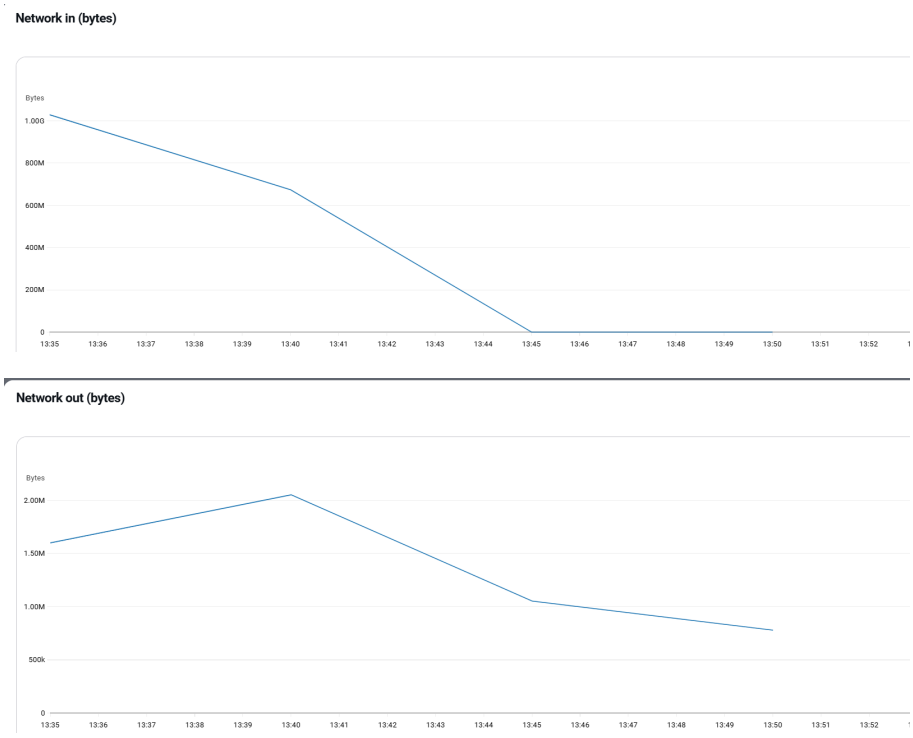
CPU usage



Memory usage



Network usage



Utilization charts

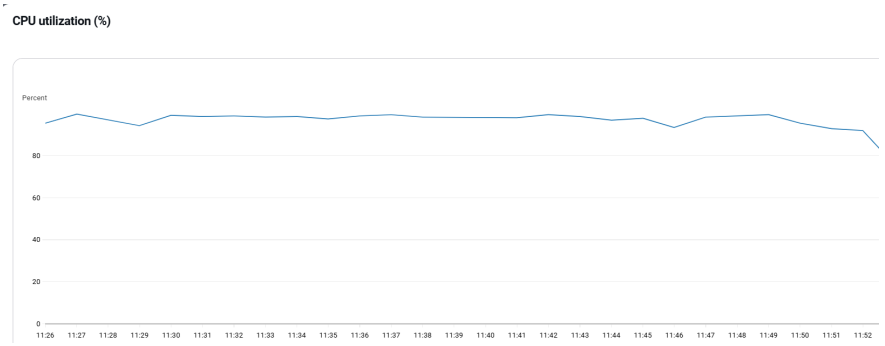
Disk usage



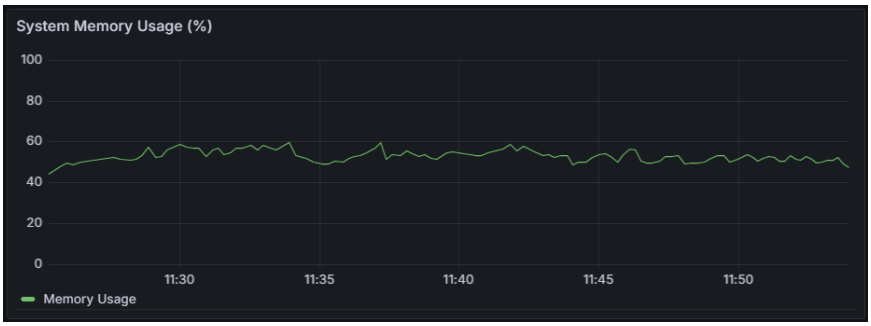
MetaDefender Core with Metascan 8 + Deep CDR + Proactive DLP + Vulnerability:

Utilization charts

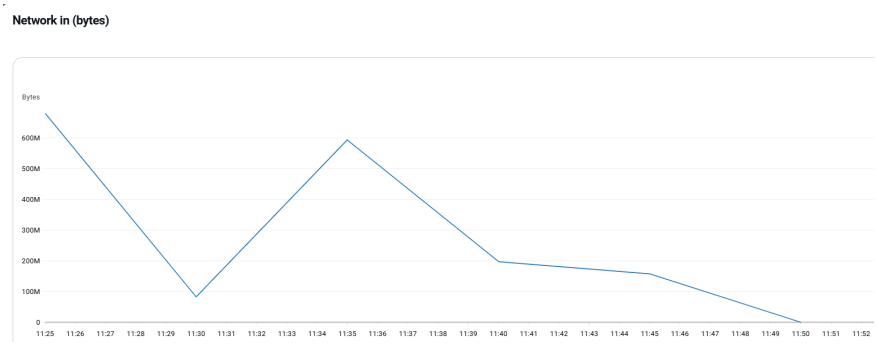
CPU usage

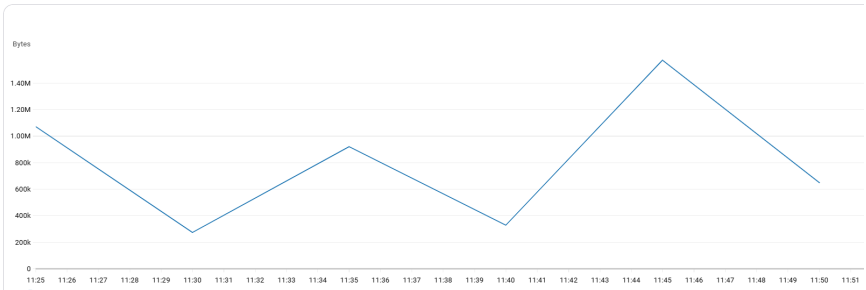


Memory usage



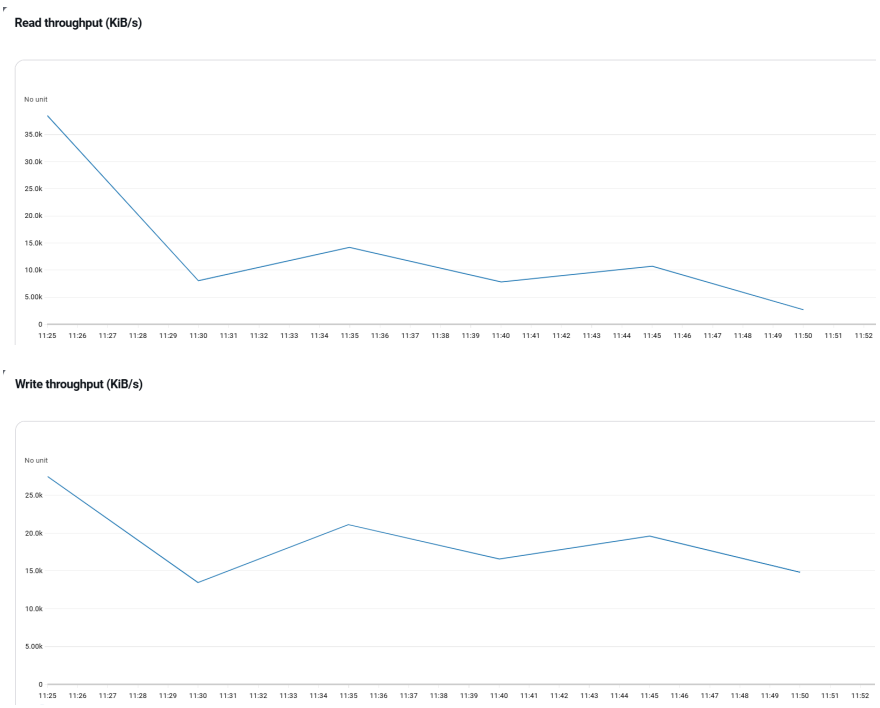
Network usage





Utilization charts

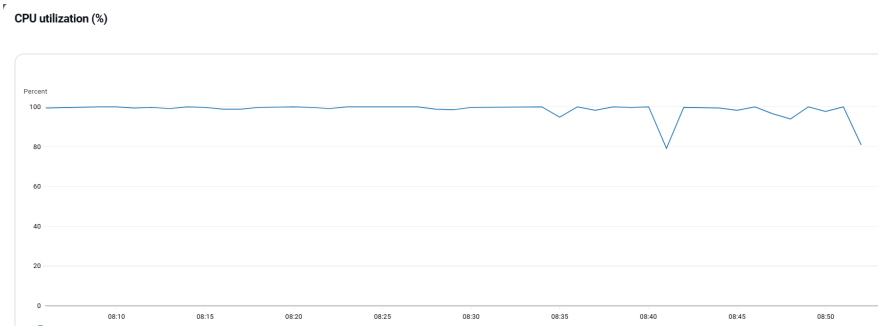
Disk usage



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

Utilization charts

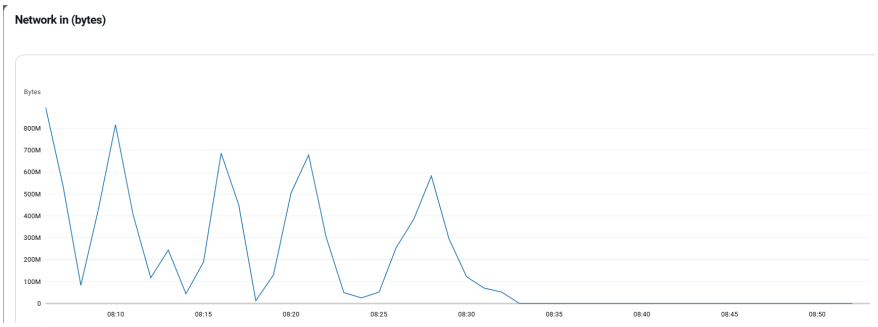
CPU usage



Memory usage



Network usage

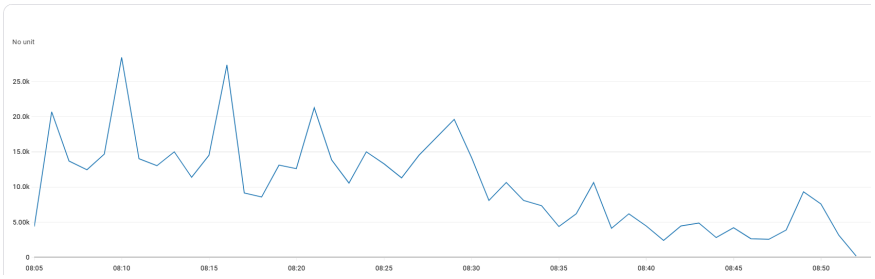




Utilization charts

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
Debian 12	c5.4xlarge	16	32	Up to 10	SSD	Amazon EC2 c5.4xlarge - Geekbench

Client tool

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

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
	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 5.14.0
- Engines:
  - Metascan 5: Ahnlab, Bitdefender, ClamAV, ESET, K7
  - Metascan 10: Metascan 5, Avira, Quick Heal, Ikarus, Cyren, Tachyon
  - Metascan MAX: Metascan 10, NANO, Sophos, McAfee, CrowdStrike Falcon ML, Lionic, Xvirus, Webroot SMD, Cylance, RocketCyber, CMC
  - Deep CDR: 7.4.1
  - Proactive DLP: 2.23.0
  - Archive: 7.4.1
  - File type analysis: 7.4.1
  - File-based vulnerability assessment: 4.57-236

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update

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

- Max file size: 99999999
- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Performance test results

Summary

MetaDefender Core with single engine (technology):

Use case	Scan duration	Throughput	Avg. processing time	Avg./Max CPU usage	Avg./Max RAM usage	Avg. Network speed
	(minutes)	*(objects/hour)*	*(seconds/object)*	*(%)*	*(%)*	*(KB/s)*
Metascan 5	17.3	441,362	0.008	42.4 / 68.6	25.8 / 27	9,742
Metascan 10	22.6	338,349	0.011	69.3 / 97	34.5 / 35.7	7,682
Metascan MAX	65	117,361	0.031	55.3 / 99.5	54.4 / 75	5,267
Deep CDR	19.4	353,031	0.010	68.7 / 91.7	51 / 54.4	8,589
Proactive DLP	16.5	434,071	0.008	45.2 / 72.1	44.2 / 51.6	8,023
Vulnerability	12	451,079	0.008	36.7 / 70.2	40.6 / 41	10,784

MetaDefender Core with common engine packages:

Use case	Scan duration	Throughput	Avg. processing time	Avg./Max CPU usage	Avg./Max RAM usage	Avg. Network speed
	(minutes)	*(objects/hour)*	*(seconds/object)*	*(%)*	*(%)*	*(KB/s)*
Metascan 5 **+ Deep CDR**	23.5	325,272.6	0.011	76 / 98.7	37.6 / 39.4	6,843
Metascan 5 **+ Deep CDR** **+ Proactive DLP**	29.5	258,531	0.014	81.7 / 99.4	42 / 50.5	8,714
Metascan 5 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	30.7	248,720	0.014	86.4 / 99.6	40.8 / 45.4	6,373
Metascan 10 **+ Deep CDR**	30.4	251,366	0.014	88.3 / 98.3	46.5 / 51.5	6,143
Metascan 10 **+ Deep CDR** **+ Proactive DLP**	34.4	221,691	0.016	93 / 99.8	50 / 59	6,017
Metascan 10 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	36.6	208,259	0.017	93.7 / 99.8	50.2 / 56.6	6,274
Metascan MAX **+ Deep CDR**	71.1	107,266	0.034	65.4 / 99.5	64.3 / 71.7	6,853

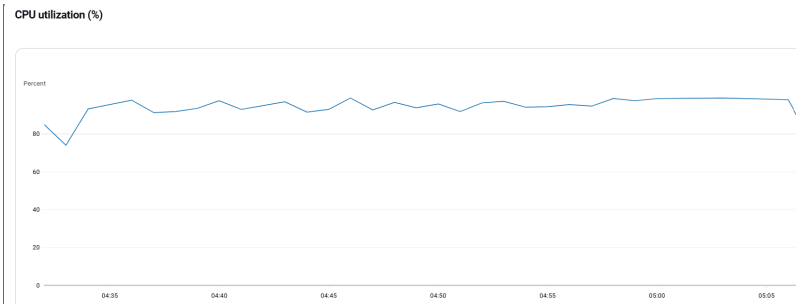
Use case	Scan duration	Throughput	Avg. processing time	Avg./Max CPU usage	Avg./Max RAM usage	Avg. Network speed
Metascan MAX **+ Deep CDR** **+ Proactive DLP**	72.1	105,831	0.034	79.8 / 99.9	65.3 / 76.7	6,467
Metascan MAX **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	72.9	104,670	0.034	80 / 99.8	66 / 76.8	6,183

System resource utilization charts

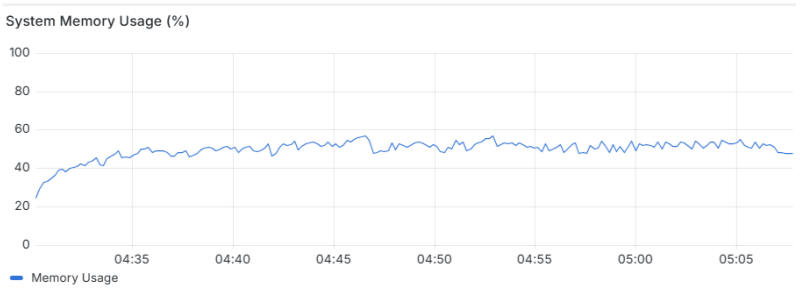
MetaDefender Core with Metascan 10 + Deep CDR + Proactive DLP + Vulnerability:

Utilization charts

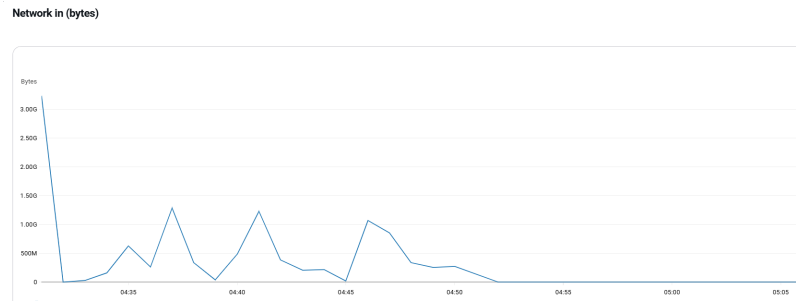
CPU usage

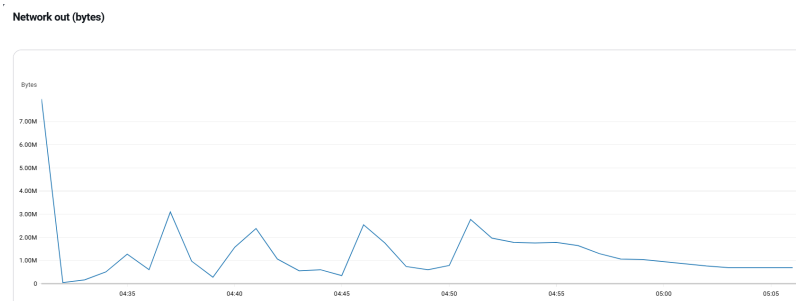


Memory usage



Network usage





Utilization charts

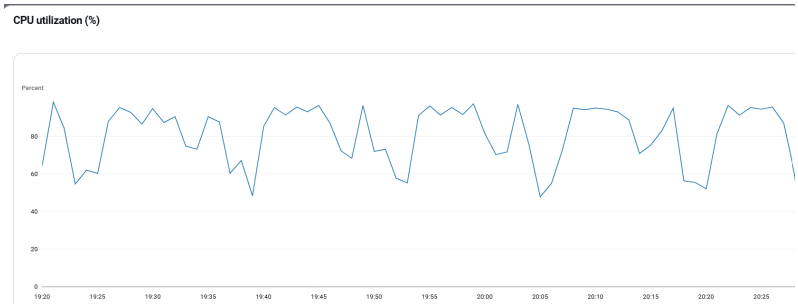
Disk usage



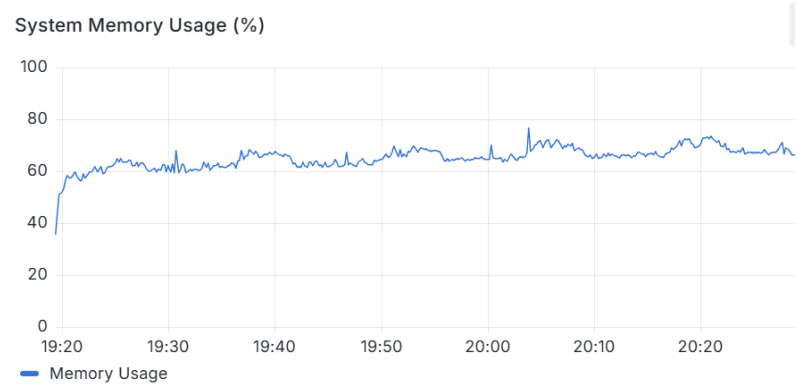
MetaDefender Core with Metascan MAX + Deep CDR + Proactive DLP + Vulnerability:

Utilization charts

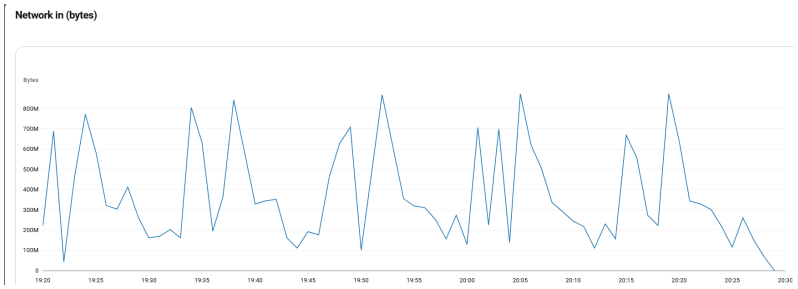
CPU usage

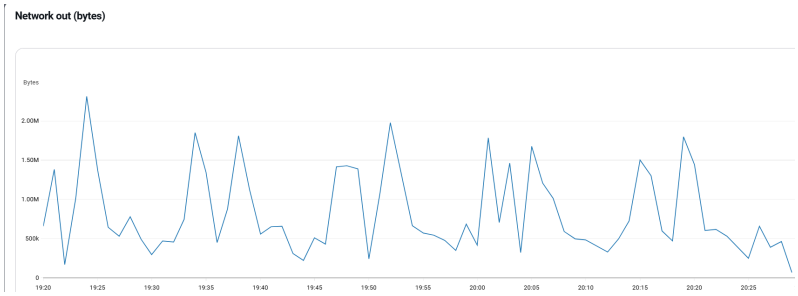


Memory usage



Network usage

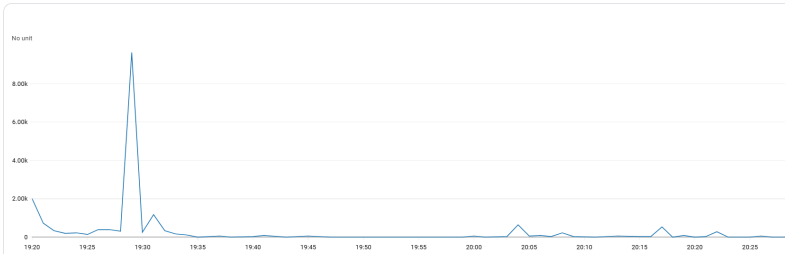




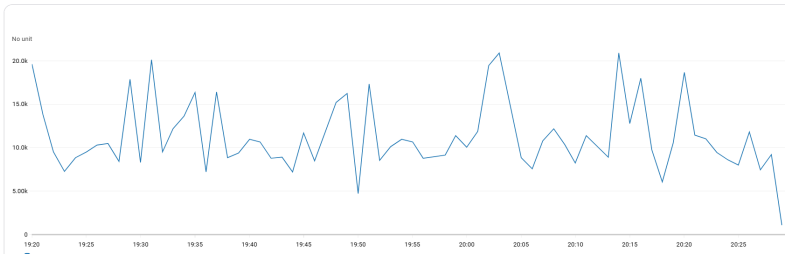
Utilization charts

Disk usage

Read throughput (KiB/s)



Write throughput (KiB/s)



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:

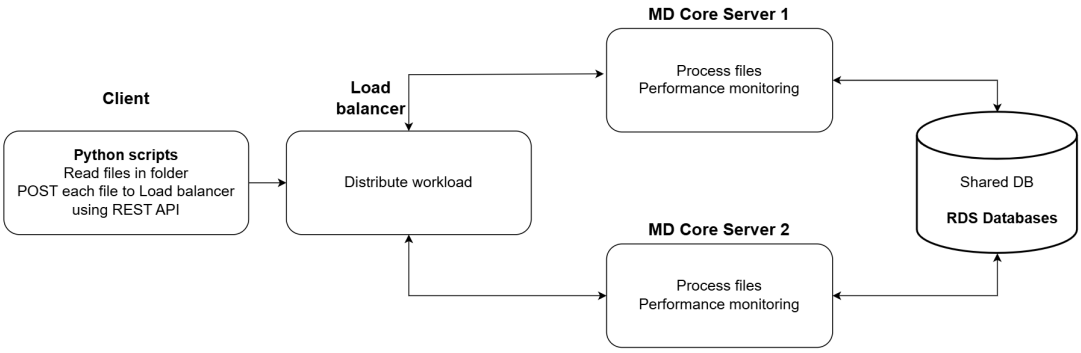
MetaDefender Core

	OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type	Benchmark
MetaDefender Core #1	Windows Server 2022	c5.4xlarge	16	32	Up to 10	SSD	Amazon EC2 c5.4xlarge - Geekbench
MetaDefender Core #2	Windows Server 2022	c5.4xlarge	16	32	Up to 10	SSD	Amazon EC2 c5.4xlarge - Geekbench

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.

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
	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 5.14.0
- 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, NANO, Comodo, VirusBlokAda, Zillya!
  - Deep CDR: 7.4.0
  - Proactive DLP: 2.23.0
  - Archive: 7.4.0
  - File type analysis: 7.4.0
  - File-based vulnerability assessment: 4.2.416.0

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update

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

- Max file size: 99999999
- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Performance test results

MetaDefender Core with single engine (technology)

Summary metrics:

Use case	Scan duration	Throughput	Avg. processing time
	(minutes)	*(processed objects/hour)*	*(seconds/object)*
Metascan 8	8	953,812	0.004
Metascan 12	14.4	529,895	0.007
Metascan 16	15.7	486,019	0.007
Deep CDR	9.51	802,435	0.004
Proactive DLP	7.2	1,047,992	0.003
Vulnerability	6.2	1,230,726	0.003

System resource utilization:

Use case	Avg./Max CPU usage	Avg./Max CPU usage	Avg./Max RAM usage	Avg./Max RAM usage	Avg. Network speed	Avg. Network speed
	Core 1	Core 2	Core 1	Core 2	Core 1	Core 2
	(%)	*(%)*	*(%)*	*(%)*	*(KB/s)*	*(KB/s)*
Metascan 8	73.5 / 92.4	75.5 / 92	49.5 / 55	43 / 52	9,570	9,145
Metascan 12	87 / 98	89.4 / 99	53 / 57.4	44 / 49	5,657	5,840
Metascan 16	80 / 98.6	80.4 / 98.2	56.8 / 59.5	50.7 / 54.2	4,168	6,031
Deep CDR	72.4 / 91.6	72 / 90.5	52.6 / 60	45.5 / 53	5,617	5,602
Proactive DLP	43.2 / 80	49.4 / 81	51.3 / 57.5	44.1 / 53.1	8,460	8,383
Vulnerability	61 / 85.5	61.7 / 88.3	48.4 / 53.5	41.5 / 49.3	11,692	12,076

MetaDefender Core with common engine packages

Summary metrics:

Use case	Scan duration	Throughput	Avg. processing time
	(minutes)	(processed objects/hour)	*(seconds/object)*
Metascan 8 + Deep CDR	13.2	578,068	0.006
Metascan 8 + Deep CDR + Proactive DLP	14.92	511,553	0.007
Metascan 8 + Deep CDR + Proactive DLP + Vulnerability	15.6	488,618	0.007
Metascan 12 + Deep CDR	18.4	414,701	0.009
Metascan 12 + Deep CDR + Proactive DLP	19.3	395,362	0.009
Metascan 12 + Deep CDR + Proactive DLP + Vulnerability	19.4	393,324	0.009
Metascan 16 + Deep CDR	21.1	361,635	0.01
Metascan 16 + Deep CDR + Proactive DLP	22.5	339,133	0.01
Metascan 16 + Deep CDR + Proactive DLP + Vulnerability	22.7	336,887	0.01

System resource utilization:

Use case	Avg./Max CPU usage	Avg./Max CPU usage	Avg./Max RAM usage	Avg./Max RAM usage	Avg. Network speed	Avg. Network speed
	Core 1	Core 2	Core 1	Core 2	Core 1	Core 2
	(%)	*(%)*	*(%)*	*(%)*	*(KB/s)*	*(KB/s)*
Metascan 8 **+ Deep CDR**	86.6 / 99	84.5 / 98.5	53.3 / 62.5	45.6 / 52	4,589	5,148
Metascan 8 **+ Deep CDR** **+ Proactive DLP**	83 / 99	83.7 / 99.4	52.7 / 61.8	46 / 54	4,124	4,038
Metascan 8 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	87 / 99.7	89 / 99.5	53 / 61.2	45.3 / 52	4,513	4,079
Metascan 12 **+ Deep CDR**	87.3 / 99	90.4 / 99.3	55.1 / 62.3	46.8 / 53.8	3,438	3,340
Metascan 12 **+ Deep CDR** **+ Proactive DLP**	90.7 / 99.3	90 / 99.5	56.8 / 63	47.3 / 56.2	4,122	3,337
Metascan 12 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	91.3 / 99.7	94.5 / 99.8	56.8 / 62.6	50.7 / 56.4	4,154	3,311
Metascan 16 **+ Deep CDR**	87.3 / 99.5	87.8 / 99.3	62.8 / 67.5	54.5 / 59.5	3,313	2,934

Use case	Avg./Max CPU usage	Avg./Max CPU usage	Avg./Max RAM usage	Avg./Max RAM usage	Avg. Network speed	Avg. Network speed
Metascan 16 **+ Deep CDR** **+ Proactive DLP**	92.2 / 99.5	91.6 / 99.7	63.3 / 69.3	56 / 66.6	3,146	2,945
Metascan 16 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	95.4 / 99.5	94.8 / 99.6	64 / 69.2	55.3 / 62.5	3,810	2,677

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:

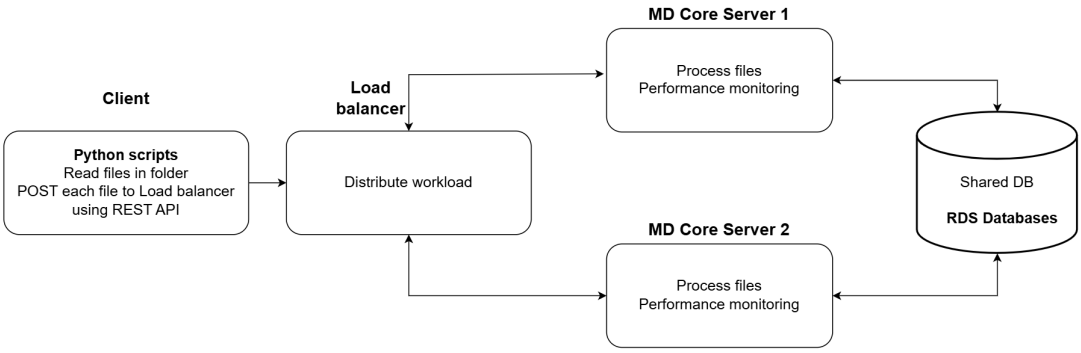
MetaDefender Core

	OS	AWS instance type	vCPU	Memory (GB)	Network bandwidth (Gbps)	Disk type	Benchmark
MetaDefender Core #1	Debian 12	c5.4xlarge	16	32	Up to 10	SSD	Amazon EC2 c5.4xlarge - Geekbench
MetaDefender Core #2	Debian 12	c5.4xlarge	16	32	Up to 10	SSD	Amazon EC2 c5.4xlarge - Geekbench

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.

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
	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 5.14.0
- Engines:
  - Metascan 5: Ahnlab, ClamAV, ESET, Bitdefender, K7.
  - Metascan 10: Metascan 5, Avira, Quick Heal, VirIT Explorer, Varist, Ikarus, Tachyon.
  - Metascan MAX: Metascan 10, NANO, Sophos, McAfee, CrowdStrike, Lionic, Xvirus, WebRoot, Cylance, RocketCyber, CMC.
  - Deep CDR: 7.4.0
  - Proactive DLP: 2.23.0
  - Archive: 7.4.0
  - File type analysis: 7.4.0
  - File-based vulnerability assessment: 4.57-236

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update

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

- Max file size: 99999999
- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Performance test results

MetaDefender Core with single engine (technology)

Summary metrics:

Use case	Scan duration	Throughput	Avg. processing time
	(minutes)	*(processed objects/hour)*	*(seconds/object)*
Metascan 5	8.1	942,037	0.004
Metascan 10	10.9	700,045	0.005
Metascan MAX	31.7	240,709	0.014
Deep CDR	11.1	687,432	0.005
Proactive DLP	8.5	897,705	0.004
Vulnerability	8.4	908,392	0.004

System resource utilization:

Use case	Avg./Max CPU usage	Avg./Max CPU usage	Avg./Max RAM usage	Avg./Max RAM usage	Avg. Network speed	Avg. Network speed
	Core 1	Core 2	Core 1	Core 2	Core 1	Core 2
	(%)	*(%)*	*(%)*	*(%)*	*(KB/s)*	*(KB/s)*
Metascan 5	32.9 / 64.8	32.2 / 64.6	36.5 / 45.6	37 / 45.8	8,765	8,426
Metascan 10	64 / 94.5	62.4 / 95.2	42.5 / 46	41 / 44.7	6,533	6,502
Metascan MAX	67.3 / 99.2	66.2 / 99.4	54 / 74	54 / 74.6	4,310	4,753
Deep CDR	63.6 / 91.5	64 / 92	50.6 / 54	51.2 / 54.7	6,005	5,601
Proactive DLP	43.5 / 72.6	46.2 / 74	43.8 / 50	44.7 / 50.4	6,789	6,235
Vulnerability	36.2 / 71	36.8 / 71	41.5 / 41.7	40.3 / 40.8	6,901	6,763

MetaDefender Core with common engine packages

Summary metrics:

Use case	Scan duration	Throughput	Avg. processing time
	(minutes)	(processed objects/hour)	*(seconds/object)*
Metascan 5 + Deep CDR	12.1	630,619	0.006
Metascan 5 + Deep CDR + Proactive DLP	14.7	519,081	0.007
Metascan 5 + Deep CDR + Proactive DLP + Vulnerability	15.3	498,725	0.007
Metascan 10 + Deep CDR	14.6	522,636	0.007
Metascan 10 + Deep CDR + Proactive DLP	17.5	436,028	0.008
Metascan 10 + Deep CDR + Proactive DLP + Vulnerability	17.9	426,284	0.008
Metascan MAX + Deep CDR	34.5	221,173	0.016
Metascan MAX + Deep CDR + Proactive DLP	35.1	217,393	0.016
Metascan MAX + Deep CDR + Proactive DLP + Vulnerability	35.7	213,739	0.016

System resource utilization:

Use case	Avg./Max CPU usage	Avg./Max CPU usage	Avg./Max RAM usage	Avg./Max RAM usage	Avg. Network speed	Avg. Network speed
	Core 1	Core 2	Core 1	Core 2	Core 1	Core 2
	(%)	*(%)*	*(%)*	*(%)*	*(KB/s)*	*(KB/s)*
Metascan 5 **+ Deep CDR**	68.8 / 93	68 / 93.2	45.7 / 49 .6	46 / 50.8	5,526	5,420
Metascan 5 **+ Deep CDR** **+ Proactive DLP**	76 / 93.6	71.3 / 94.8	47.2 / 51.5	48.3 / 51.3	5,996	5,195.1
Metascan 5 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	82.5 / 96.4	81.4 / 96.2	49.6 / 53	49.5 / 54.8	5,482	4,579
Metascan 10 **+ Deep CDR**	83 / 95.8	85.7 / 95	47.5 / 51	47 / 49.8	4,794	4,777
Metascan 10 **+ Deep CDR** **+ Proactive DLP**	89 / 97	86.7 / 95.7	49.3 / 57.5	50 / 56.4	4,468	4,352
Metascan 10 **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	89.5 / 99	92.5 / 99.3	50.6 / 58.3	51.5 / 59	3,879	4,404
Metascan MAX **+ Deep CDR**	83 / 98.5	83 / 99.2	64 / 74.3	64 / 76.3	4,254	4,205

Use case	Avg./Max CPU usage	Avg./Max CPU usage	Avg./Max RAM usage	Avg./Max RAM usage	Avg. Network speed	Avg. Network speed
Metascan MAX **+ Deep CDR** **+ Proactive DLP**	84.8 / 99	84.5 / 99.3	64 / 75	64.3 / 76.8	4,167	4,018
Metascan MAX **+ Deep CDR** **+ Proactive DLP** **+ Vulnerability**	86.3 / 99.2	87 / 99.6	64.3 / 75	65.6 / 78.2	4,192	4,073

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.

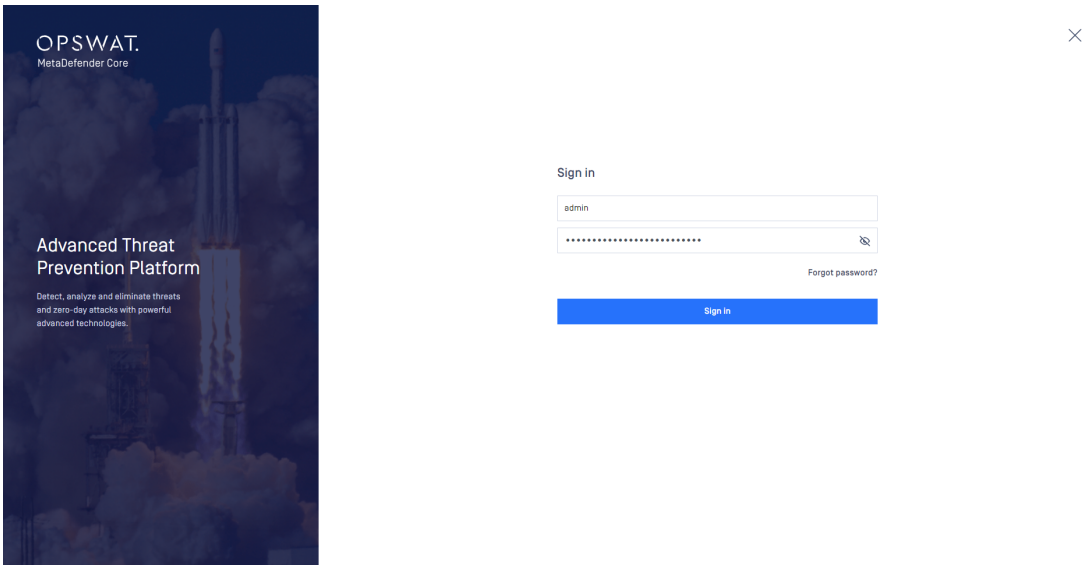
MetaDefender Core Documentation

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

Document	SHA256
MetaDefender Core manual	
OPSWAT MetaDefender AGD Documentation_v1.3	DA7AB11F8195F0E49A2D4ED36CA3C99510F77B5A1881DF937B14F5AFFC3A20C4

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



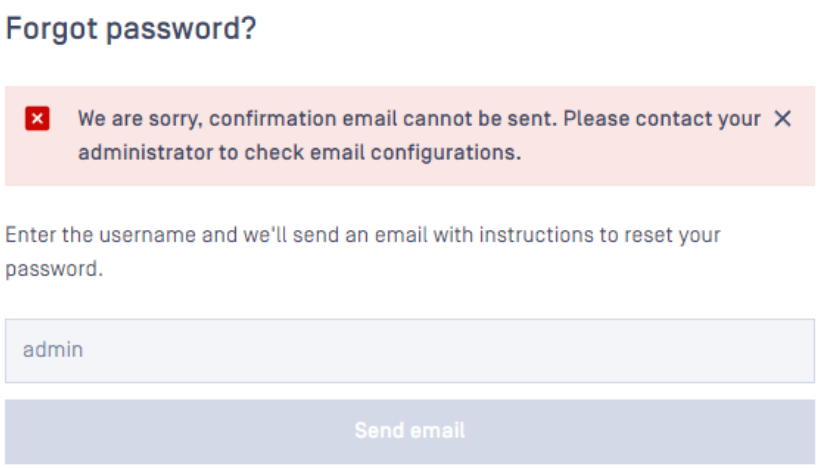
The "Forgot Password" feature requires authentication via email, and force the affected user to change their password at the first login time for security reason. More details at Password Recovery.

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.

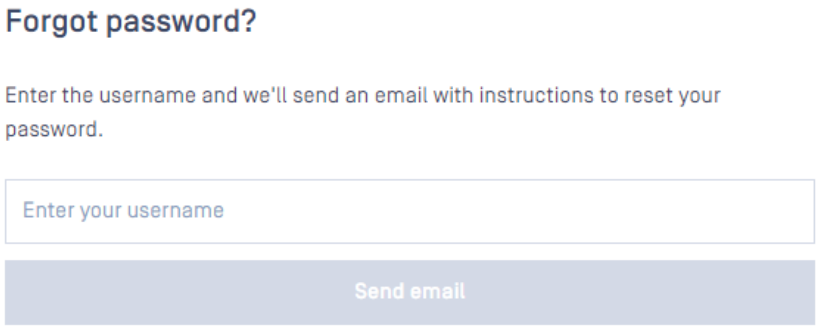
Both methods requires authentication via email, and force the affected user to change their password at the first login time for security reason.

Forgot Password

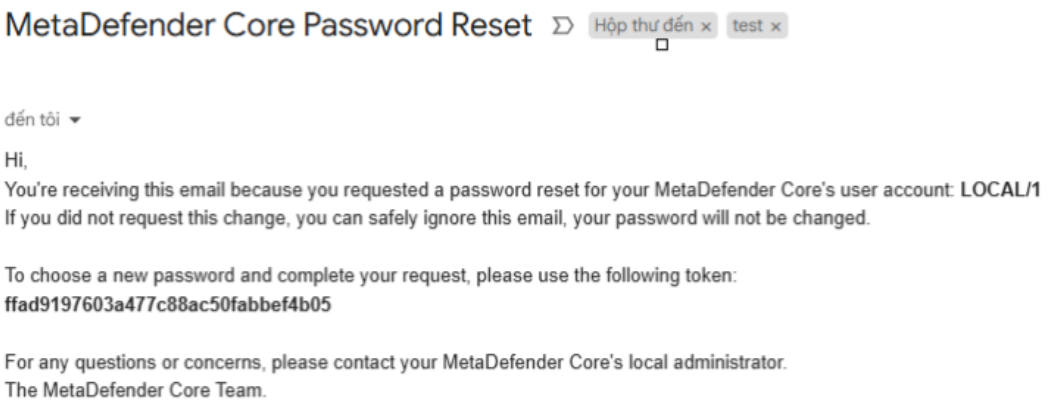
Any local user registered with email on MetaDefender Core should be able to reset their own password by clicking on **Forgot password?** link on login page.



You will be redirected to Recover Password page.

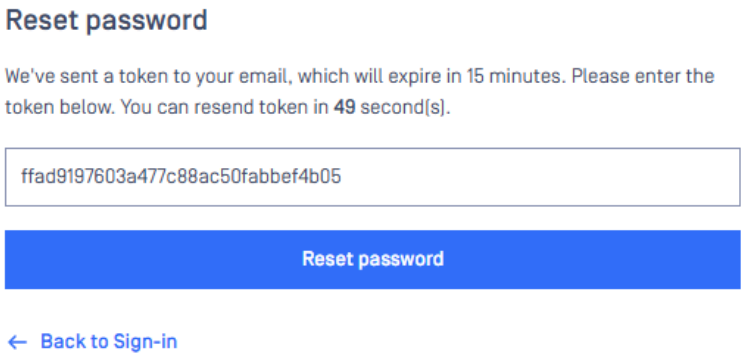


An email with a token will be sent to user's registered email entitled `MetaDefender Core Password Reset`

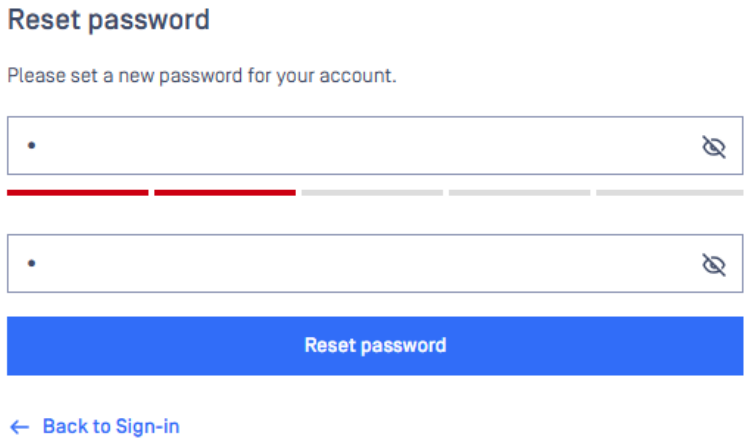


> **Warning**
> If that user does not take any action, the token will be expired in 15 minutes

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



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

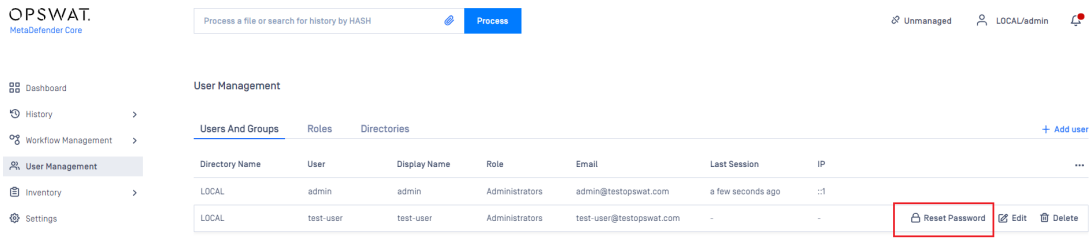


You will be automatically redirected to MetaDefender Core dashboard after resetting password successfully.

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.

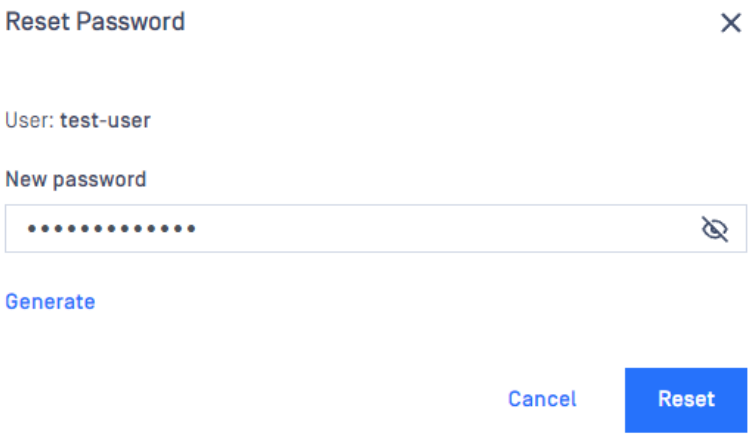
Go to "User Management" → under "Users And Groups", mouse over on user to chose "Reset Password" option



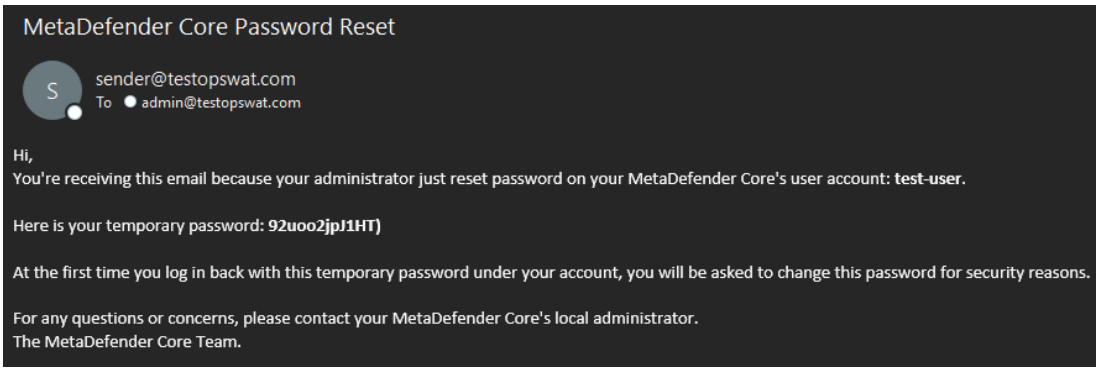
Admin to RESET PASSWORD

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

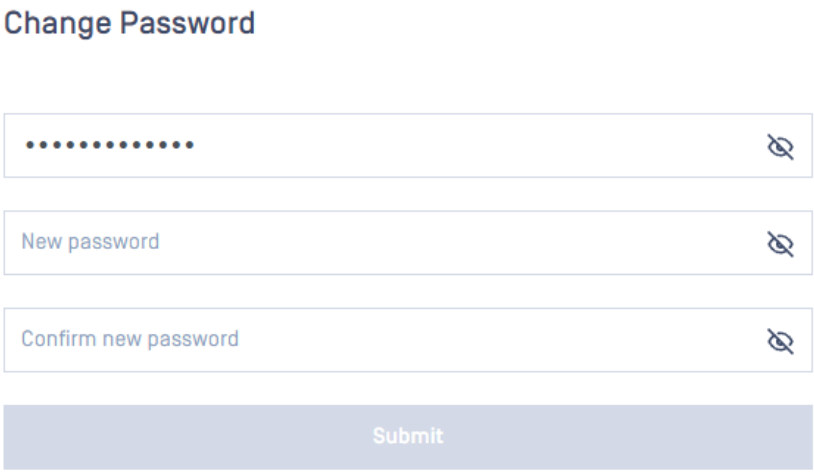
Click RESET PASSWORD button once done.



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	default value	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.
`restport`	8008	required	Designated port number for the web management console and REST interface
`report_engine_issue`	true	optional	Enable reporting of engine issue count. (possible values: "true" or "false")
`dlppath`	[Data directory]/dlp	optional	Directory for DLP-processed database and items
`quarantinepath`	[Data directory]/quarantine	optional	Directory for quarantine database and quarantined items
`sanitizepath`	[Data directory]/sanitized	optional	Directory for sanitized database and sanitized items

parameter	default value	required	description
`dbmode`	1	optional	Support database mode, possible values: • 0: Not used for now, reserved • 1: Standalone Core (default) • 2: Core instance in Central Hub 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 MetaDefender Core to access device and create temporary folder on it.

[logger] section

key	default value	required	description
`logfile`	/var/log/ometascan/ometascan.log	optional	Full path of a logfile to write log messages to
`loglevel`	info	optional	Level of logging. Supported values are: debug, info, warning, error
`log_rotation`	1	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: ◦ 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`	false	optional	Set local timezone for events sending to local syslog server

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

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

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

[internal] section

key	default value	required	description
`db_connection`	10	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
`data_directory`	/var/lib/ometascan	optional	Full path for MD Core's data (database, updates etc.) E.g. /var/lib/ometascan/test
`skip_upgrade_scan_data`	0	optional	When enabled (set to 1), upgrading MetaDefender Core will auto skip 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.
`parallelcount`	20	optional	Set maximum number of threads (files) sending to engine at the same time, applicable to all engines Exception: • Archive engine (extraction): default = -1 (unlimited) • Archive engine (compression): default = 20 • Proactive DLP engine: default = 5 • Sandbox engine: default=5

key	default value	required	description

`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
			• **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`	1000	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.

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	default value	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.
`restport`	8008	string value	required	Designated port number for the web and REST interface
`report_engine_issue`	true	string value	optional	Enable reporting of engine issue count. (possible values: "true" or "false").
`dlppath`	[installdir]\data\dlp	string value	optional	Directory for DLP-processed database and items
`quarantinepath`	[installdir]\data\quarantine	string value	optional	Directory for quarantine database and quarantined items
`sanitizepath`	[installdir]\data\sanitized	string value	optional	Directory for sanitized database and sanitized items

parameter	default value	type	required	description
`dbmode`	1	string value	optional	Support database mode, possible values: • 0: Not used for now, reserved • 1: Standalone Core (default) • 2: Core instance in Central Hub 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`		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.

Logger

`HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`

key	default value	type	required	description
`logfile`		string value	optional	Location of a logfile to write log messages to.
`loglevel`		string value	optional	Level of logging. Supported values are: debug, info, warning, error. Must set value on this key when **logfile** key is also set accordingly.
`log_rotation`	1	string value	optional	This setting is only applicable on Windows only (on Linux, we use built-in OS log rotation). Should only set this key when **logfile** key is also set accordingly. Supported values: • 0: Core logs are not rotated. • 1 (default mode), enable to rotate log: ◦ 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.
`wineventlog_level`	info	string value	optional	Level of logging. Supported values are: debug, info, warning, error.
`syslog`		string value	optional	Value can only by in form of 'protocol://<hostname>:<port>' (Multiple server can be specified separated with comma). More details: HERE
`syslog_level`		string value	optional	Level of logging. Supported values are: debug, info, warning, error. Must set value on this key when **syslog** key is also set accordingly.
`local_timezone`	false	string value	optional	Set local timezone for events sending to local syslog server.

key	default value	type	required	description
`override`		string value	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.
`cef`	false	string value	optional	If true, the log format is Common Event Format.
`nginx_logfile`	[installdir] ginx ginx.log	string value	optional	File name and path to store the NGINX logs.
`nginx_log_rotation`	1	string value	optional	This setting is only applicable on Windows only (on Linux, we use built-in OS log rotation). Should only set this key when **nginx_logfile** key is also set accordingly. Supported values: • 0: Nginx logs are not rotated. • 1 (default), enable to rotate log: ◦ Rotation process will be performed every day, regardless of file size. ◦ 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.: nginxlog.log-20200330.gz), all saved in same location with what you set in **nginx_logfile.** ◦ All generated log packages included in MetaDefender Core support package

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	default value	type	required	description
`db_connection`	10	string value	optional	Define maximum number of concurrent connections allows MetaDefender Core to open to work with PostgreSQL database server. Only available since MetaDefender Core 4.19.1
`data_directory`	<MD Core installation folder>\data	string value	optional	Full path for MD Core's data (database, updates etc.) E.g. D:\custom_path
`skip_upgrade_scan_data`	0	string value	optional	When enabled (set to 1), upgrading MetaDefender Core will auto skip 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.
`parallelcount`	20	string value	optional	Set maximum number of threads (files) sending to engine at the same time, applicable to all engines **Exception:** • Archive engine (extraction): default = -1 (unlimited) • Archive engine (compression): default = 20 • Proactive DLP engine: default = 5 • Sandbox engine: default=5

key	default value	type	required	description

`parallelcount_<enginename>`		string value	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
				• **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`	1000	string value	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.

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.

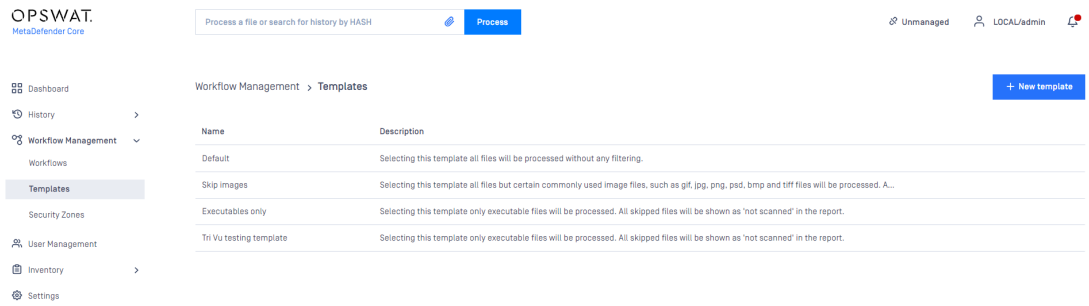
Workflow Configuration Template

The Workflow templates page is found under **Policy > Workflow 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 can not 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

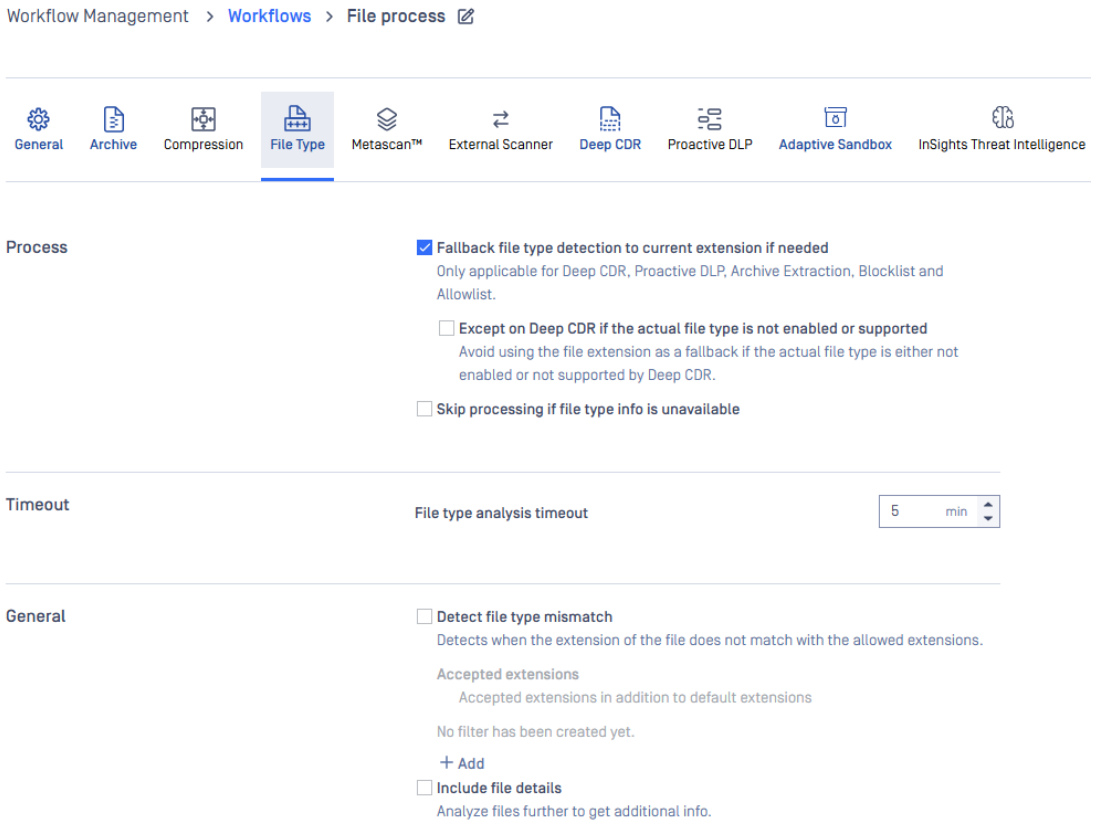
When clicking on a workflow template a windows pops up showing different tabs related to the workflow templates different kind of properties.

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.

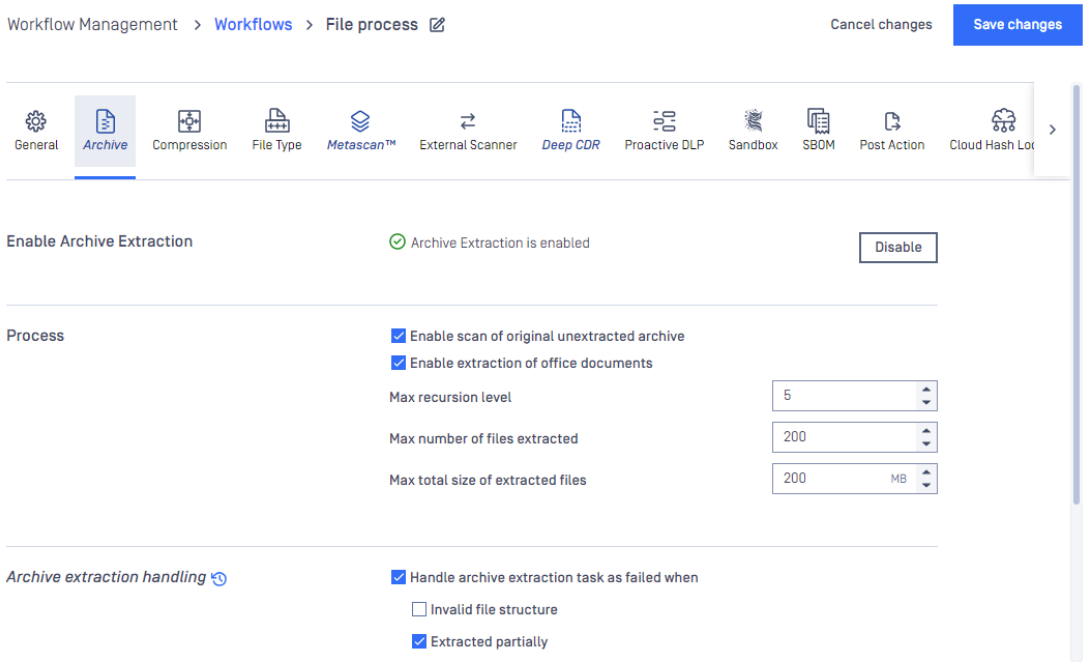


Archive

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

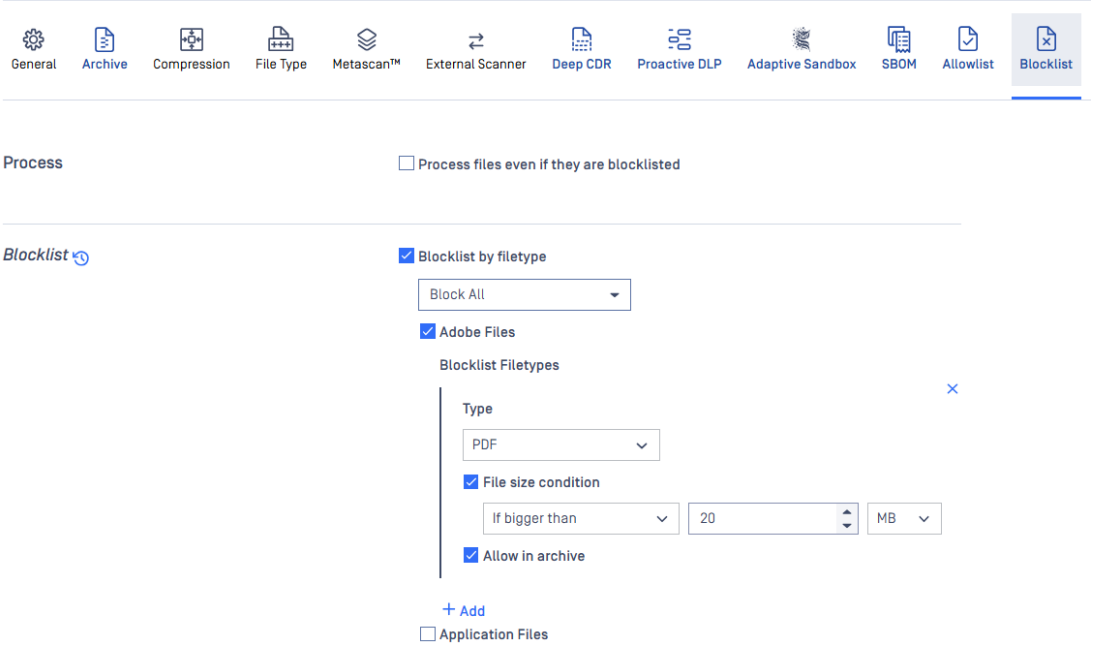
The max recursion level defines how deep extraction should go into the archive, the number of maximum extracted files also can be set as well as the overall maximum size of these files.

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.

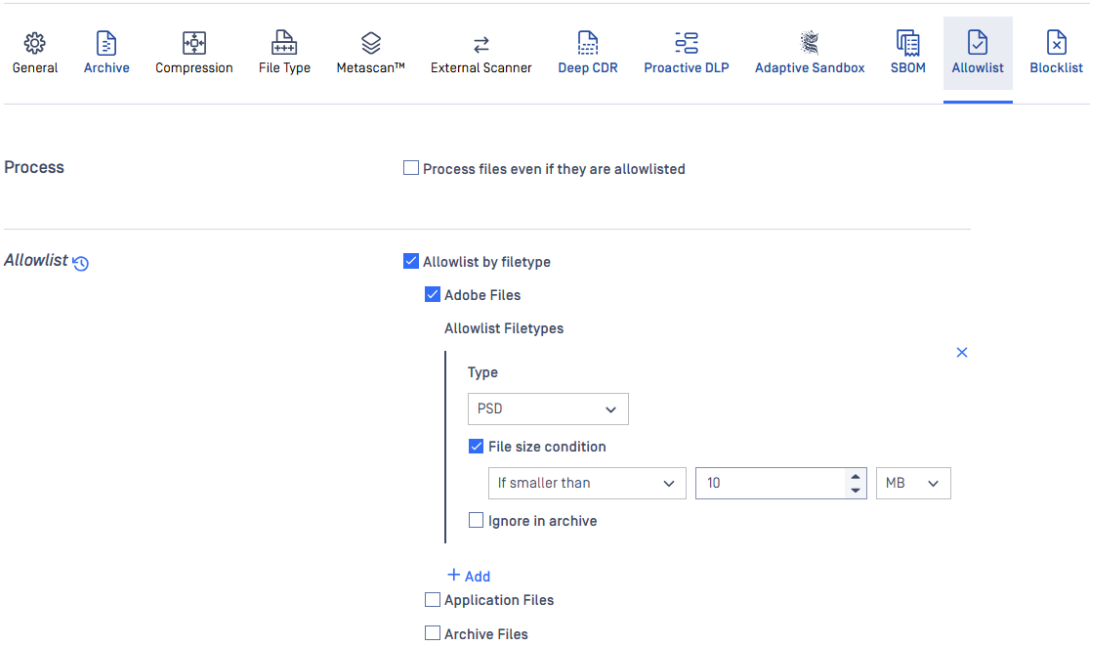


During scan, it is possible to create blocklists where files depending on their file type, checksum (hash), filename, extension, file size or MIME-TYPE. All of these can be stored in the fields on the Blocklist tab. Administrators can select tactic "Block All" or "Block All Except". Also it is available 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.

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".

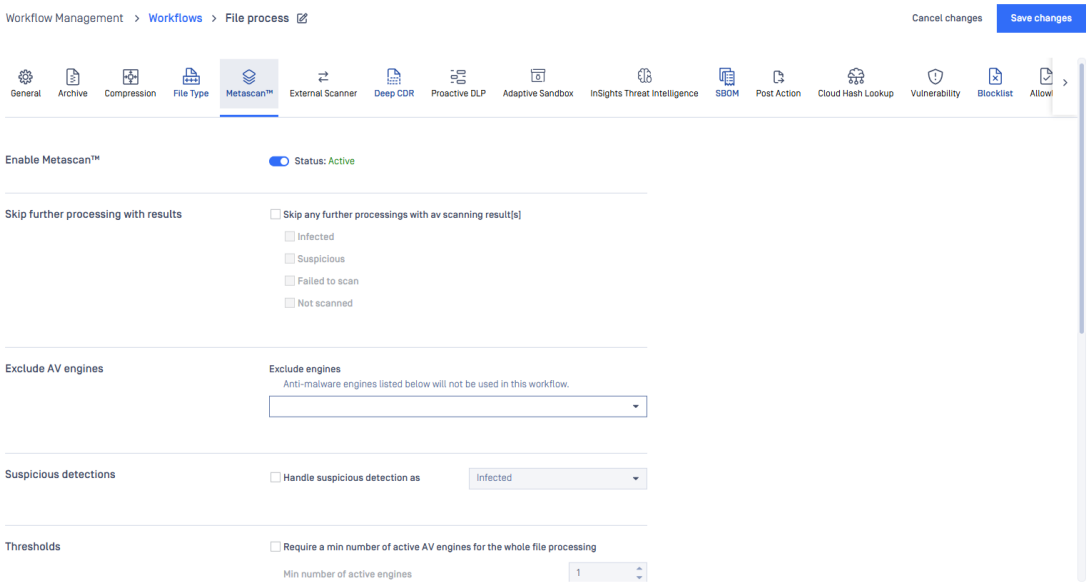
THREAT DETECTED THRESHOLD: When checked, this setting supports two configuration options INFECTED LIMIT and the SUSPICIOUS LIMIT when SUSPICIOUS DETECTED HANDLED AS is disabled, and its handling logic will be described as following:

- 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.

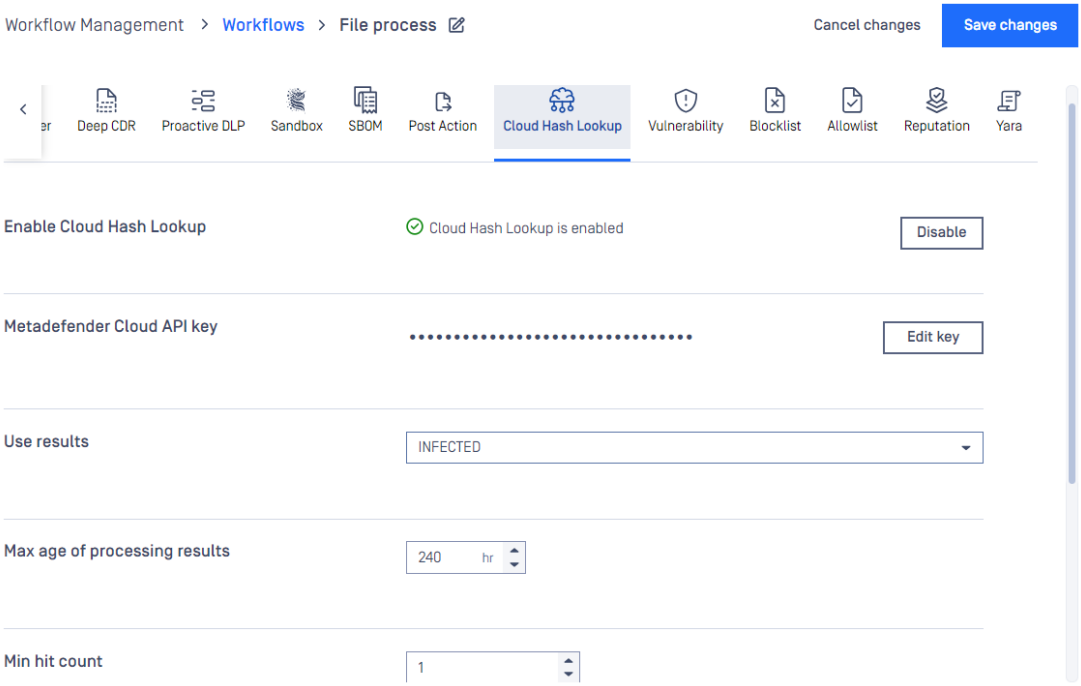


Cloud Hash Lookup

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**".

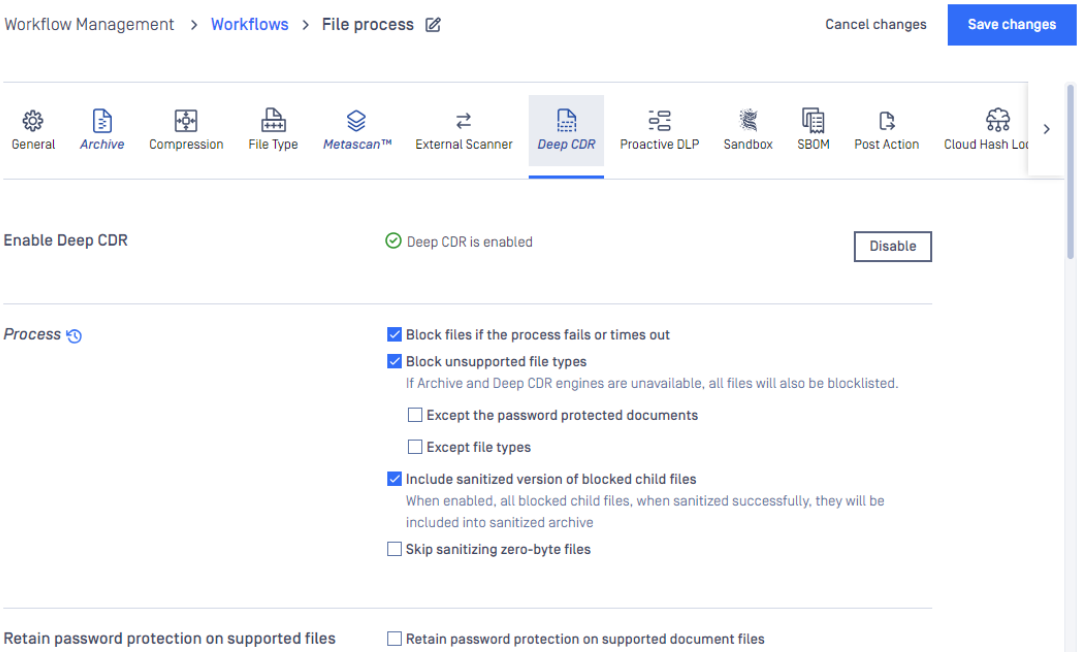
The maximum allowed time for data sanitization to be made can be configured through the "**DEEP CDR TIMEOUT**" and "**TRY COUNT**" options, where first one means that data sanitization should finish within the configured time frame, otherwise abort the conversion and latter means the number of times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

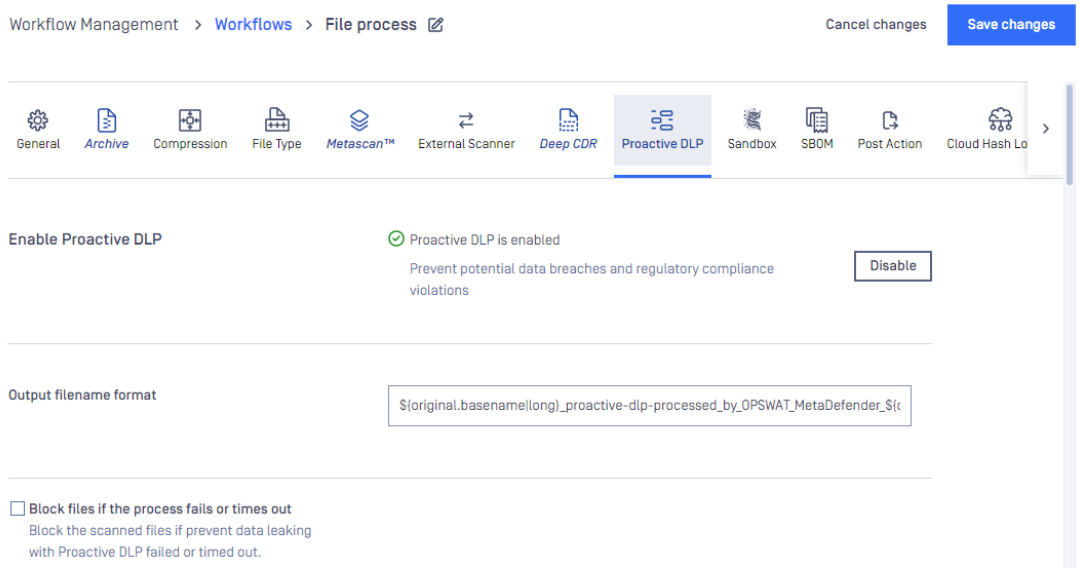
Result of sanitization can be either downloaded on the scan page or retrieved the data ID via REST. See Fetch processing result. Note that /hash API does not provide such information.

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



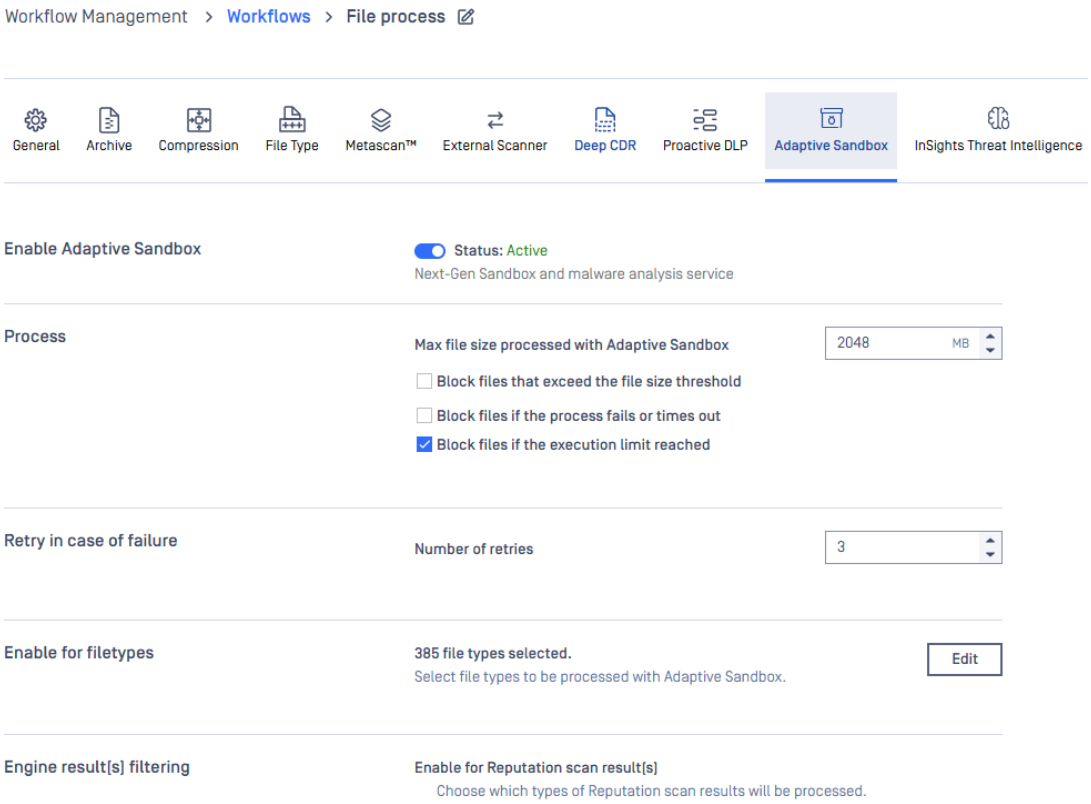
Proactive DLP

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



Adaptive Sandbox

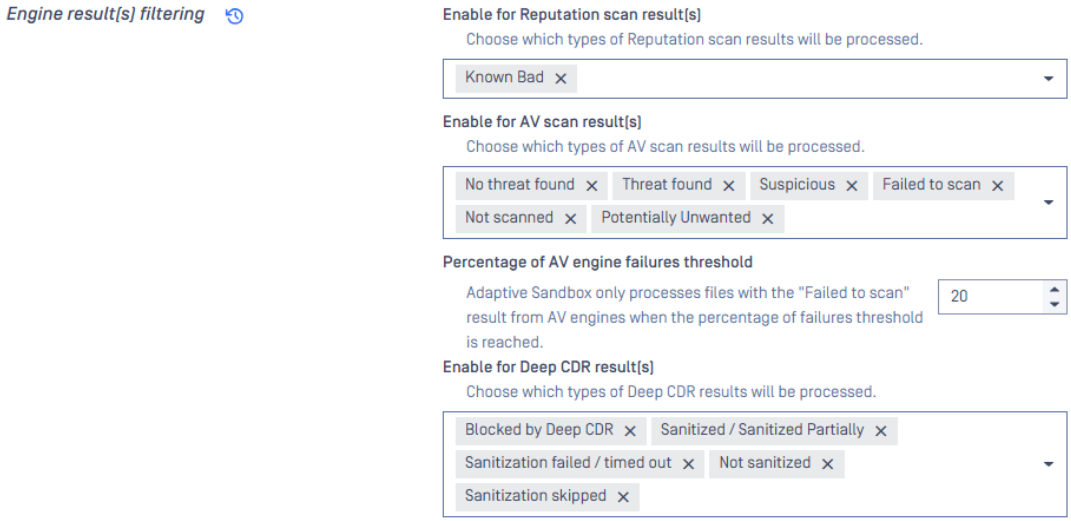
For all information about features powered by Adaptive Sandbox, please learn more at Adaptive Sandbox overview.



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.

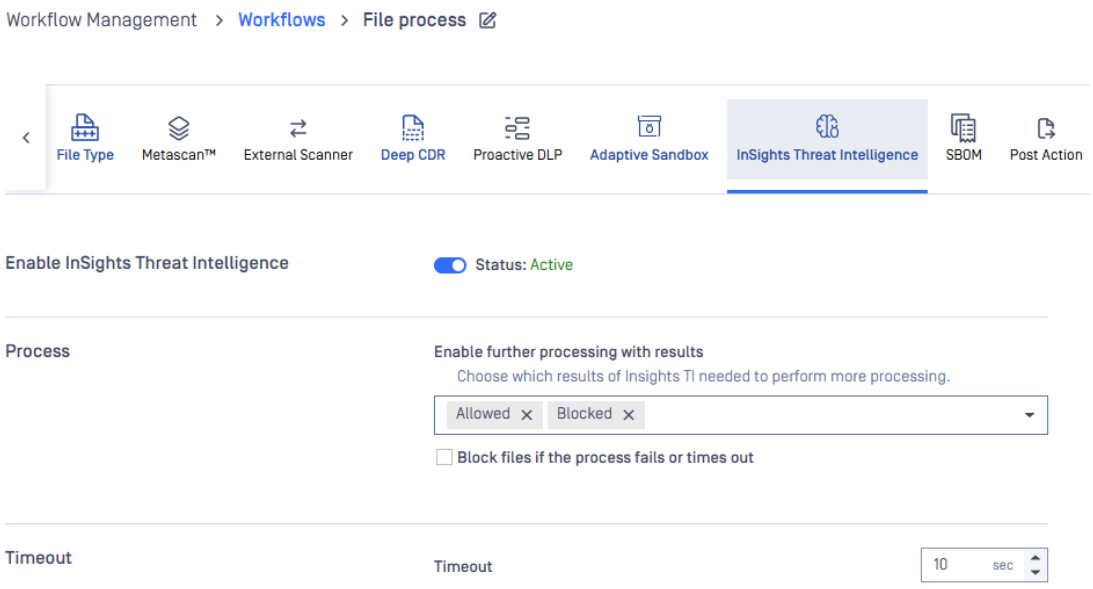
With the Result Filtering configuration, you can configure Adaptive Sandbox engine to process files based on

- 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.



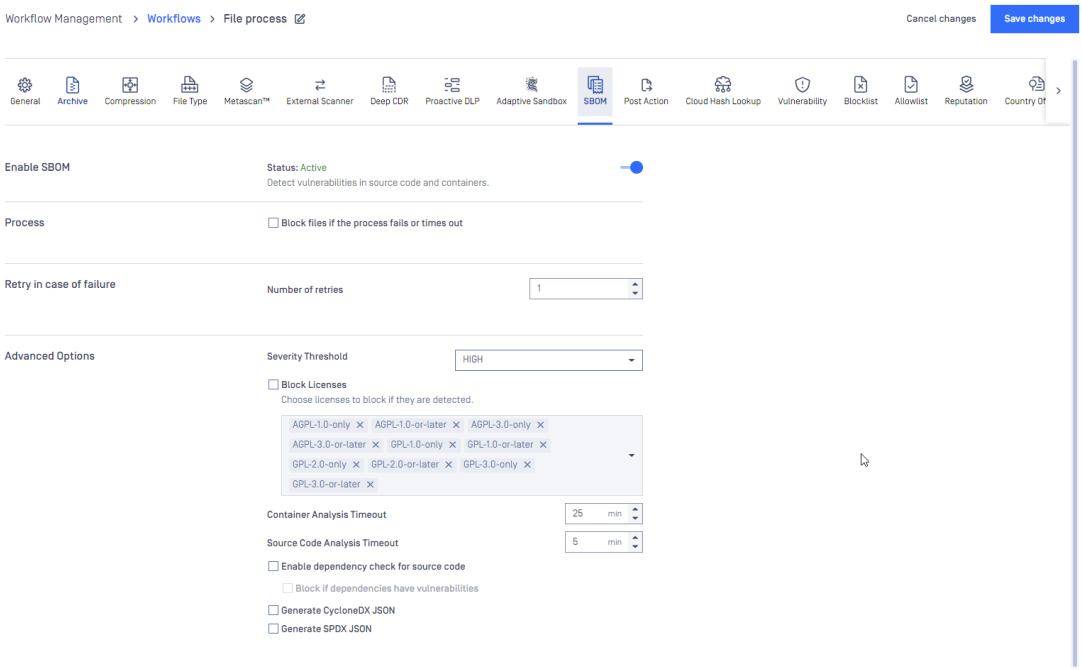
InSights Threat Intelligence

For all information about features powered by SBOM, please learn more at (Link Removed).

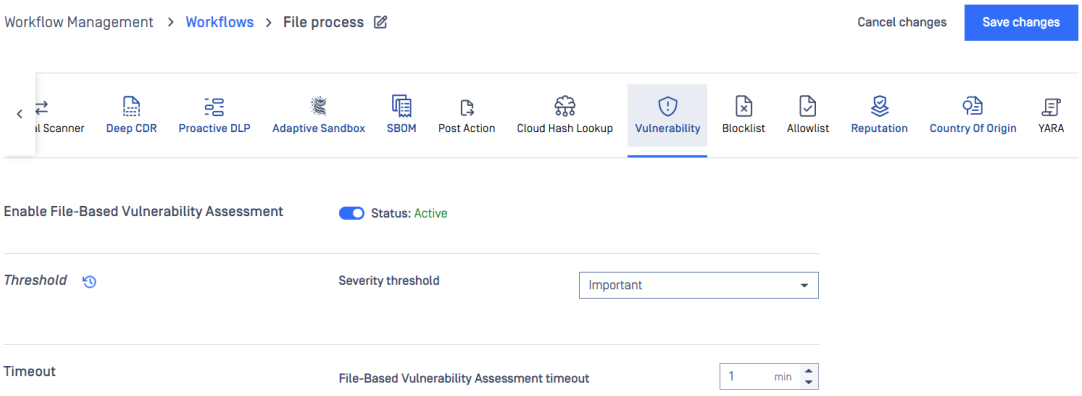


SBOM

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

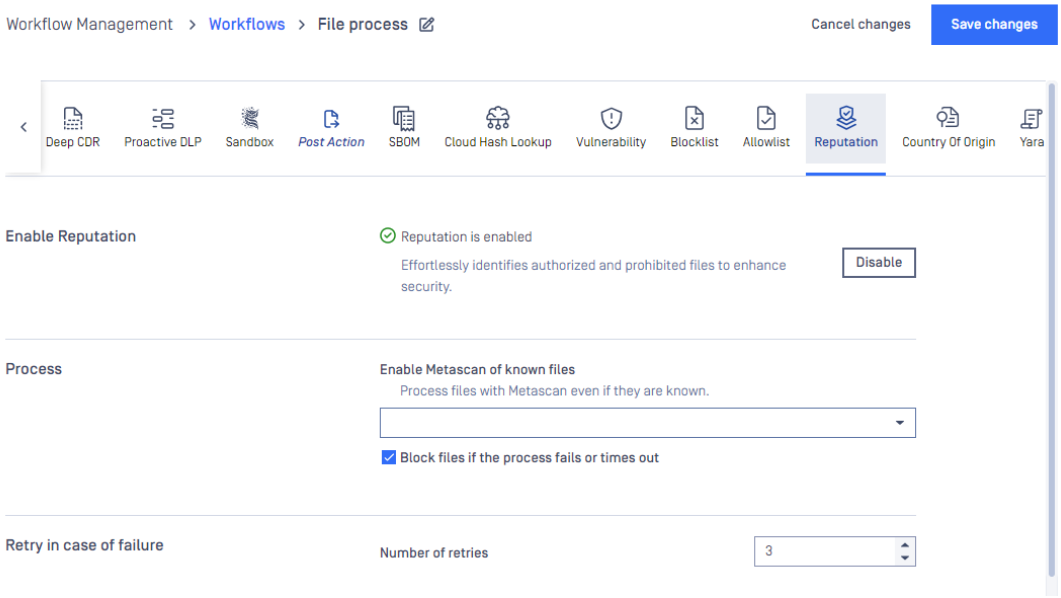


Vulnerability



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.

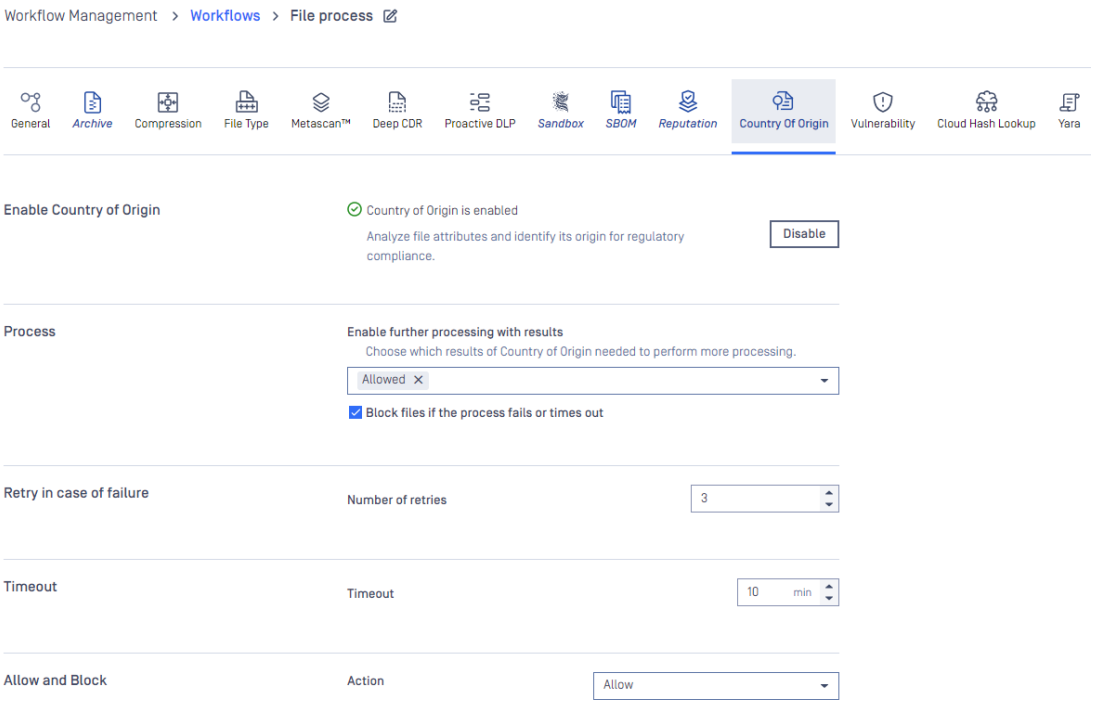
Reputation



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

Country of Origin

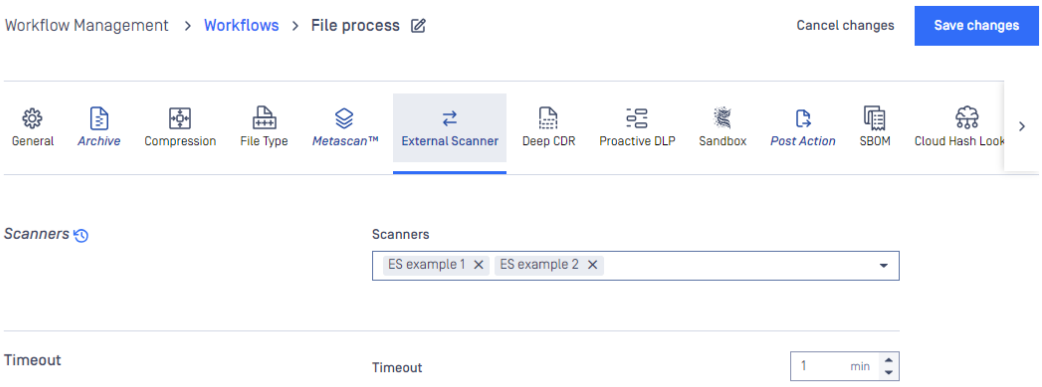
For all information about features powered by Country of Origin, please learn more at Country of Origin



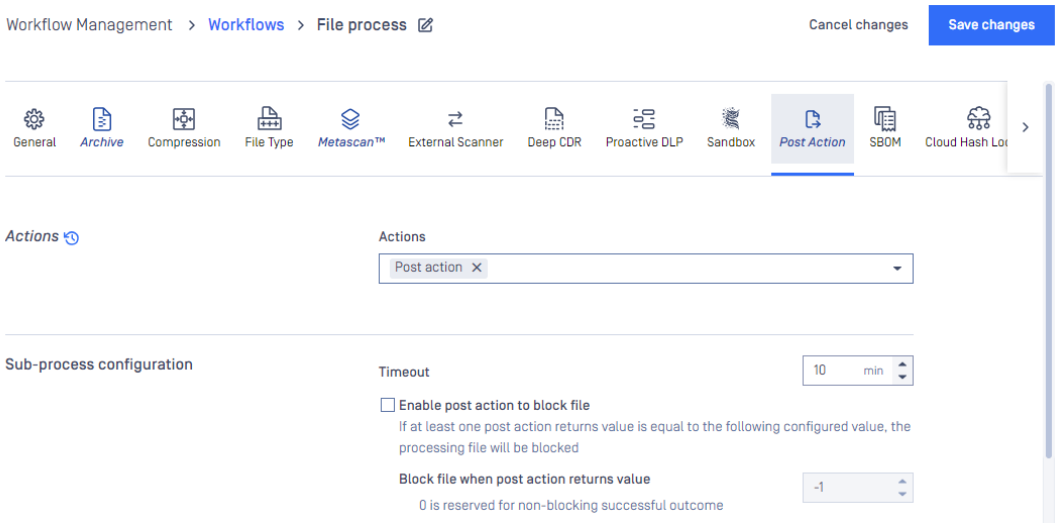
External Scanner and Post Action

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).

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" API, 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.

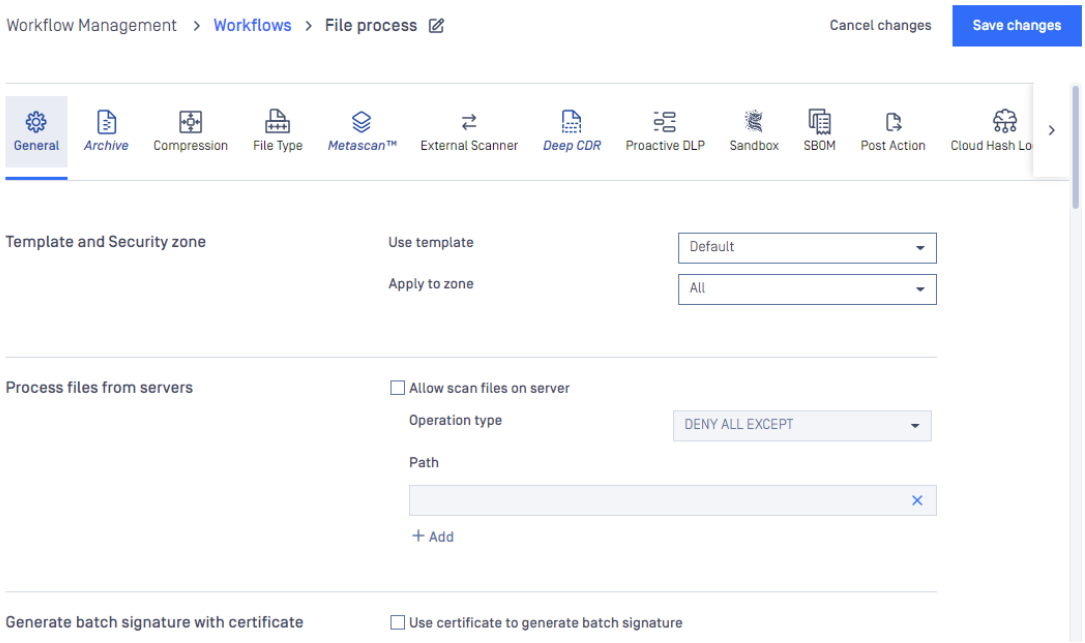
- This is only applicable to Asynchronous mode.
- When a scan request is timed out by this global processing timeout setting, the scan request will be blocked with "Global timeout exceeded" as blocked reason.
- **Note:** *This global processing timeout, when enabled, it should be greater or equal to 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

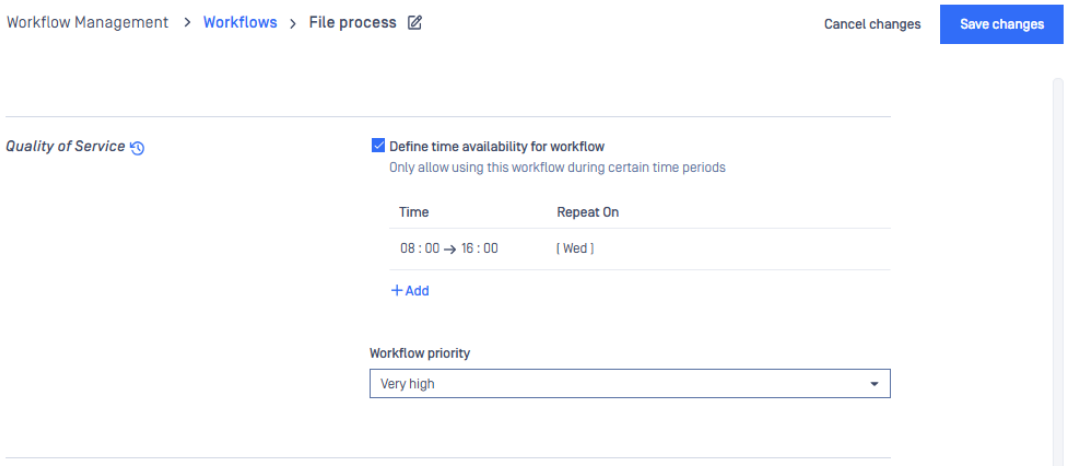


By enabling '**Define time availability for workflow**', it is possible for clients to define time frames in which a specific workflow is ready to use. MetaDefender Core only accepts requests of the workflow rule if they are within available time frames defined by the rule and responds HTTP code 403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

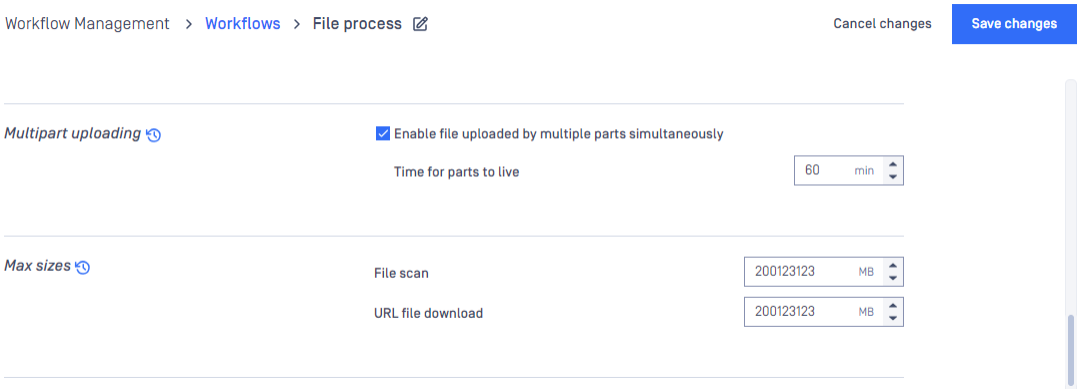
By selecting one of values of '**Workflow priority**', it is possible for clients to prioritize a task coming to engine.

- 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.

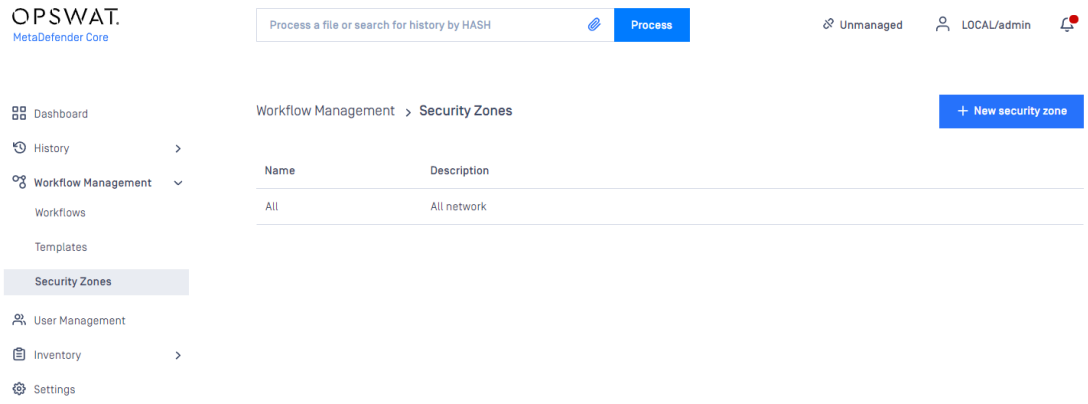


By enabling '**Multipart uploading**', it is possible for clients to upload files for scan by multiple parts simultaneously. By default, parts have at most 60 minutes to live until the file is combined successfully to scan.



Security zone configuration

The Security zone page is found under **Policy > Security zones** after successful login.



The following actions are available:

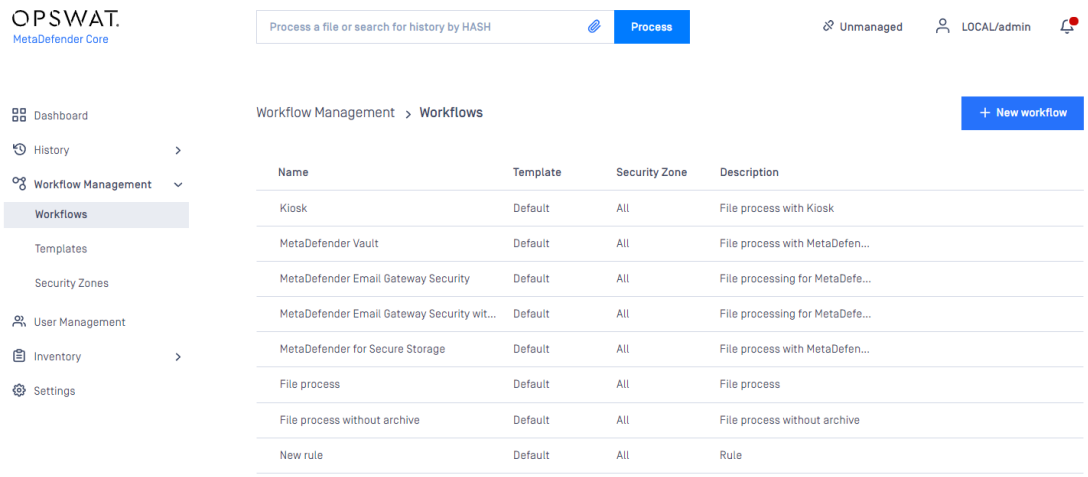
- new zones can be added
- existing zones can be viewed
- existing zones can be modified
- existing zones can be deleted

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 **Policy > Workflow rules** 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.

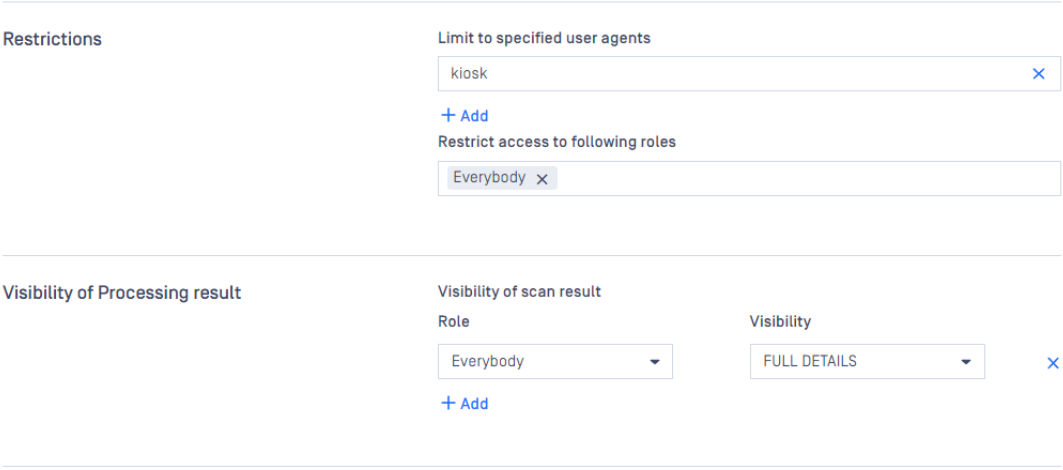
By enabling this feature, a local scan node is able to scan the files at their original place if the files' location is allowed in the list below the checkbox. For example if this list has *C:\data* in it, then all files under that folder (e.g.: *C:\data\not_scanned\JPG_213134.jpg*) are allowed to be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

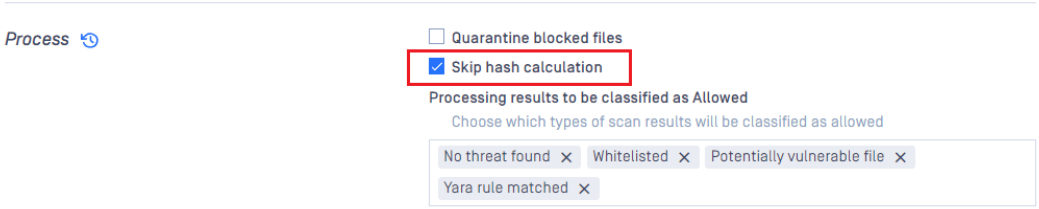
Rules are processed in order, the first matching rule will be used for the request. You can change order of rules via drag&drop in the Web Management Console. If there is no rule that matches for the client (source IP address), then the scan request will be denied.

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,

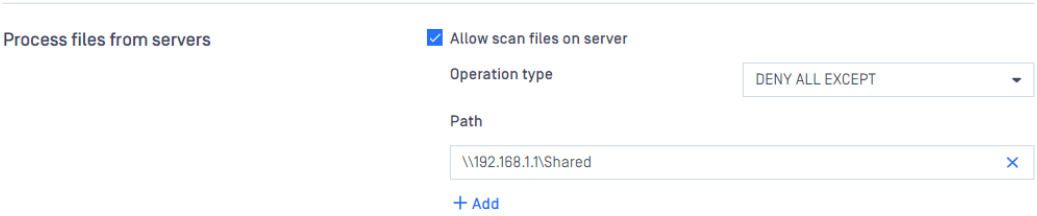
- it reveals empty values at md5, sha1, sha256.
- is_skip___hash key will be set to true should be used as a flag for client integration.

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

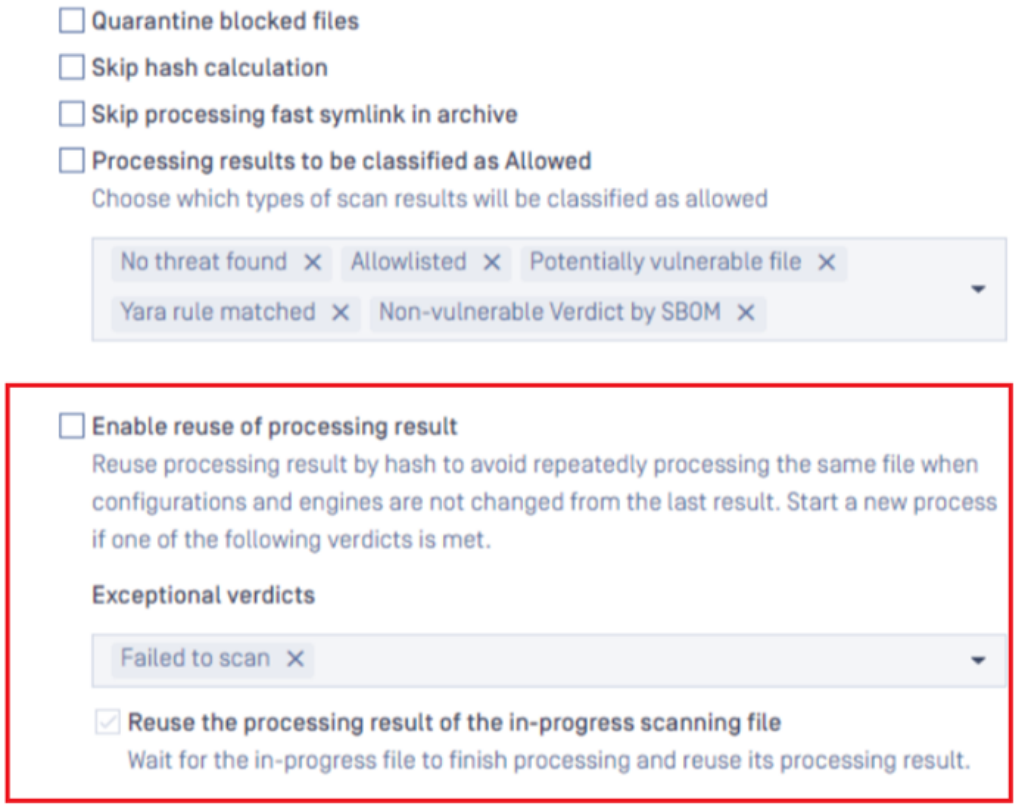


Info
Starting with MetaDefender Core 4.19.1, the path under above setting could accept UNC path (besides local folder path)

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

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.

> **Warning**
> Regarding to external scanner and post action, MetaDefender Core can not 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 in stead of keeping the old name for new scanner/script.

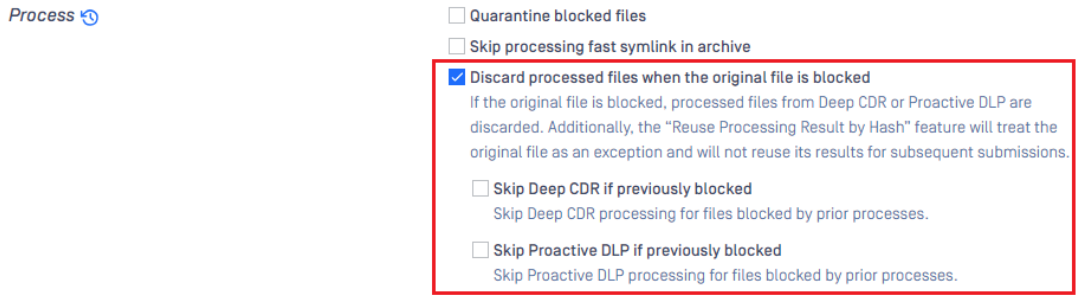
Besides that, MetaDefender Core customers have an option to let the product wait for other in-progress request (with the same hash), until it finishes and then automatically resuse to apply the final result on those waiting requests.

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

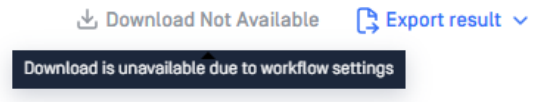
- The file completing its processing will wake all other requests waiting for its result. If the current request reuses the processing result successfully, it will wake all others and so on.
- Reaching the wait timeout (5 minutes, cannot adjust for now), do rescan the file.

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.



When processed files are discarded, the top-right corner of the scan result detail screen on the MD Core UI will display as shown below:



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.

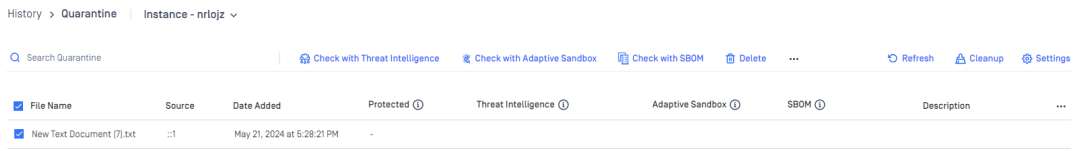
In cases where a processed file is discarded, the scan results of the original file will not be available for 'Reuse scan result', as mentioned earlier.

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.
5. Send to MetaDefender Cloud for Threat Intelligence. For details see the next section.

The *Check with MetaDefender Cloud*, *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 Threat Intelligence

Files in the quarantine can be uploaded to MetaDefender Cloud to get threat intelligence on them.

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

Quarantine items may be sent to MetaDefender Cloud:

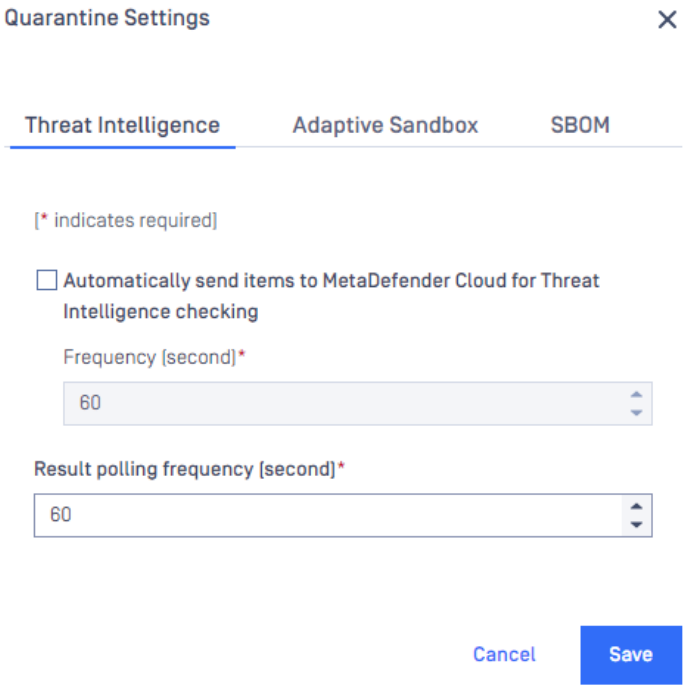
1. Manually using the Send to MetaDefender Cloud, or
2. Automatically, driven by the configuration under Quarantine settings

Quarantine settings

To edit quarantine settings, click SETTINGS in **History > Quarantine**. The following options are available:

1. AUTOMATICALLY SEND ITEMS TO METADEFENDER CLOUD: If enabled, all new quarantine items will be uploaded to MetaDefender Cloud for threat intelligence information.
    a. CHECK QUARANTINE FOR NEW ITEMS TO SEND: The frequency (in seconds) to check for new quarantine items to upload to MetaDefender Cloud.

2. RESULT POLLING: Once a quarantine item is uploaded to MetaDefender Cloud, MetaDefender must poll the Cloud for results. The polling frequency (in seconds) can be set here.



Threat intelligence details

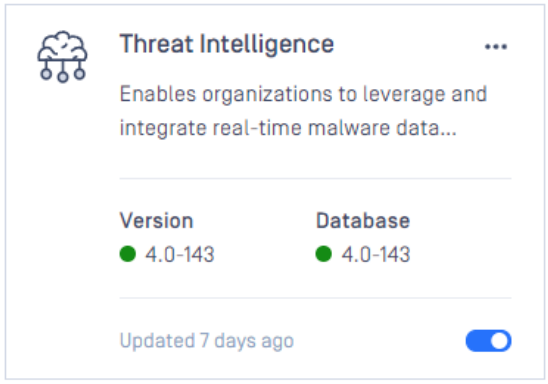
Clicking the *Show details* function, the *Quarantine item details* view is shown. Clicking the THREAT INTELLIGENCE RESULTS tab, further details from MetaDefender Cloud are shown:

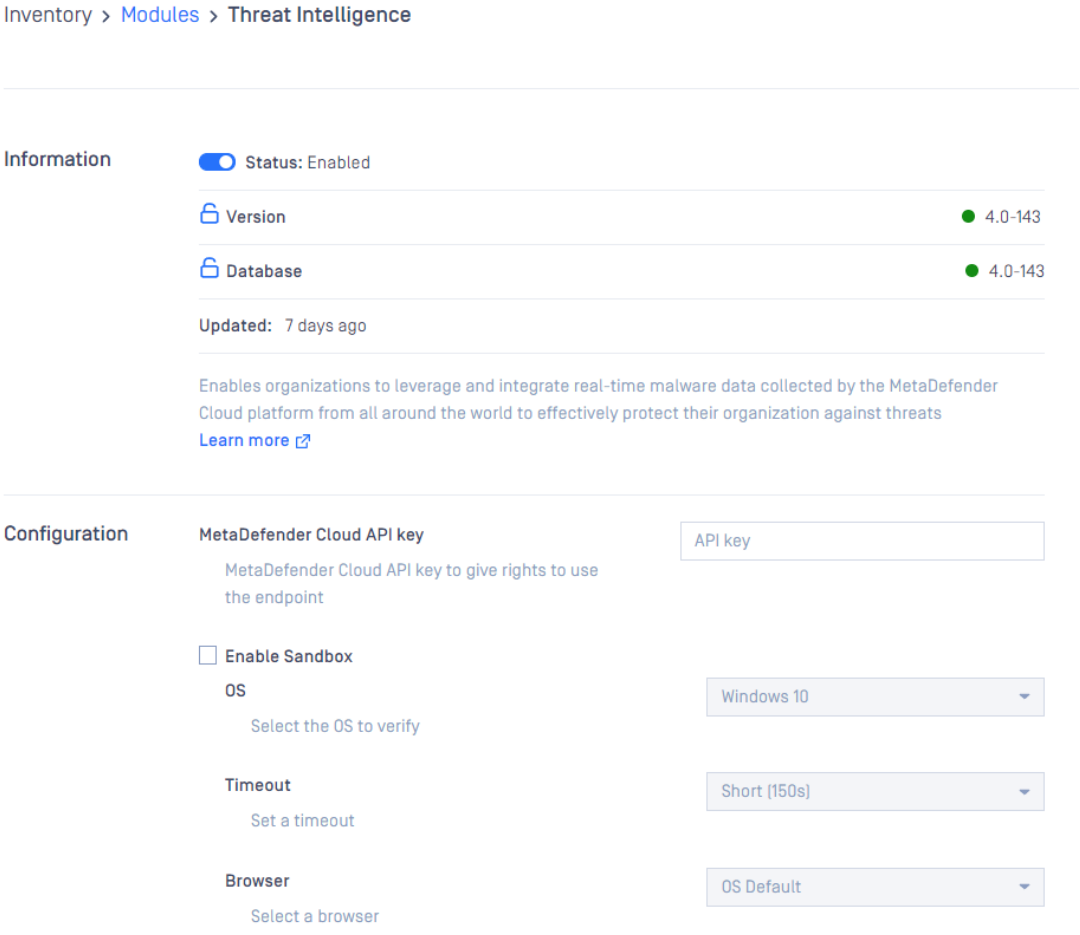
- RESULT: Processing summary if the entry was blocked or allowed.
- VERDICT: A more verbose details about the processing results.
- AV ENGINES: Number of anti-virus engines that were used for scanning this item.
- TOTAL TIME: Total processing time of this item for this scan.
- RESULT LINK: Link to the processing results on MetaDefender Cloud.

If this quarantine item was uploaded to the Cloud multiple times, then there will be multiple THREAT INTELLIGENCE RESULTS pages in the tab.

Enabling MetaDefender Cloud integration

MetaDefender Cloud integration requires the *Threat Intelligence* technology to be licensed, and enabled under **Inventory > Modules**:





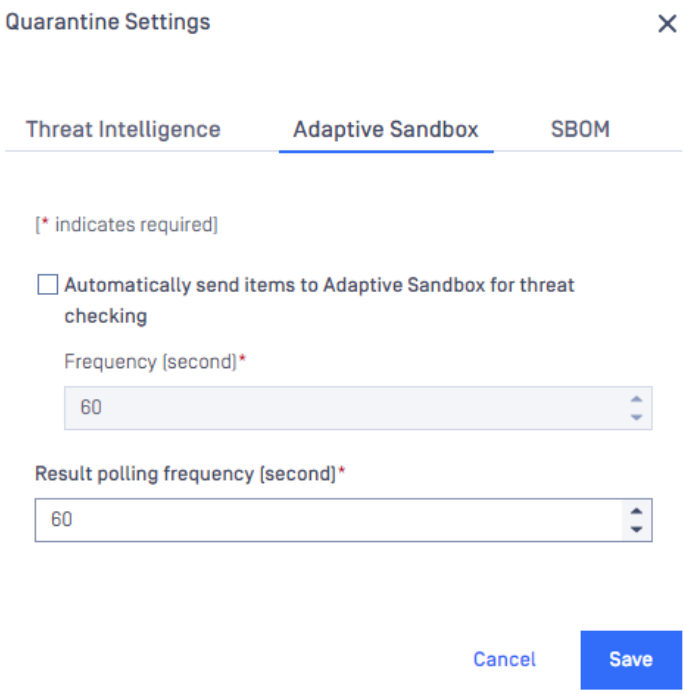
Check with Adaptive Sandbox

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



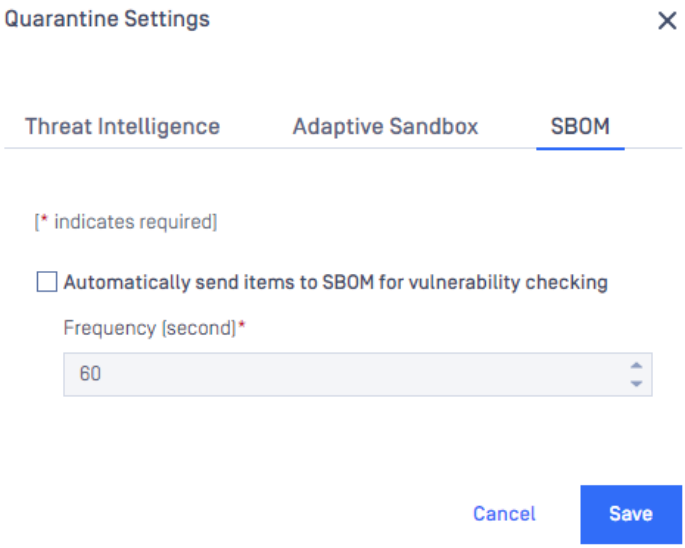
Check with SBOM

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


Quarantine Settings
Threat Intelligence
Adaptive Sandbox
SBOM
[* indicates required]
Automatically send items to SBOM for vulnerability checking
Frequency (second)*
60
Cancel
Save

##

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 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
	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	

Environment	Use Case	CPU Cores	Memory	Free Disk Space*	Additional Environment tuning
CDR and DLP	Deep CDR and Proactive DLP	+4	+8 GB		There are additional vCPU and memory requirements for different configurations. 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:**
    a. Identify key performance indicators (KPIs) with metrics such as throughput, total processing time, and resource utilization thresholds.
    b. 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:**
    a. Analyze logs to identify culprit(s) causing performance bottlenecks, while monitoring resource utilization (CPU, memory, disk, network) to identify resource constraints.
    b. 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.

Setting Name and how to set	Description	Default value	Recommended settings and when to change
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.
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>>

Setting Name and how to set	Description	Default value	Recommended settings and when to change
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 (configurations and engines are not changed from the last result)	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
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.

Setting Name and how to set	Description	Default value	Recommended settings and when to change
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. We recommend lowering the parallel count for the sandbox engine when MetaDefender Core uses remote sandbox.	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
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) file storage Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global\sanitizepath` Linux: File `/etc/ometascan/ometascan.conf` => `sanitizepath`	Similar to temporary file 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.	Inside installation folder	Recommended to use 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.

Setting Name and how to set	Description	Default value	Recommended settings and when to change
Max number of database connections Windows: Registry `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global\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 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.	• Less transparent backend infrastructure for client integrations. • Requires additional setup for centralized scan data repository. • Requires additional setup for configurations auto sync.	• Scanning performance is the top priority. • Client-side integration is flexible (able to setup sticky session or webhook callback 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:

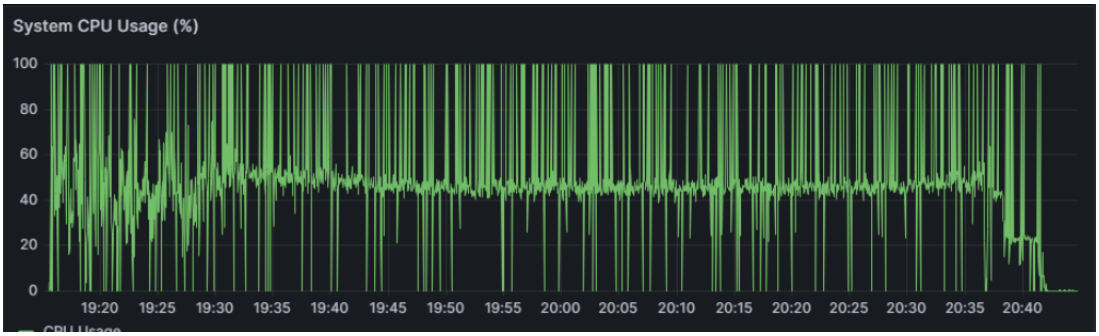
- **Performance goals:**
  - KPIs: the crucial metric to be measured here is the total processing time. CPU resource utilization should be also captured.
  - Performance target: total processing time is 1 hour or less, with an acceptable margin of error of 10% (i.e. 1 hour 6 minutes), and average CPU 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

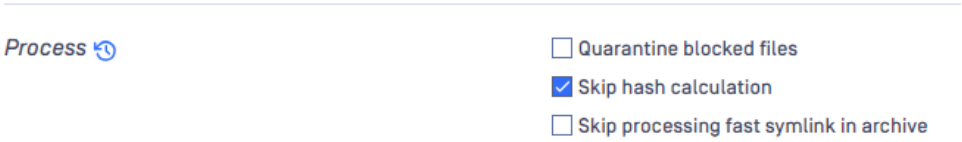
The next step is identifying bottlenecks. Equipped with our baseline data, there are a few key log items we want to analyze logs further in this step:

- 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

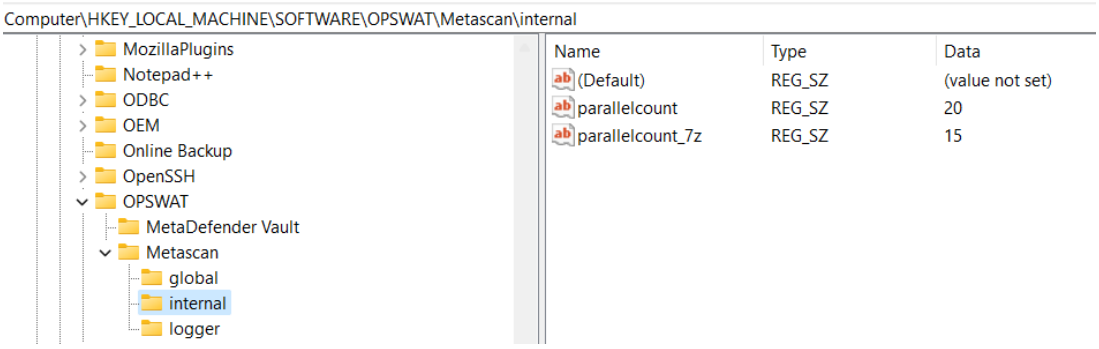
related to engine timeouts, and consequently adding significant overhead on overall performance time.

Based on this analysis, we may implement the following remediation plan:

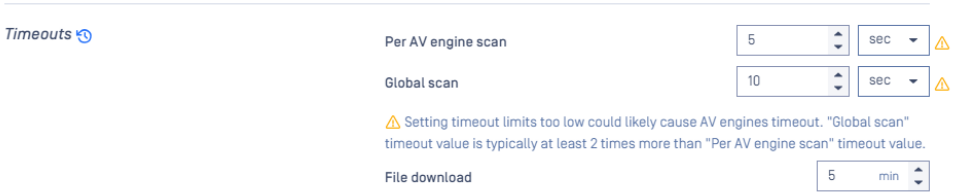
- Queue size: increased to 10,000 to ensure enough slots in the queue are allocated for archive extraction.
- Hash calculation: enabled “Skip hash calculation.”



- 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.

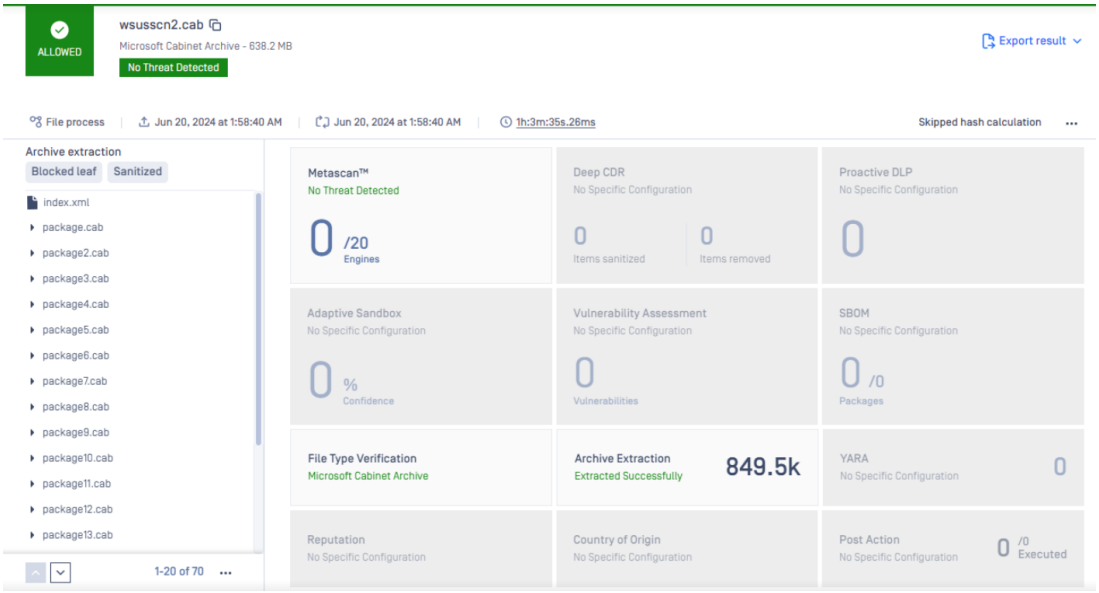


Design test case and perform load testing

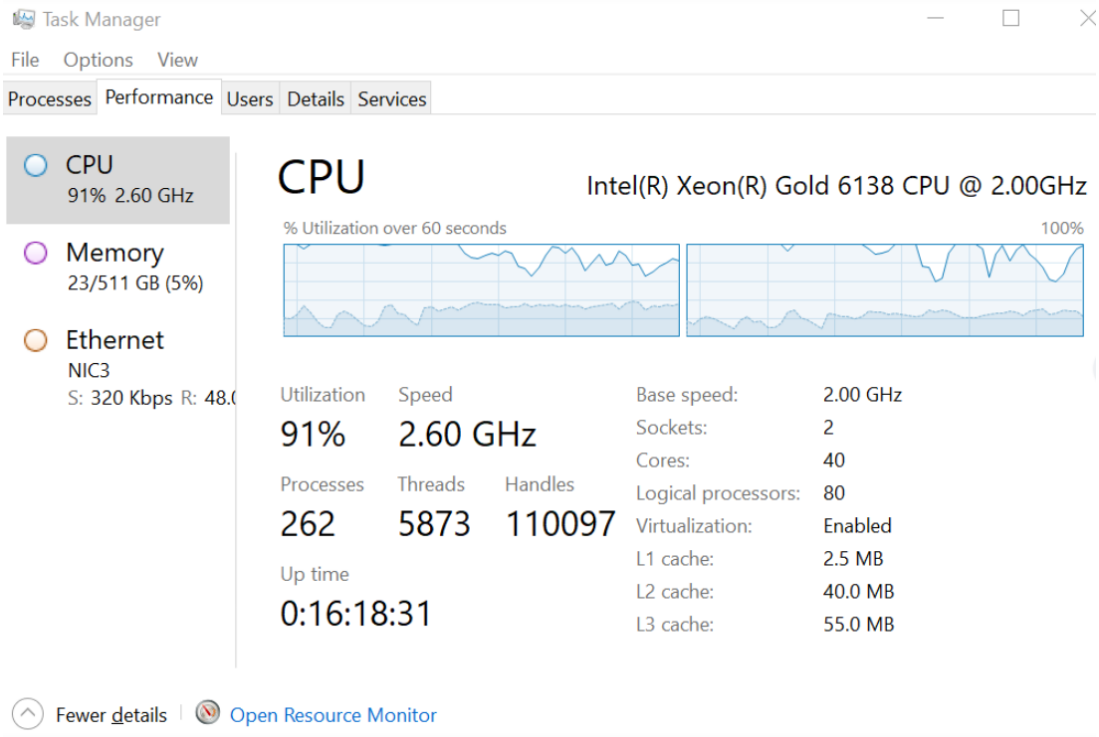
The same test case should be used, now with fine-tuned settings value as mentioned in previous step.

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 **Settings > 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	User directory
management	*N/A*	Metadefender Central Management	management@localhost	Administrators	SYSTEM

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).

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
    a. In this case the highest available permission applies to each function. Example:

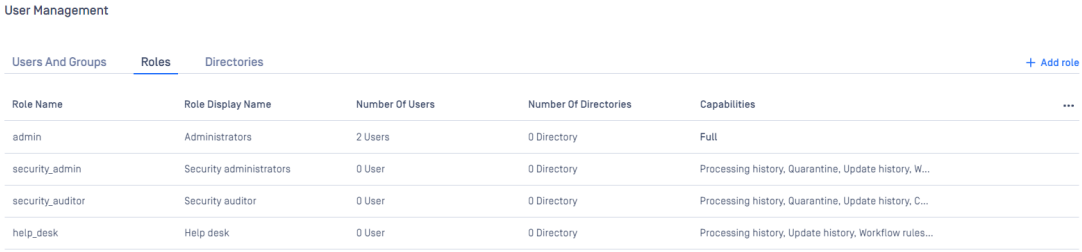
Roles assigned	Effective permissions	
	Full permission	Read only permission
security_admin	Scan history, Update history, Security rules, Security zones, Analysis workflows, Scan nodes, Engines, Update settings, Scan settings	
security_auditor		All except External settings
security_admin AND security_auditor	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:

Rolename	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 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:

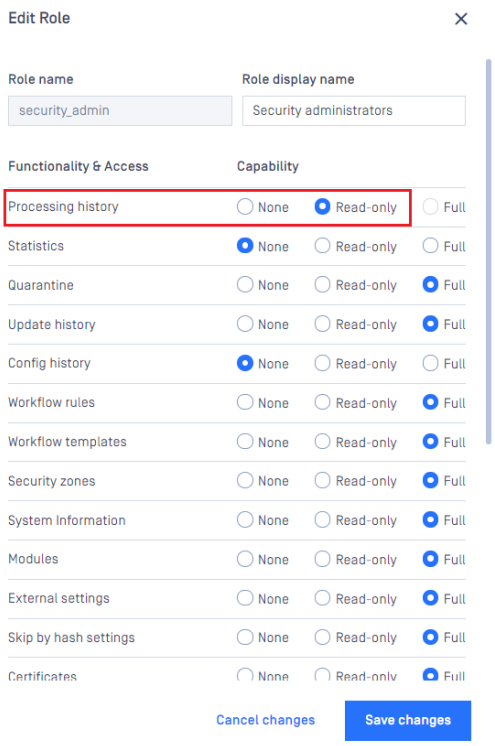
- GET /hash/<md5, sha1, sha256> (GET File Analysis API)
- GET /file/<data_id> (GET File Analysis API)
- GET /file/batch/<batch_id> (GET Status of Batch Analysis API)
- GET /stat/log/scan (Leveraged by Core management console)
- GET /stat/log/scan/export (Leveraged by Core management console)

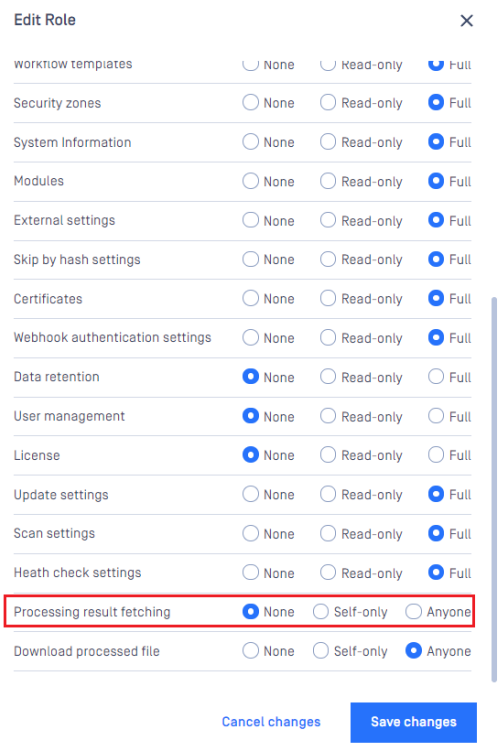
Download processed file:

- GET /file/converted/<data_id> (GET Download Sanitized Files API)
- GET /file/processed/<data_id> (Leveraged by Core management console)

A right can be set to one of three different states:

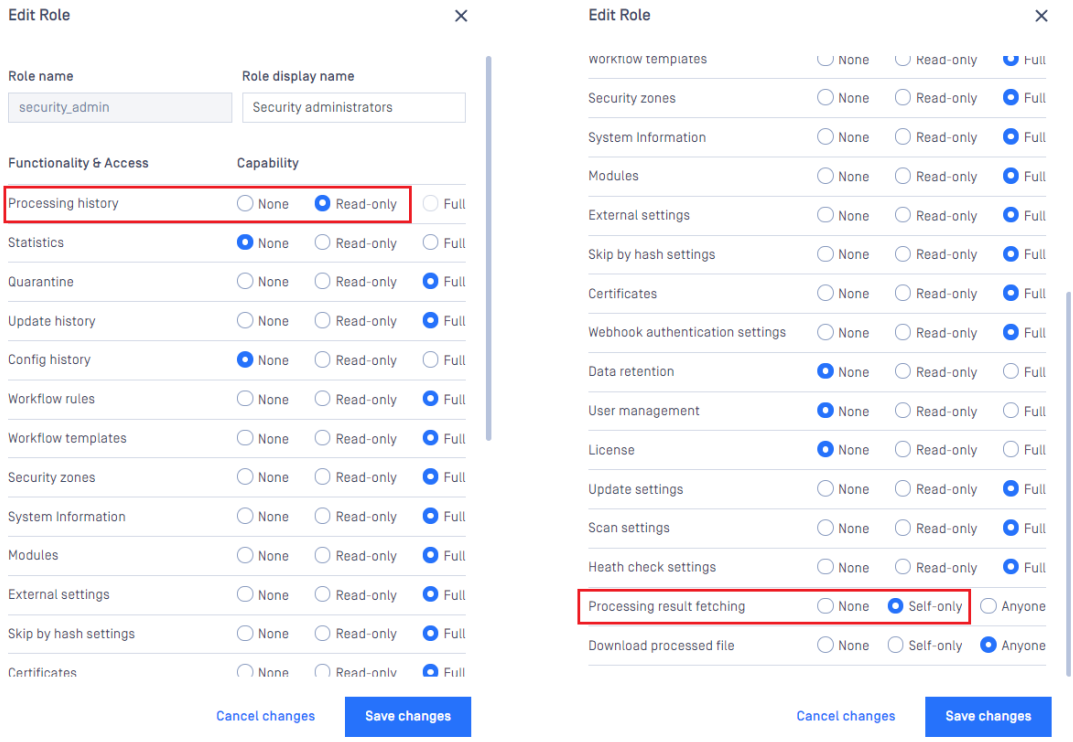
- **None:**
  - Users of this role have no right to access the given REST APIs (return "Access denied" error) and relevant functionalities on Metadefender Core Web Management Console
  - **Note:** When "NONE" is selected for "Processing result fetching", "Processing history" menu item under Dashboard 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** can not 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 can not be deleted as long as it is assigned to any user.

As a consequence deleting a role can not 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

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.

LDAP and Active Directory 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

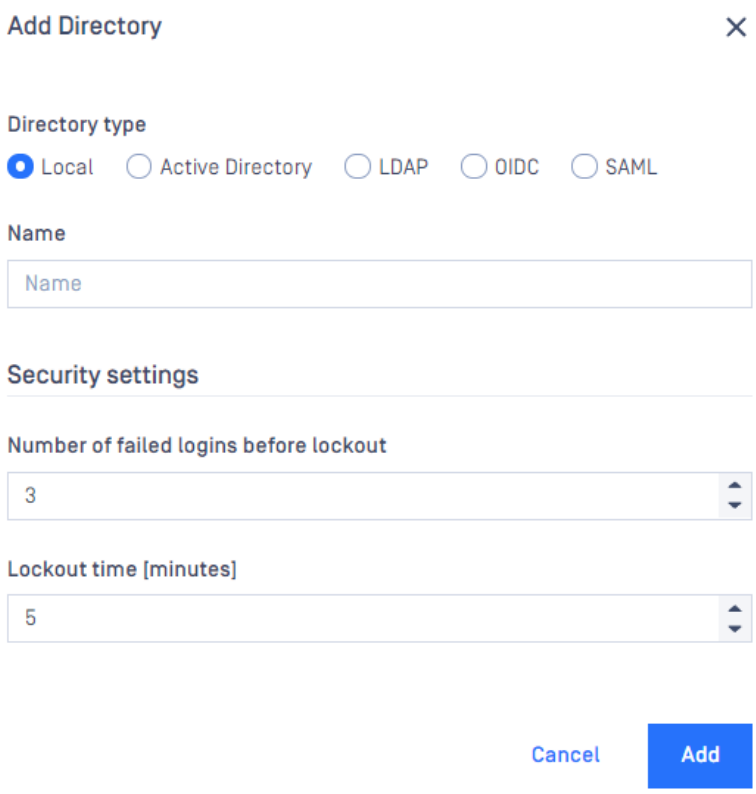
Besides listing existing user directories the **User directories** tab provides the following functions:

- 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 type user 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.

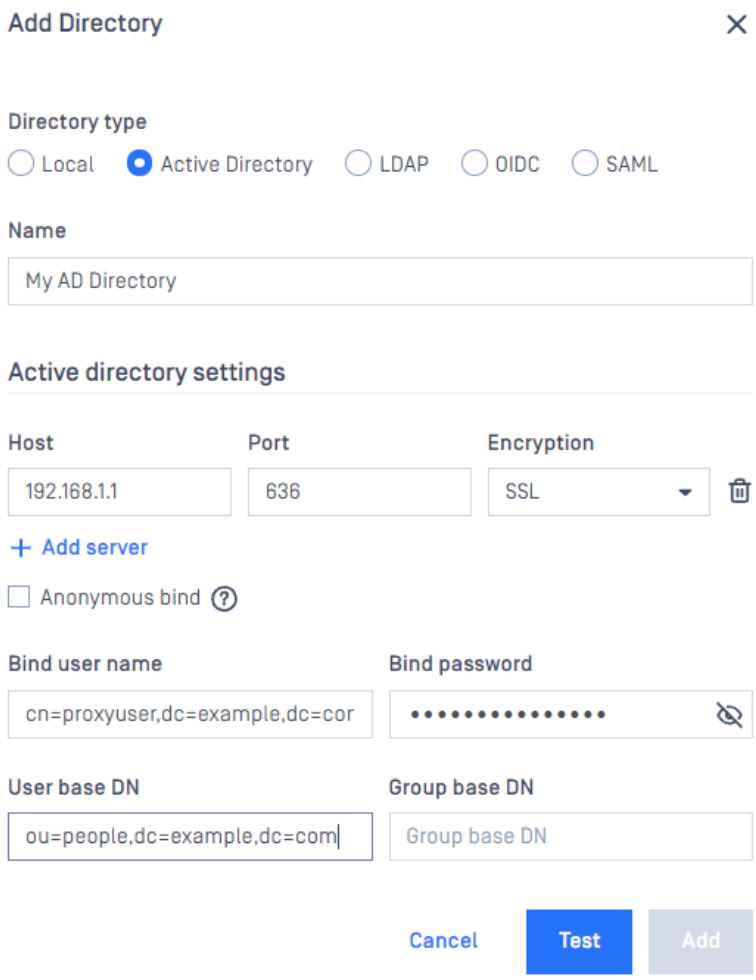
Warning
As long as TLS is not configured for the Web Management Console, passwords are sent clear-text over the network. To set up TLS see Enabling HTTPS.

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 attributes.

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.

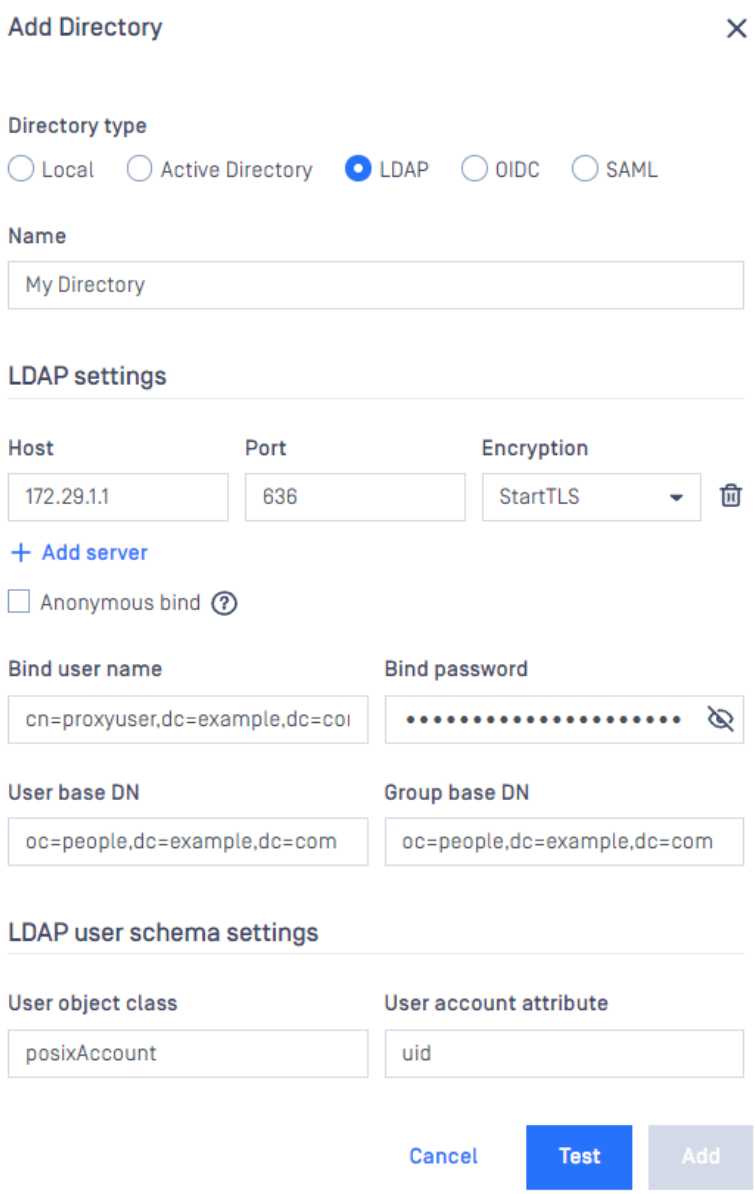


Add new Active Directory type user directory

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.



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

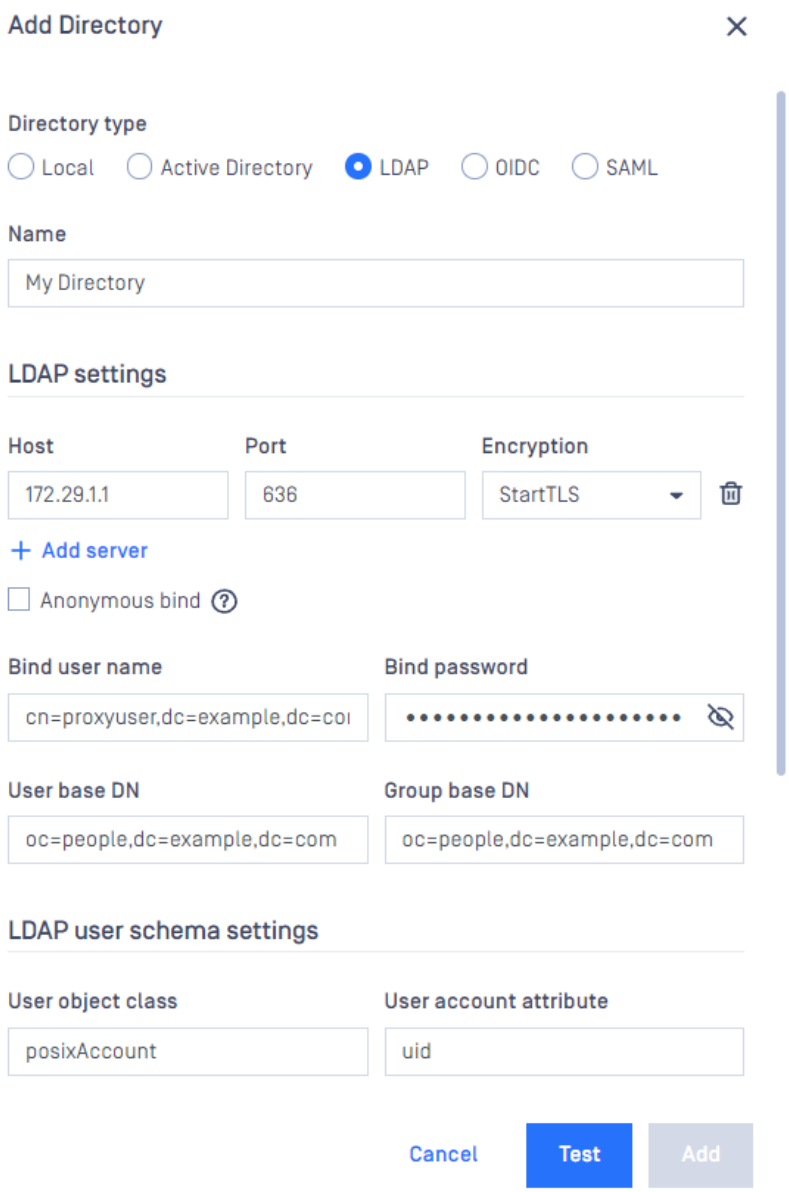
The currently logged on user can not disable the user directory to which her account is assigned to. For example the admin user can not disable the LOCAL user directory.

The currently logged on user can not delete the following:

- Her own user account. For example the admin user can not delete the admin user account.
- The user directory to which her account is assigned to. For example the admin user can not delete the LOCAL user directory.

Active Directory 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.



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:

none

```
> dsquery user domainroot -samid john.smith
```

or

none

```
> 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:

none

```
DC=example,DC=com
```

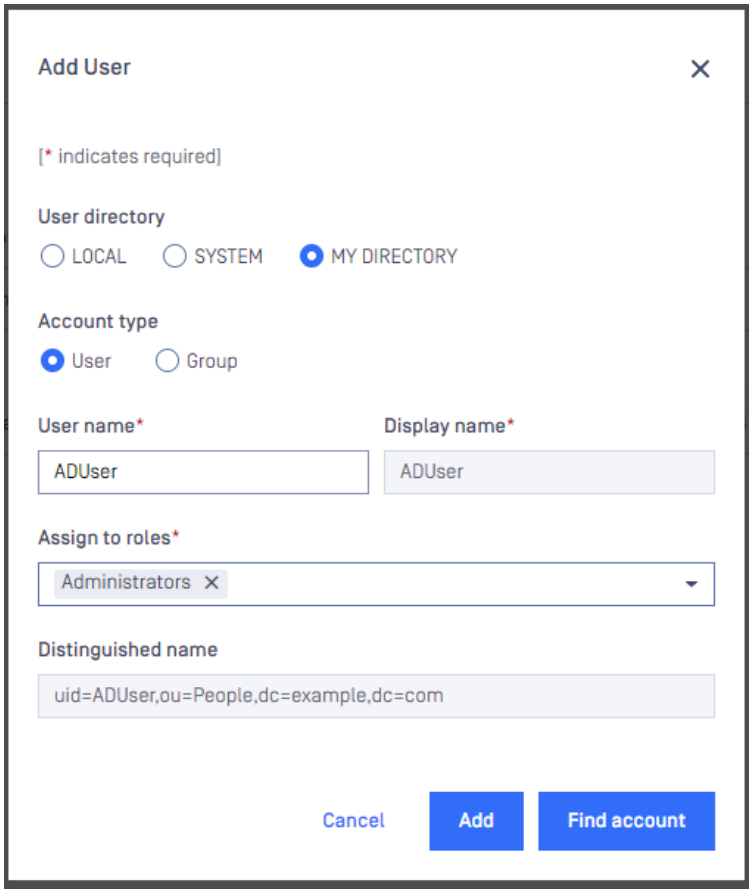
Note
Please note that using only DC components for the user/group DNs may result in searches to be executed from the top of the directory information tree and potentially slow down AD server responses a lot and thus have an impact on Metadefender Core password validation. The rule of thumb here is that 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
    a. the name of your AD directory - the name that user has configured on management console
    b. 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:

none

```
[DirectoryName]\[AD account]
```

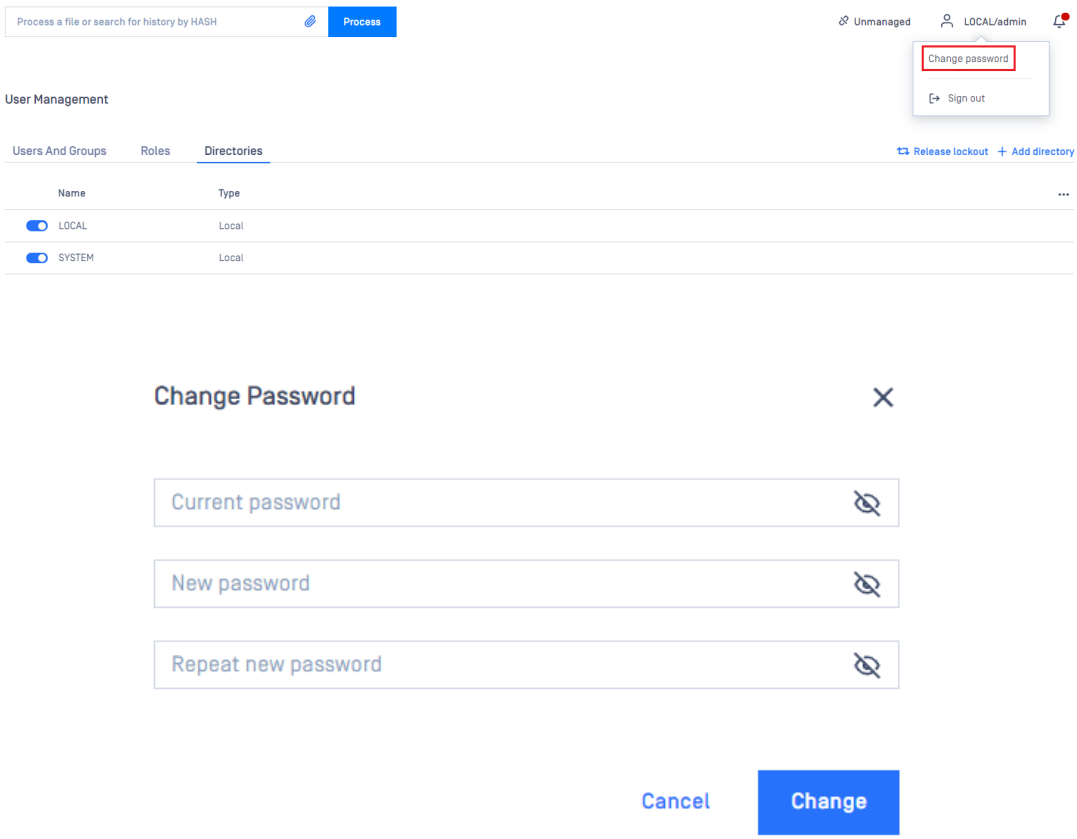
For example, if your AD directory is `My Directory` and your domain name is `mycompany` and your username is `ADUser`, you would log in as:

none

```
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.

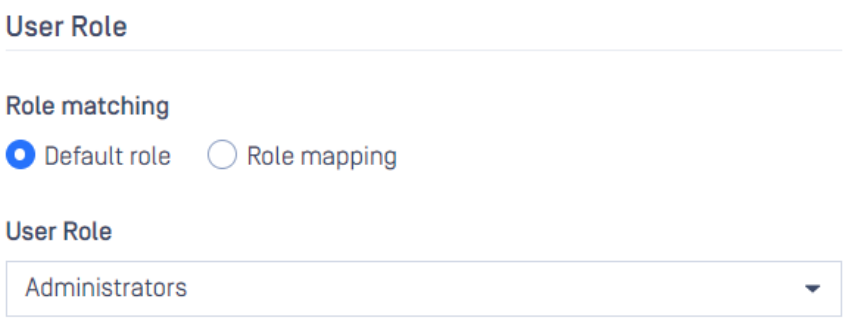
Learn more integration guidelines with the following specific IdPs.

IdP	OIDC	SAML
Amazon Cognito	Supported	
Auth0	Soon	Soon
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 role mapping

Motivation

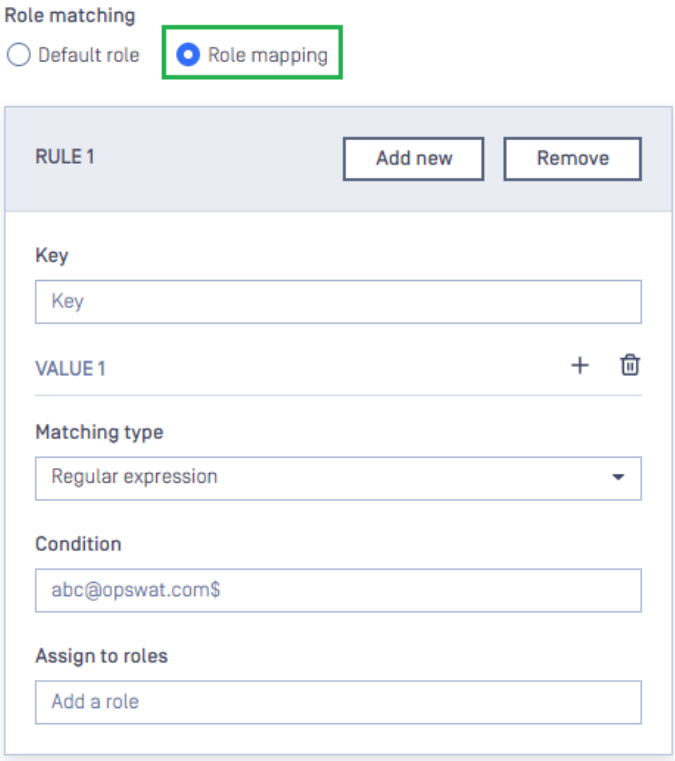
MetaDefender Core supports two types of role assignments for signed-in users. The first is `Default role`, which is the standard option and fits most simple use cases. If this option is selected, a fixed role is assigned to users when they successfully sign in to MetaDefender Core with their accounts. 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. If the condition is met, the associated roles specified by `Assign to roles` are assigned to the user. 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, until the first match is found or none match.

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. If multiple conditions match the value of the key, only the first in their defined order is selected.

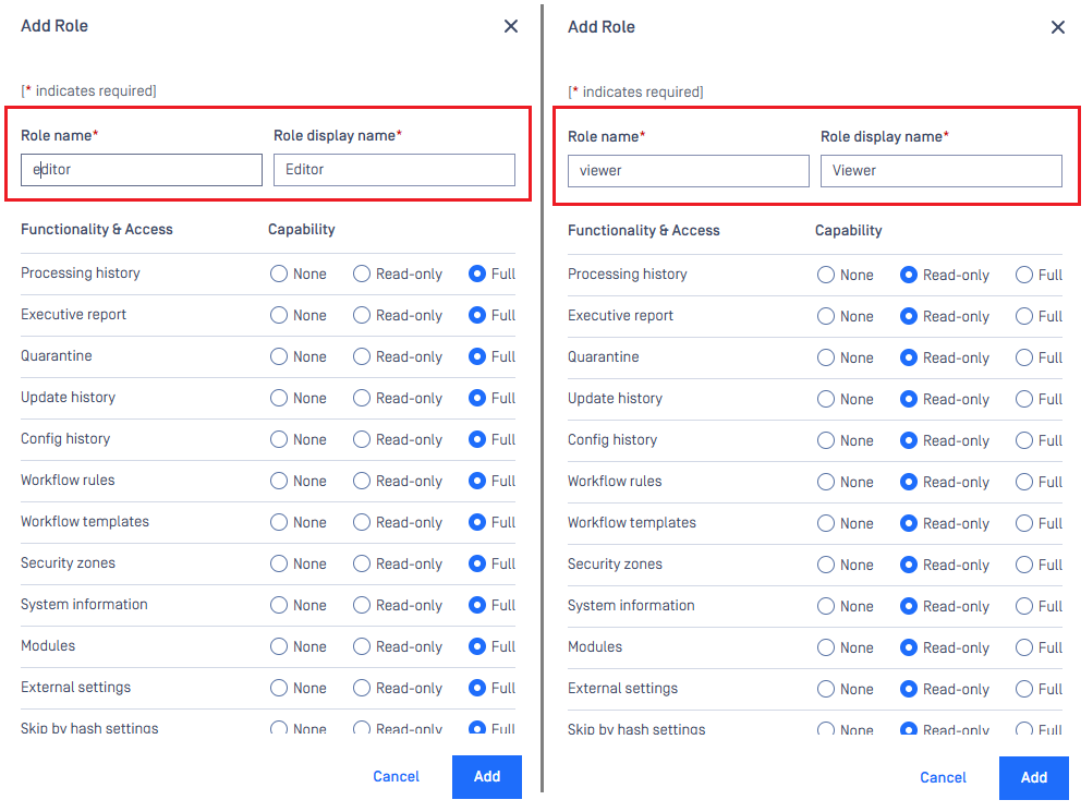
Building rules

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.

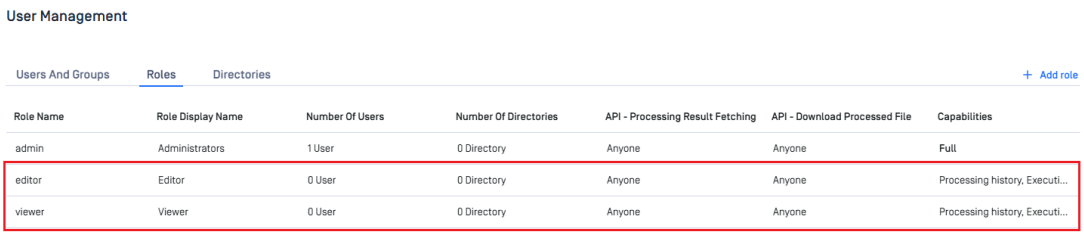
Info
You are advised to successfully set up SSO integration between your preferred IdP and MetaDefender Core using the default role option before attempting to apply role mapping.

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.

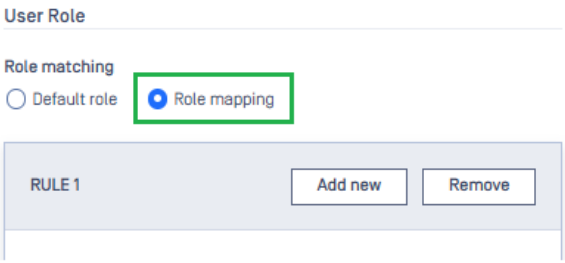
1. Sign in to MetaDefender Core.
2. From the dashboard, navigate to `User Management` > `Roles`.
3. Click `Add Role` to add Editor role and assign it appropriate rights.
4. Do the same for Viewer role.



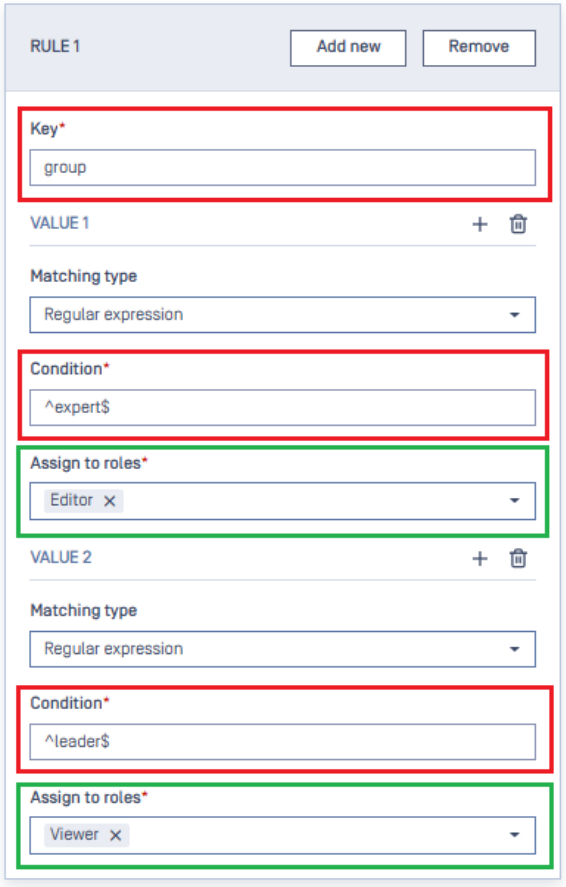
5. Click `Add` to complete the assignment of rights to the role.



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.



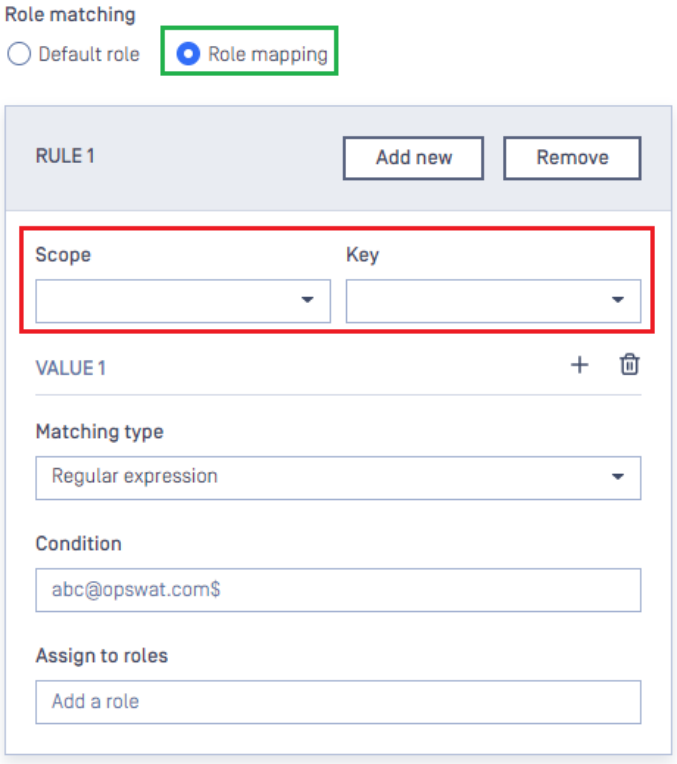
7. Fill in `Key` field with "group".
8. Click the plus button to define an additional `VALUE 2`.
9. `VALUE 1` will attempt to match the value of "group" key against the regular expression ^expert$, while the second will do so against ^leader$. If the first matches perfectly, the user will be assigned 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.



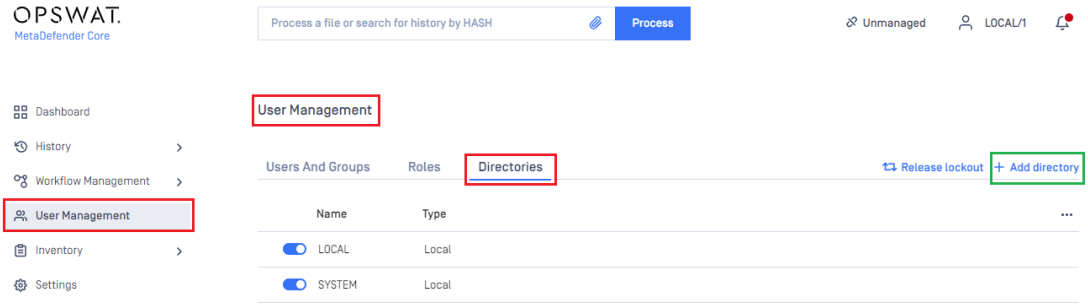
Amazon Cognito

The instructions guide users on how to configure Amazon Cognito OIDC Single Sign-on for MetaDefender Core.

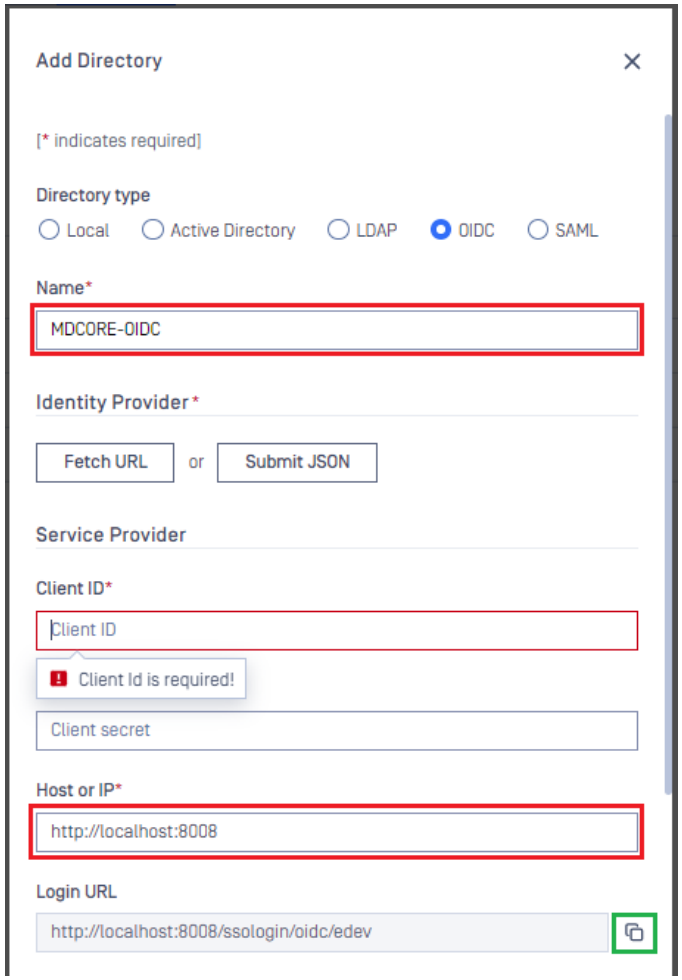
Configuring OIDC Single Sign-on

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.

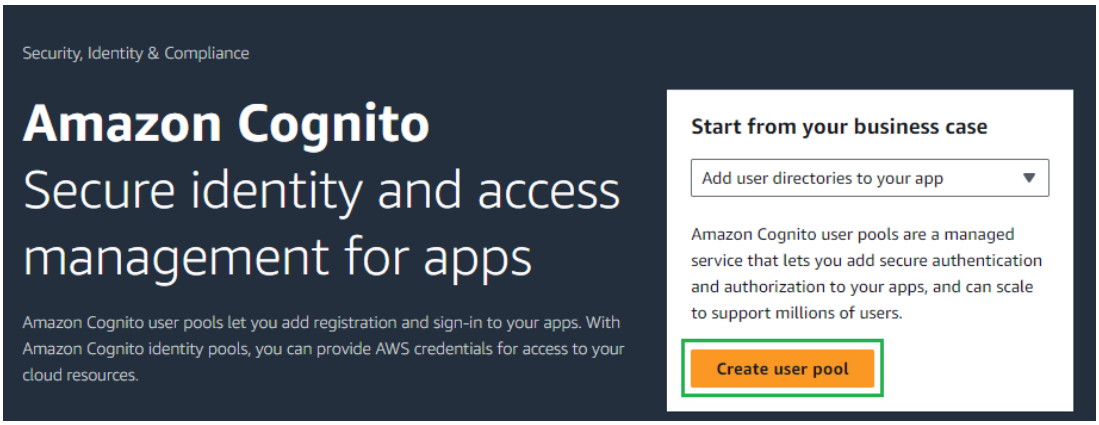


4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCore-OIDC.
5. Enter `Host or IP` where MetaDefender Core is hosted, for this example use https://127.0.0.1:8008.
6. Copy the string generated under `Login URL` and store it as *reply_uri*.

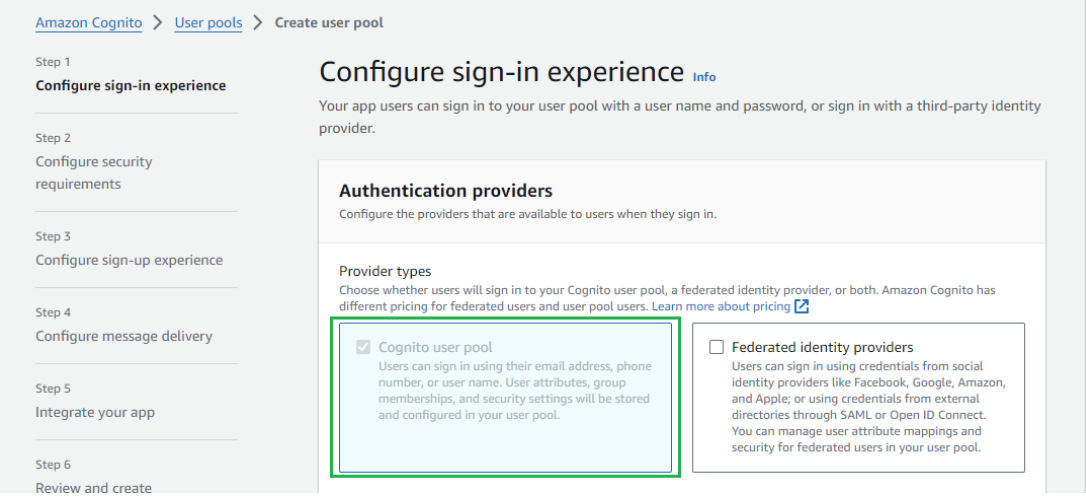


Create Amazon Cognito user pool

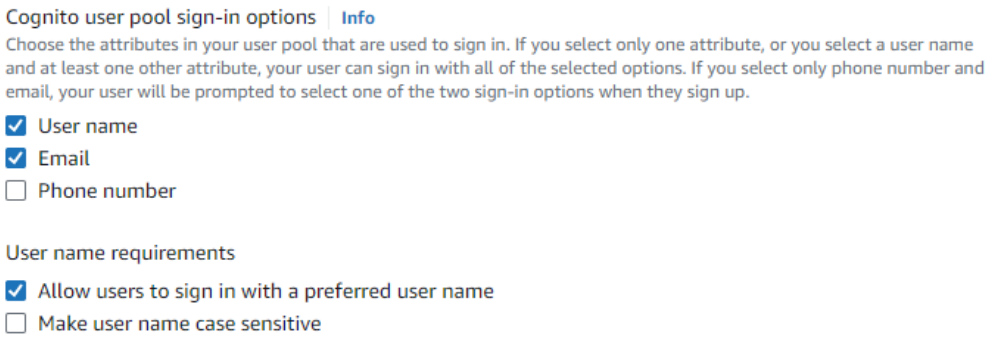
1. Access https://aws.amazon.com/cognito/v2/home and sign in with your account.
2. On the main page, click `Create user pool`.



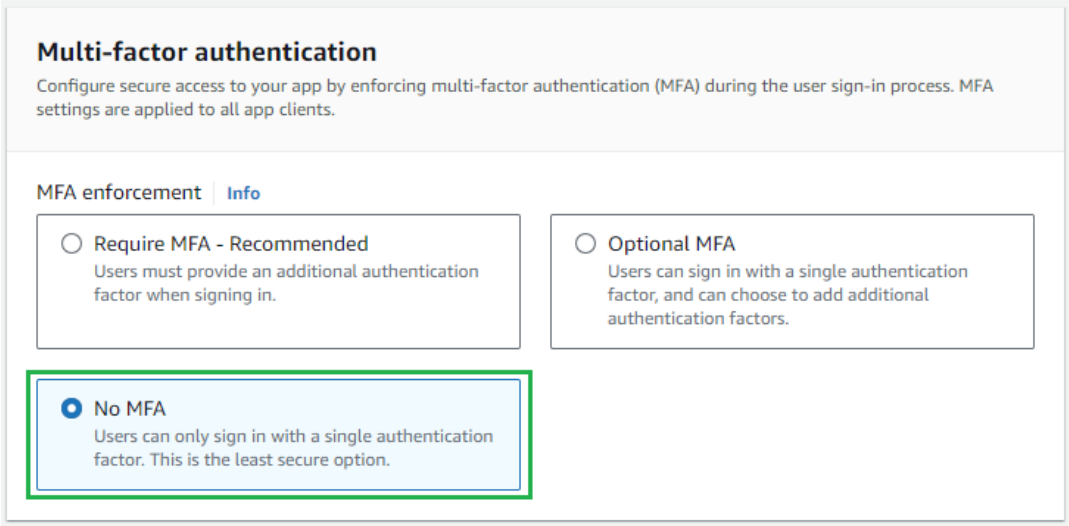
3. In `Step 1 - Configure sign-in experience`, select `Cognito user pool` for `Provider types`.



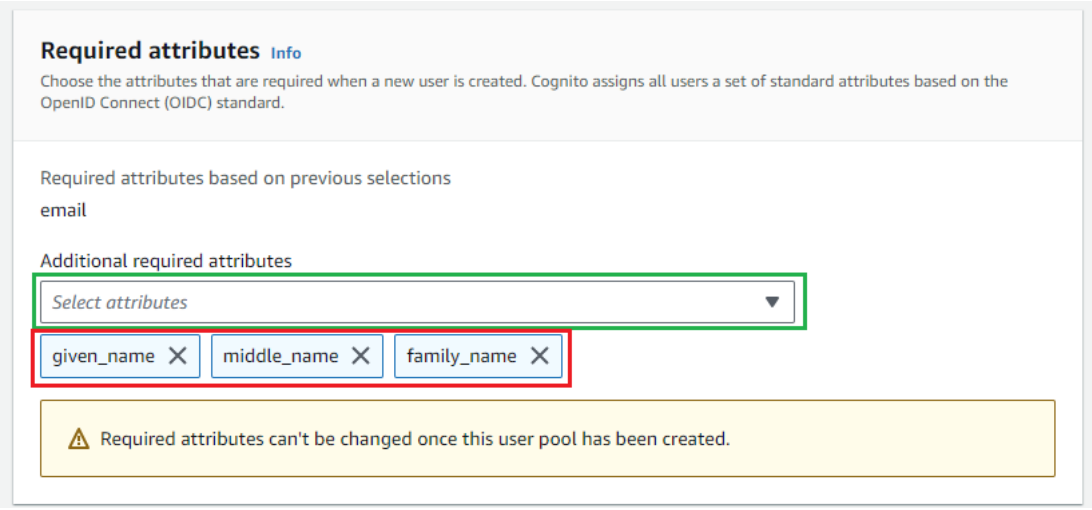
4. Tick `User name`, `Email`, and `Allow users to sign in with a preferred user name`, then click `Next`.



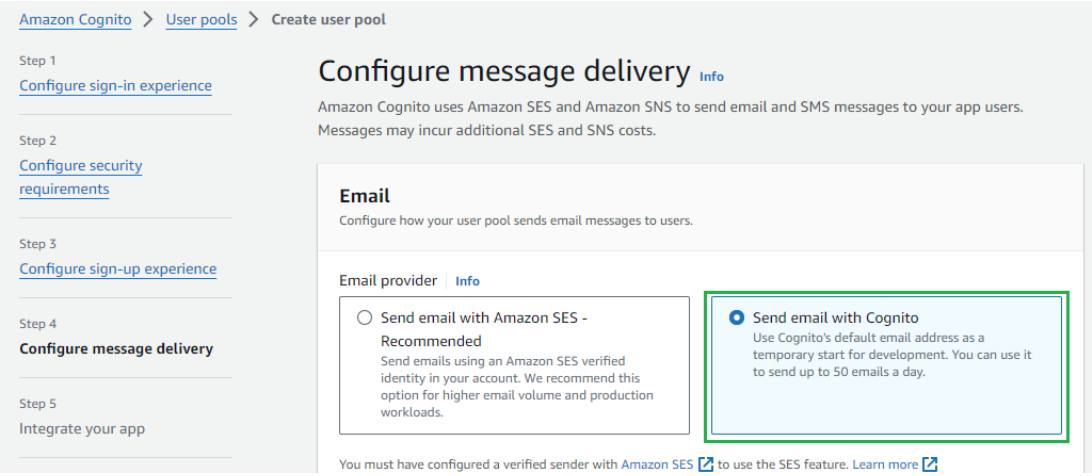
5. In `Step 2 - Configure security requirements`, select `No MFA` for `MFA enforcement`, leave all others as default , and click `Next`.



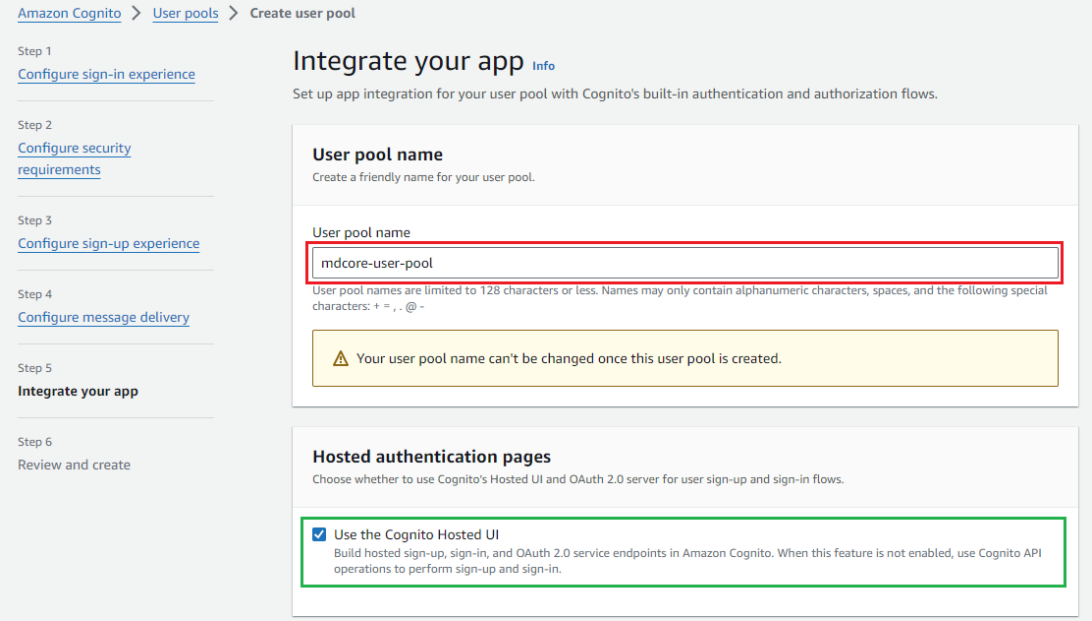
6. In `Step 3 - Configure sign-up experience`, navigate to `Required attributes` and add additional attributes: given_name, middle_name , and family_name. Click `Next`.



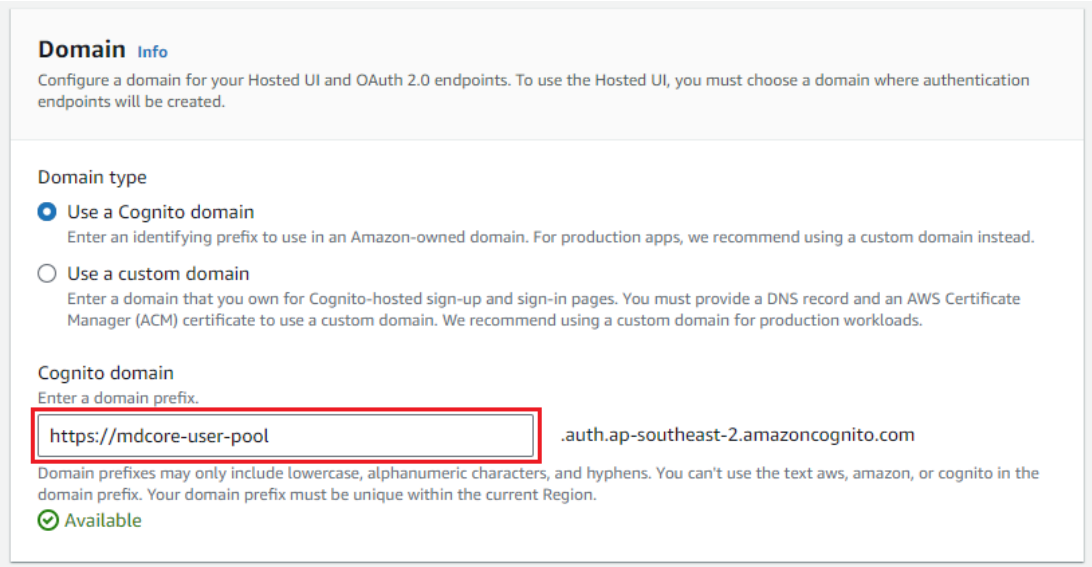
7. In `Step 4 - Configure message delivery`, choose `Send email with Cognito` and click `Next`.



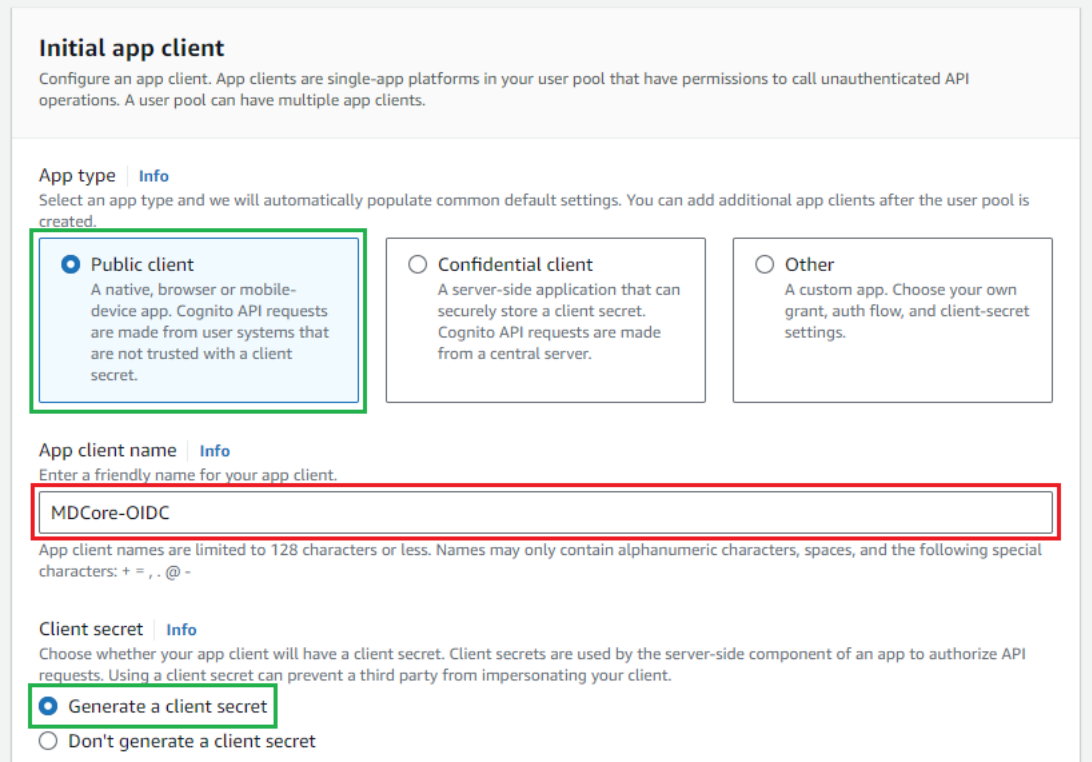
8. In `Step 5 - Integrate your app`, enter the user pool name and tick `Use the Cognito Hosted UI`.



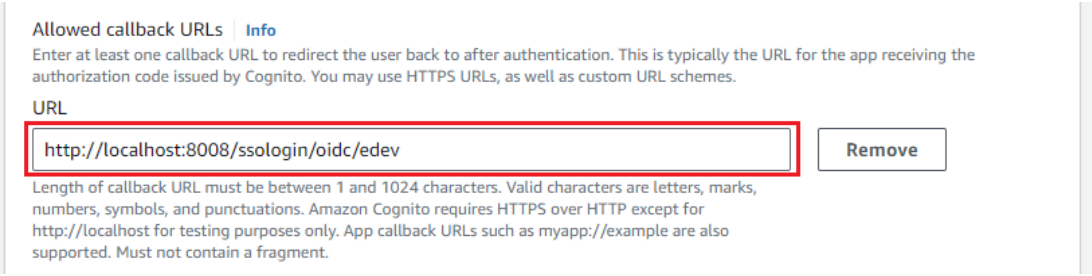
9. Enter your domain to `Cognito domain`.



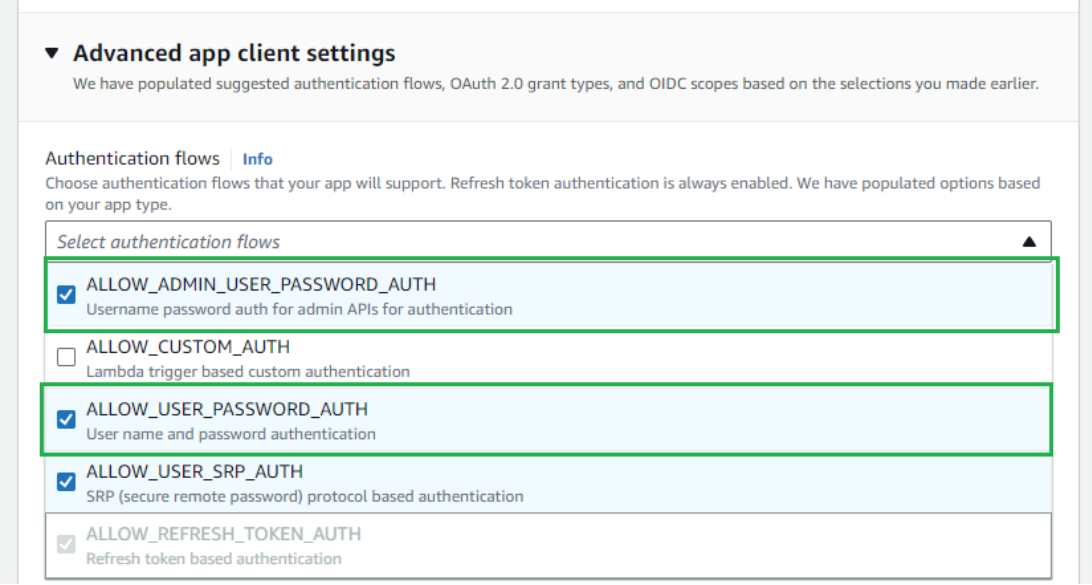
10. Under `Initial app client` section, select `Public client` for `App type`, enter `App client name`, and select `Generate a client secret` for `Client secret`.



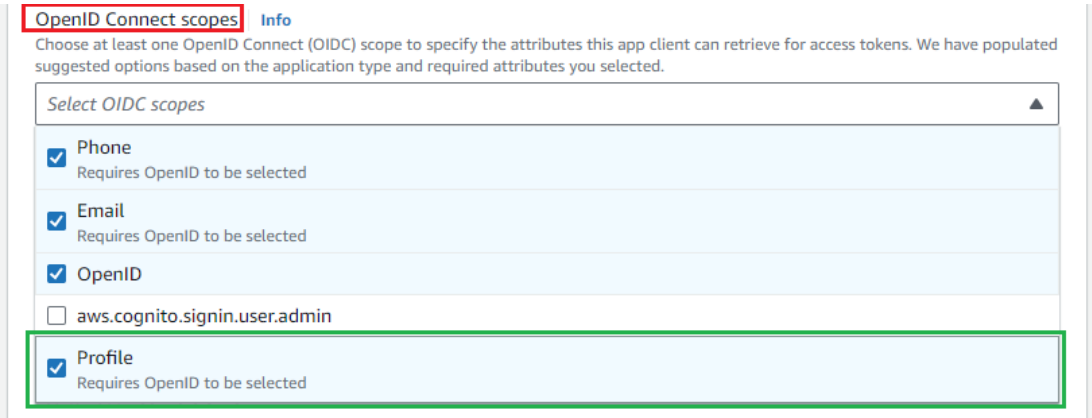
11. Under `Allowed callback URLs` section, fill in `URL` box with the value of *reply_uri*.



12. Expand `Advanced app client settings`, then stick to ALLOW_ADMIN_USER_PASSWORD_AUTH and ALLOW_USER_PASSWORD_AUTH for `Authentication flows`.



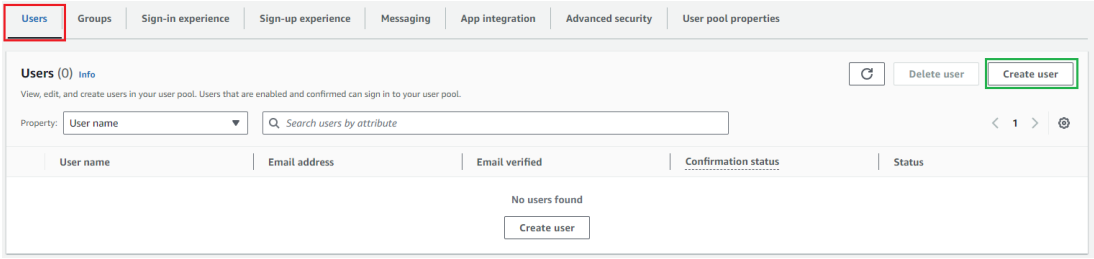
13. Navigate to `OpenID Connect scopes` and add `Profile` scope.



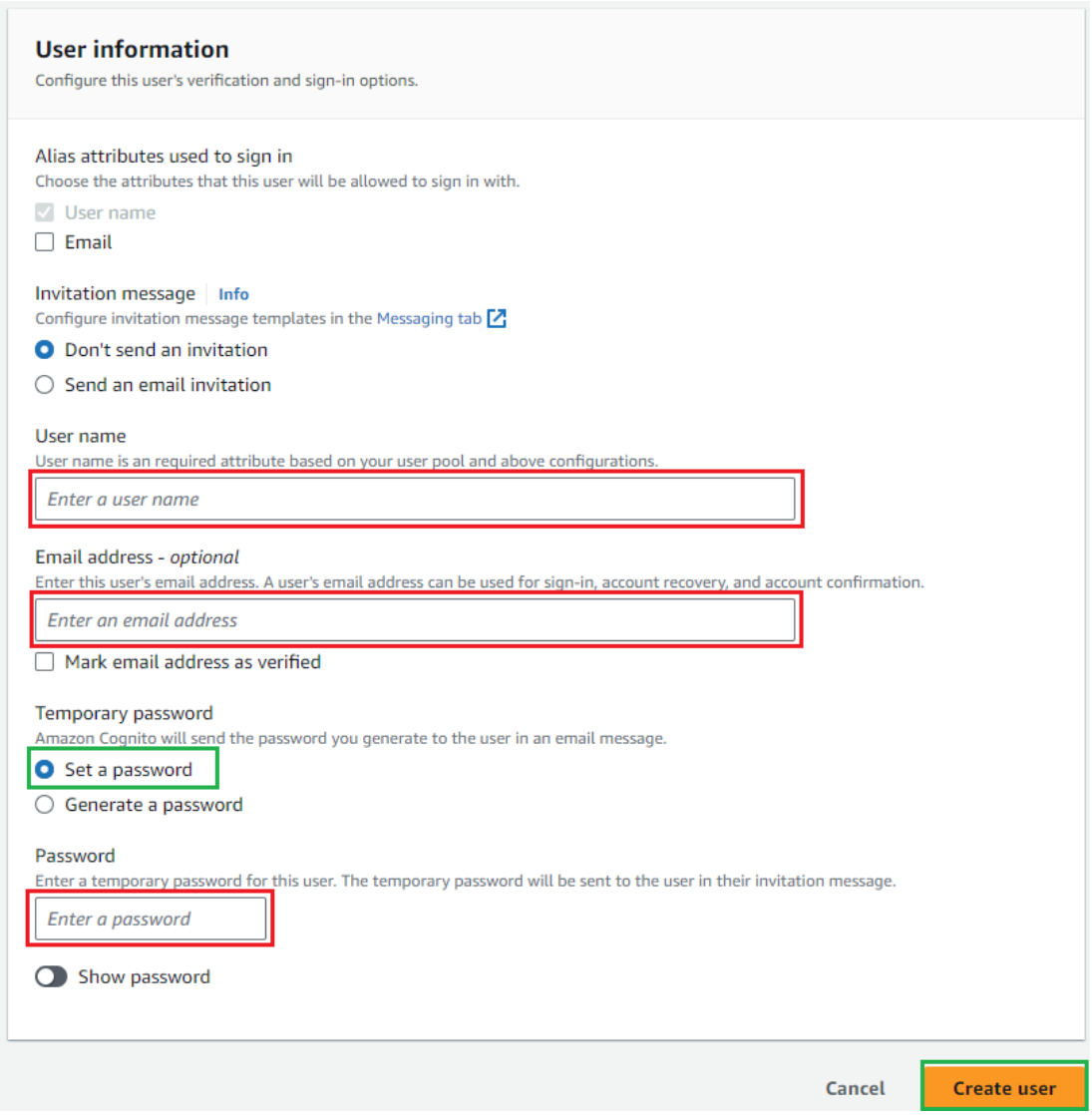
14. Click `Next` in the final review page and click `Create user pool`.

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`.

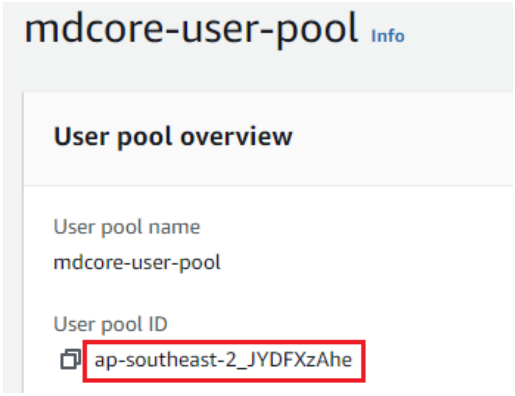


3. Enter the user name, and optional email address; stick to `Set a password`, and enter your temporary password. Click `Create User` to complete.



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.

json

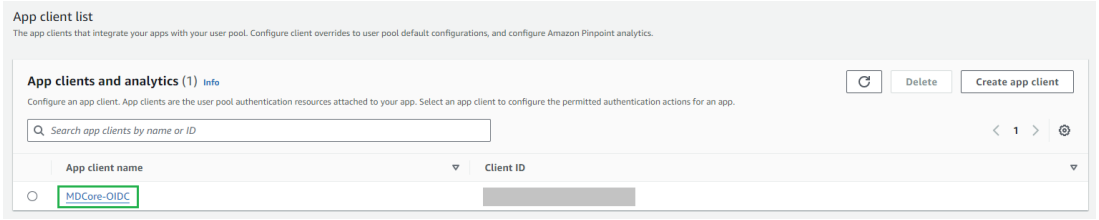
```
"claims_supported": ["middle_name", "given_name", "family_name"],
```

5. Save the file.

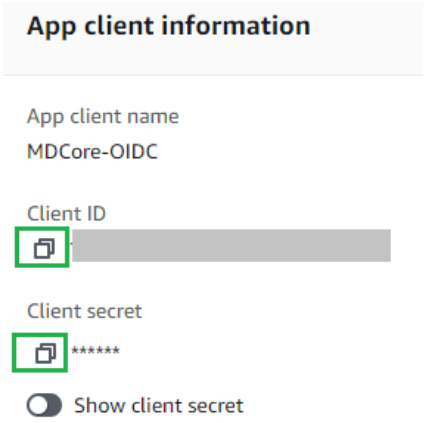
> **Info**
> `claims_supported` includes items that are selected for `Required Attributes` at step 6 of Create Amazon Cognito user pool stage.

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.



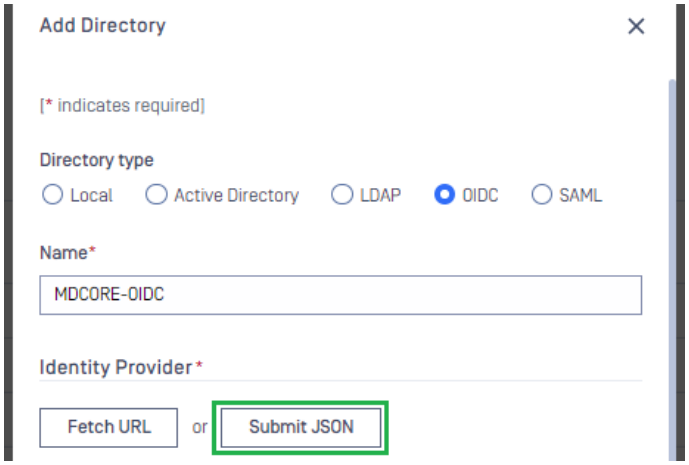
3. Copy `Client ID` and `Client Secret`.



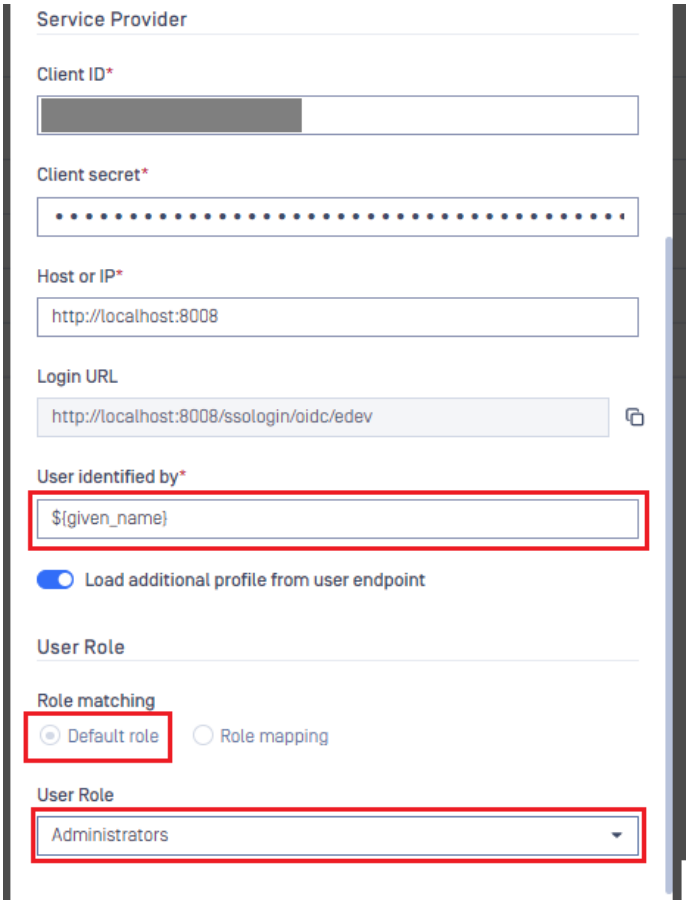
4. Switch to MetaDefender Core, under `Service Provider`, paste `Client ID` and `Client Secret` from the previous step into their respective fields.



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.

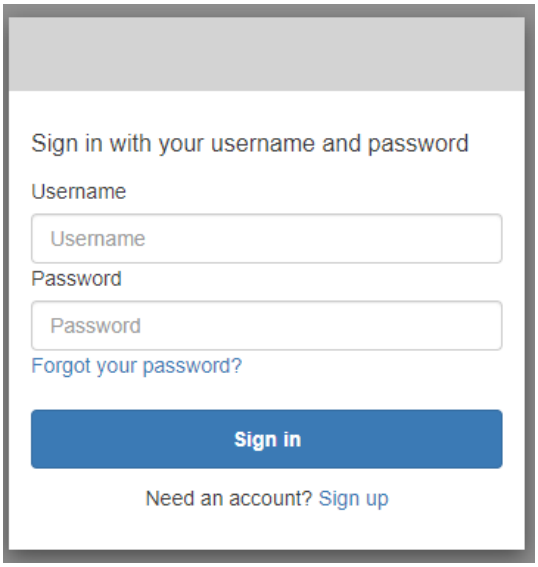


7. Click `Add` to complete.
8. In `User Management`, toggle the new directory. A dialog box will appear to confirm the action. Once `Enable` is clicked, all sessions will be expired immediately.

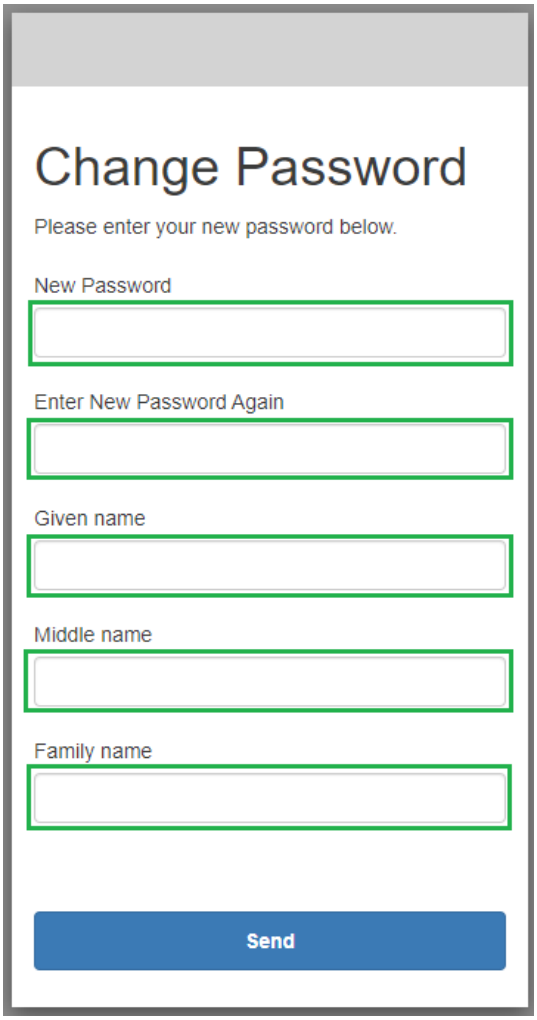


Test the integration

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`.



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

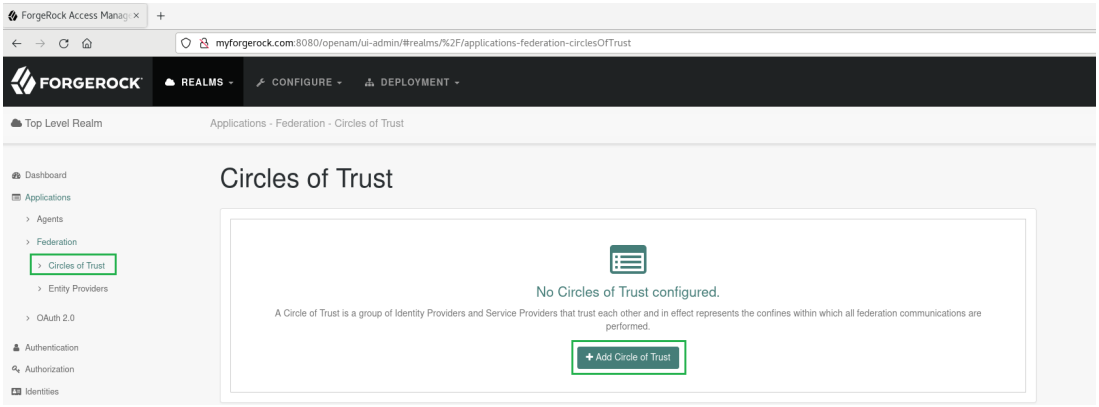
The instructions guide users on how to configure ForgeRock SAML Single Sign-on for MetaDefender Core.

Configuring SAML Single Sign-on

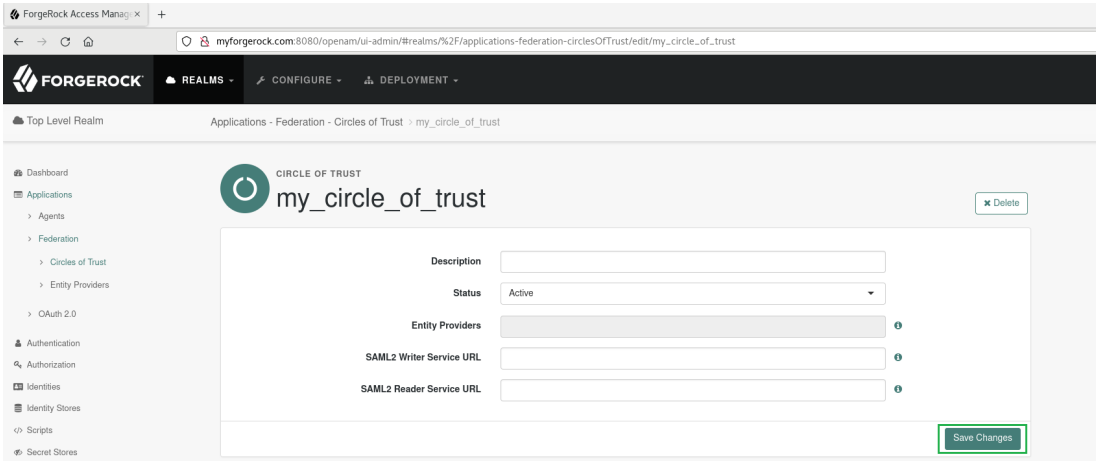
> **Info**
> The integration is demonstrated using ForgeRock version 7.2.0.

Configure Identity Provider in ForgeRock

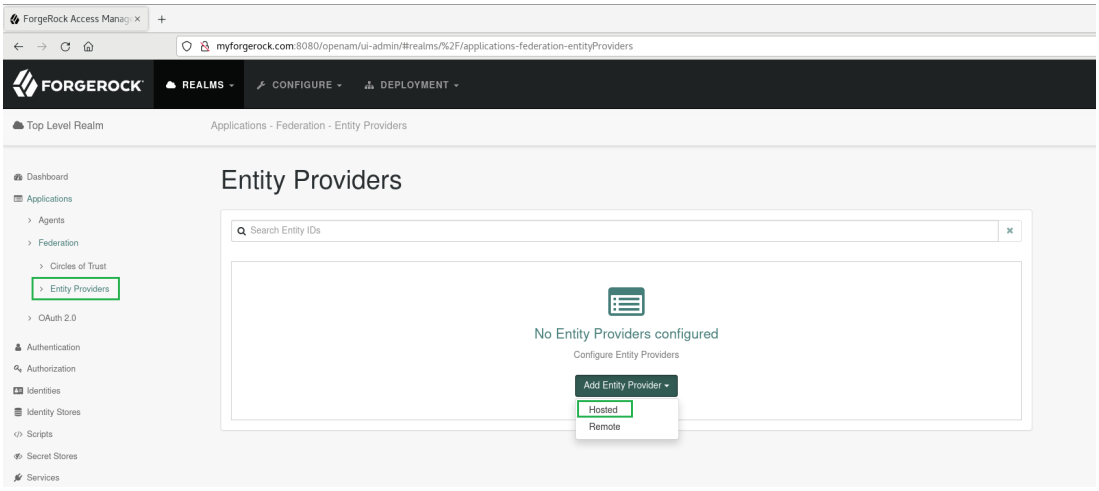
1. Access ForgeRock console.
2. On `Realms` page, select `Top Level Realm` located in the upper left corner.
3. Go to `Applications` in the sidebar, expand `Federation`, choose `Circles of Trust`, and press `Add Circle of Trust`.



4. Input the name, 'my_circle_of_trust' for instance, then select `Create` and `Save Changes`.

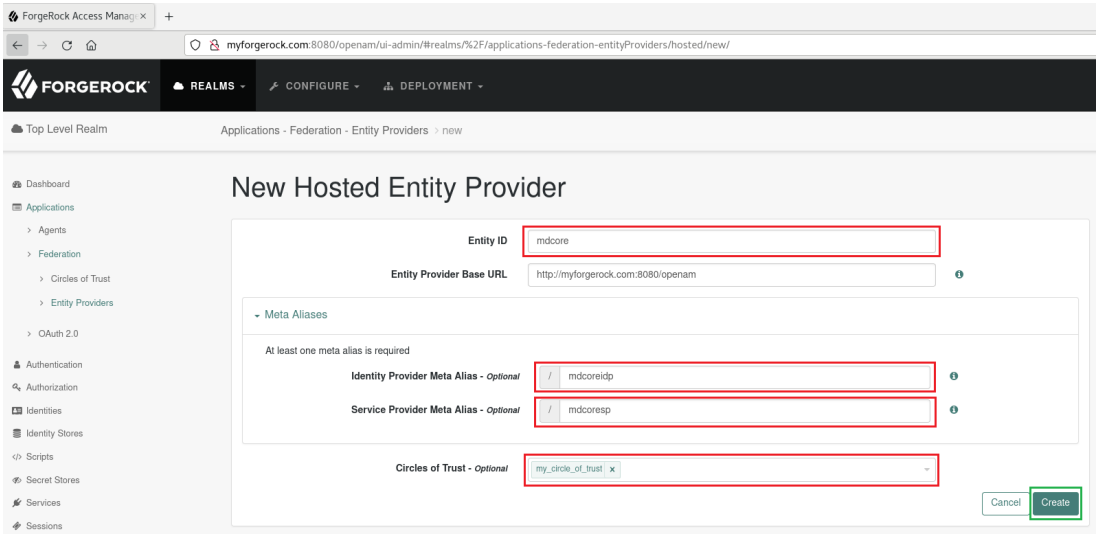


5. Select `Entity Providers` from `Federation` in the side bar, select `Add Entity Provider`, and pick `Hosted`.



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.

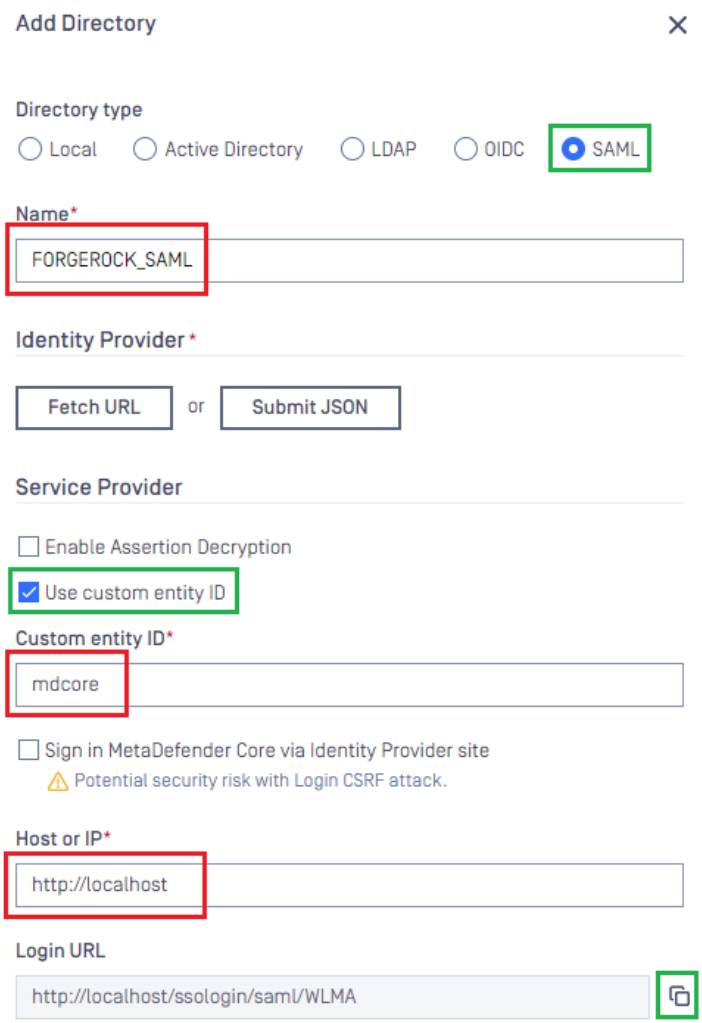


Create SAML directory in MetaDefender Core

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`.

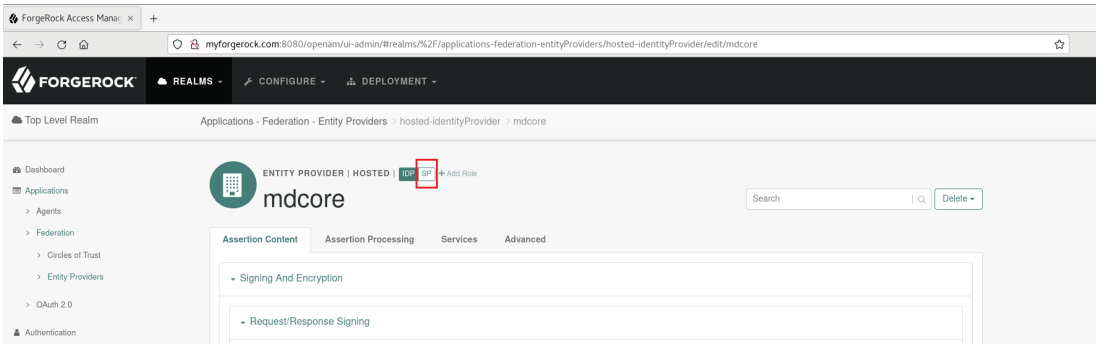
> **Info**
> The copied login URL will be used subsequently in `Location` setting of ForgeRock.



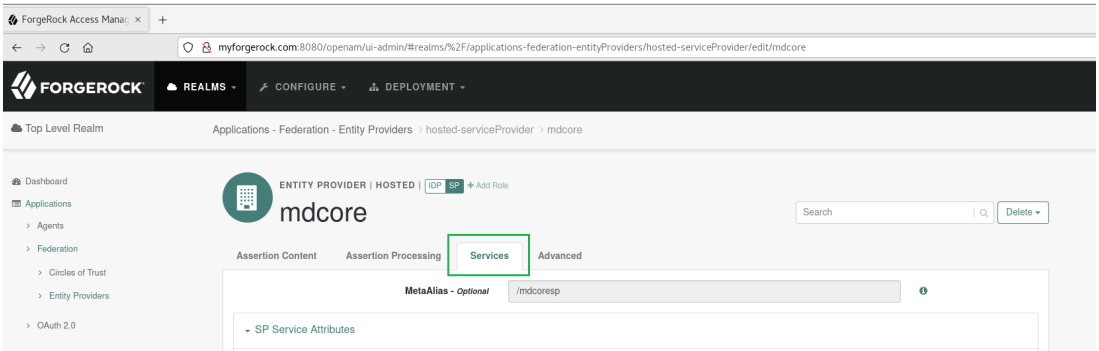
> **Warning**
> The setup in MetaDefender Core is incomplete, please refrain from clicking outside of `Add Directory` screen.

Configure Service Provider in ForgeRock

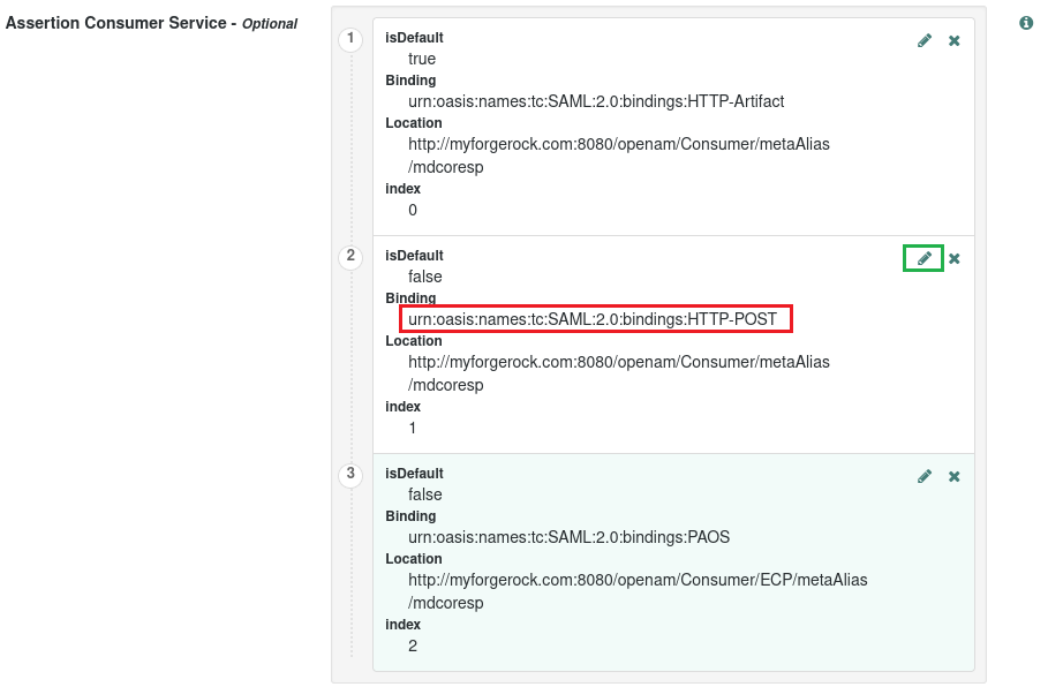
1. In the ForgeRock console, switch to `SP` for the entity provider 'mdcore'.



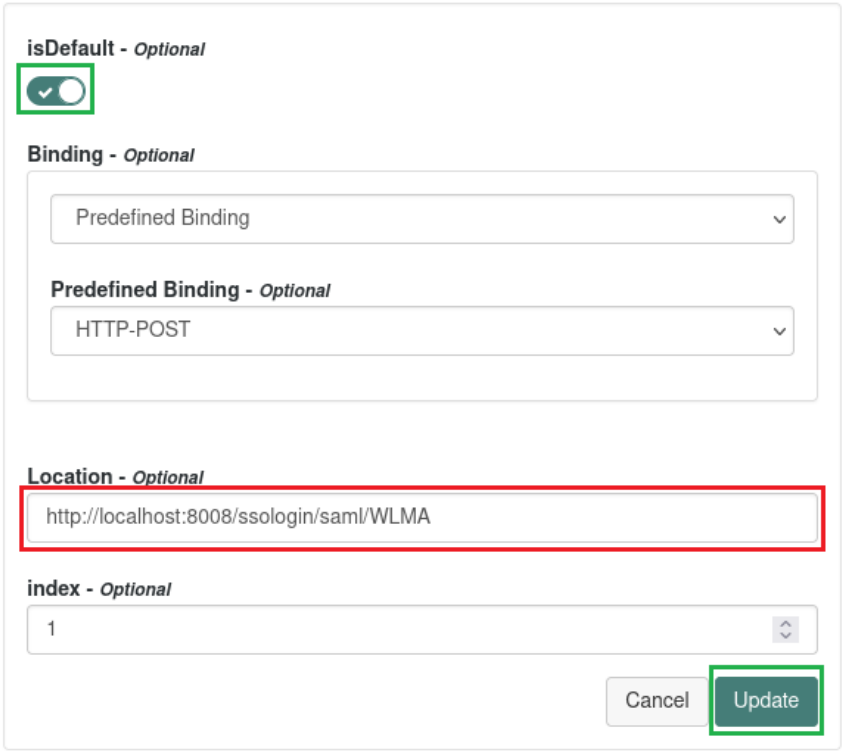
2. Click on `Services` tab.



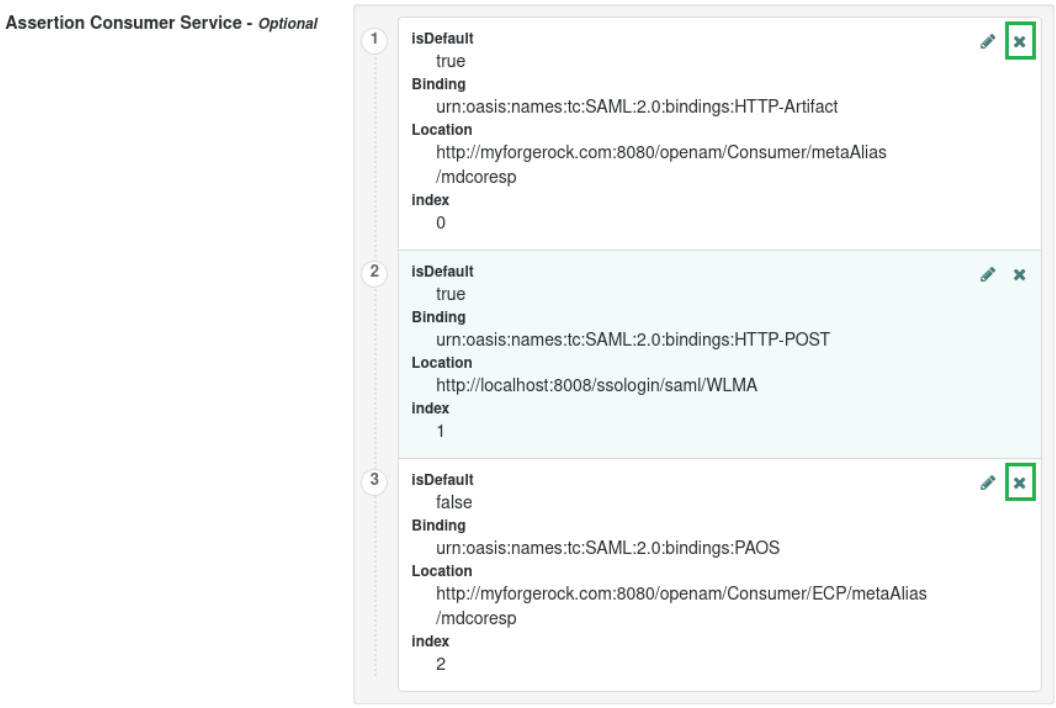
3. Go to `Assertion Consumer Service` area, choose HTTP-POST binding entry, and tap the edit icon located in the top right.



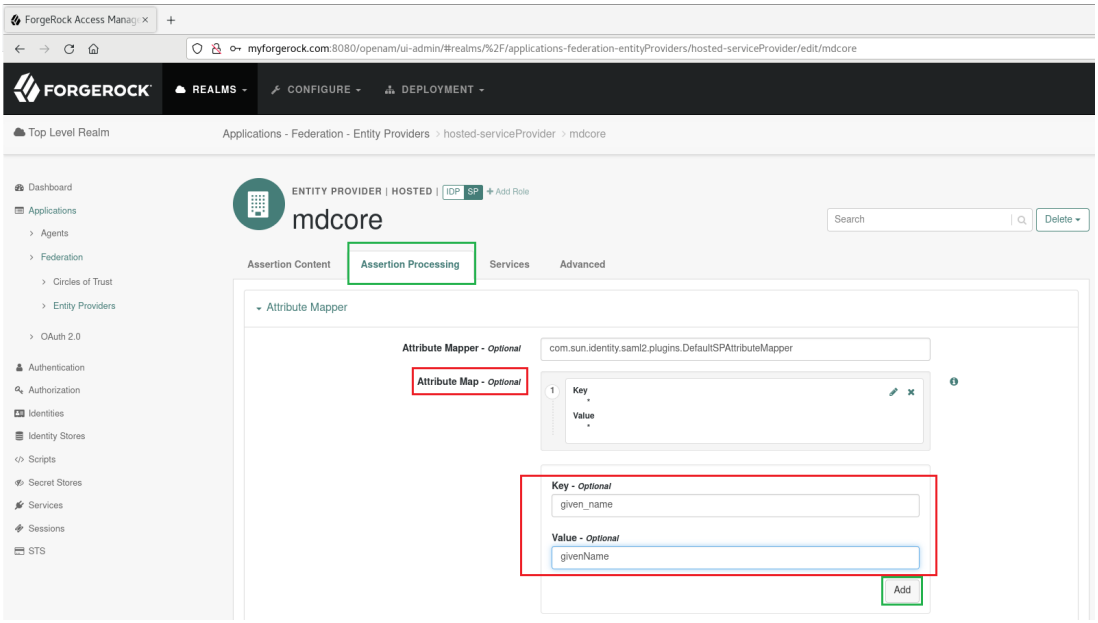
4. Input login URL from MetaDefender Core into `Location`, toggle `isDefault` switch in the upper left, and press `Update`.



5. Remove HTTP-Artifact and PAOS bindings, then press `Save Changes`.



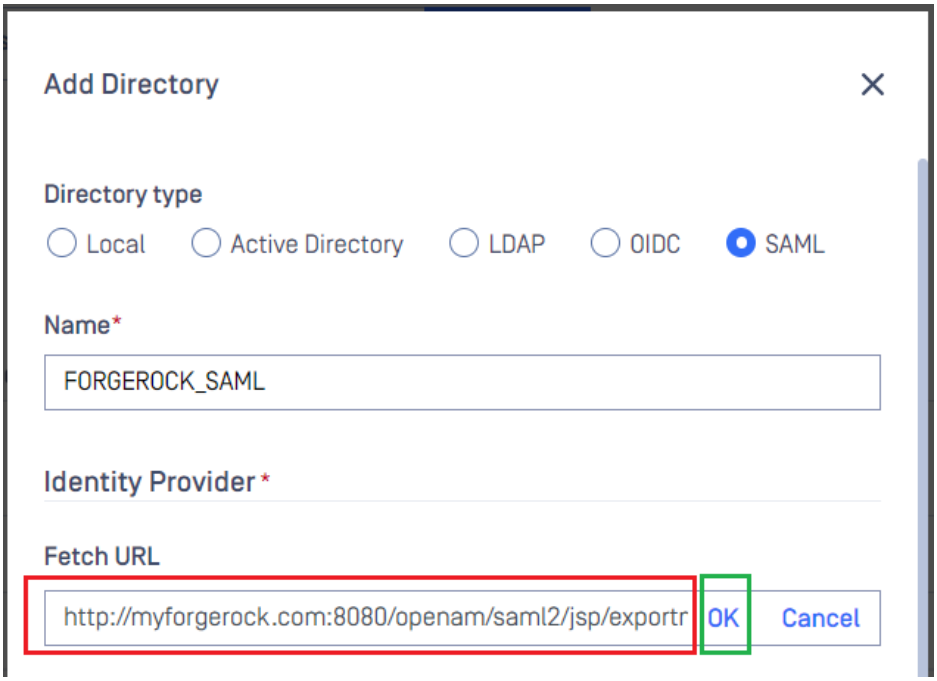
6. Select `Assertion Processing` tab, navigate to `Attribute Map`, enter 'given_name' and 'givenName' as `Key` and `Value`, accordingly.



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 `Fetch 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

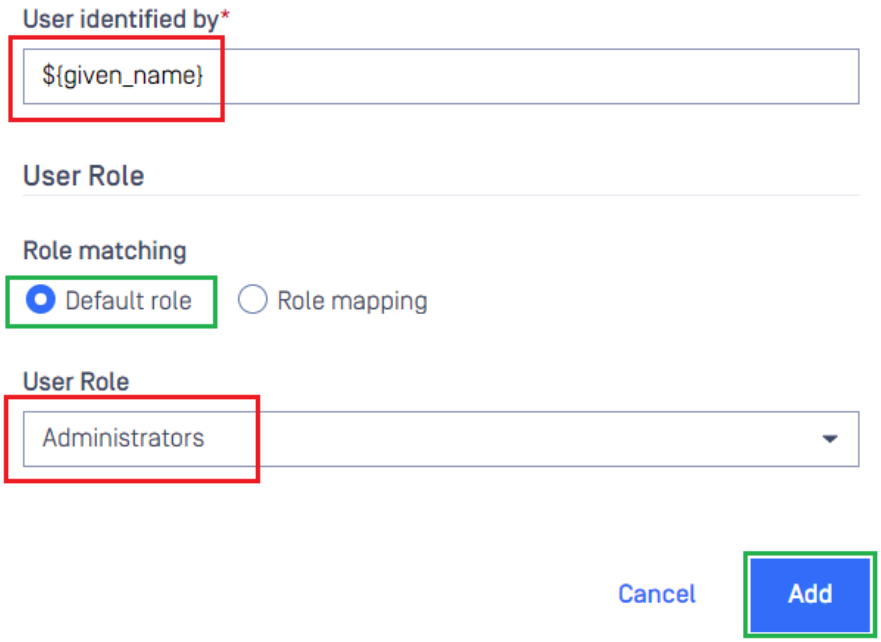
The structure of the metadata URL is as outlined below:

`http://<forgerock_server>:<port>/openam/saml2/jsp/exportmetadata.jsp?entityid=<entity_id>`

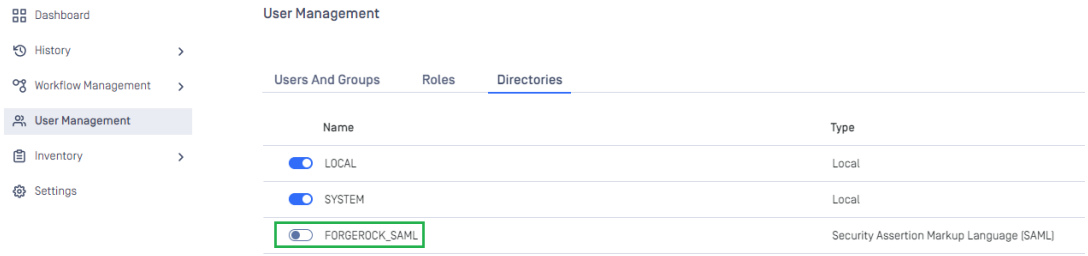
In which, `forgerock_server` and `port` correspond to the host and port of the machine hosting ForgeRock server; `entity_id` refers to the value of `Entity ID` configured in ForgeRock. In this guide, ForgeRock is located at 'myforgerock.com:8080' and the `Entity ID` is 'mdcore'; thus, the metadata URL 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.



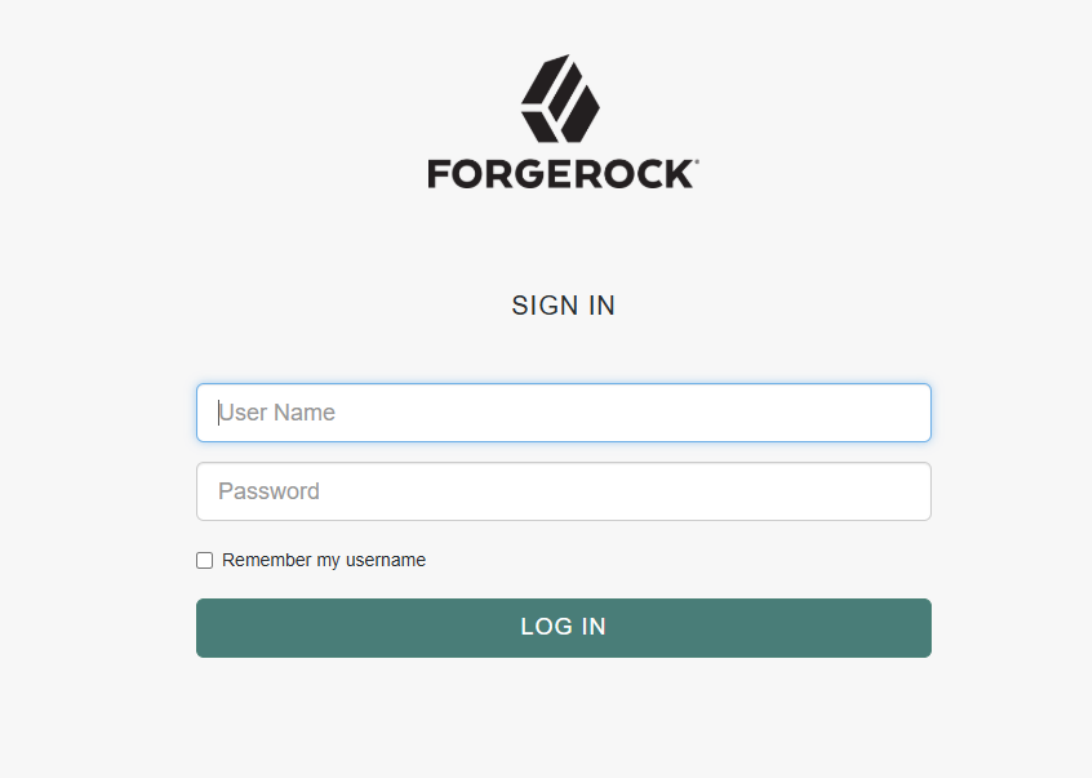
5. On `User Management` screen, switch on the new directory, 'FORGEROCK_SAML' in this case.



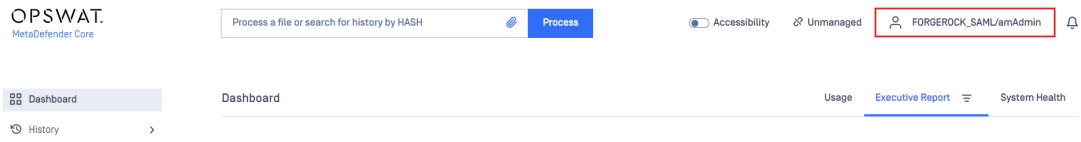
6. A confirmation dialog box will pop up to verify the action. Upon clicking `Enable`, all sessions will be terminated instantly, and ForgeRock will be utilized for user authentication.

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

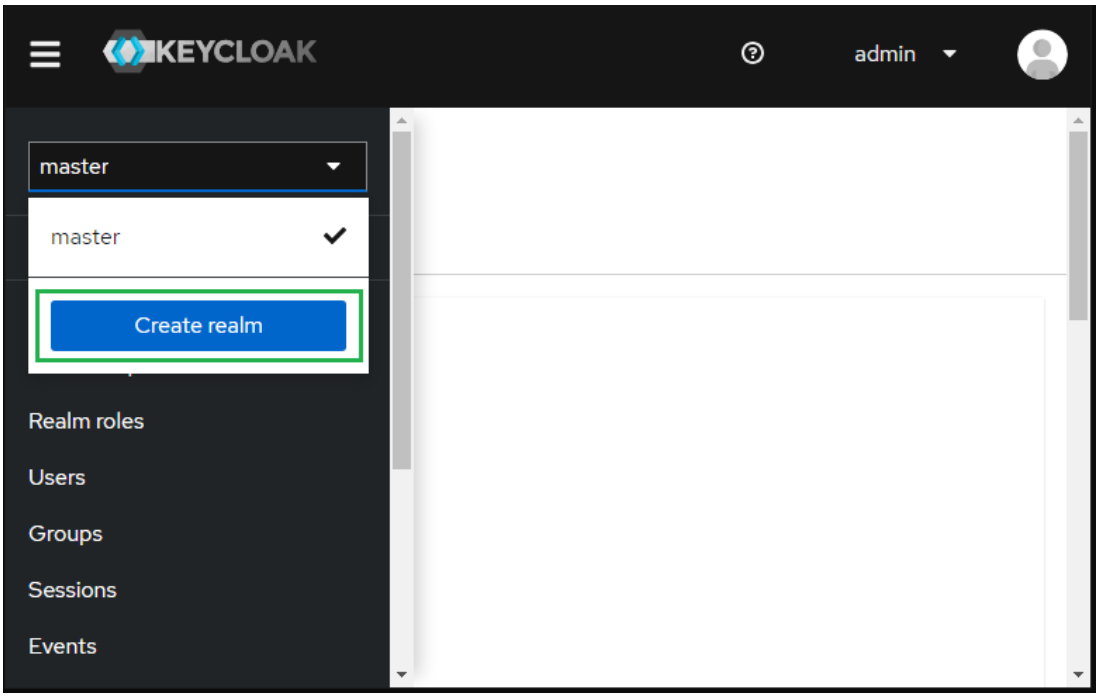
The instruction guides users on how to configure Keycloak SAML Single Sign-on for MetaDefender Core.

Configuring SAML Single Sign-on

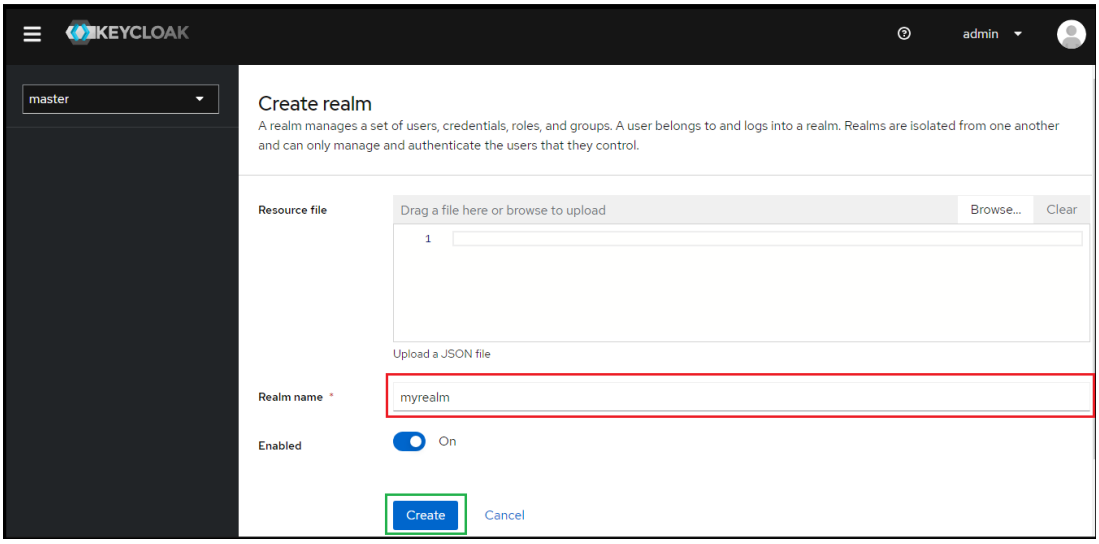
Create a realm in Keycloak

1. Sign in to Administrator Console, drop the list in the top left corner, and click `Create realm`.

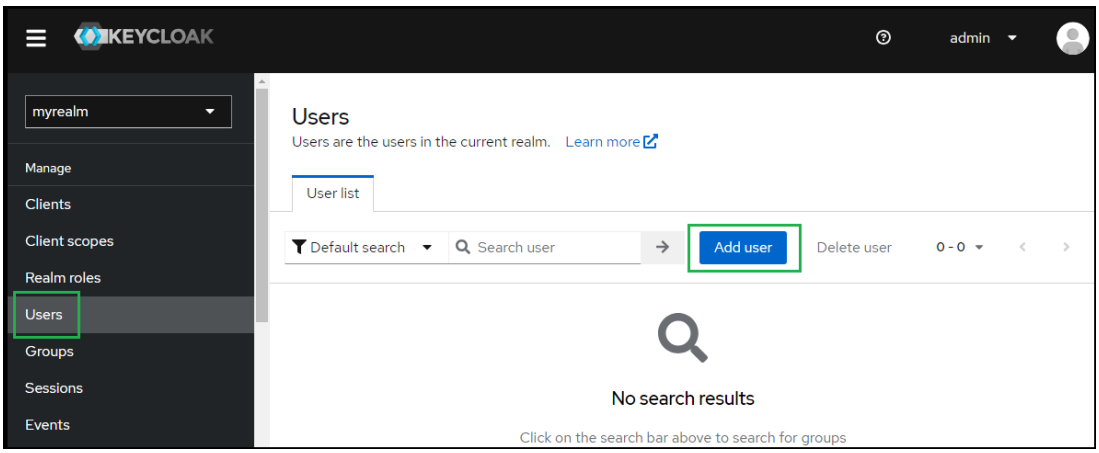
> **Info**
> Keycloak initially includes a single realm `master` which is used for managing Keycloak only and not for any applications.



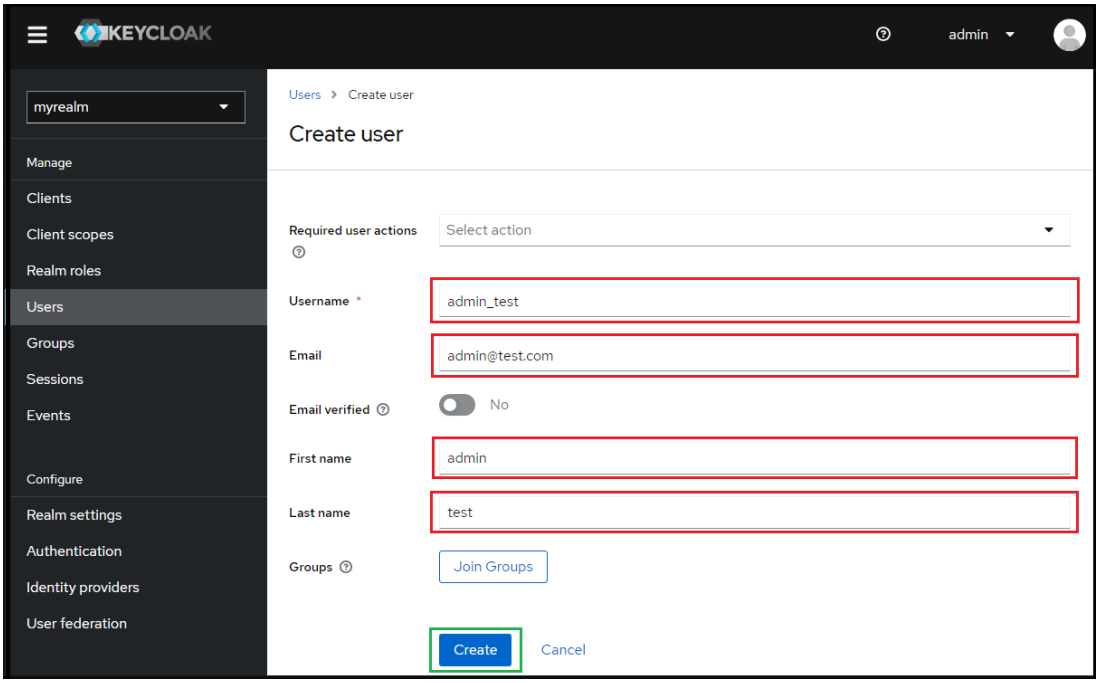
2. Enter `Realm name` e.g. myrealm and click `Create`.



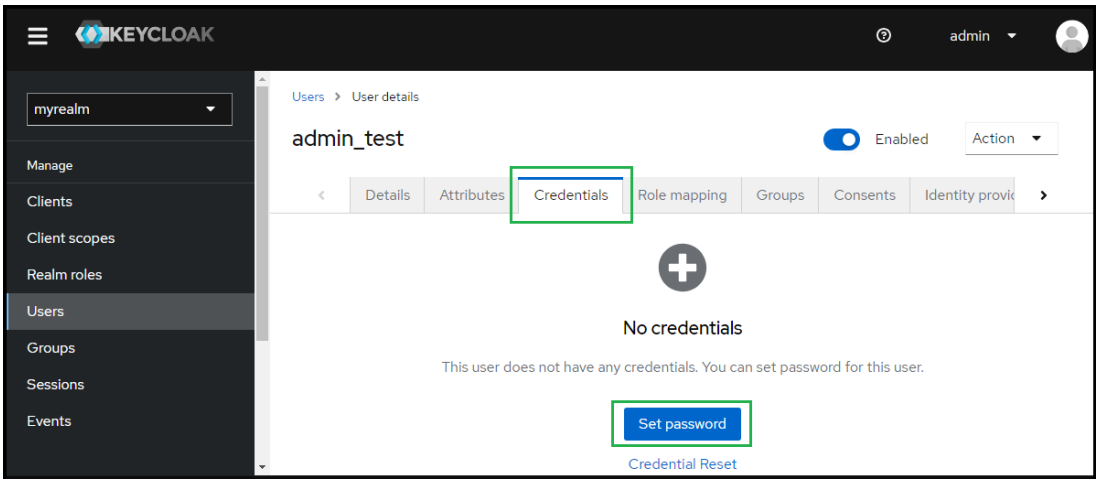
3. Select `Users` in the left sidebar and click `Add user`.



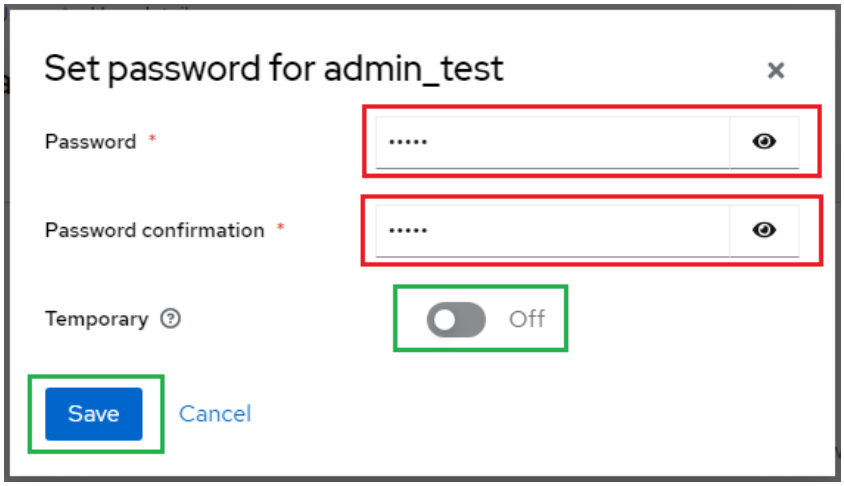
4. Enter values for `Username`, `Email`, `First name` and `Last name`; then click `Create`.



5. Under `User details`, select `Credentials` tab and click `Set password` to create a password for the user created in the previous step.

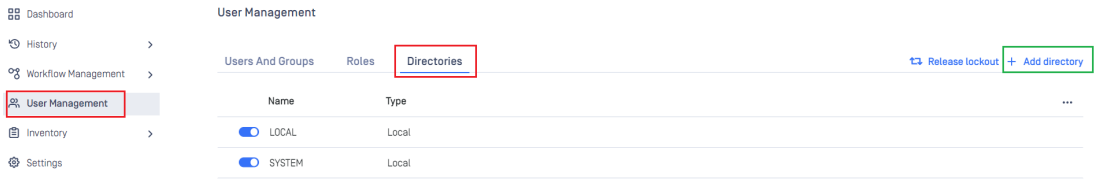


6. Enter the password and toggle `Temporary` to Off, then click `Save`.

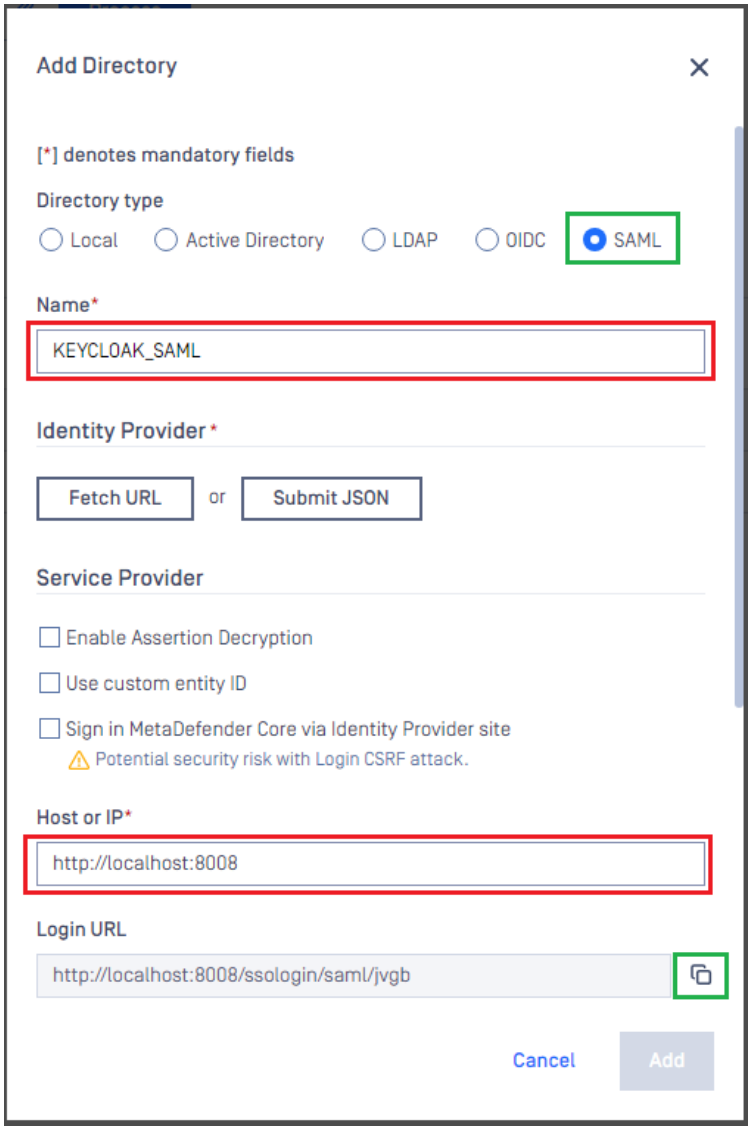


Create SAML directory in MetaDefender Core

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.

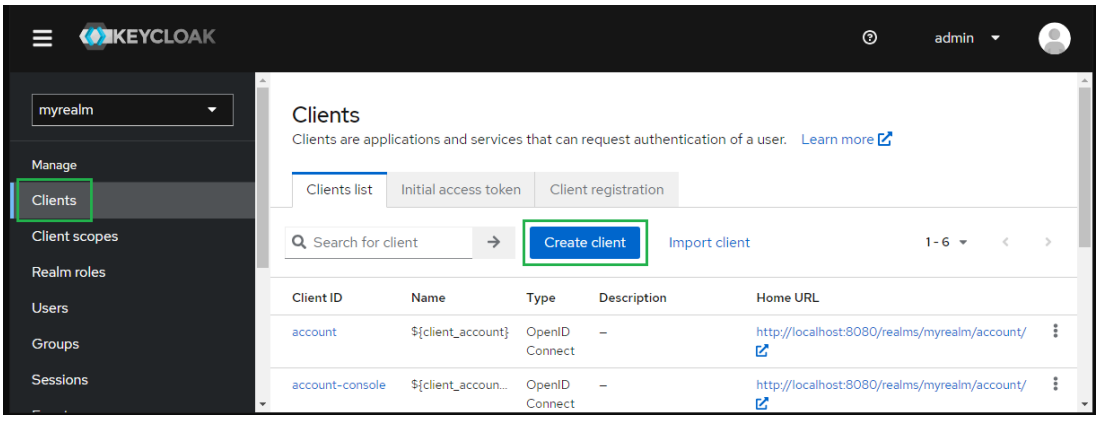


4. On `Add Directory` page, select SAML in `Directory type`.
5. Fill `Name` of the new directory, such as KEYCLOAK_SAML.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, using https://localhost:8008 as an example.
7. Copy the value of `Login URL`.

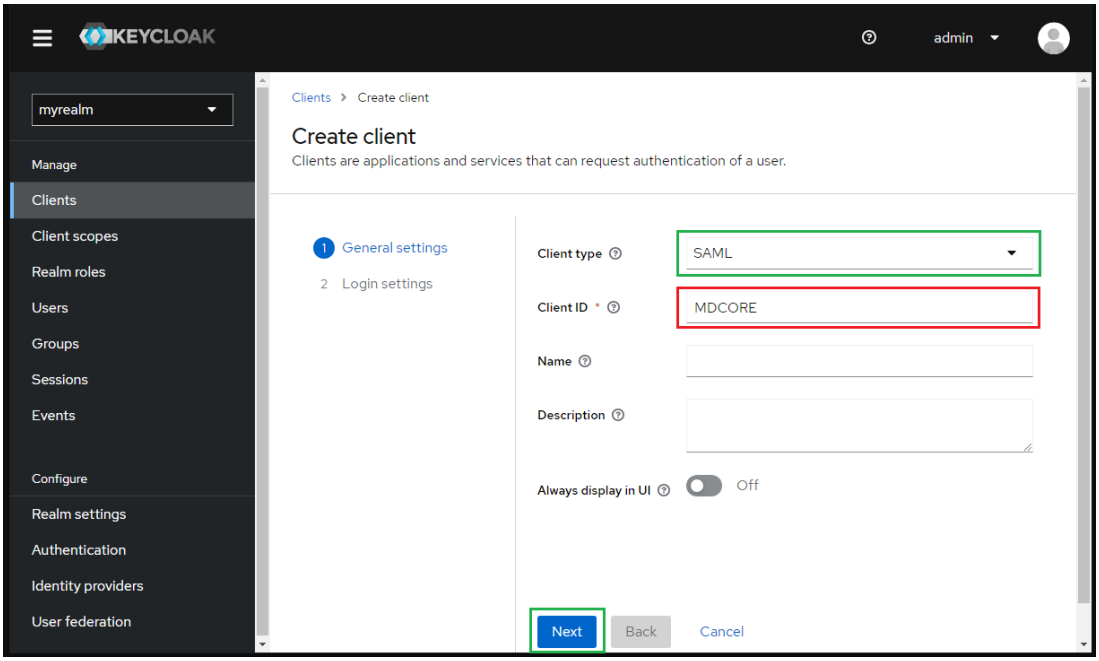


Create Keycloak application

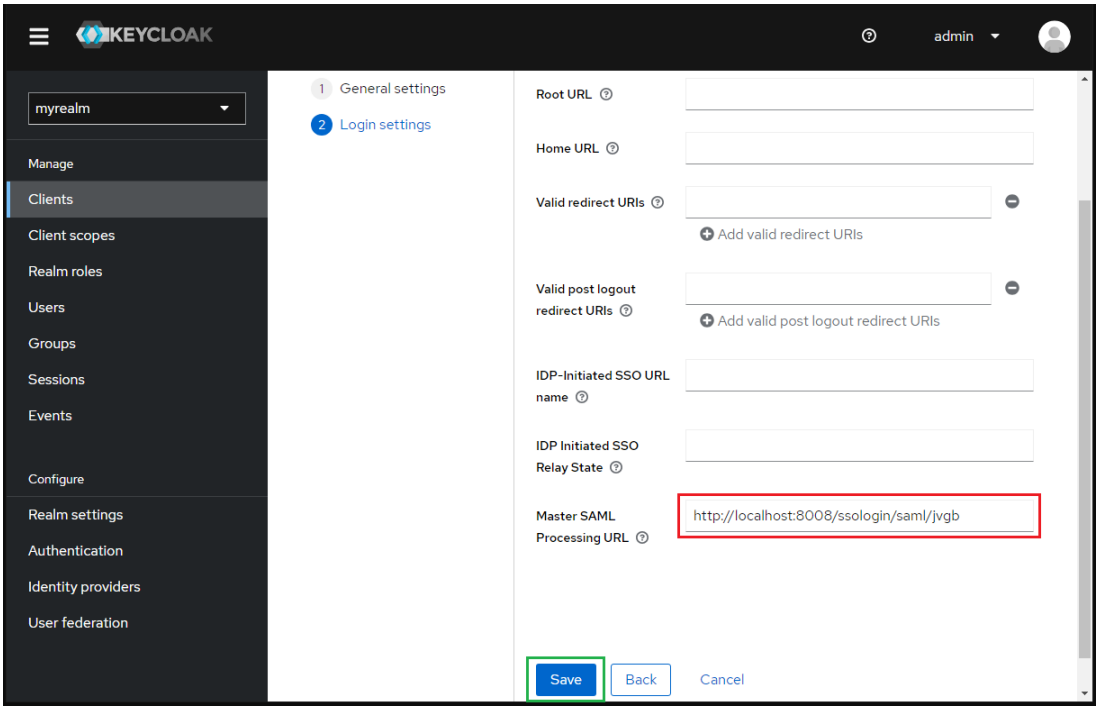
1. On screen `myrealm`, select `Clients` in the sidebar and click `Create client`.



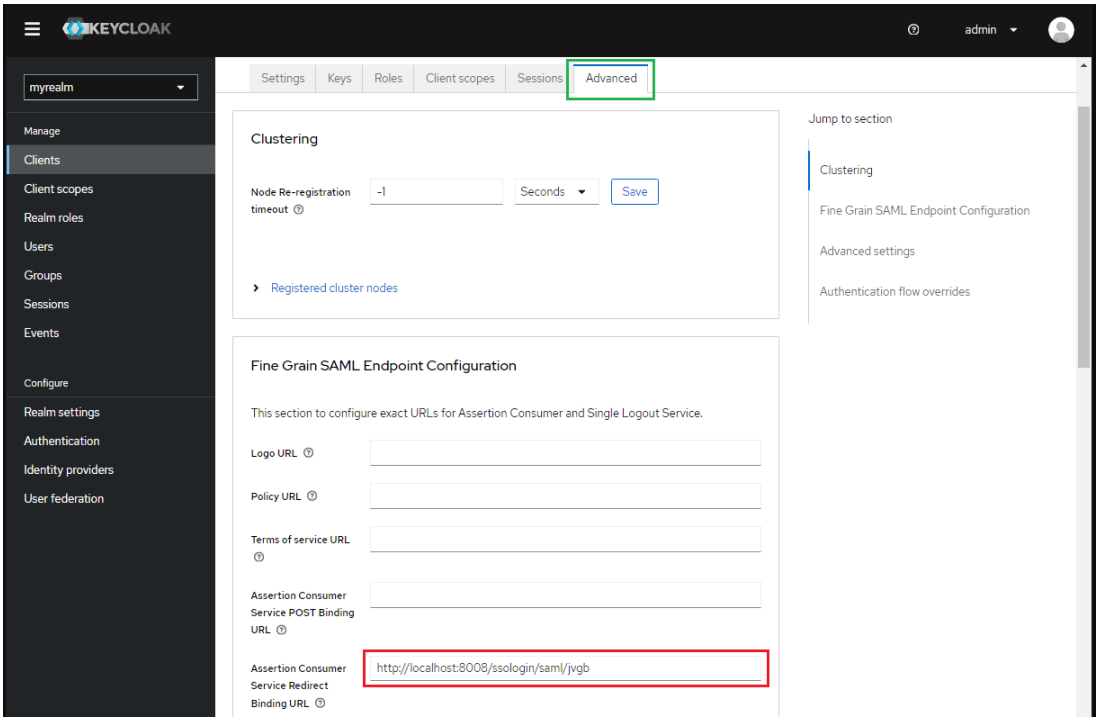
2. Choose SAML for `Client type` and enter MDCORE for `Client ID` then click `Next`.



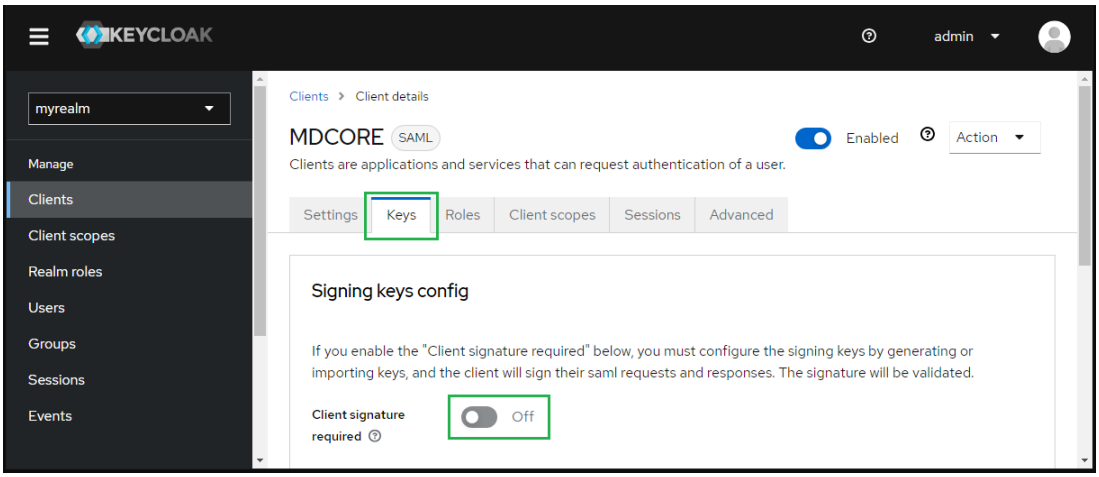
3. Paste the value of `Login URL` from MetaDefender Core into `Master SAML Processing URL` and click `Save`.



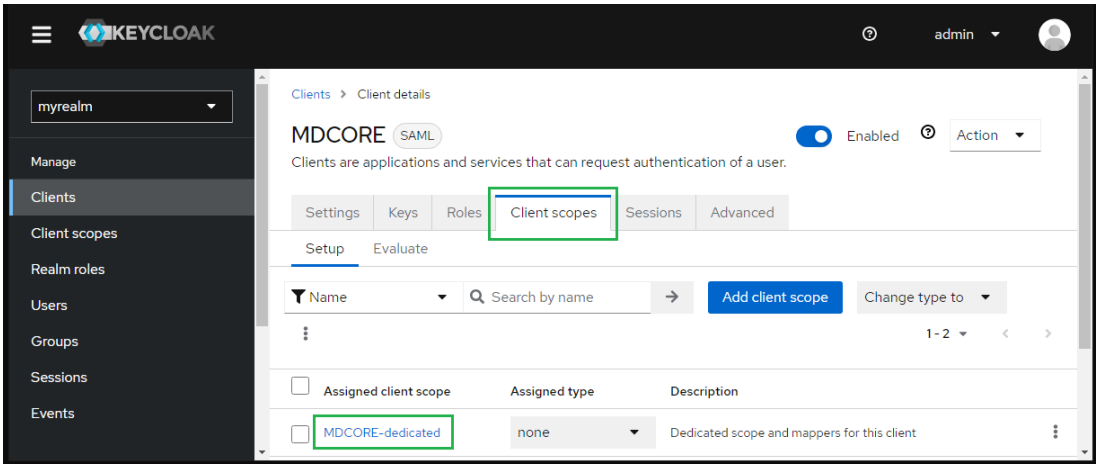
4. Go to `Advance` tab and paste the value of `Login URL` from MetaDefender Core to `Assertion Consumer Service Redirect Binding URL` and click `Save`.



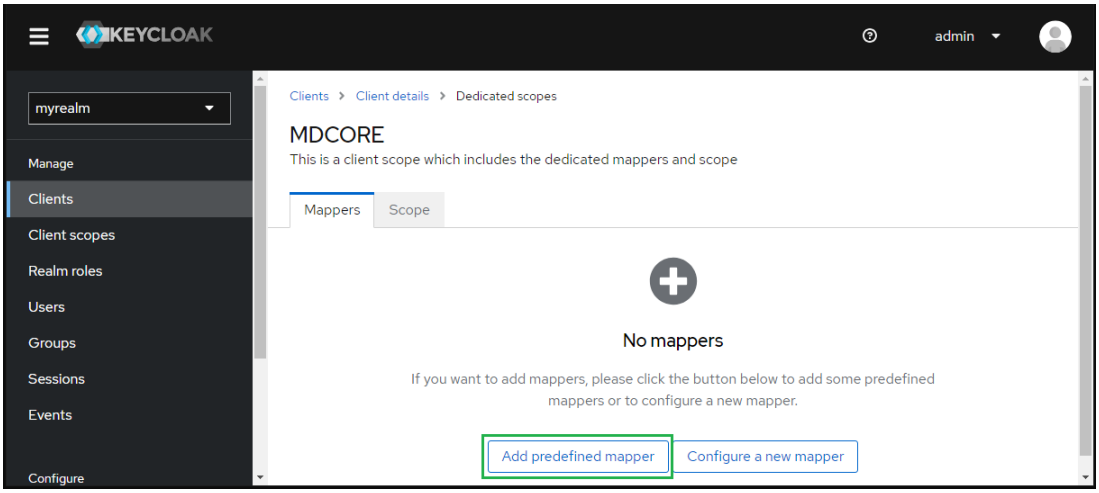
5. In `Keys` tab, toggle `Client signature required` to Off.



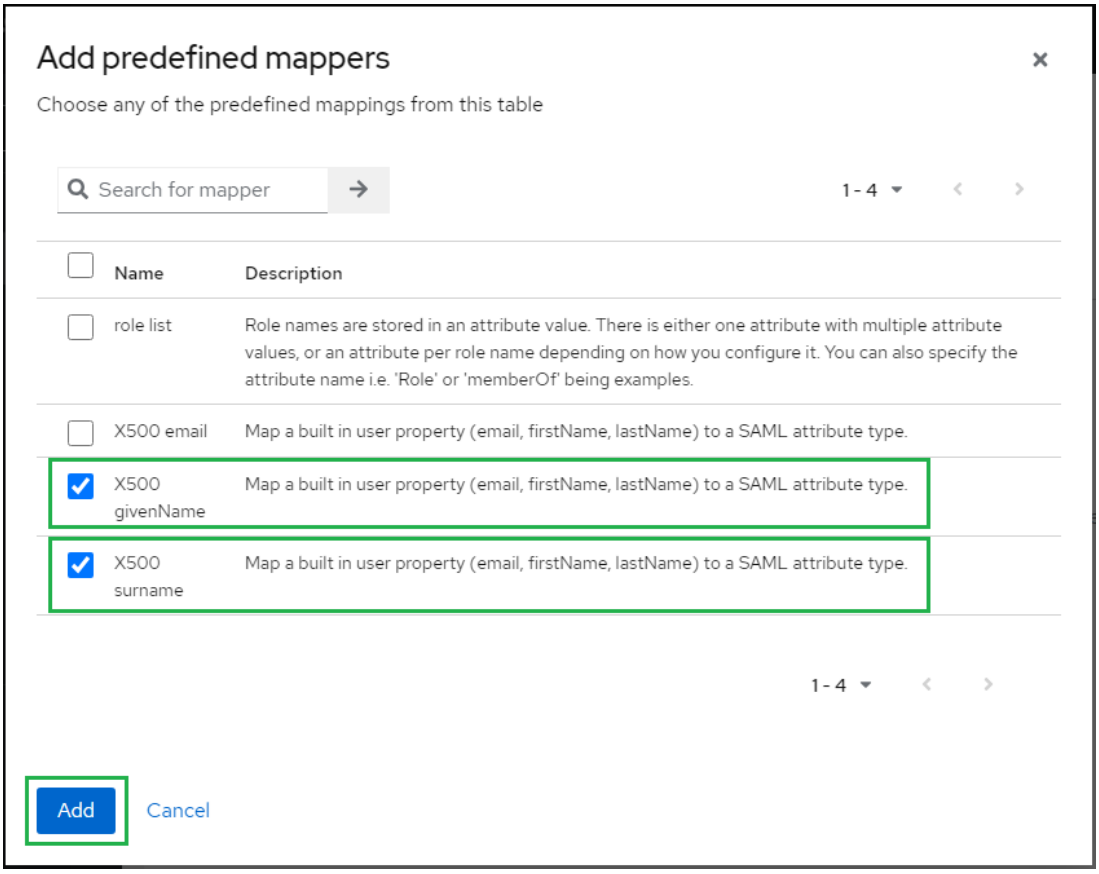
6. In `Client scopes` tab, select `MDCORE-dedicated`.



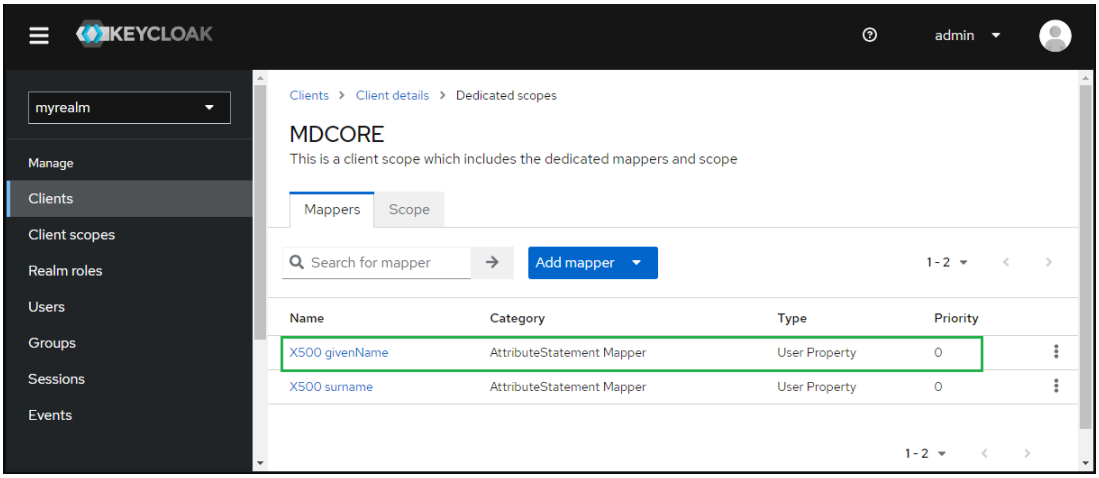
7. Under `Dedicated scopes`, navigate to `Mappers` tab and click `Add predefines mapper`.



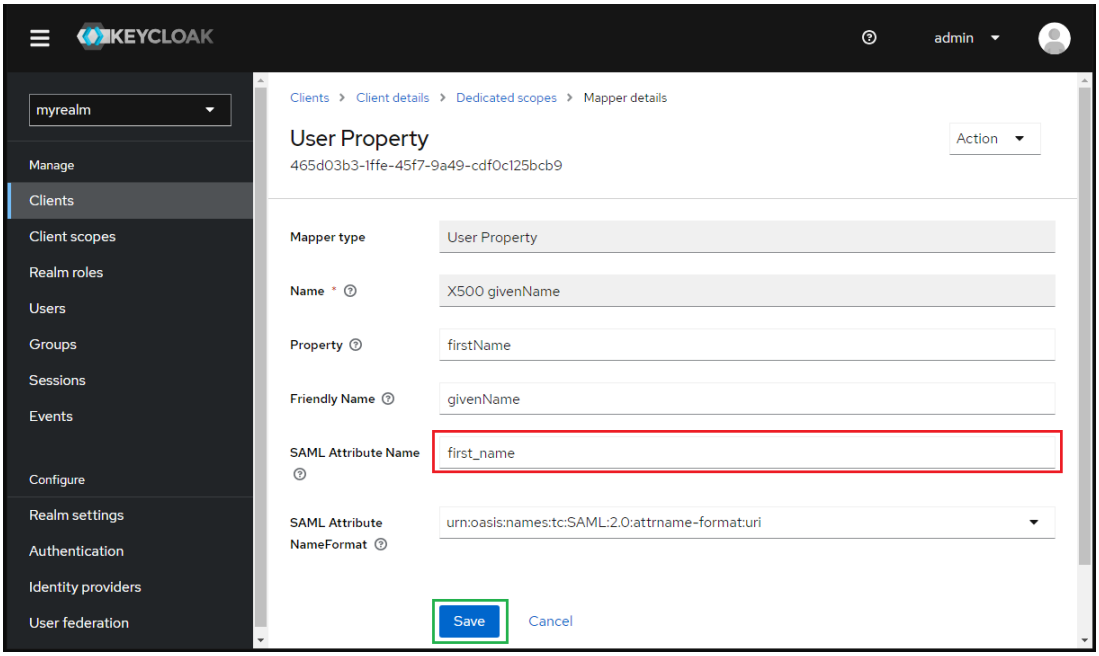
8. Check `X500 givenName` and `X500 surname`, then click `Add`.



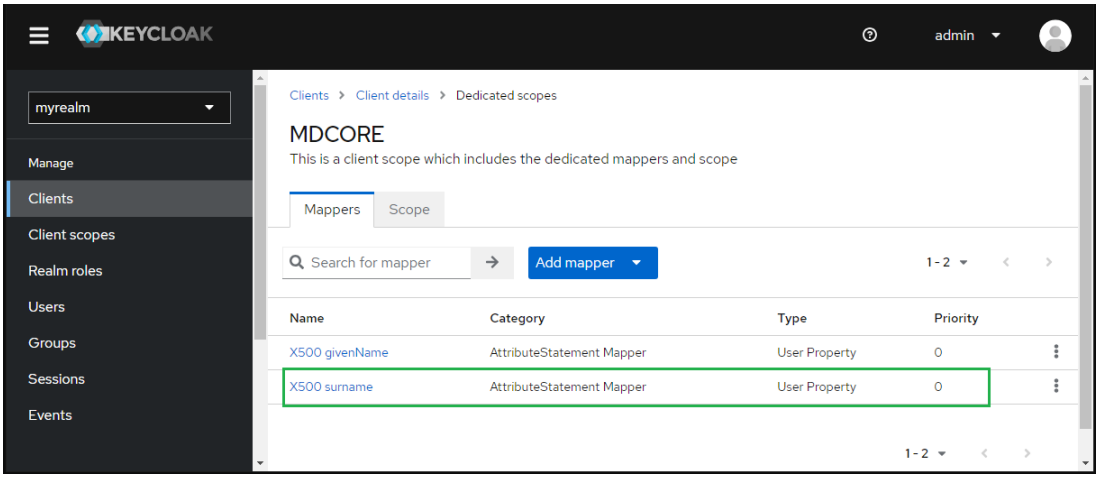
9. Back to `Mappers` tabs, click on `X500 givenName`.



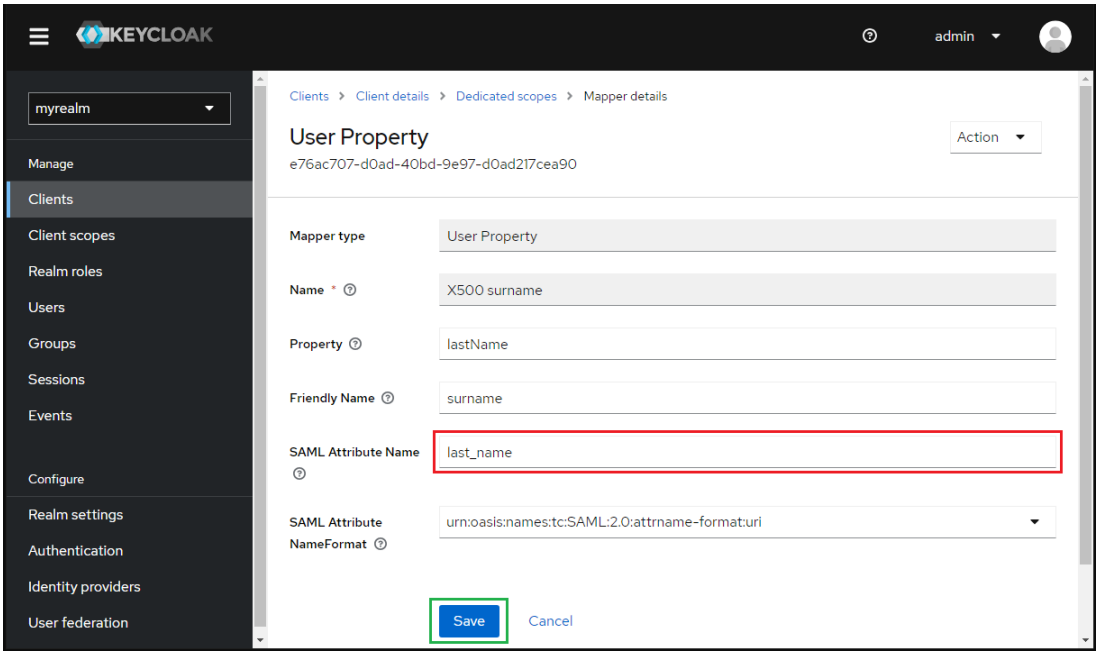
10. Enter first_name to `SAML Attribute Name` and click `Save`.



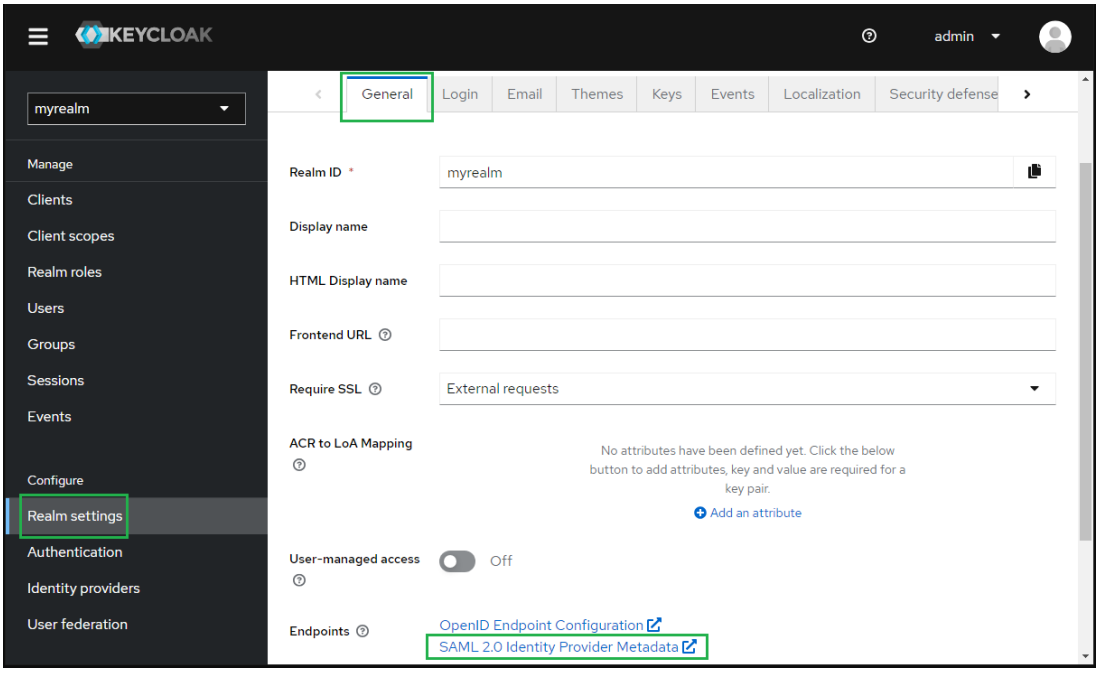
11. Click `X500 surname` in `Mappers` tab.



12. Enter last_name for `SAML Attribute Name` and click `Save`.

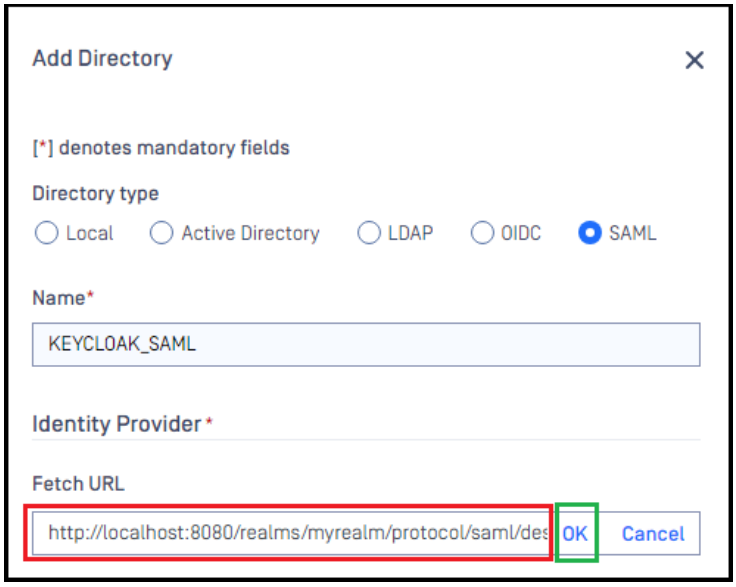


13. Select `Realm settings` in the sidebar, navigate to `General` tab, click `SAML 2.0 Identity Provider Metadata` and copy the metadata link.



Complete configuration in MetaDefender Core

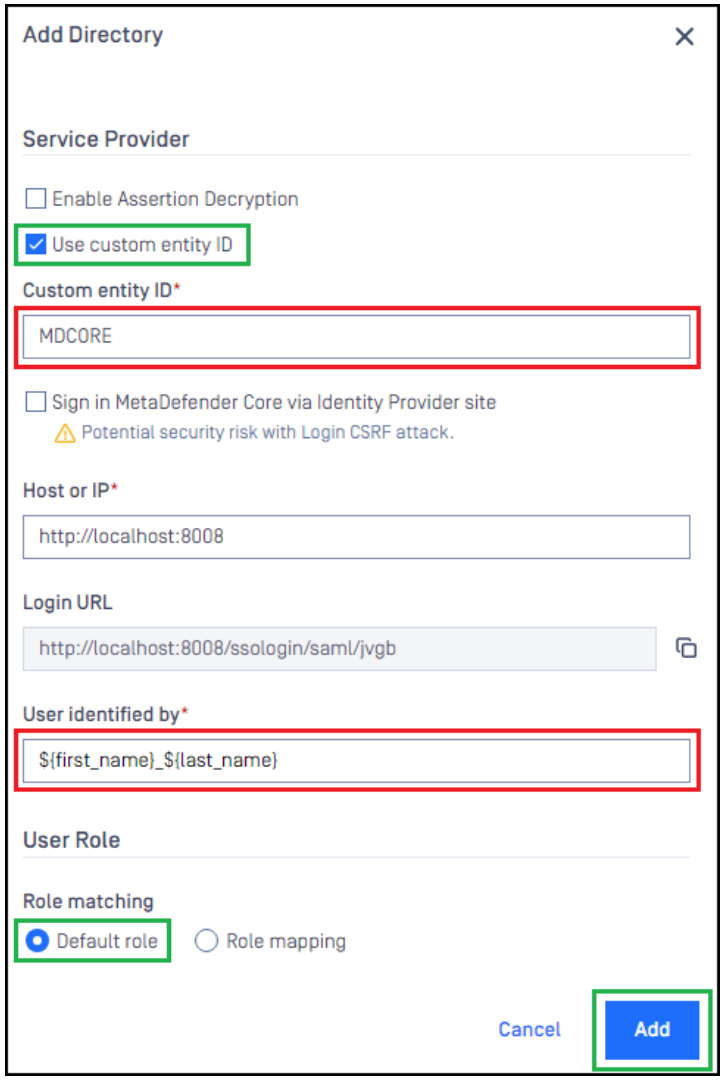
1. Switch to MetaDefender Core screen, under `Identity Provider`, click `Fetch URL`.
2. Paste the metadata link from Keycloak to the box under `Fetch URL` and click `OK` to ensure MetaDefender Core can set Keycloak as its IdP.



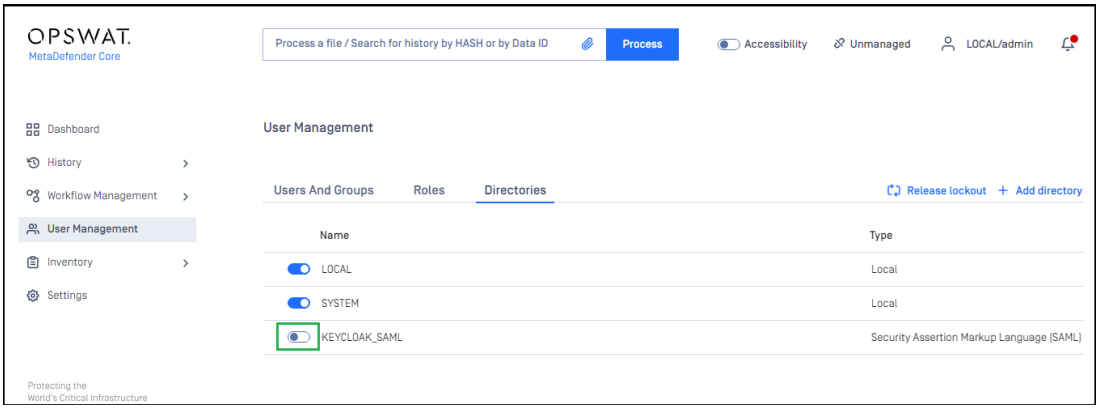
3. Under `Service Provider` section, enable `Use Custom Entity ID` and enter MDCORE in `Custom Entity ID` field.

> **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.

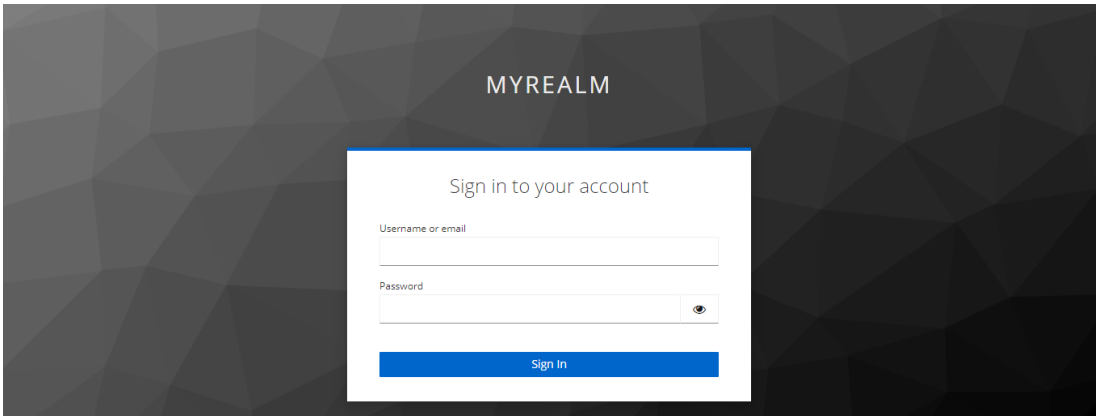


7. On `User Management` screen, toggle the new directory, *KEYCLOAK-SAML* in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all existing sessions will expire immediately.

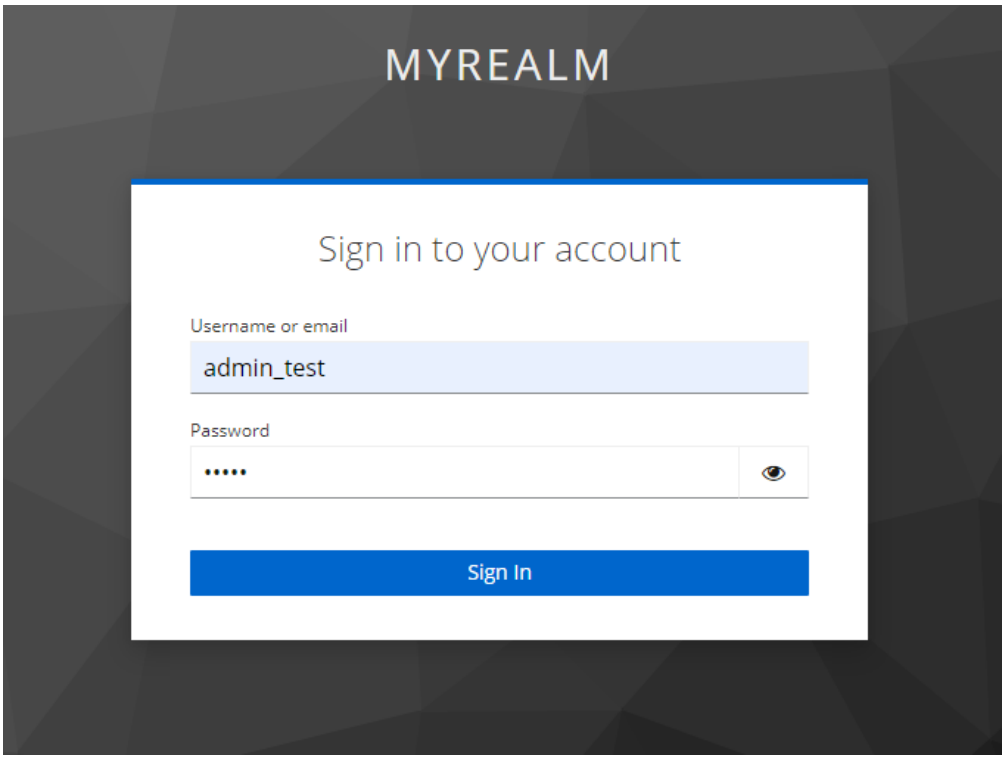


Test the integration

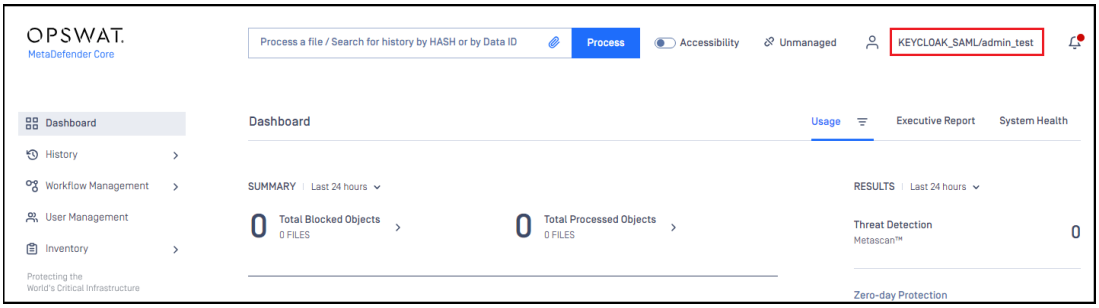
1. Click `Login` from the home screen of MetaDefender Core; the user is redirected to Keycloak page.



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.

Microsoft Active Directory Federation Services (ADFS)

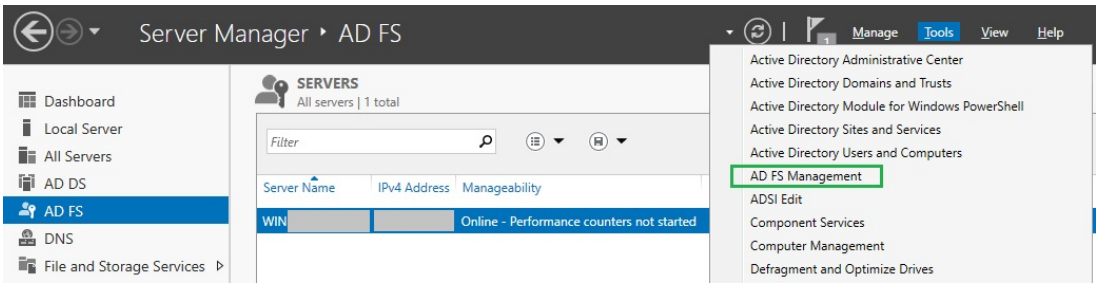
The instructions guide users on how to configure Microsoft Active Directory Federation Services SAML and OIDC Single Sign-on for MetaDefender Core.

Configuring SAML Single Sign-on

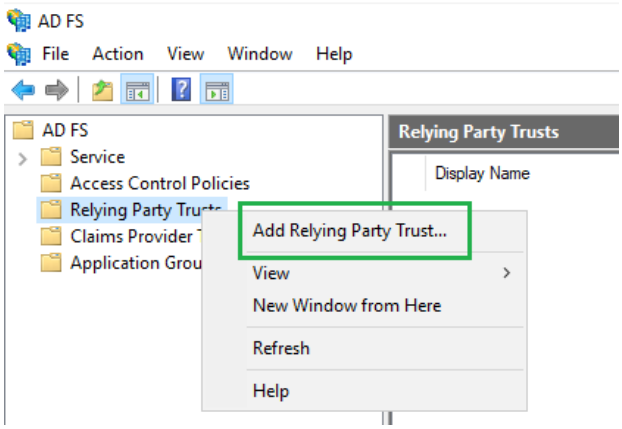
> **Info**
> ADFS requires MetaDefender Core to enable HTTPS for connections. Please follow these steps to first enable HTTPS on MetaDefender Core.

Add Relying Party Trust for ADFS

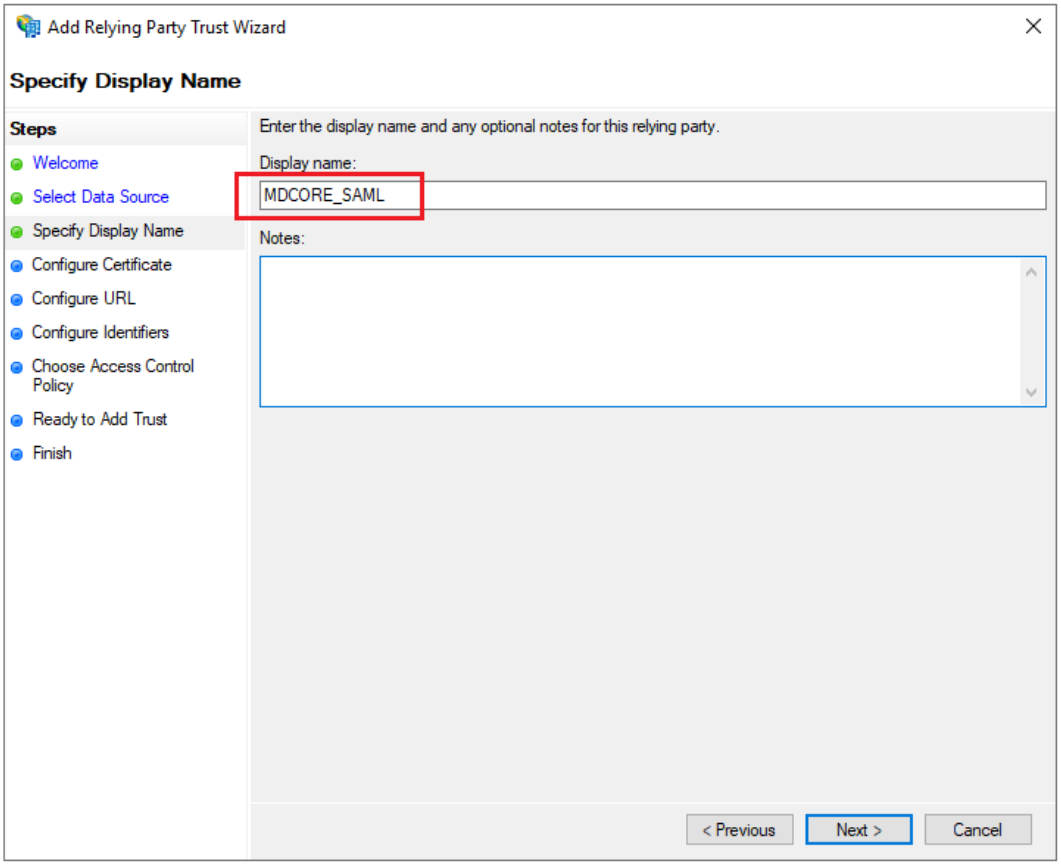
1. In the top right corner of `Server Manager`, click `Tools` and select `AD FS Management` from the drop-down list.



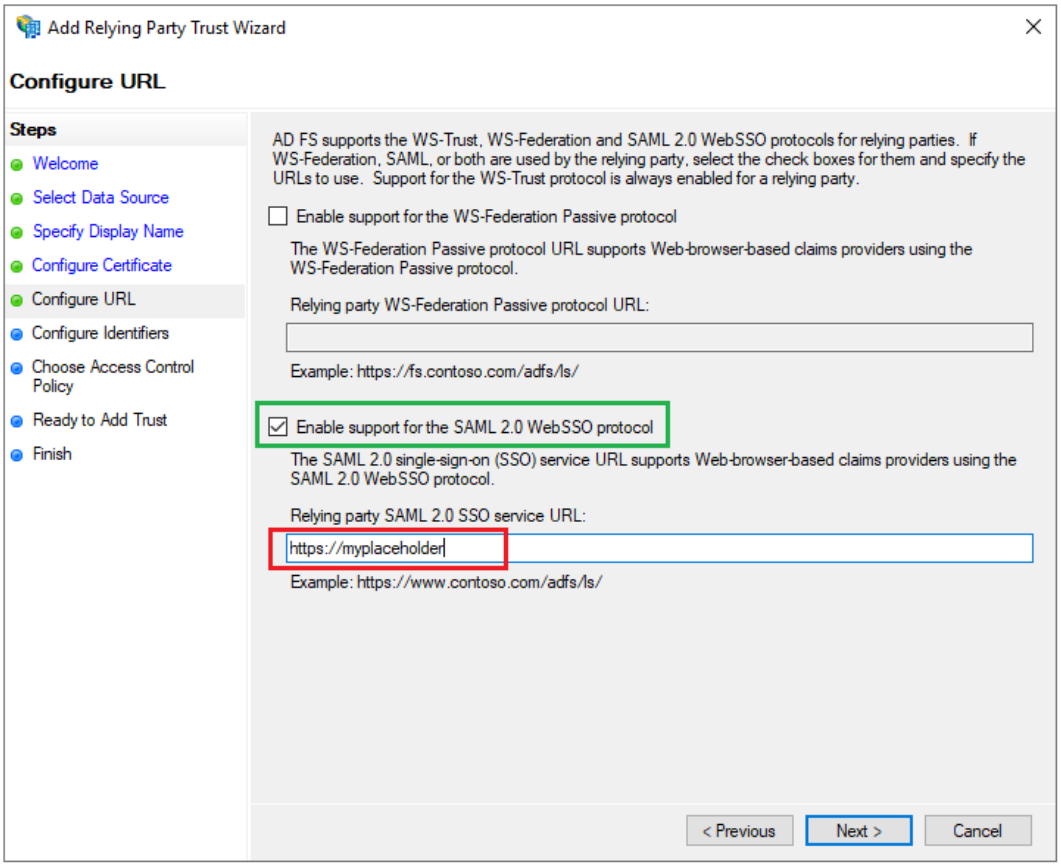
2. In `AD FS` window, right click on `Relying Party Trusts` and select `Add Relying Party Trust` from the drop-down menu.



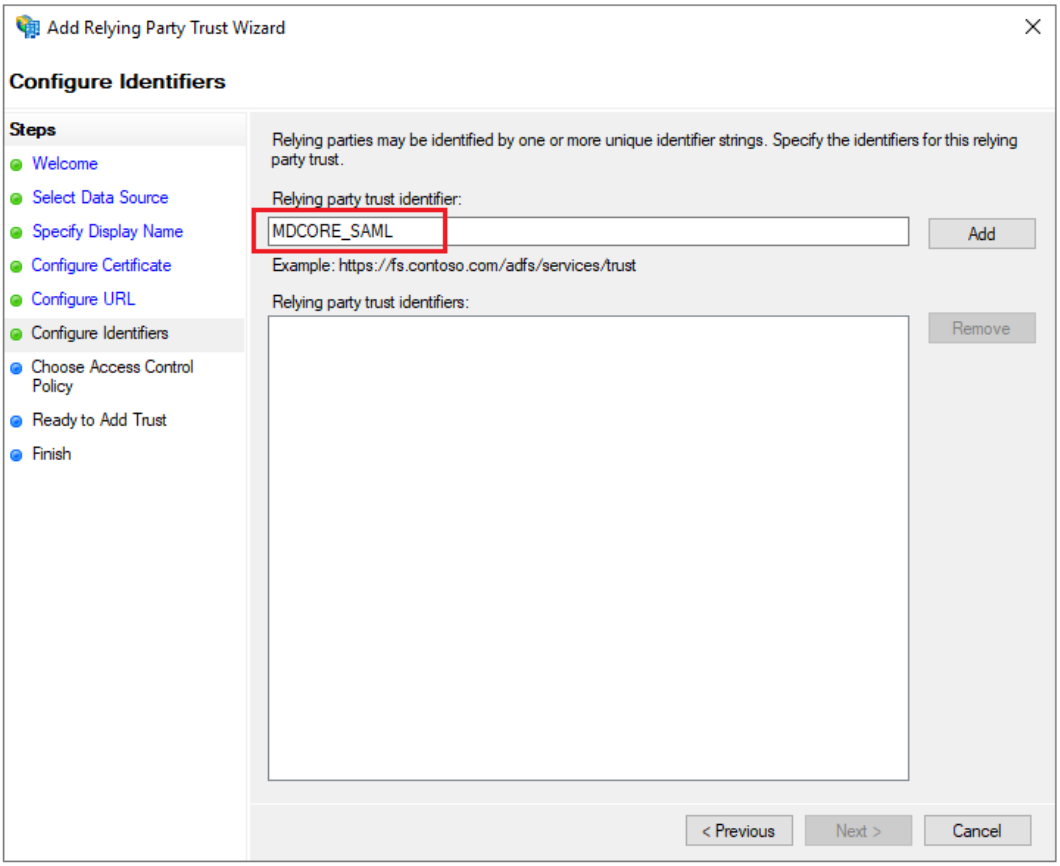
3. In `Add Relying Party Trust Wizard`, select `Claims aware` on the `Welcome` page and click `Start`.
4. On `Select Data Source` page, choose `Enter data about the relying party manually` and click `Next`.
5. On `Specify Display Name` page, provide a descriptive name for your relying party, e.g. MDCORE_SAML, and click `Next`.



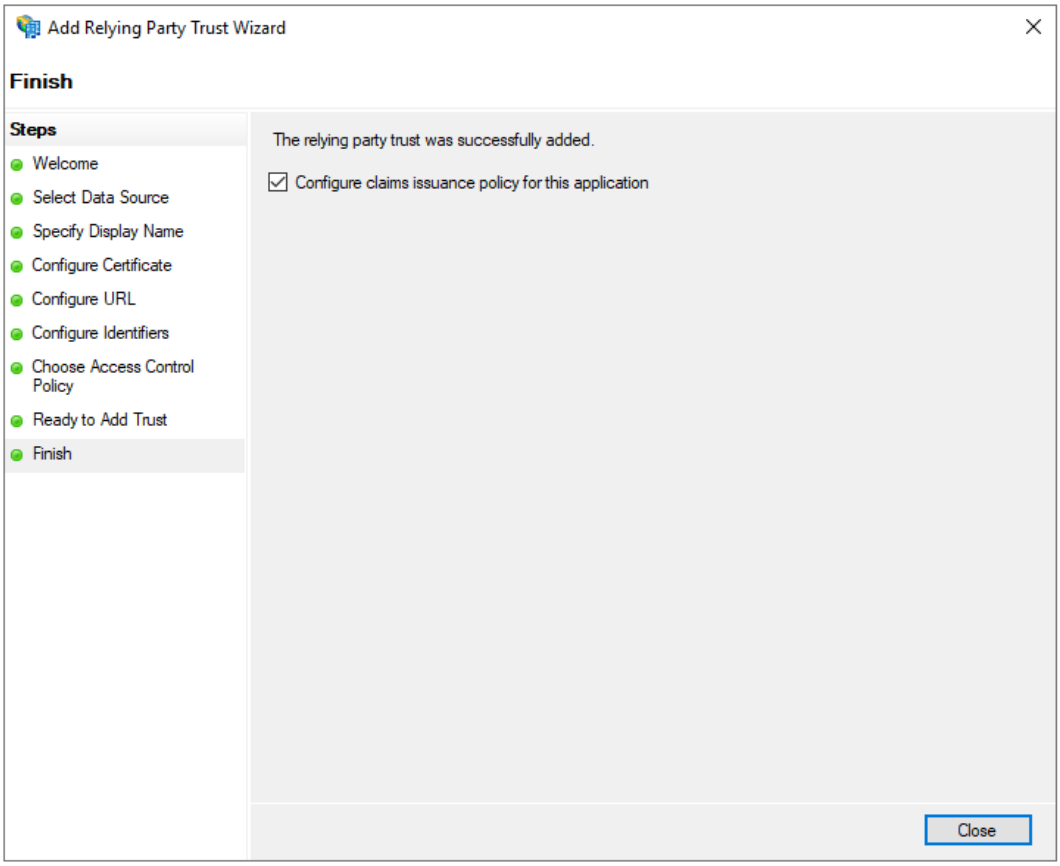
6. Click `Next` on `Configure Certificate` page.
7. On `Configure URL` page, check the box for `Enable support for the SAML 2.0 WebSSO protocol` and set a placeholder, using https://myplaceholder as an example, for `Relying party SAML 2.0 SSO service URL`.



8. On `Configure Identifiers` page, set MDCORE_SAML as `Relying party trust identifier`, then click `Add` and `Next`.

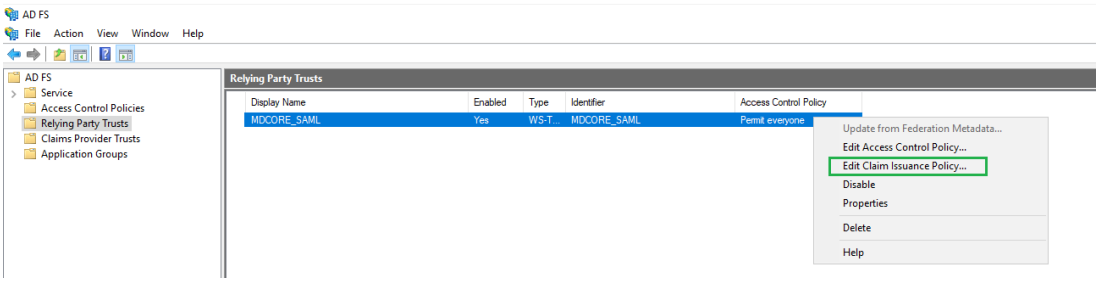


9. On `Choose Access Control Policy` page, select `Permit everyone` and click `Next`.
10. Click `Next` on `Ready to Add Trust` page to save your information.
11. If everything goes well, `Finish` page will be displayed for you to click `Close`.

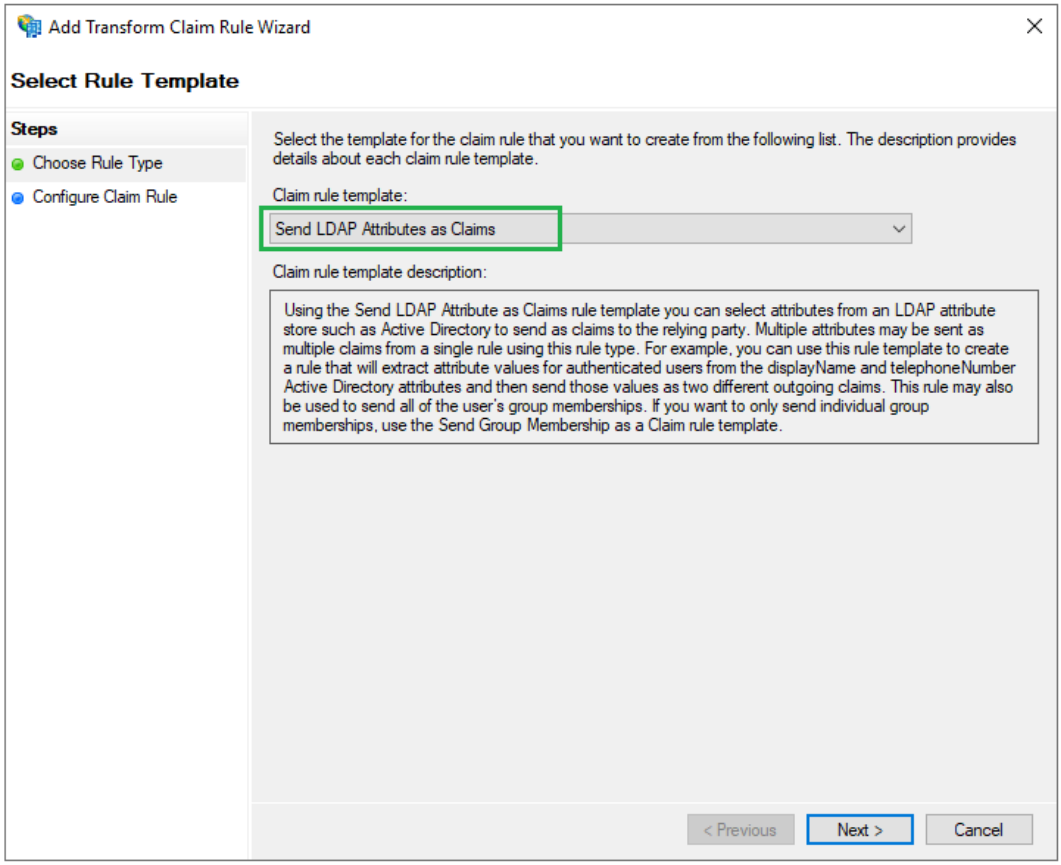


Configure claims Issuance Policy for ADFS

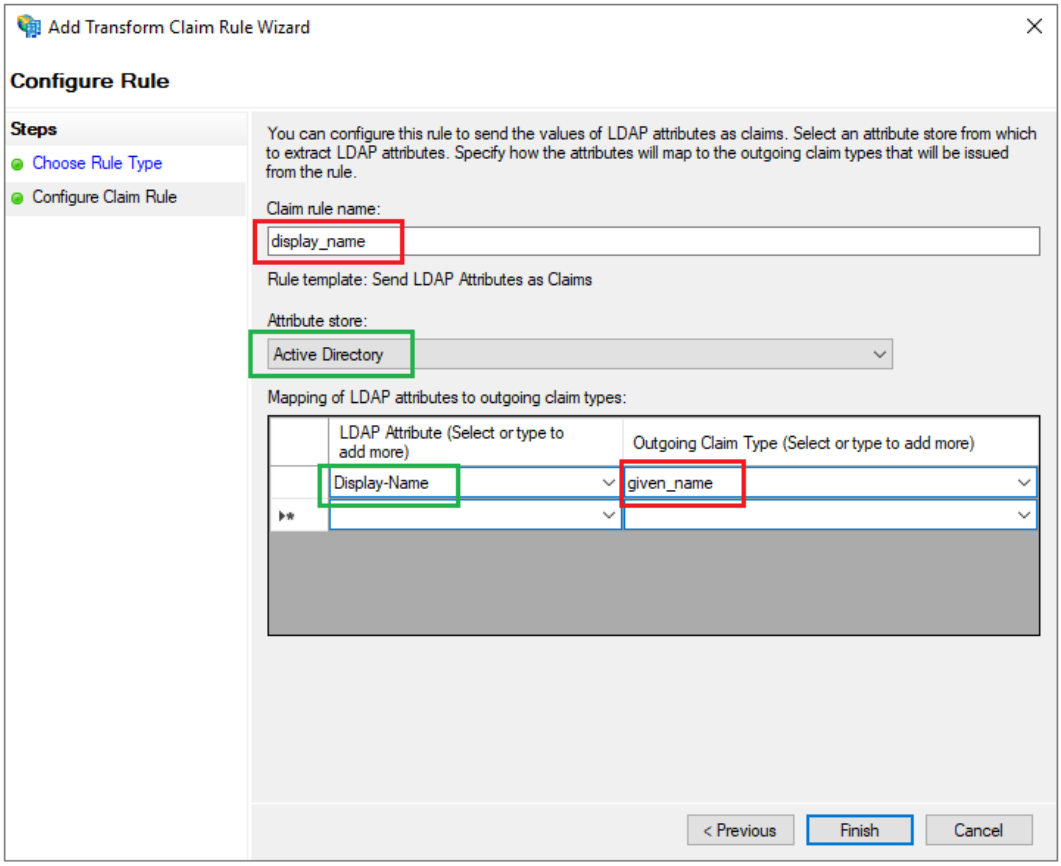
1. In `Relying Party Trusts` table of `AD FS` window, navigate to the item MDCORE_SAML that was added in the previous step, right-click on it, and select `Edit Claim Issuance Policy...` from the drop-down menu.



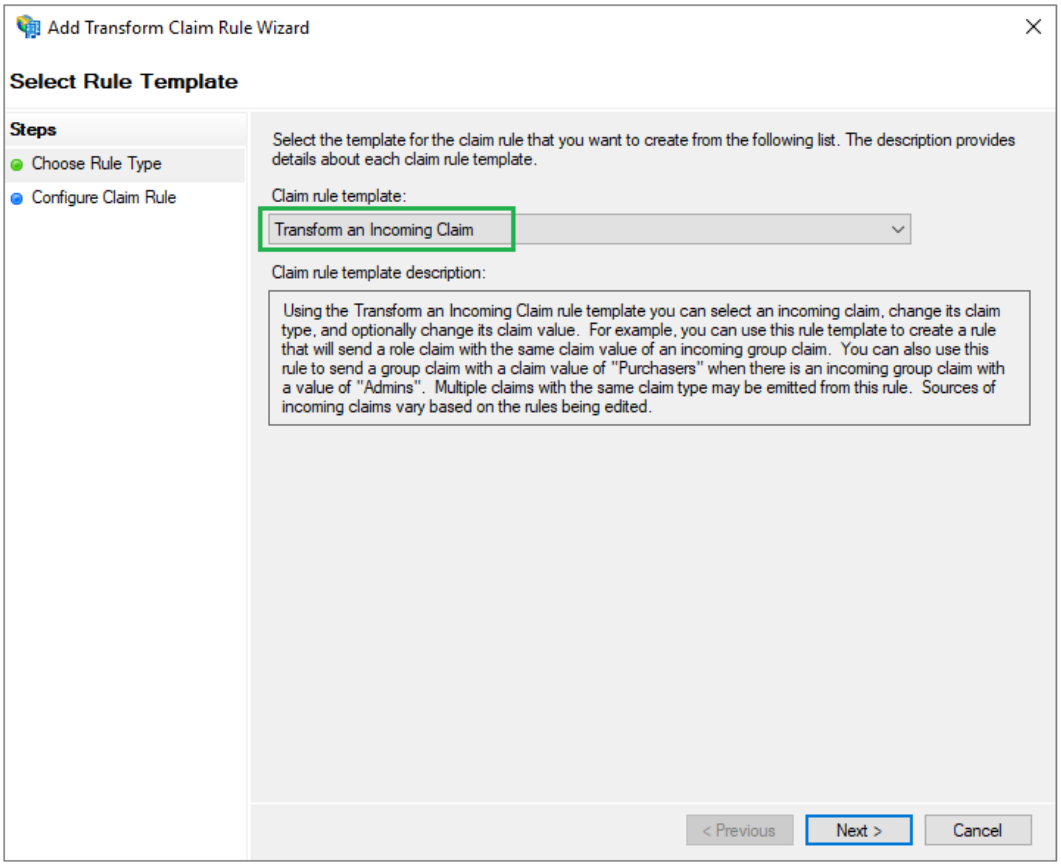
2. Click `Add Rule`.
3. Select `Send LDAP Attributes as Claims` for `Claim rule template`, and click `Next`.



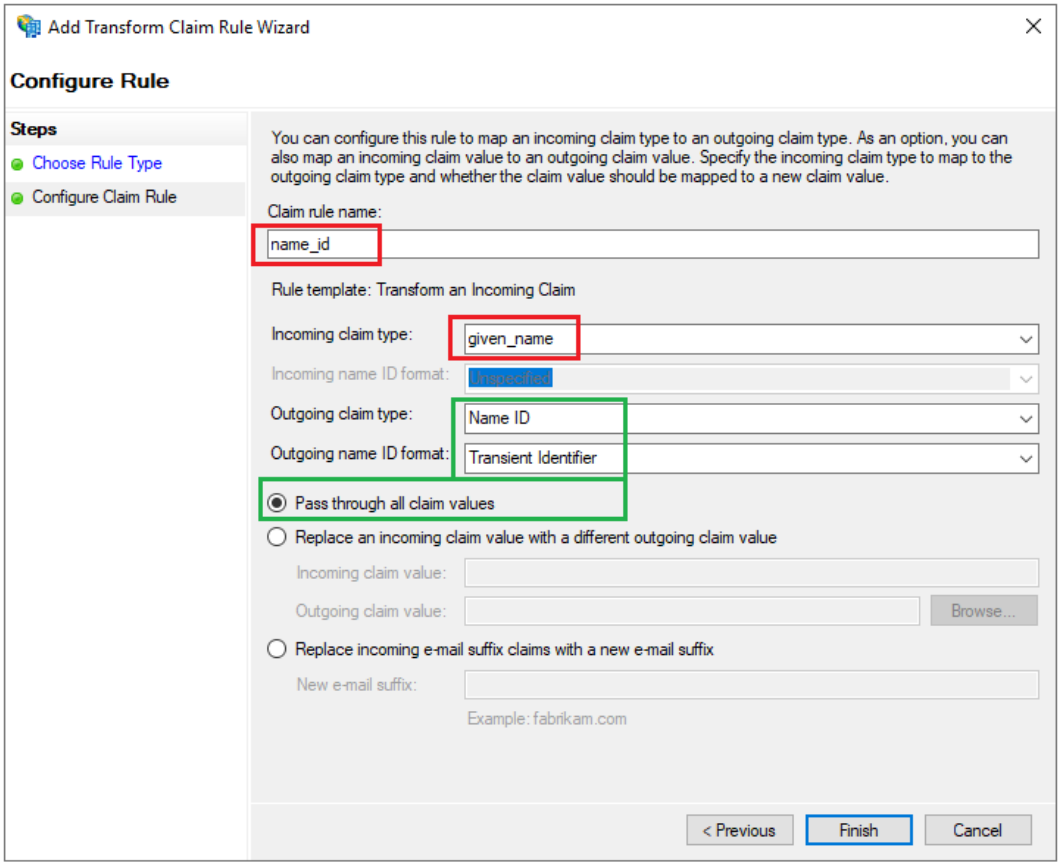
4. Enter a name for `Claim rule name`, for instance, display_name. Select `Active Directory` for `Attribute Store`. Under `LDAP Attribute` column, choose `Display-Name` from the drop-down list, type given_name for the `Outgoing Claim Type` column, and click `Finish`.



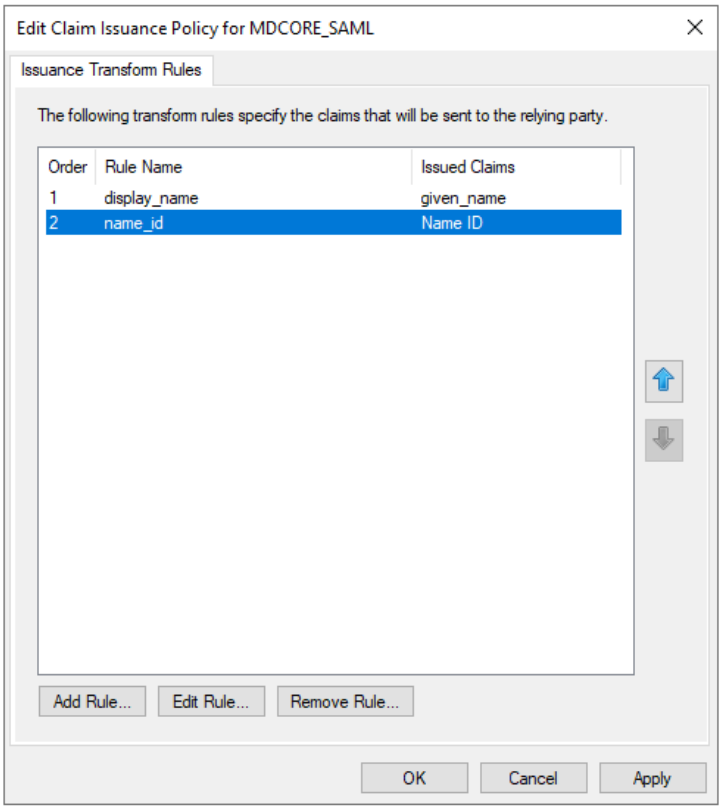
5. Click `Add Rule` once more. At `Choose Rule Type` option, select `Transform an Incoming Claim` for `Claim rule template` and click `Next`.



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`.

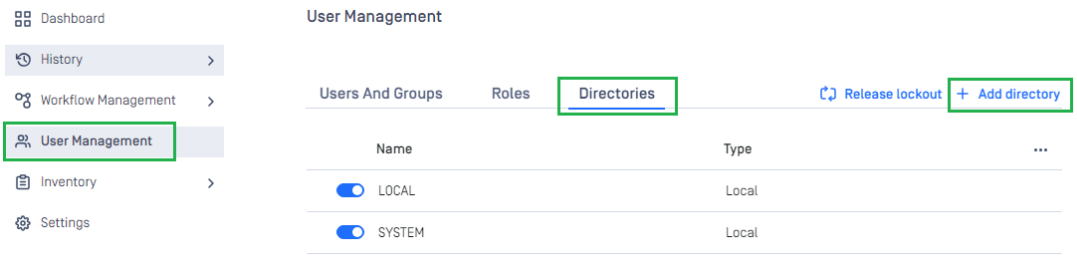


7. Click `Apply` on the next page and `OK` to close the wizard.

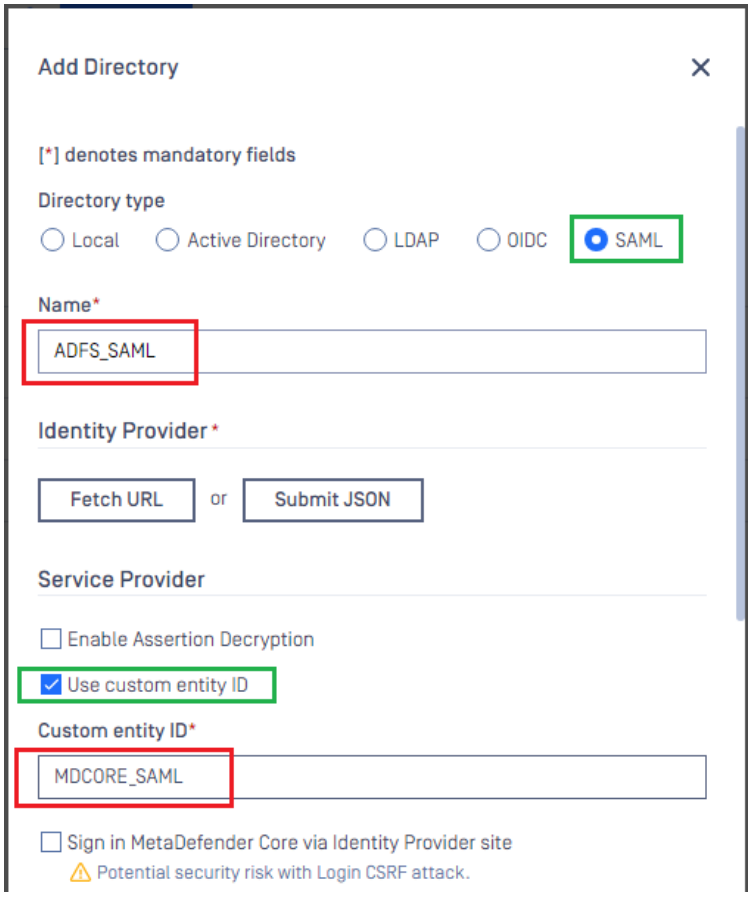


Create a SAML directory in MetaDefender Core

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.

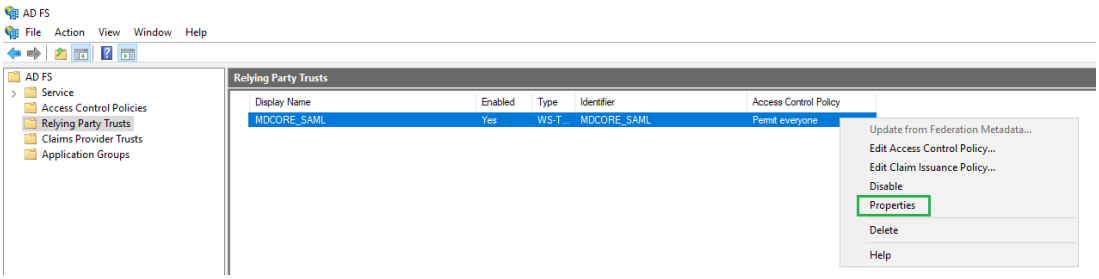


7. In `Service Provider` section, fill in `Host or IP` filed with the address where MetaDefender Core is hosted , using https://localhost:8008 as an example.
8. Copy `Login URL`.

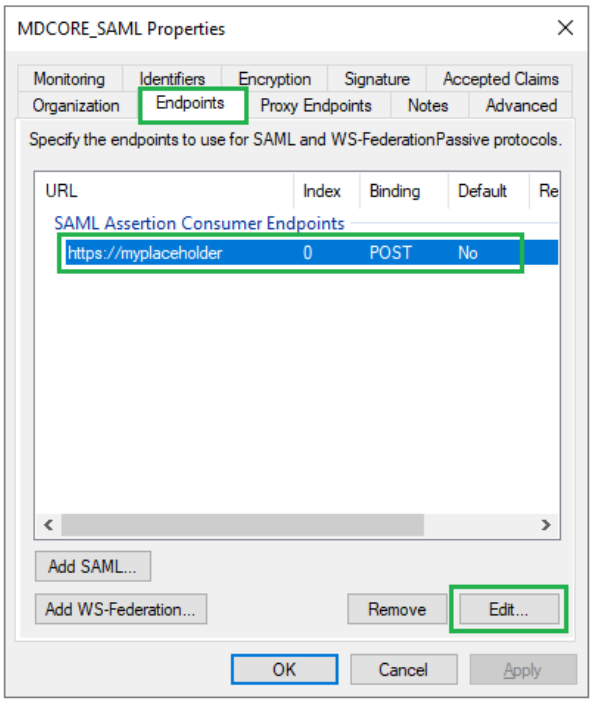


Modify Replying Party Trust on ADFS

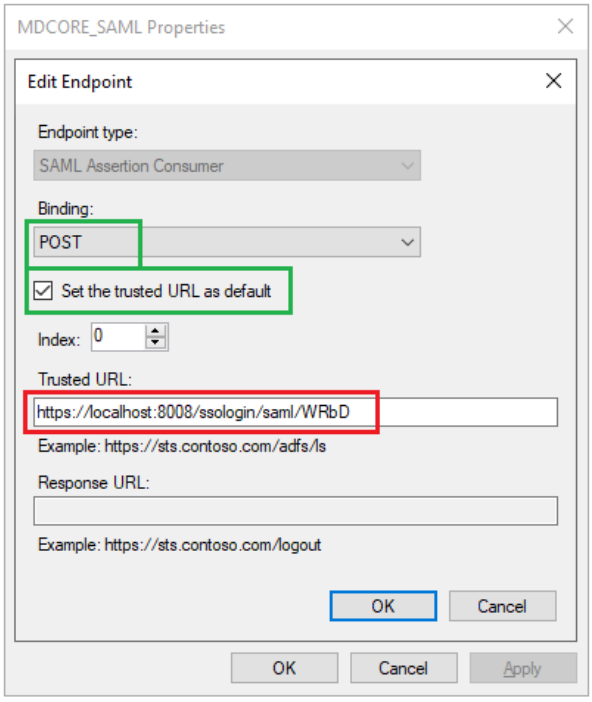
1. Access `AD FS Management` tool from `Server Manager`, select `Relying Party Trusts`, right click on the item MDCORE_SAML, and choose `Properties`.



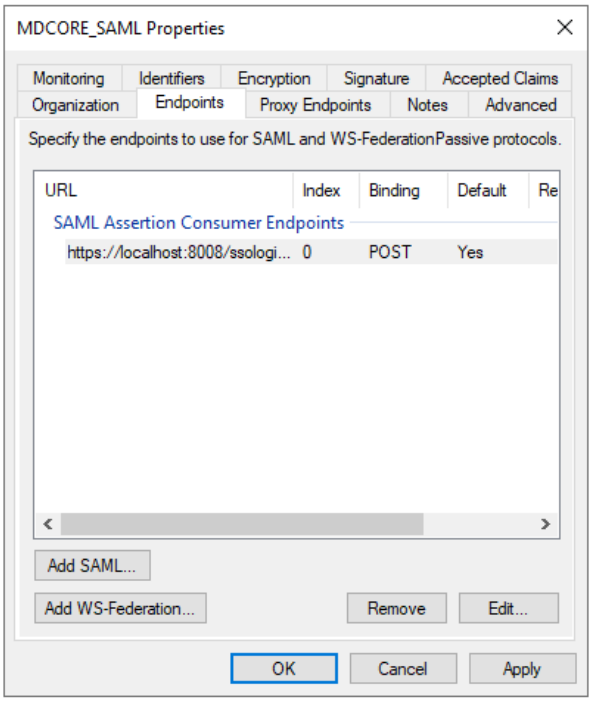
2. Navigate to `Endpoints` tab, select https://myplaceholder, and click `Edit`.



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`.

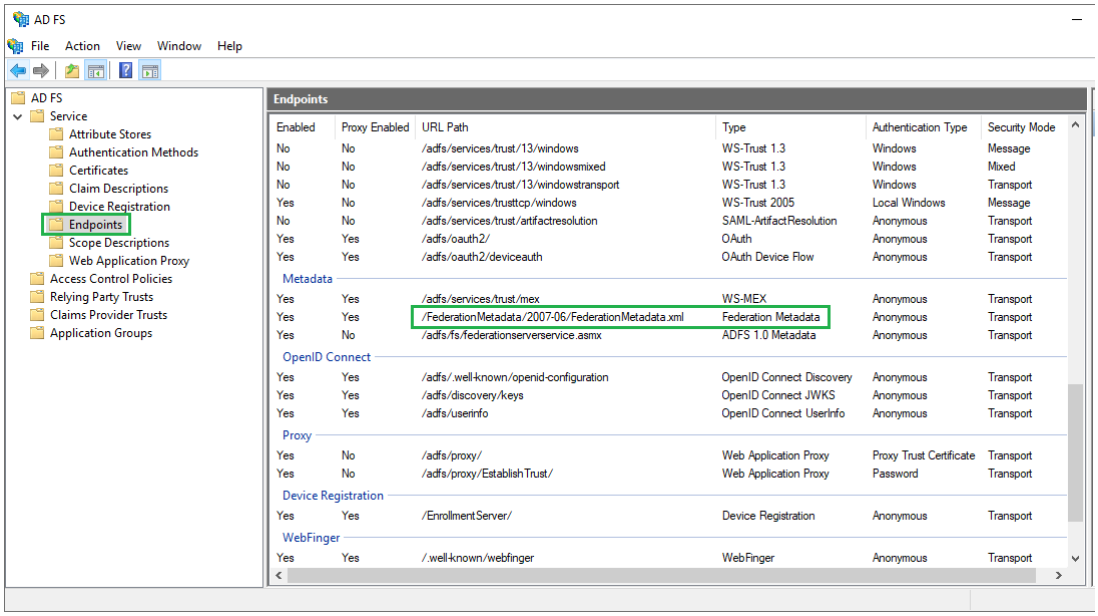


4. Click `Apply` and `OK` to complete the Relying Party Trust modification.



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.

html

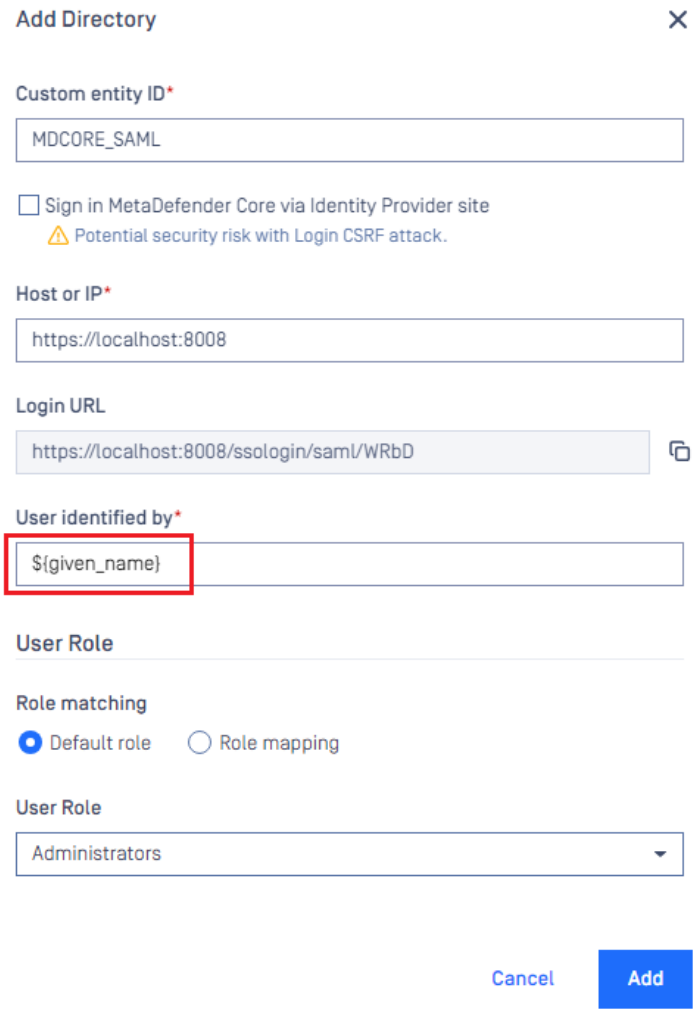
```
https://<adfs_fully_qualified_domain_name>/<metadata_endpoint_path>
```

- For example:

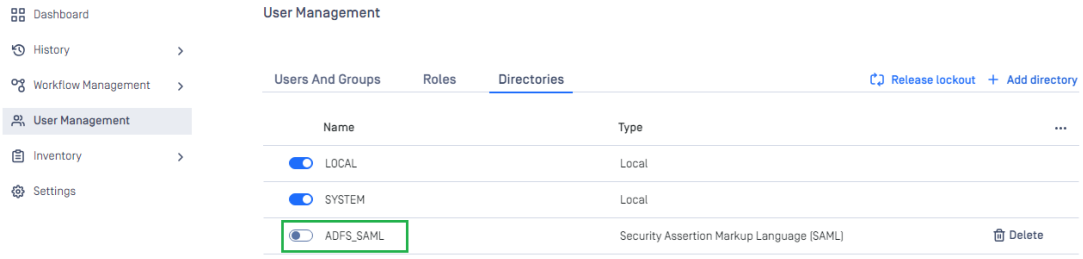
html

```
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.

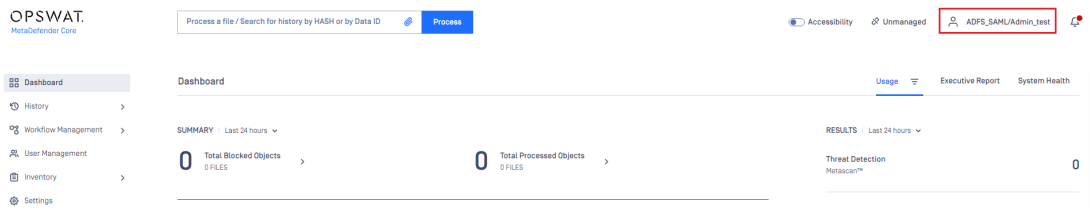


Test the integration

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.



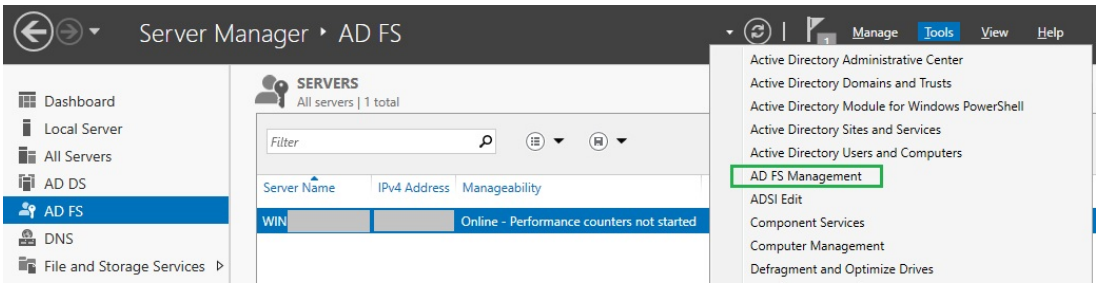
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Configuring OIDC Single Sign-on

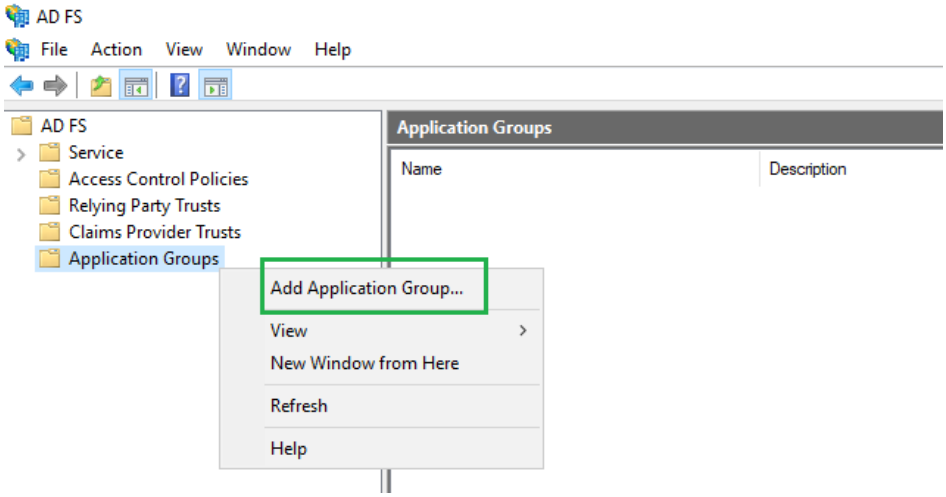
> **Info**
> ADFS requires MetaDefender Core to enable HTTPS for connections. Please follow these steps to first enable HTTPS on MetaDefender Core.

Set up OIDC in ADFS

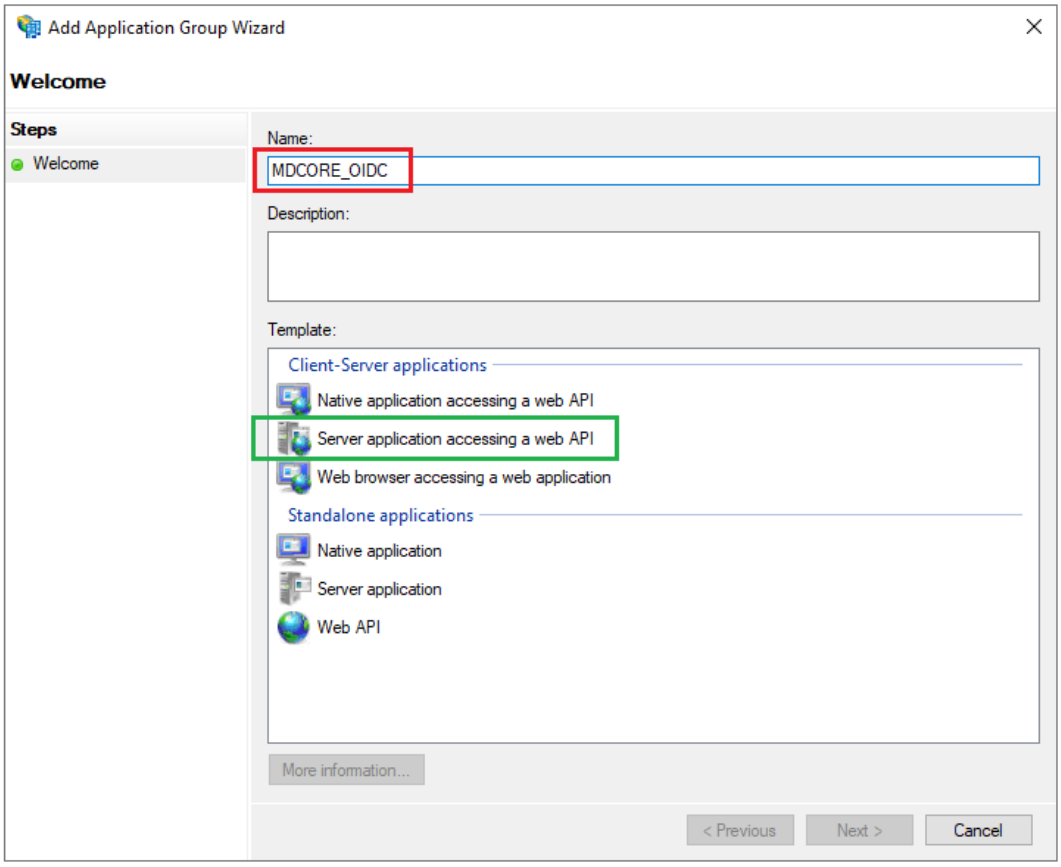
1. In the top right corner of `Server Manager`, click on `Tools` and select `AD FS Management` from the drop-down menu.



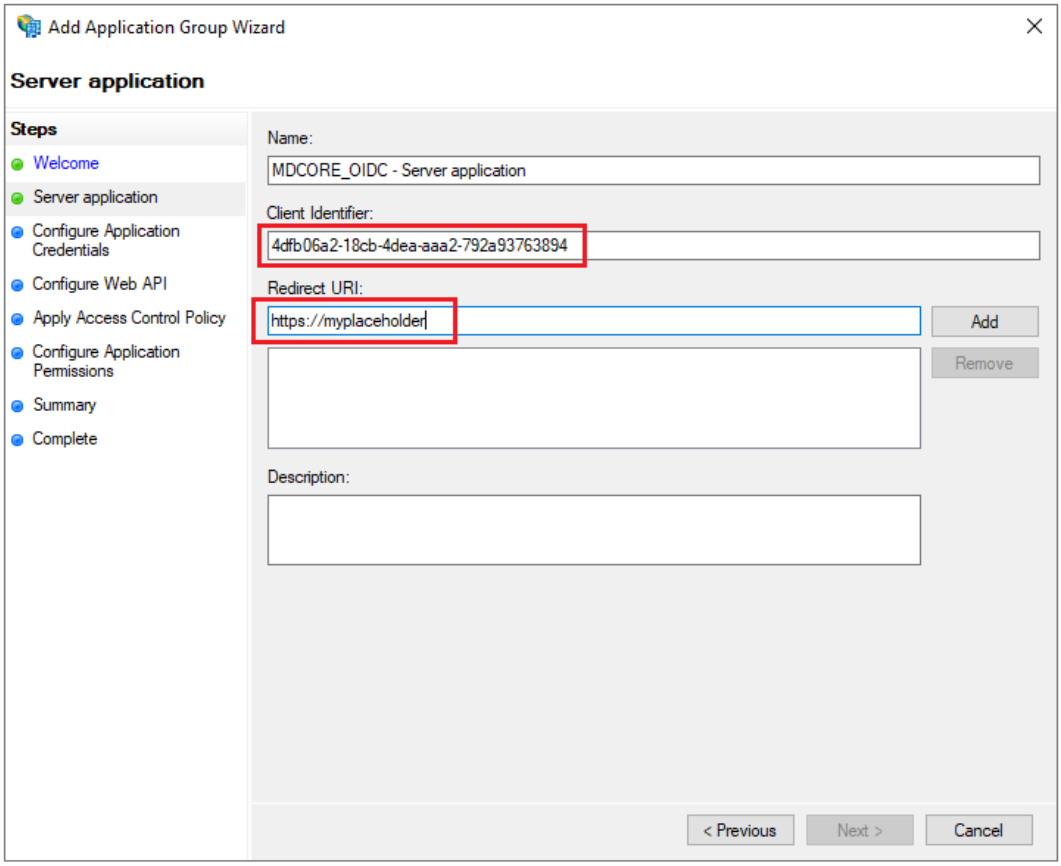
2. In `AD FS` window, right click on `Application Groups` and select `Add Application Group` from the drop-down menu.



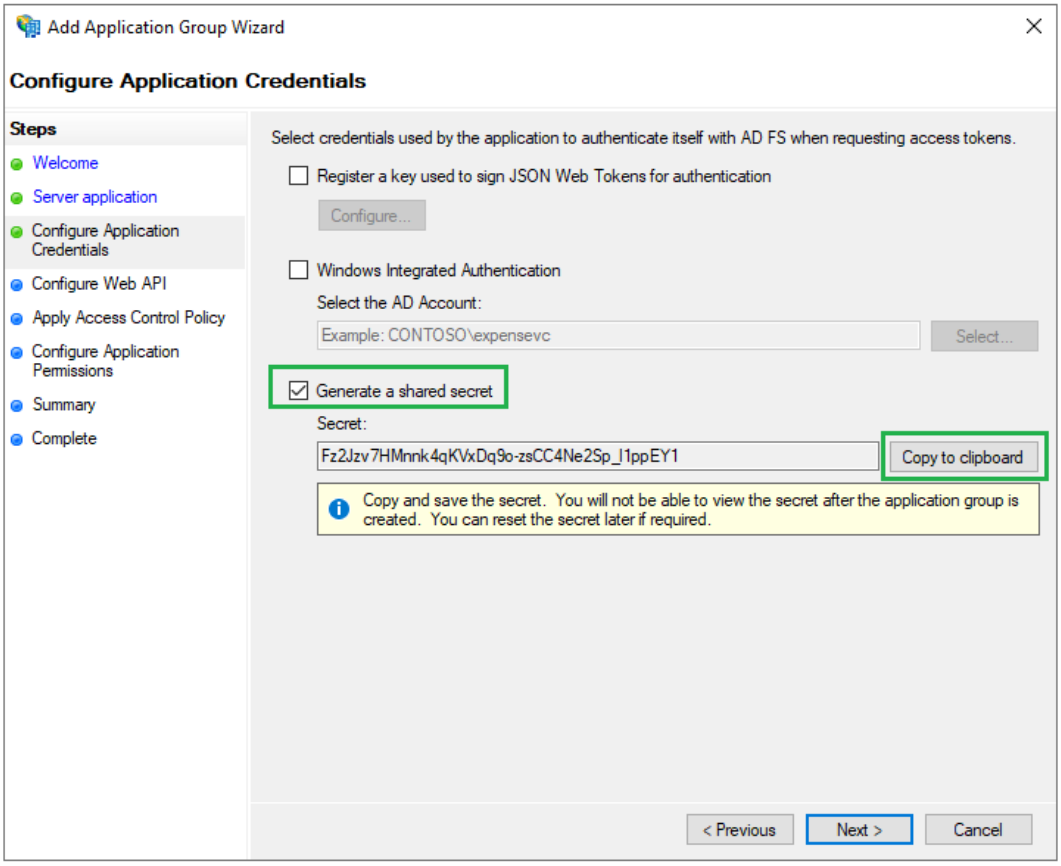
3. On `Welcome` page, provide a descriptive name for your application , e.g. MDCORE_OIDC, select `Server application accessing a web API` under `Template`, and click `Next`.



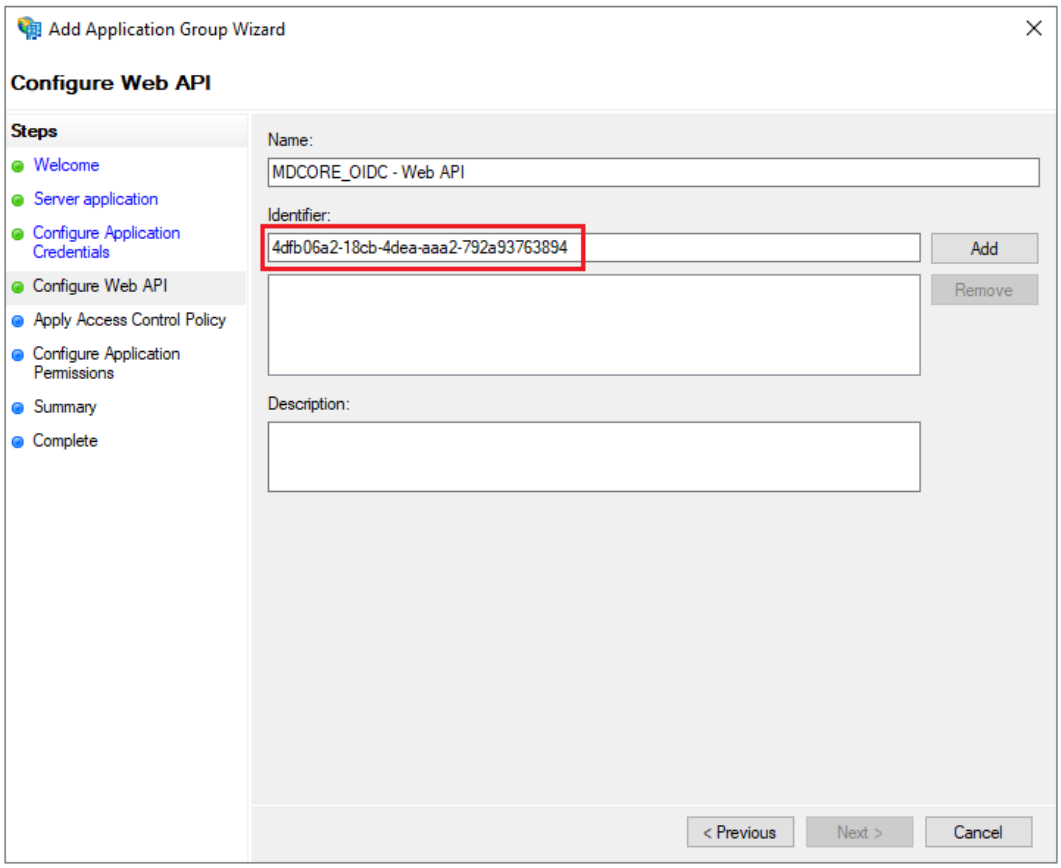
4. On `Server application` page, copy the string under `Client Identifier` for later use, set a placeholder https://myplaceholder for `Redirect URI`, then click `Add` and `Next`.



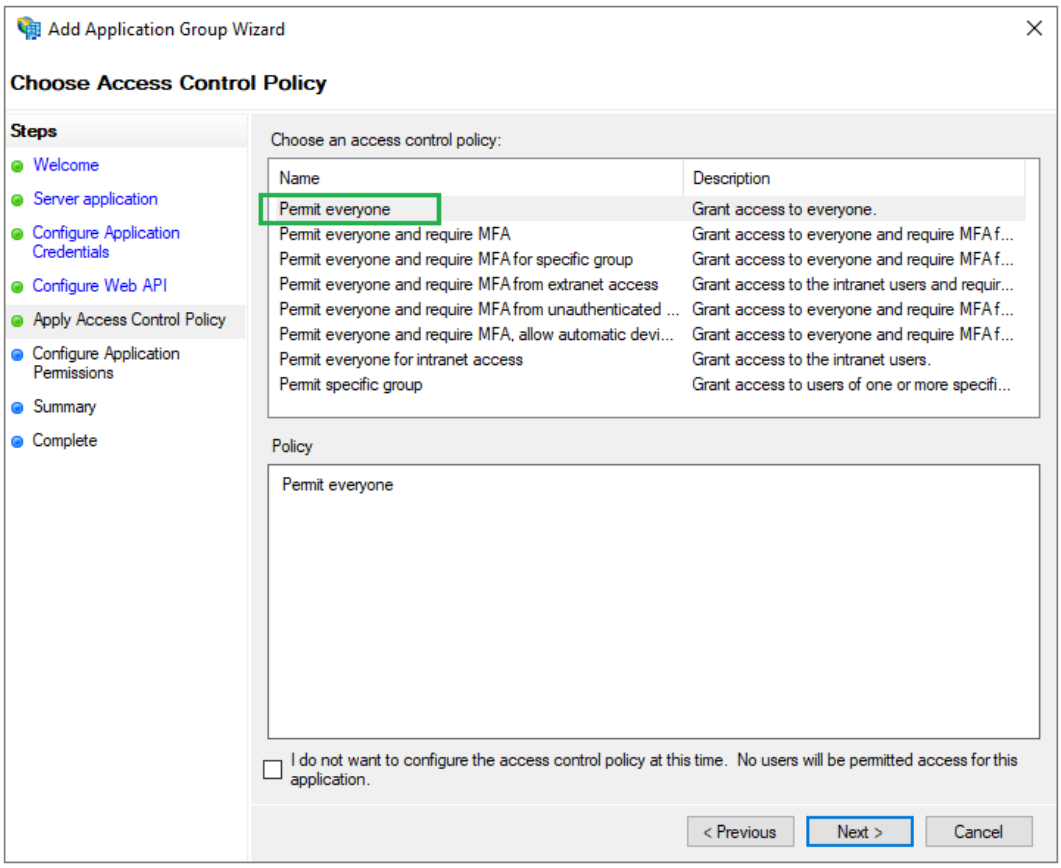
5. On `Configure Application Credentials` page, check `Generate a shared secret`, click `Copy to clipboard`, then save the secret string for later use and click `Next`.



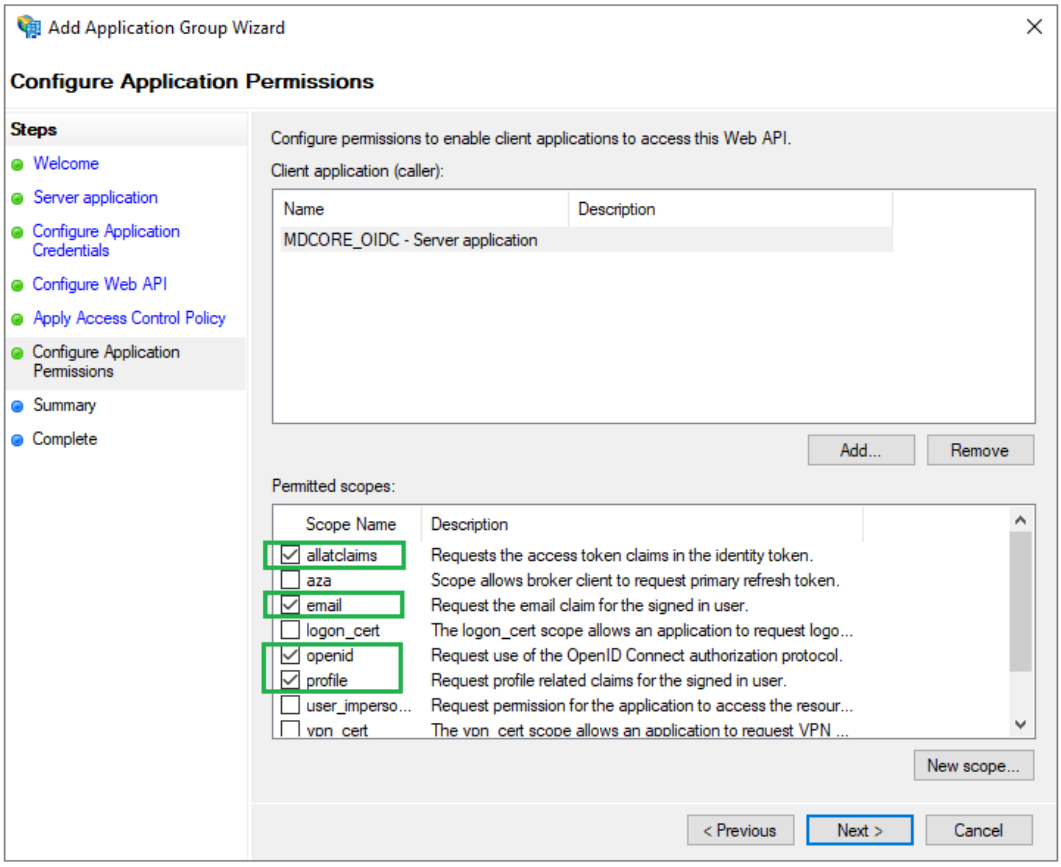
6. On `Configure Web API` page, enter the string of `Client Identifier` from step 4 into the box under `Identifier`, click `Add` , and `Next`.



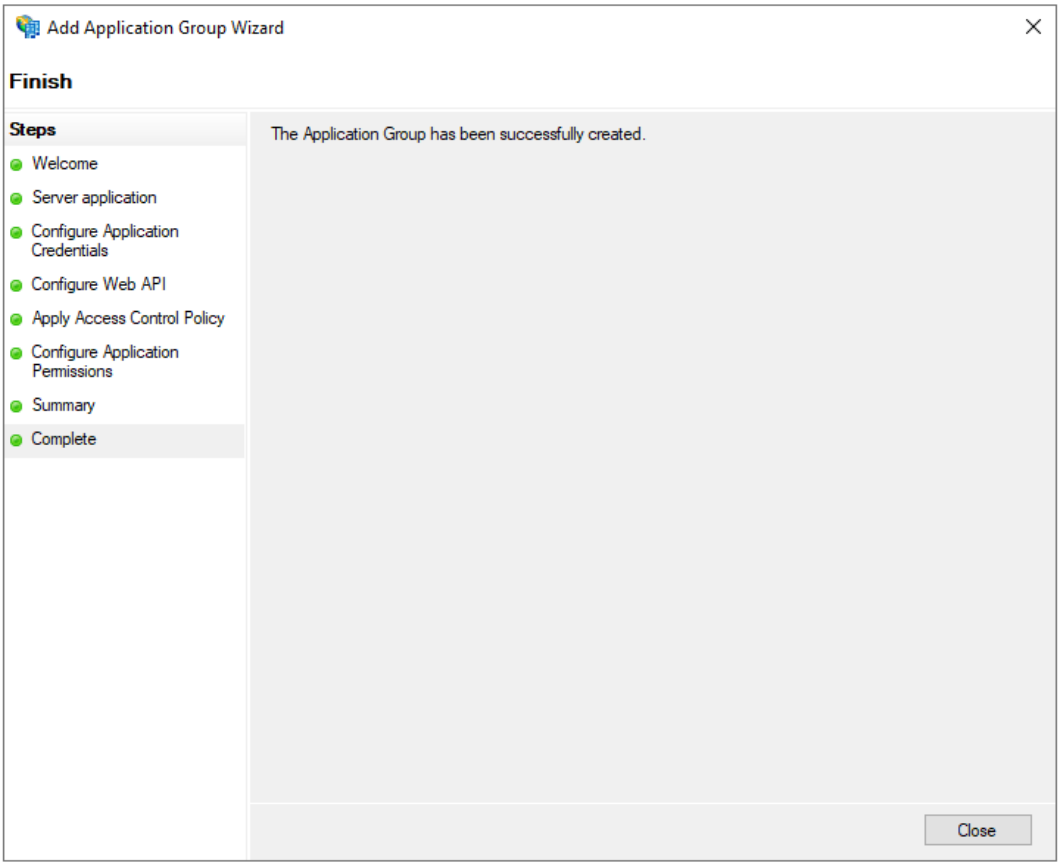
7. On `Choose Access Control Policy` page, select `Permit everyone` and click `Next`.



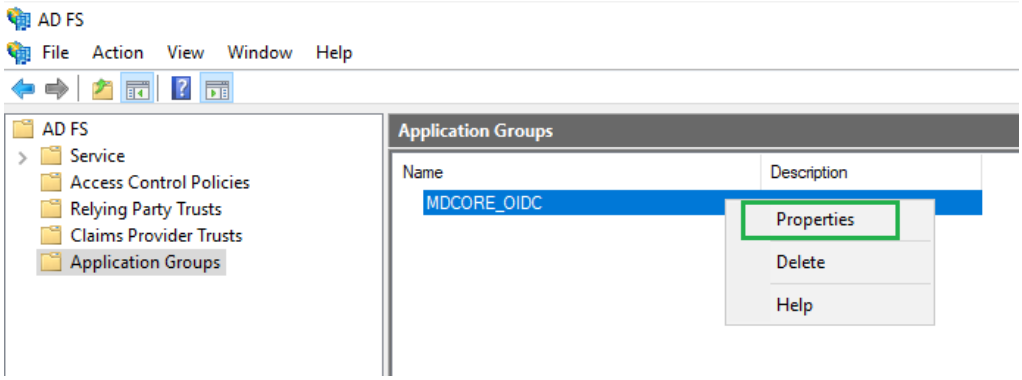
8. On `Configure Application Permissions` page, select the scopes `allatclaims`, `email`, `openid`, and `profile` under `Permitted scopes`, and then click `Next`.



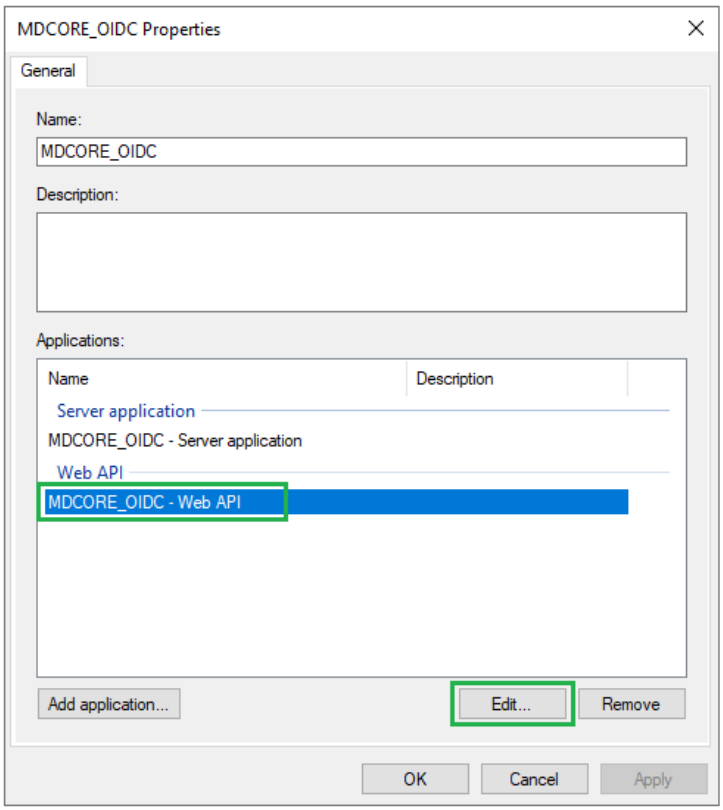
9. Click `Next` on the following page and `Close` to finish.



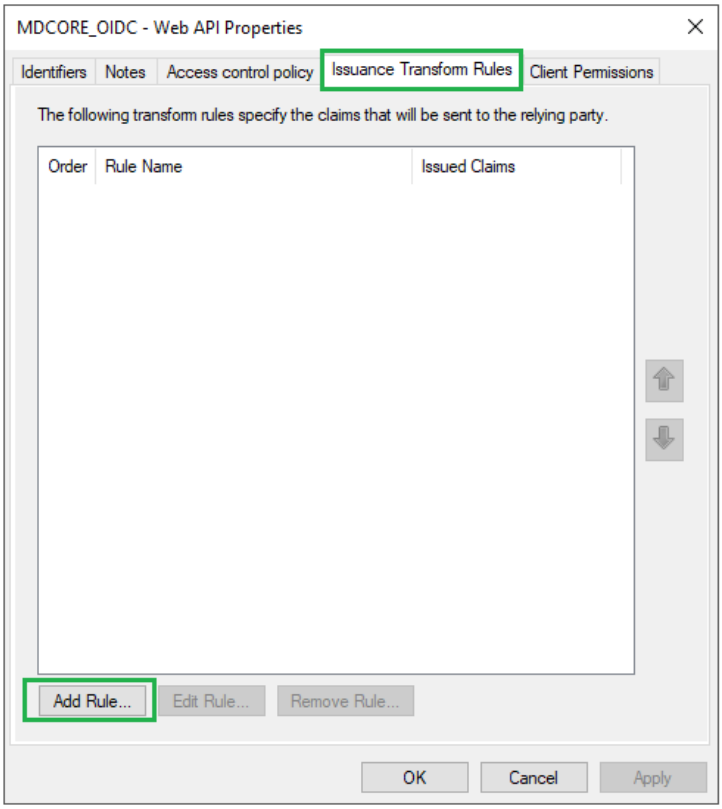
10. Back in `AD FS` window, right click on the item MDCORE_OIDC and select `Properties`.



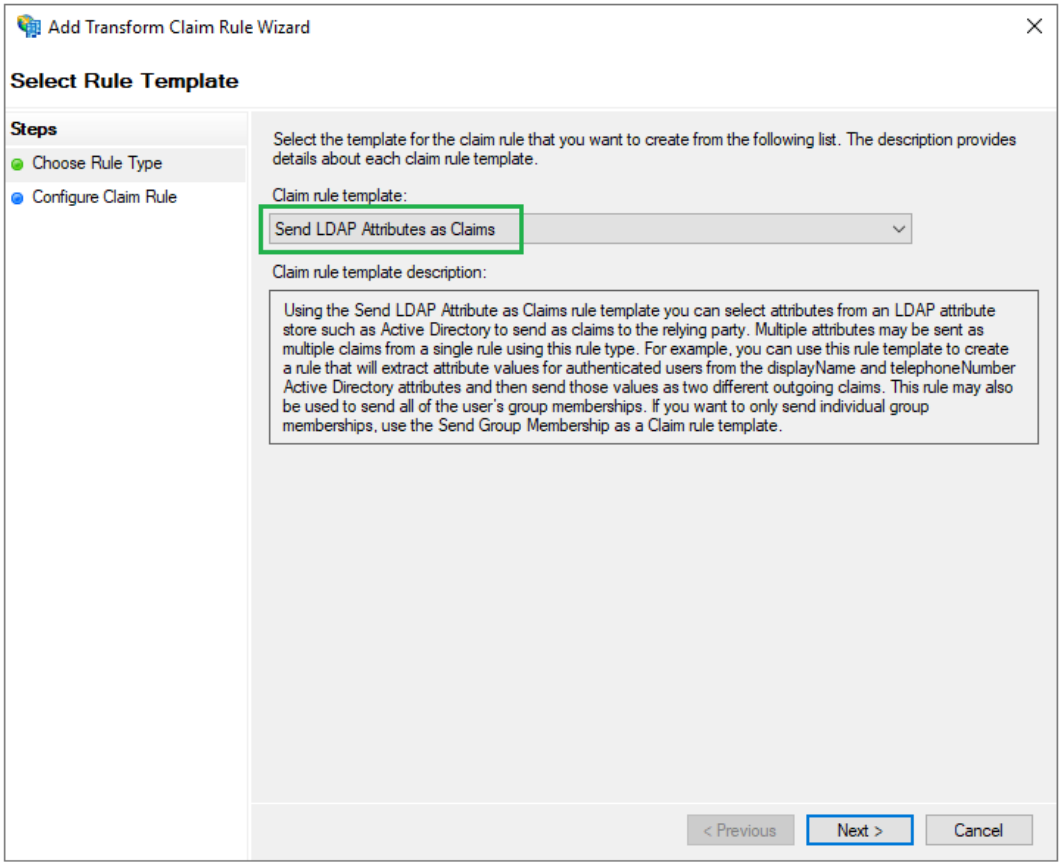
11. Select `MDCORE_OIDC - Web API` and click `Edit`.



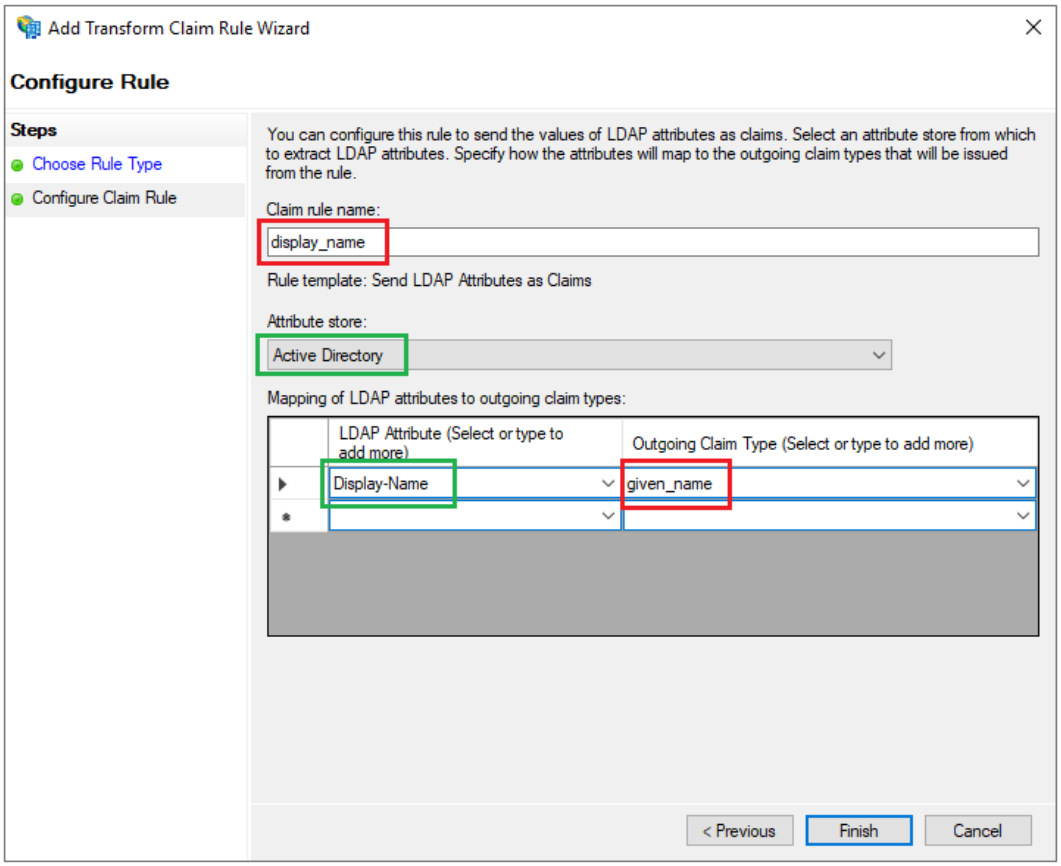
12. Go to `Issuance Transform Rules` tab and click `Add Rule`.



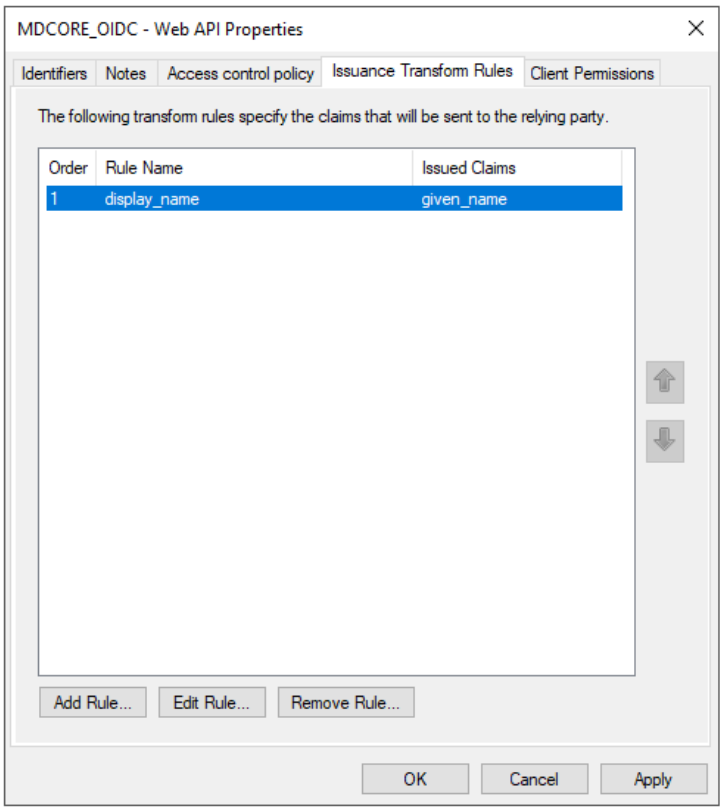
13. Select `Send LDAP Attributes as Claims` for `Claim rule template` and click `Next`.



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`.

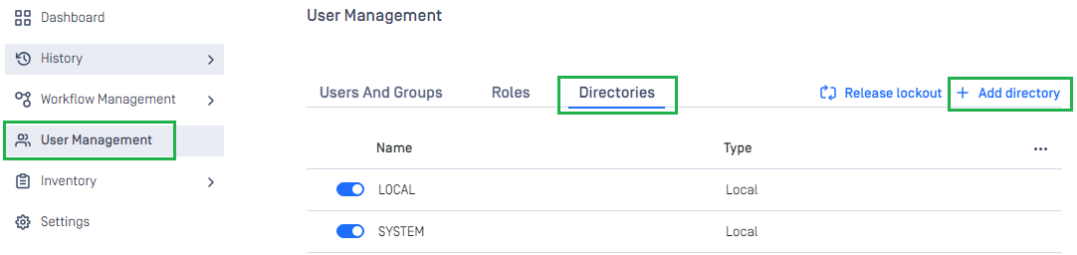


15. Click `Apply` on the next page and `OK` to close the wizard.

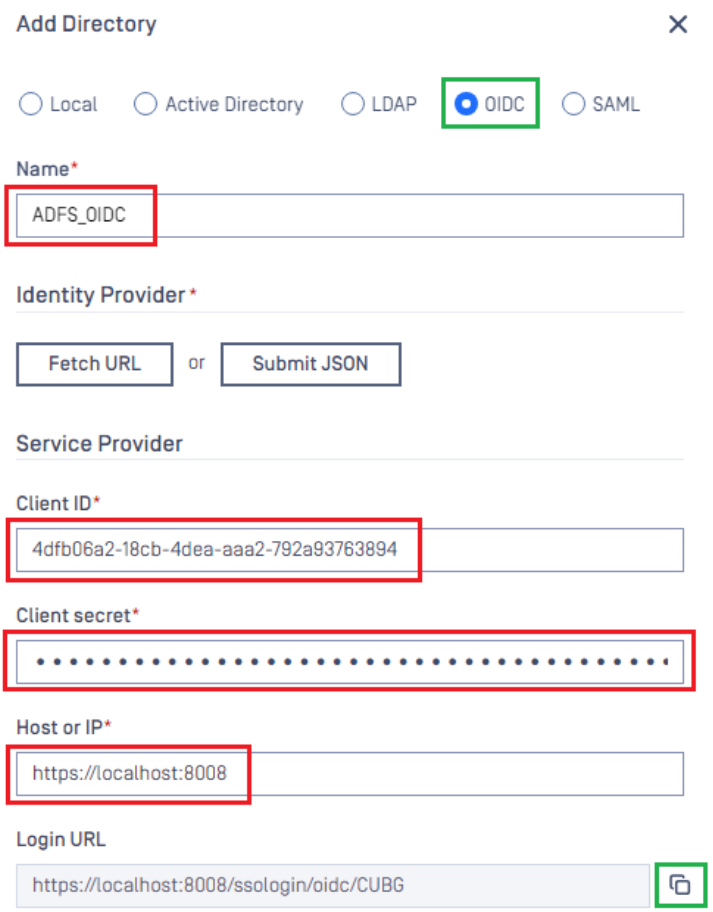


Create OIDC directory in MetaDefender Core

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.

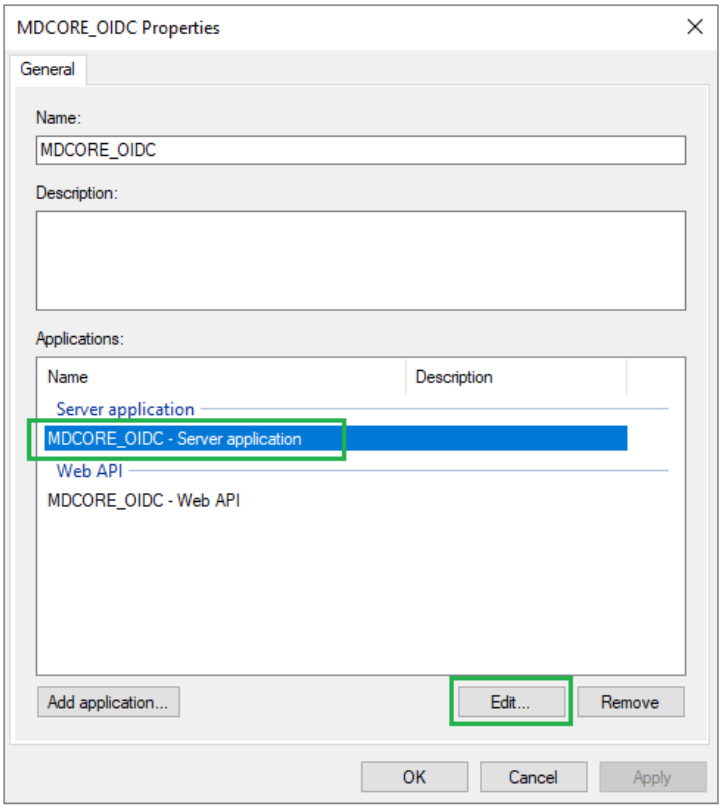


4. In `Add Directory` page, choose OIDC as `Directory type`.
5. Enter the name of the new directory, such as ADFS_OIDC.
6. In `Service Provider` section, paste the values of `Client ID` and `Client Secret` copied from ADFS, respectively.
7. In `Service Provider` section, fill in `Host or IP` with the host or IP address where MetaDefender Core is hosted; for this example, it is https://localhost:8008.
8. Copy `Login URL`.

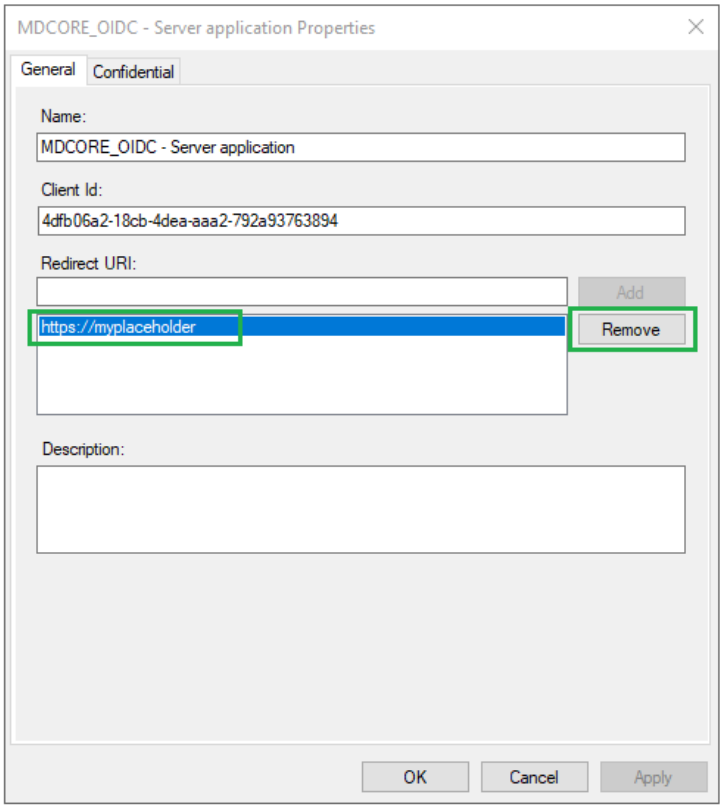


Complete configuration in ADFS

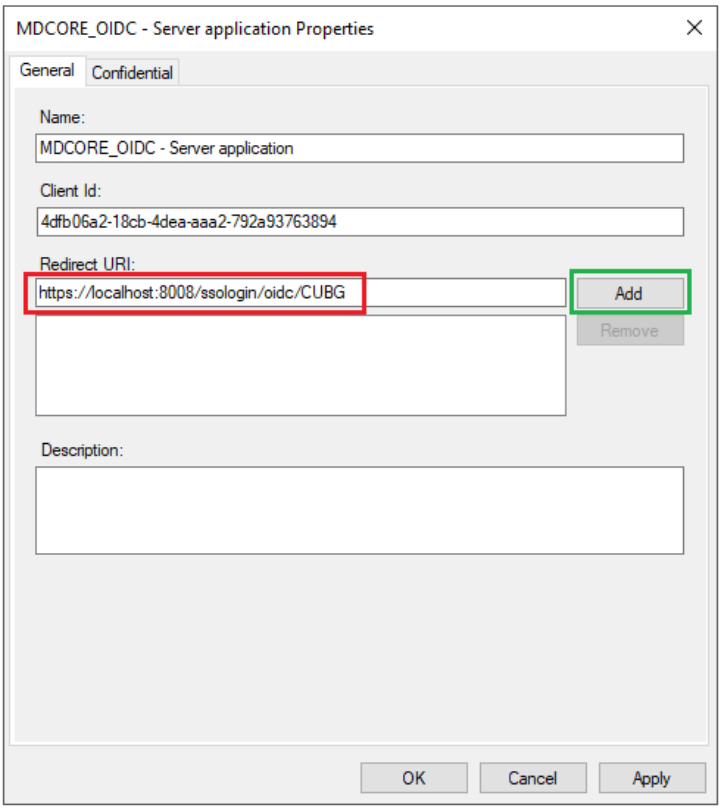
1. Back to `AD FS`, right click on item MDCORE_OIDC and select `Properties`, then choose `MDCORE OIDC - Server application` and click `Edit`.



2. Remove the placeholder https://myplaceholder by selecting it and clicking `Remove`.



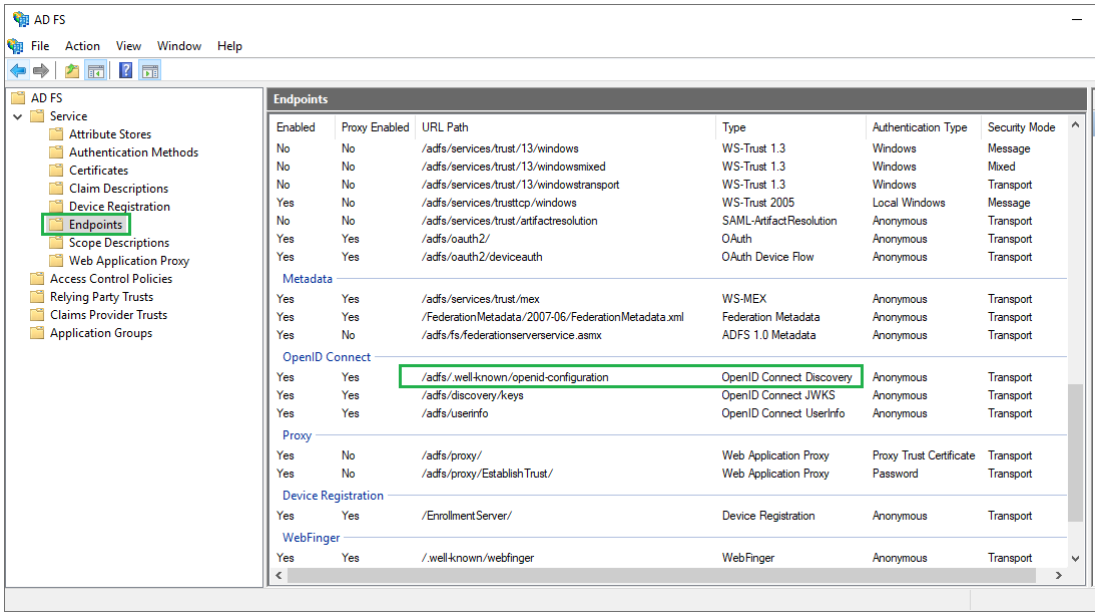
3. Paste the `Login URL` from MetaDefender Core into `Redirect URI`, click `Add`, then `Apply` and `OK`.



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

html

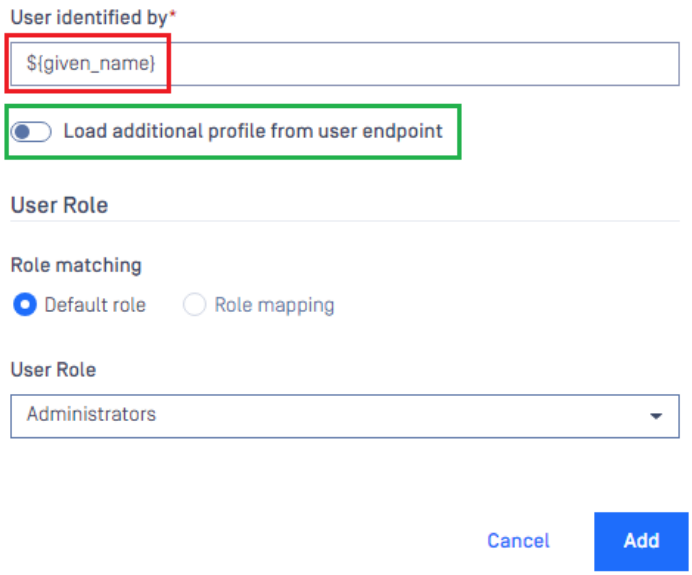
```
https://<adfs_fully_qualified_domain_name>/<metadata_endpoint_path>
```

- For example:

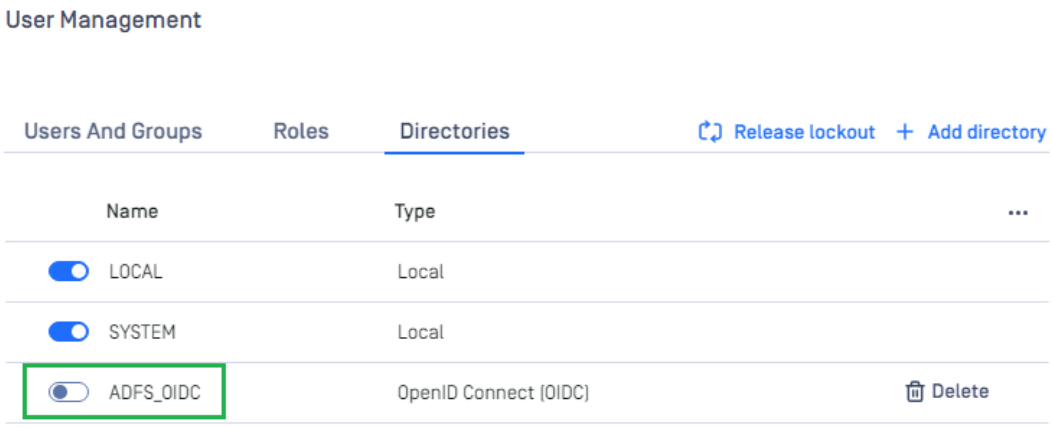
html

```
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.

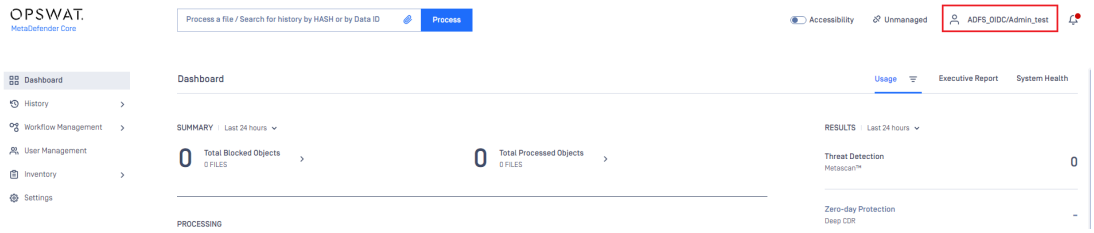


Test the integration

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

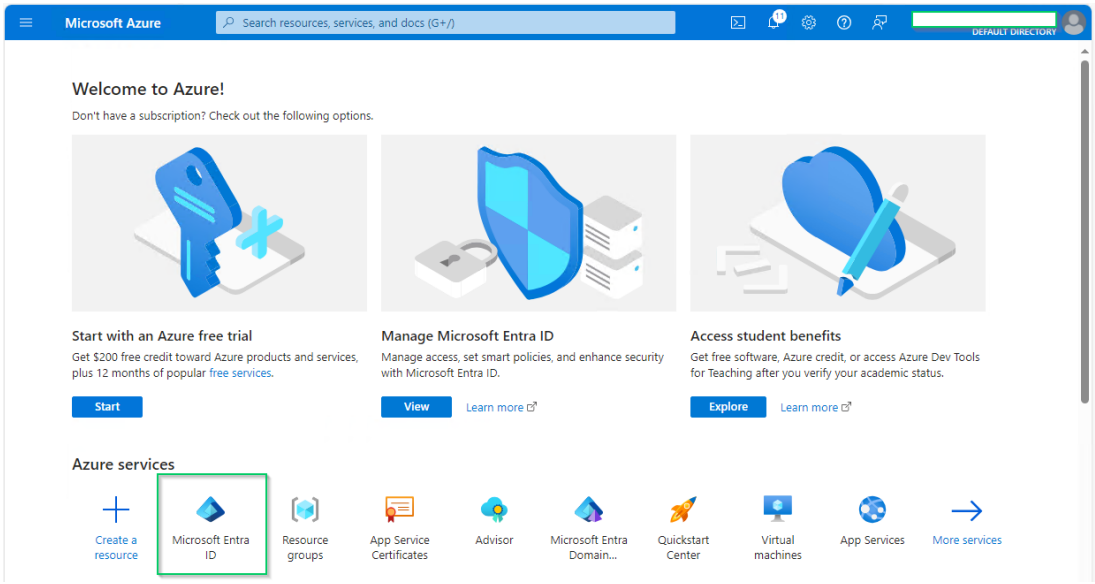
The instructions guide users on how to configure Microsoft Entra ID SAML and OIDC Single Sign-on for MetaDefender Core.

Configuring SAML Single Sign-on

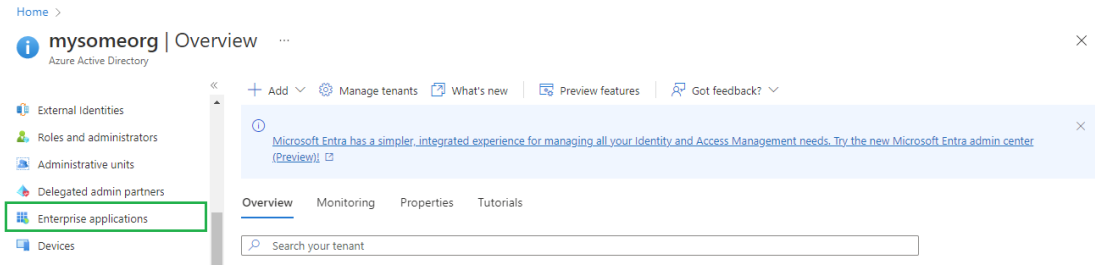
> **Info**
> Microsoft Entra ID requires MetaDefender Core to enable HTTPS for connections. Please follow these steps to first enable HTTPS on MetaDefender Core.

Register a new application in Azure

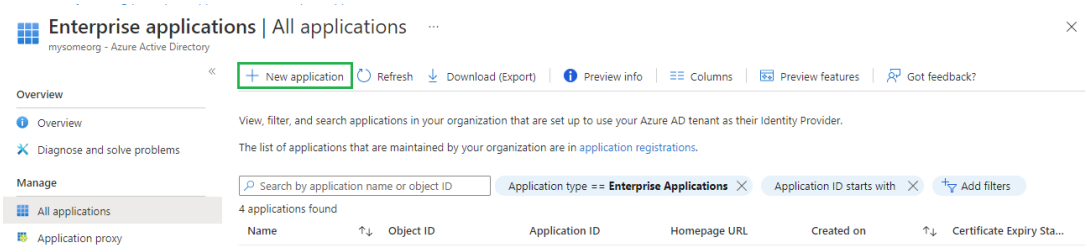
1. Access Azure portal and sign in.
2. On the homepage, select `Microsoft Entra ID` under `Azure services`.



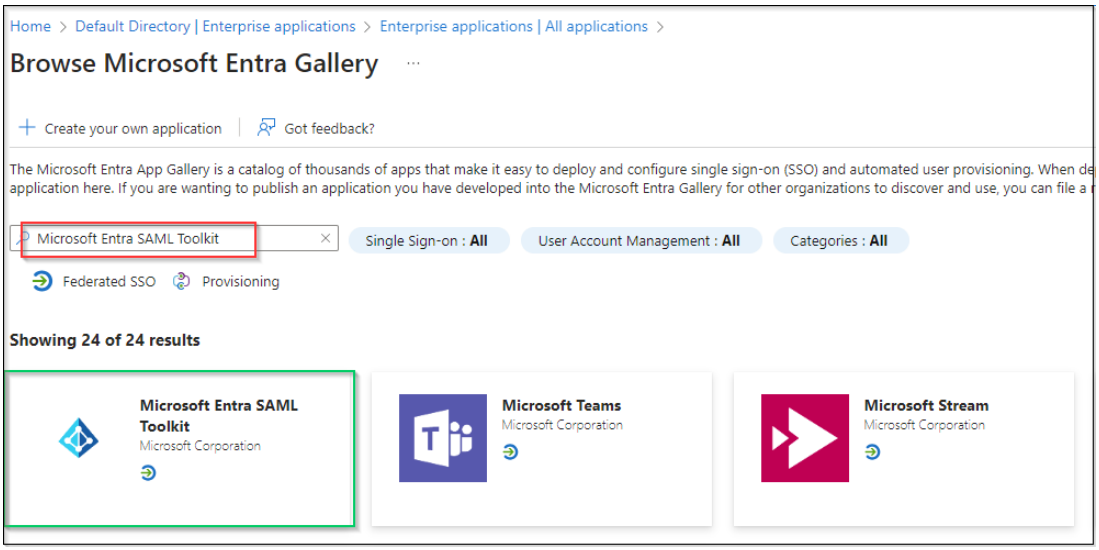
3. Select `Enterprise applications` in the left sidebar.



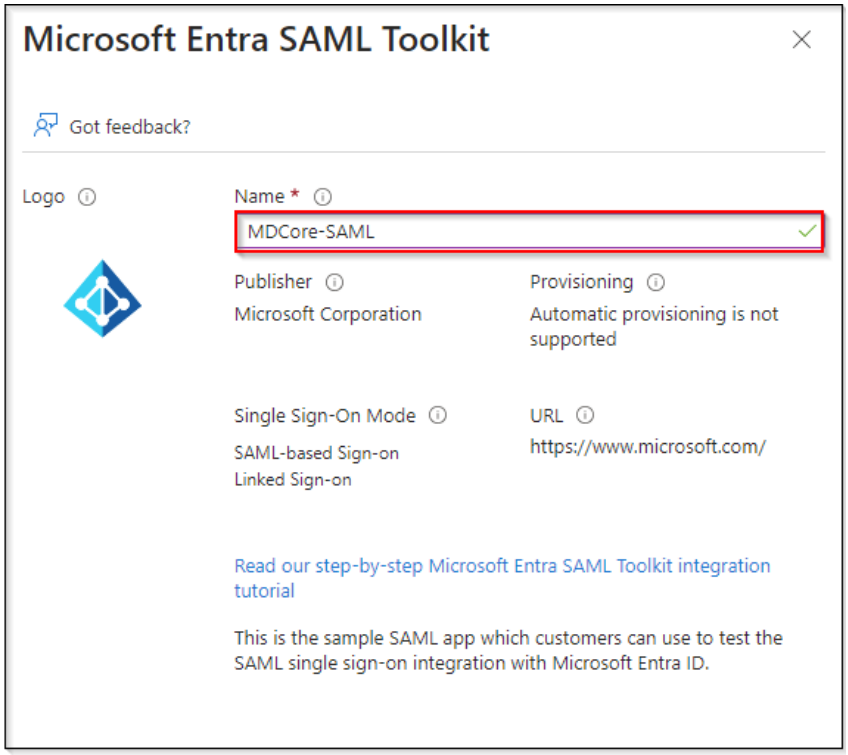
4. On `All applications` page, click `New application`.



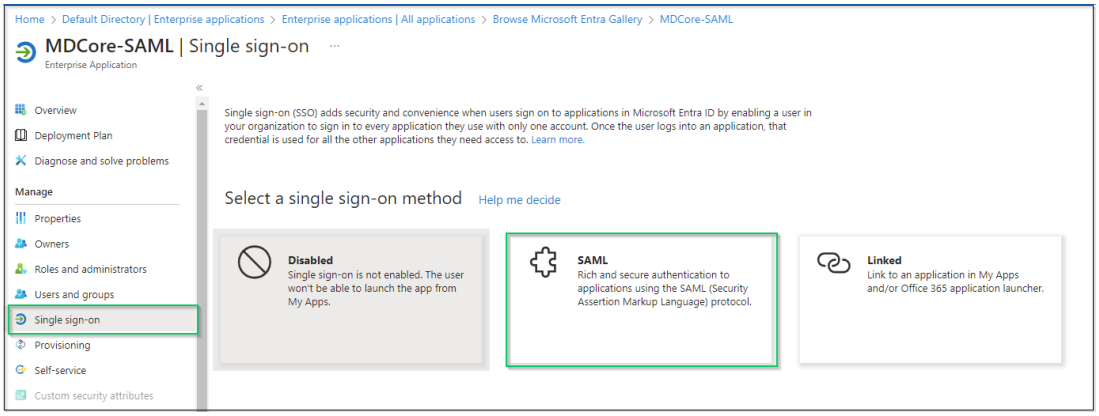
5. On `Browse Azure AD Gallery` page, search for "Microsoft Entra SAML Toolkit" and click on `Microsoft Entra SAML Toolkit` in the result panel.



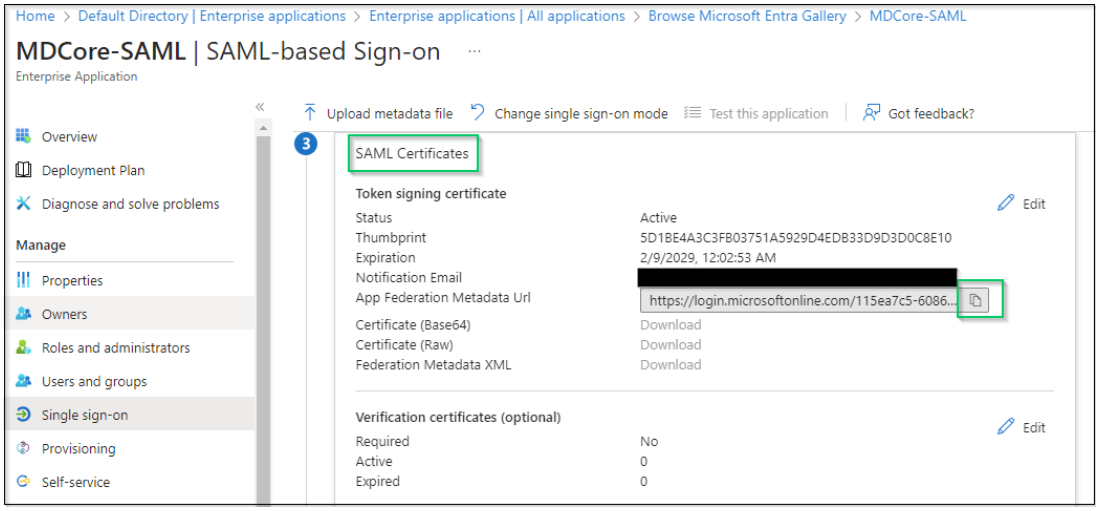
6. Fill in `Name` field with MDCore-SAML, for example, and click `Create` in the right sidebar.



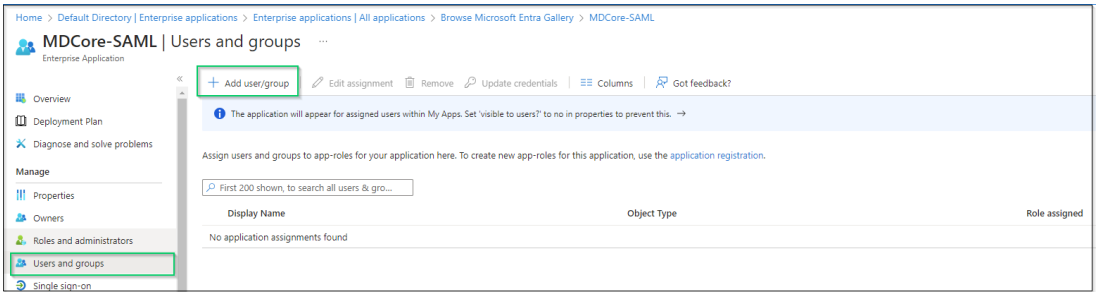
7. Navigate to `Single sign-on` on the left sidebar and click `SAML`.



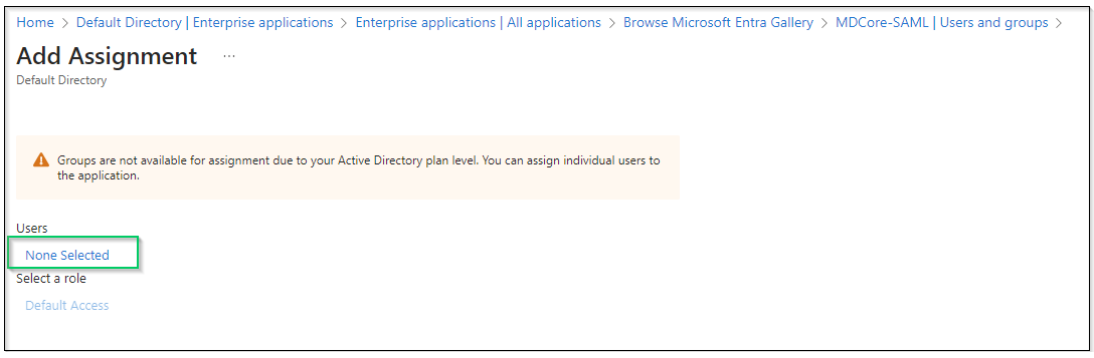
8. Go to `SAML Certificates` section, click the copy button at the far right of `App Federation Metadata Url` and store as *metadata_uri*.



9. Navigate to `Users and groups` in the left sidebar, click `Add user/group`.



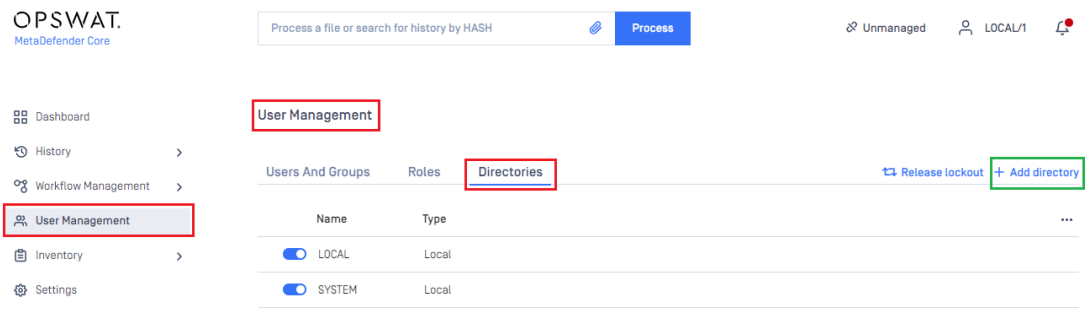
10. On `Add Assignment` screen, click `Non Selected` and add users who are allowed to log in to the app, then click `Select` in the right panel.



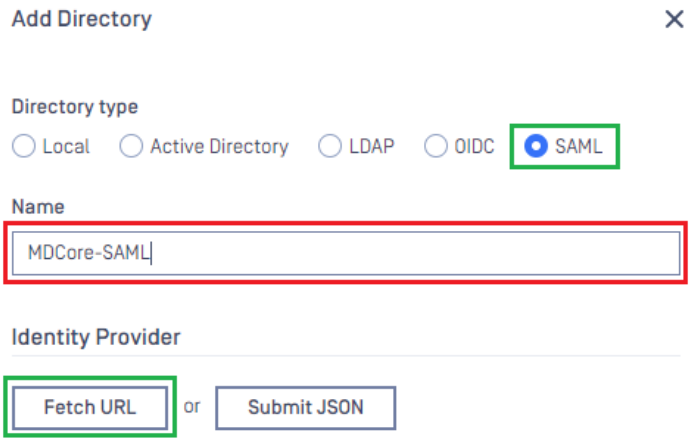
11. Finally, click `Assign` to complete.

Create SAML directory in MetaDefender Core

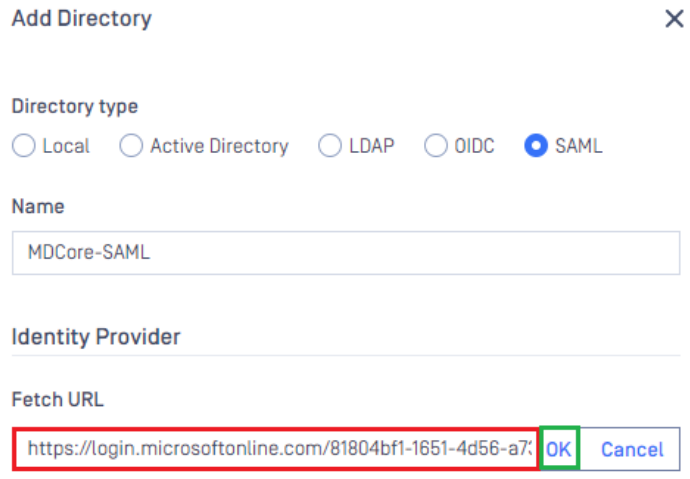
1. Sign in to MetaDefender Core management console.
2. On the dashboard, click `User Management` in the left sidebar.
3. In `User Management` page, select `Directories` tab and click on `Add directory` in the top right corner.



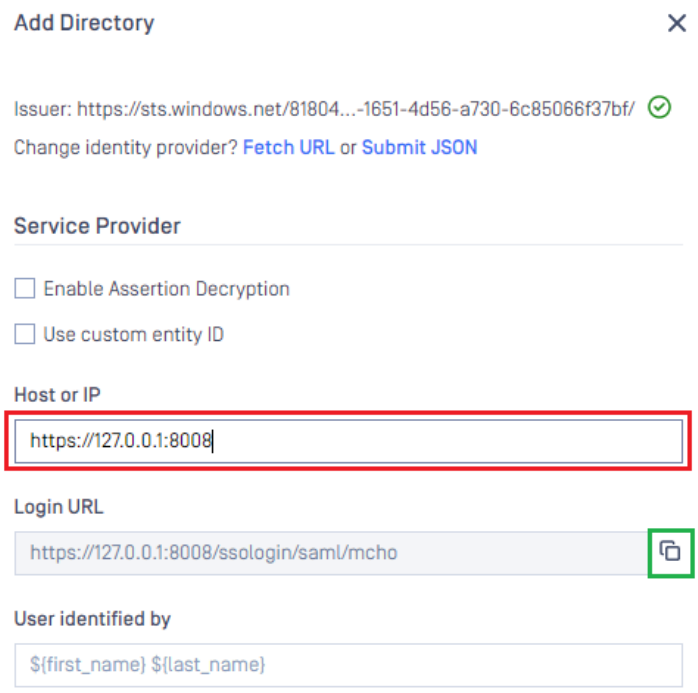
4. In `Add Directory` page, choose SAML in `Directory type`, enter a name for the new directory, such as MDCore-SAML and click `Fetch URL`.



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.

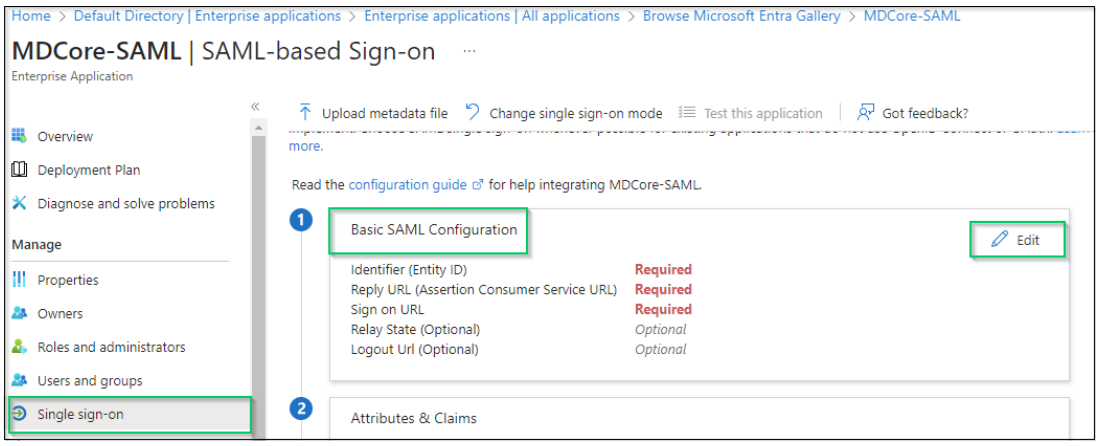


6. In `Service Provider` section, fill in `Host or IP` with the address where MetaDefender Core is hosted, for example https://127.0.0.1:8008.
7. Copy the `Login URL` and store in *reply_uri* for the later steps.



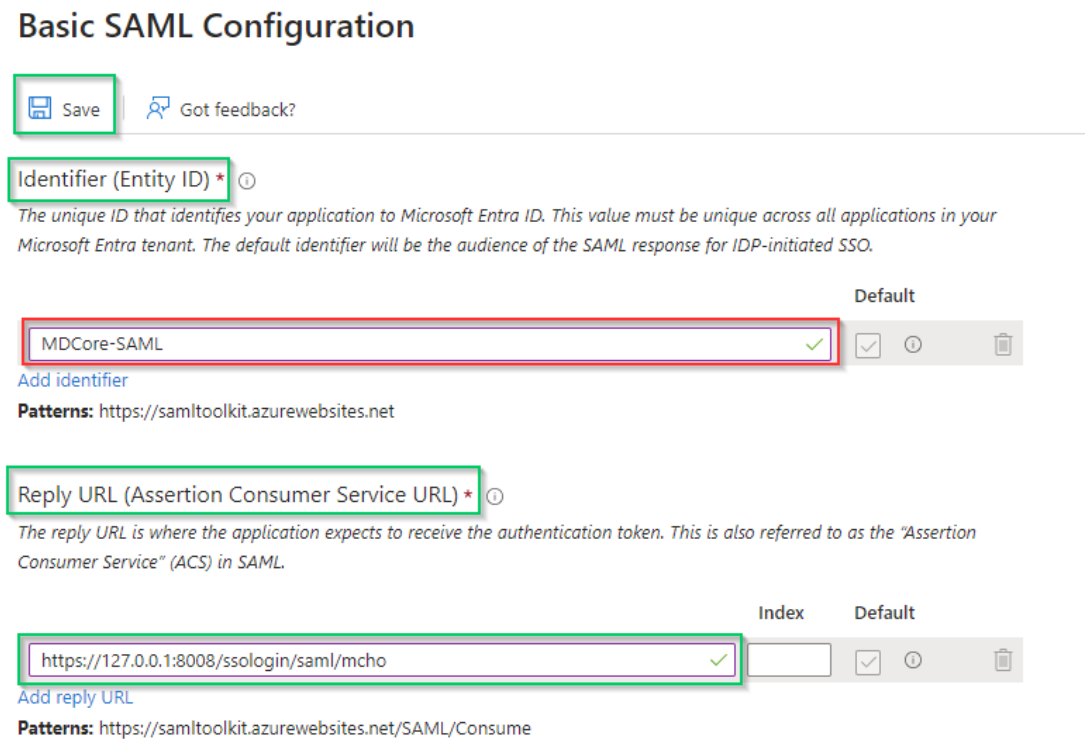
Complete configuration in Entra ID

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.

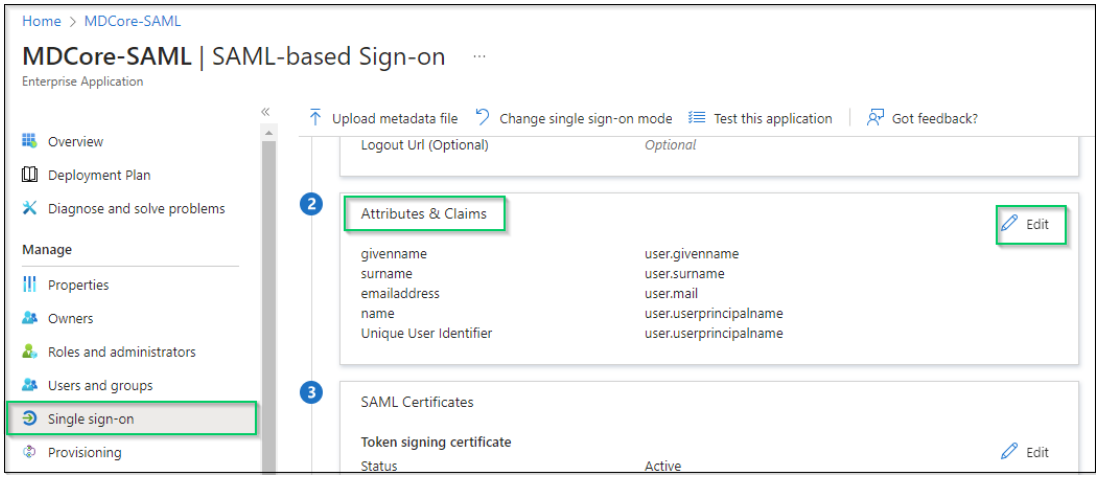


2. Navigate to `Identifier (Entity ID)` in the right sidebar, click `Add identifier`, then enter a unique ID to identify MDCore, such as MDCore-SAML. Store the identifier to *identifier.*

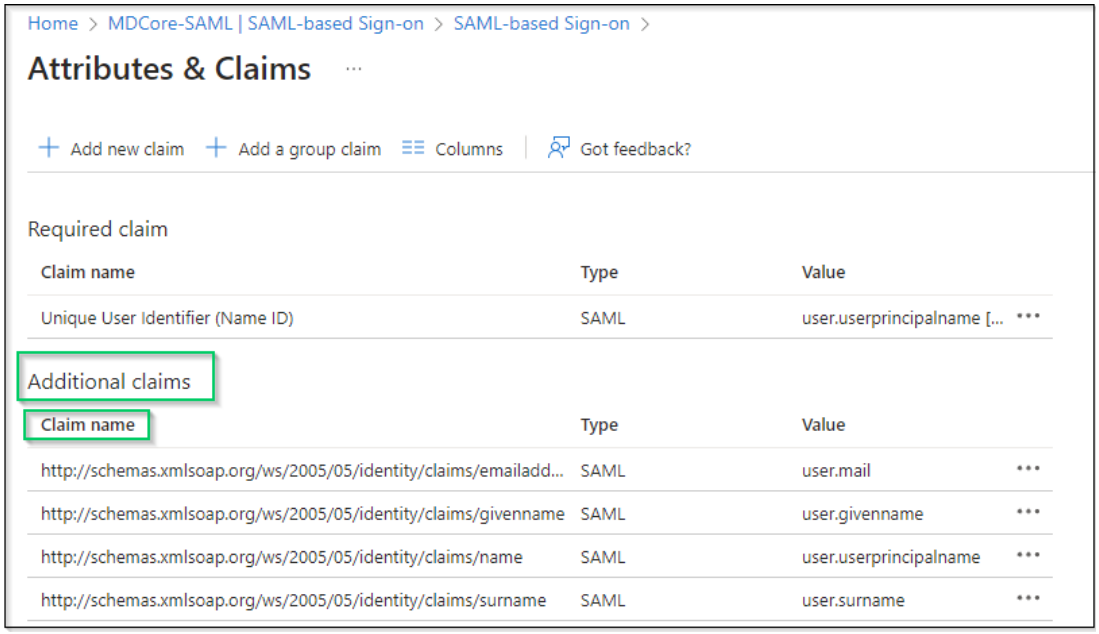
3. Navigate to `Reply URL(Assertion Consumer Service URL)`, click `Add reply URL` and fill in the URI with the value stored in *reply_uri* and click `Save`.



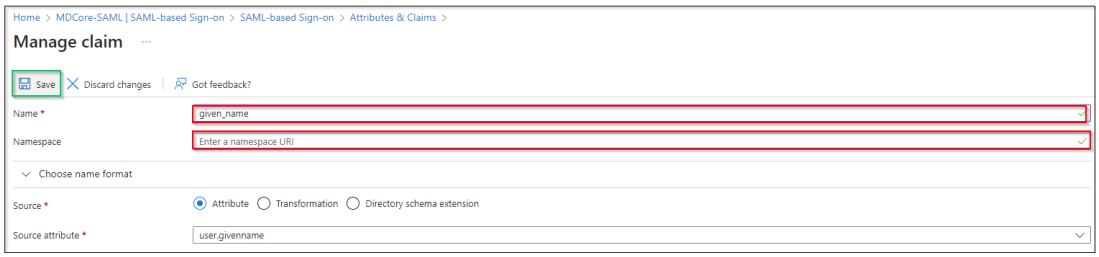
4. Navigate to `Attributes & Claims`, then click `Edit`.

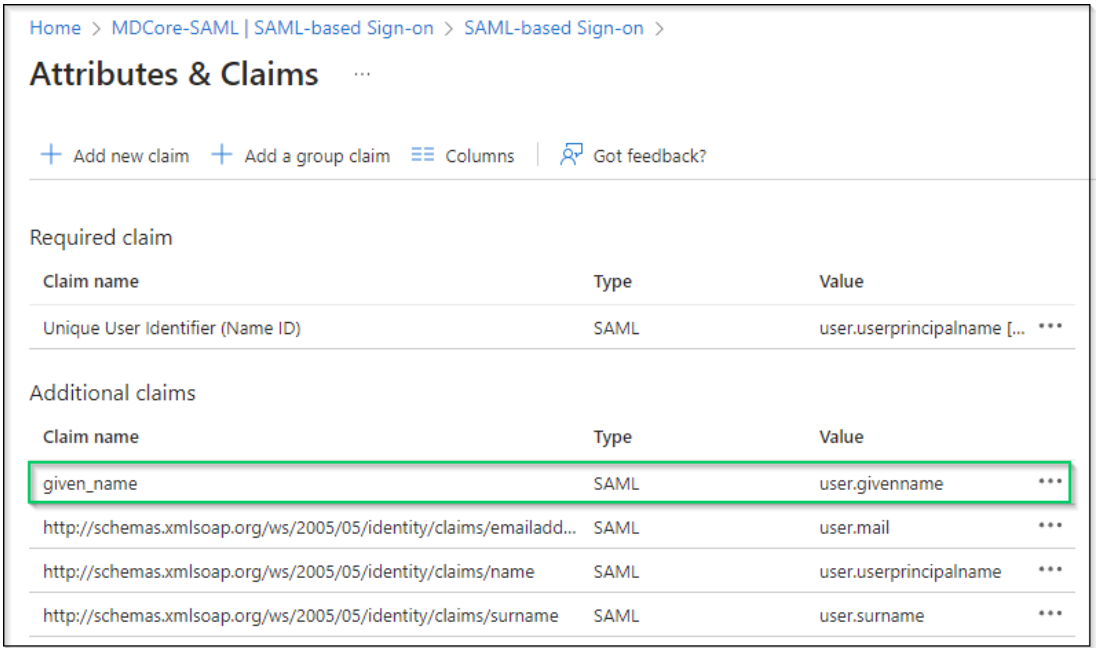


5. In `Attributes & Claims` page, go to `Additional claims`, and click on any item under `Claim name` to change its name.



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.

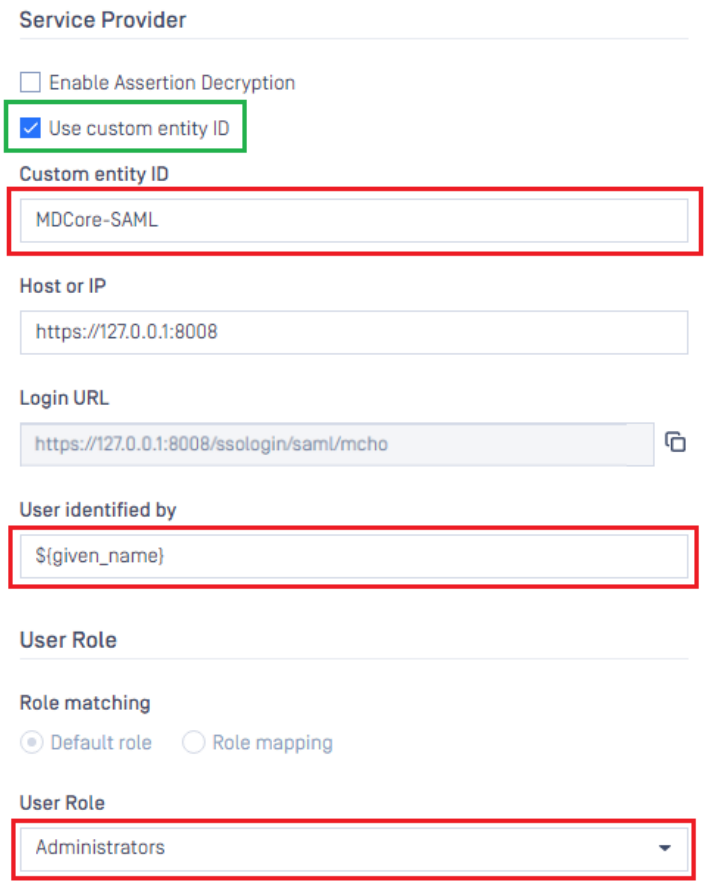
Complete configuration in MetaDefender Core

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}.

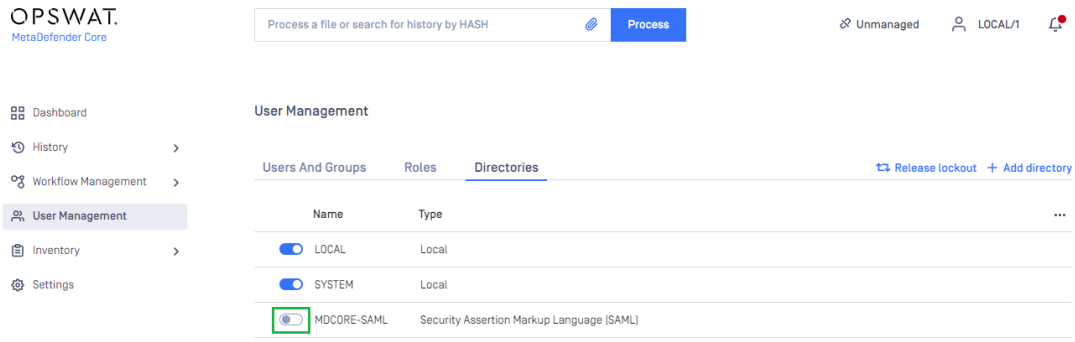
Info
If the namespace is not removed from the claim name in step 6 of the previous section, the full claim name including the namespace must be used here to build user identity.

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.

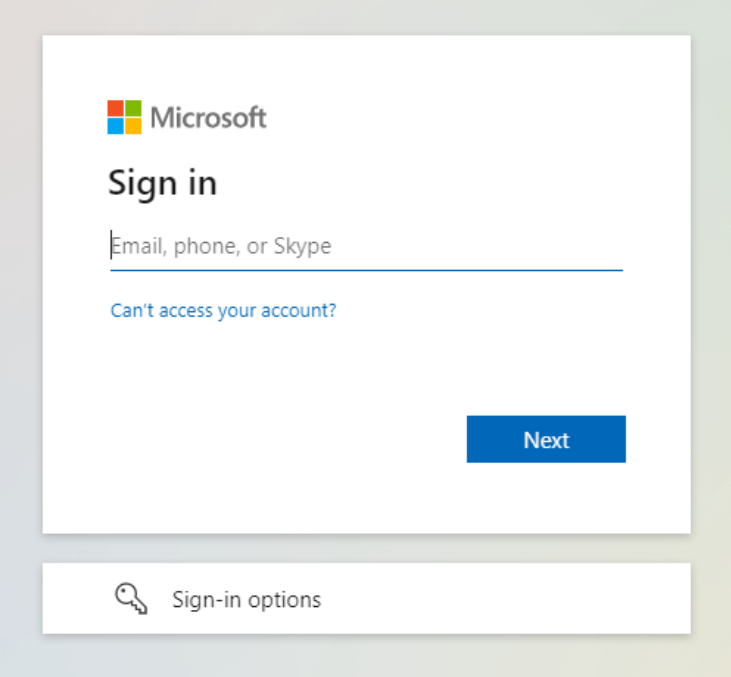


5. In `User Management` screen of MetaDefender Core, toggle the new directory named MDCORE-SAML. A dialog box appears to confirm the action. Once `Enable` is clicked, all sessions are expired immediately.



Test the integration

1. At the home screen of MetaDefender Core, click `Login`; the user is redirected to Microsoft Entra ID sign-in page.

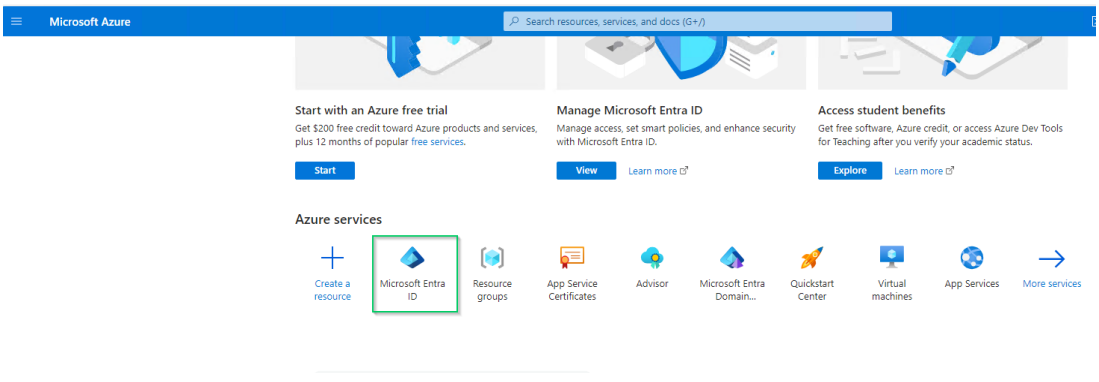


2. Sign in using the account registered with Microsoft Entra ID.
3. If everything goes correctly, MetaDefender Core dashboard is displayed with user's identity shown in the top right corner.
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

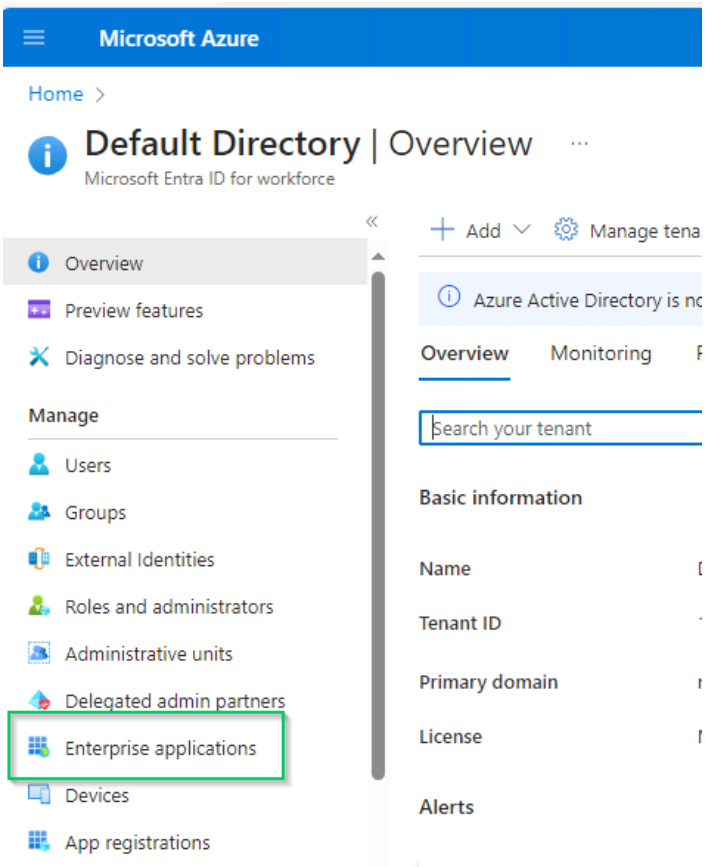
Setup pseudo IdP-initiated SSO

> **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.

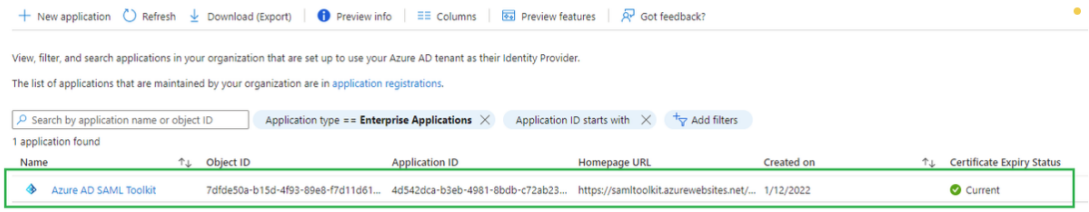
1. Sign in to Azure management page.
2. Navigate to `Azure services` section, click on `Microsoft Entra ID`.



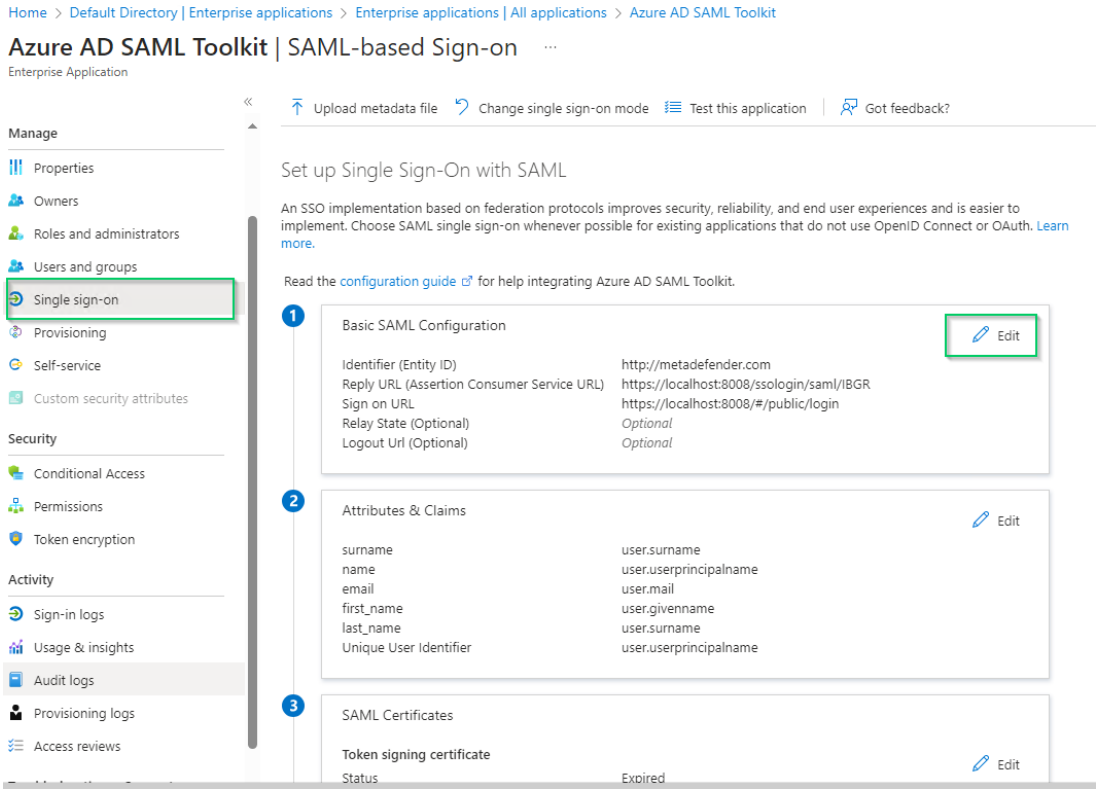
3. In the left sidebar, click on `Enterprise applications`.



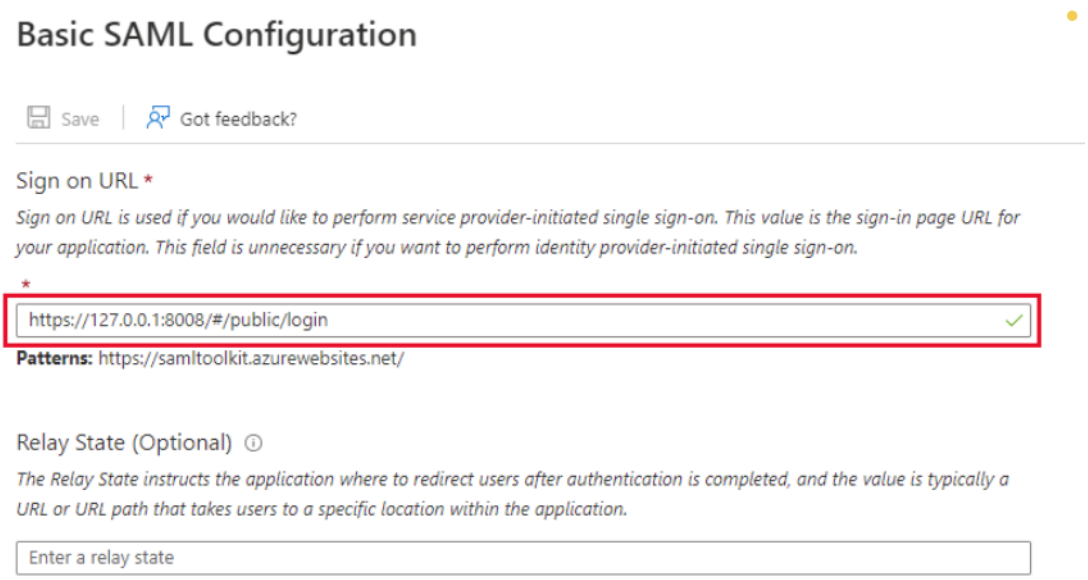
4. Select `Azure AD SAML Toolkit` from the list of enterprise applications.



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`.

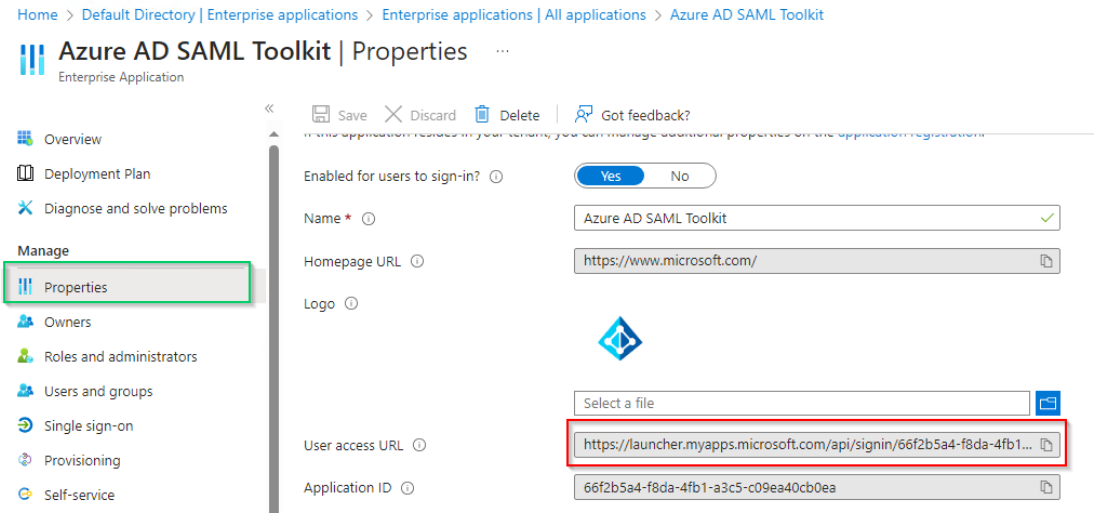


7. Click `Save` to complete.

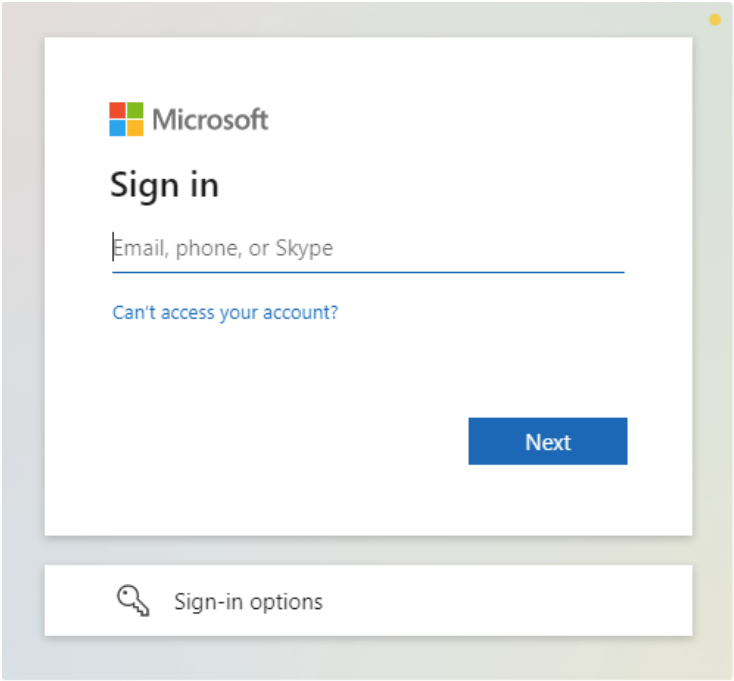
Test IdP-initiated SSO

1. Sign in to Azure management page.

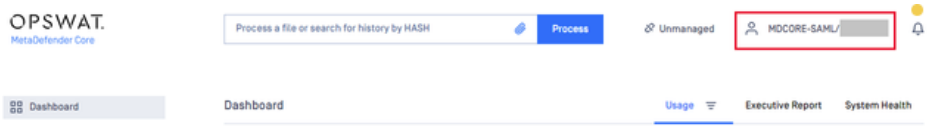
2. Navigate to `Azure services` section, and select `Microsoft Entra ID`.
3. In the left sidebar, click on `Enterprise applications`.
4. Choose `Azure AD SAML Toolkit` from the list of enterprise applications.
5. Go to `Properties` tab.



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.

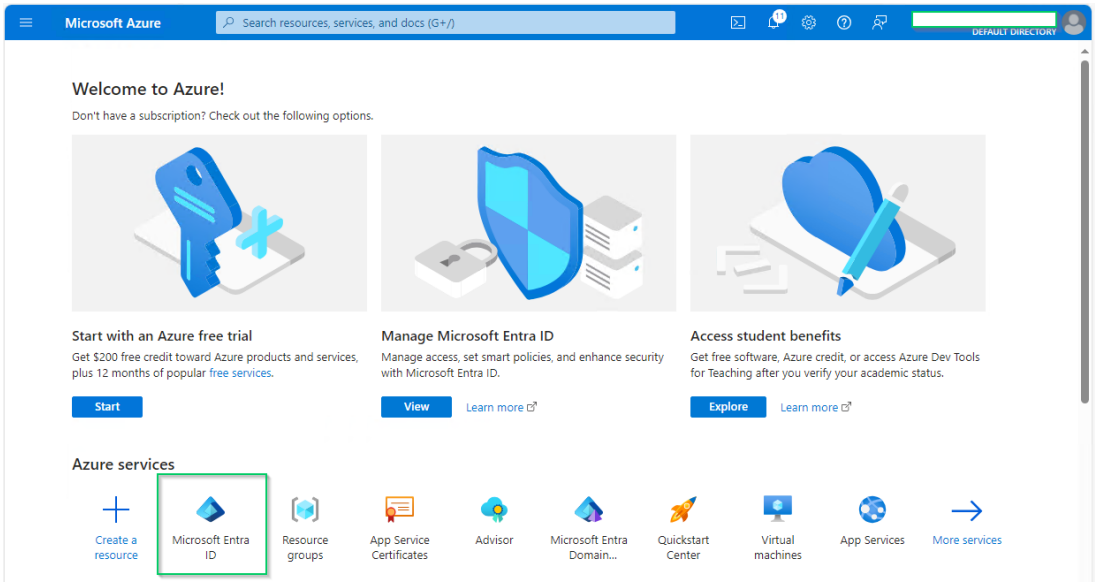


9. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

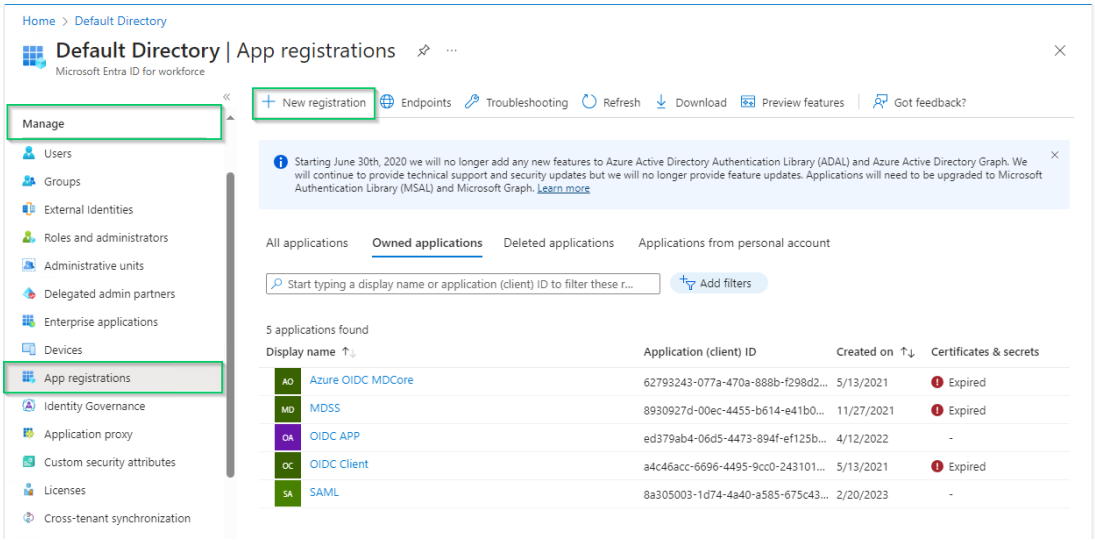
Configuring OIDC Single Sign-on

Register a new application in Azure

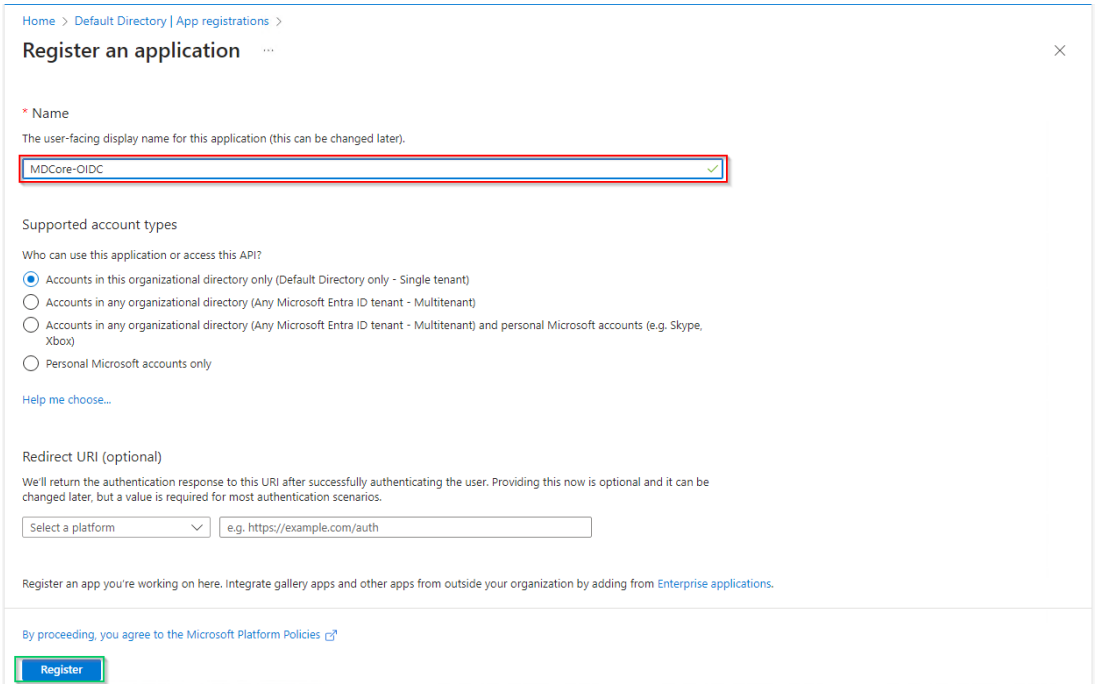
1. Access Azure portal and sign in.
2. On the home page, select `Microsoft Entra ID` under `Azure services`.



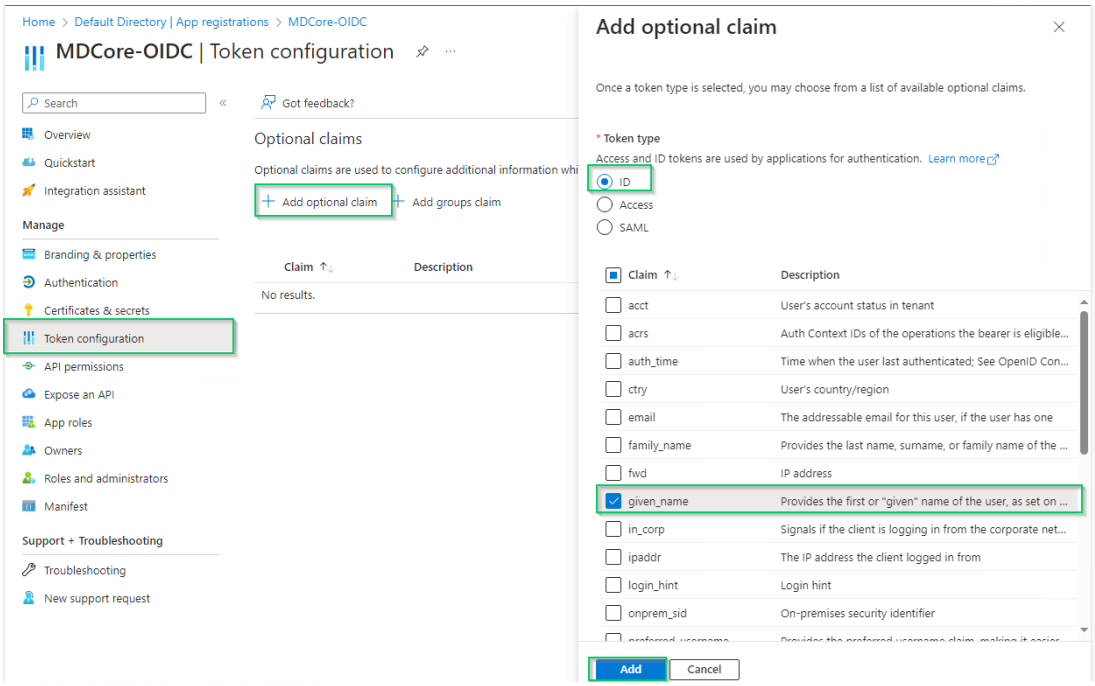
3. Select `App registrations` under `Manage` section on the left sidebar, then click on `New registration` on the top menu bar.



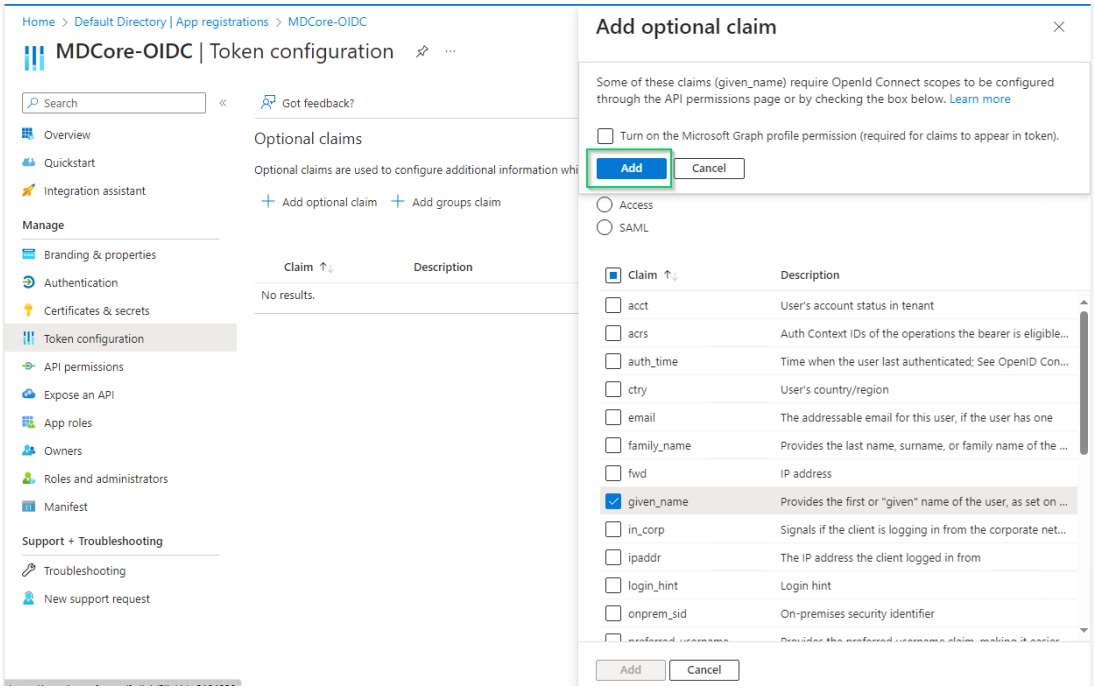
4. In `Register an application` page, fill in the name for the new application, using MDCore-OIDC as an example, and click `Register` to complete.



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.



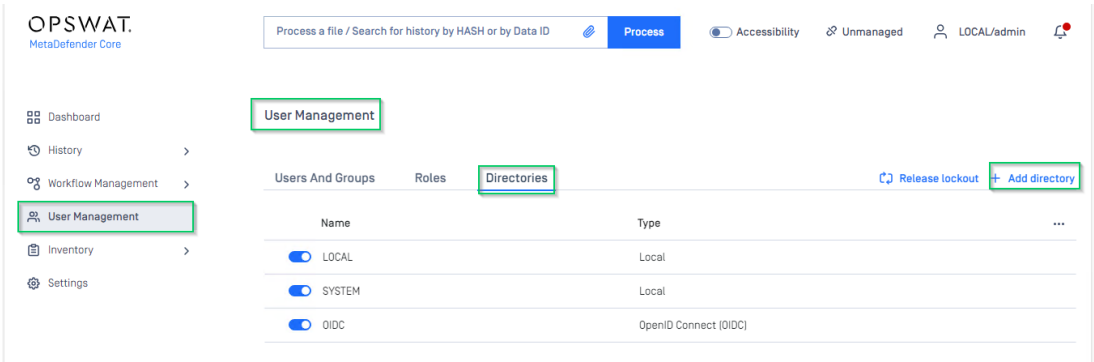
6. Click `Add` on the popup to finish.



Info
Enabled claims will be used later by MetaDefender Core to identify logged-in user.

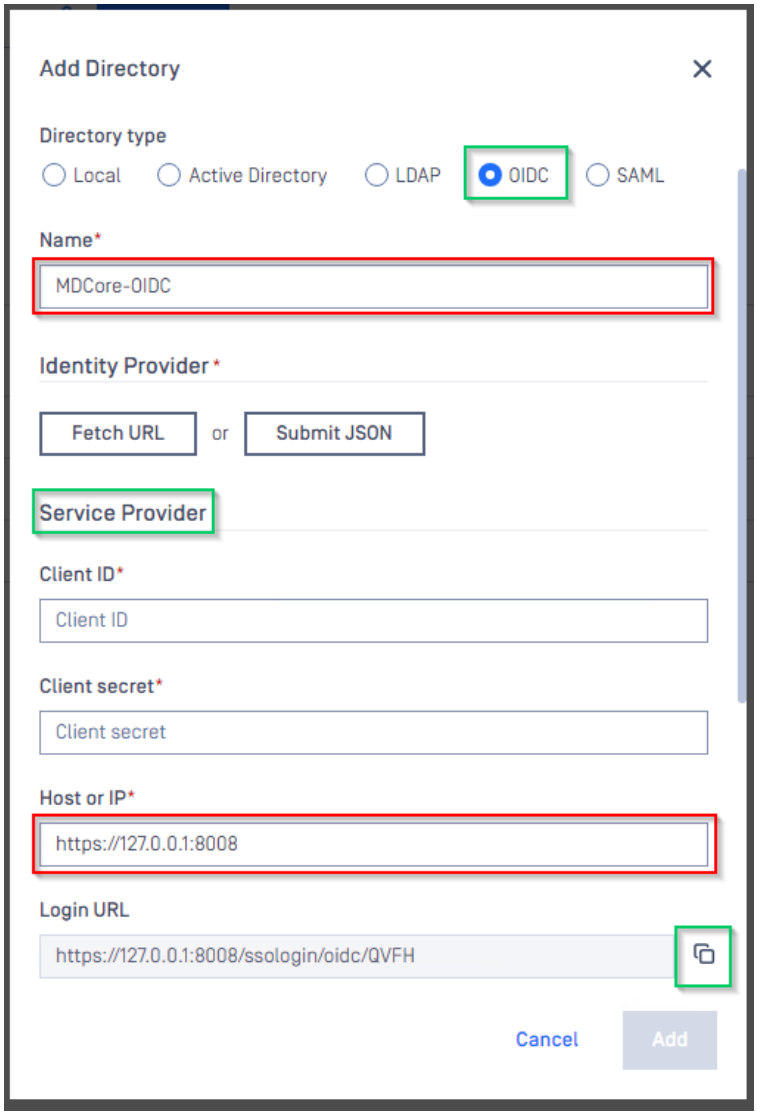
Create OIDC directory in MetaDefender Core

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.



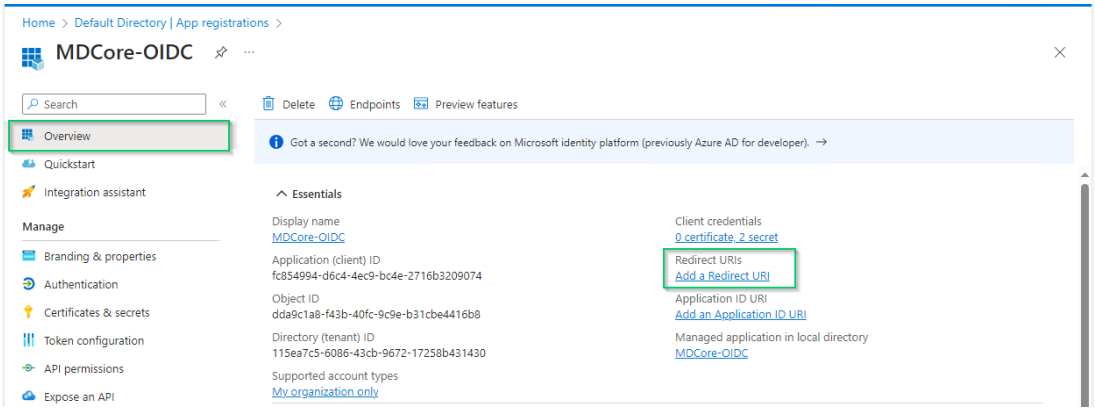
4. In `Add Directory` page, choose OIDC in `Directory type`.
5. Fill in the name the new directory, such as MDCore-OIDC.
6. In `Service Provider` section, fill in `Host or IP` with the host or IP address where MetaDefender Core is hosted; for this example, it is https://localhost:8008.

7. Copy the `Login URL` and store it somewhere for later use.

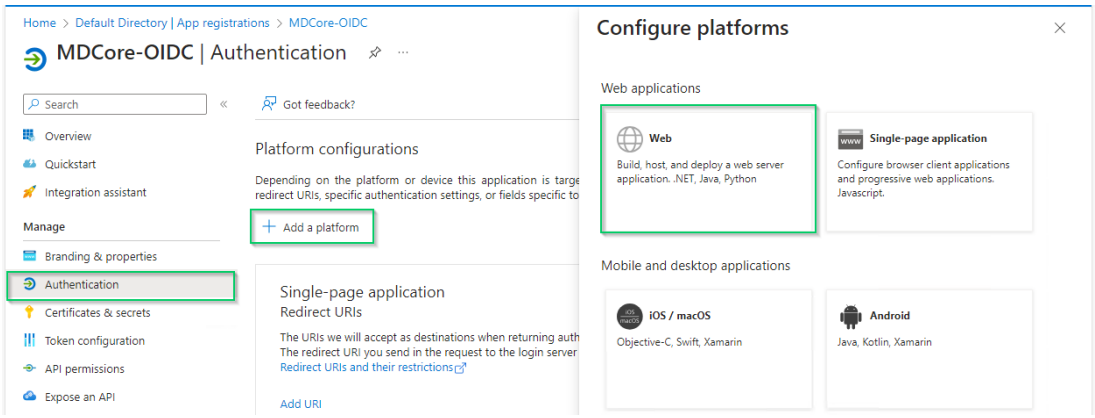


Complete configuration in Entra ID

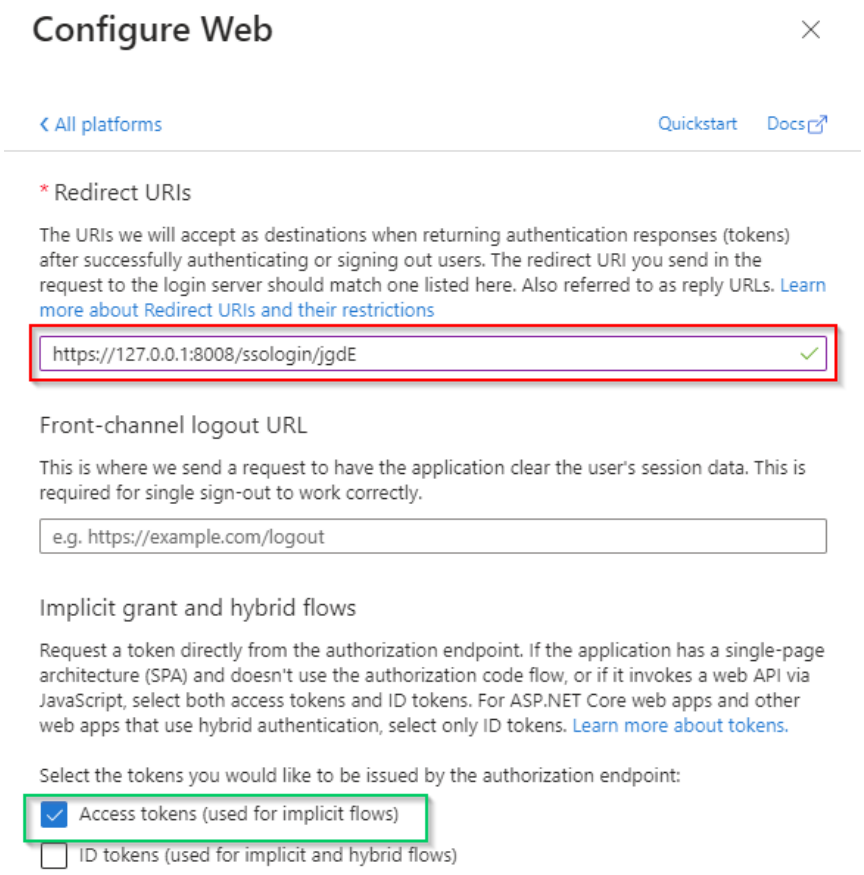
1. Back to Microsoft Entra ID, on `MDCore-OIDC` page, select `Overview` section in the left sidebar and click on `Add a Redirect URI` link to switch to `Authentication` page.



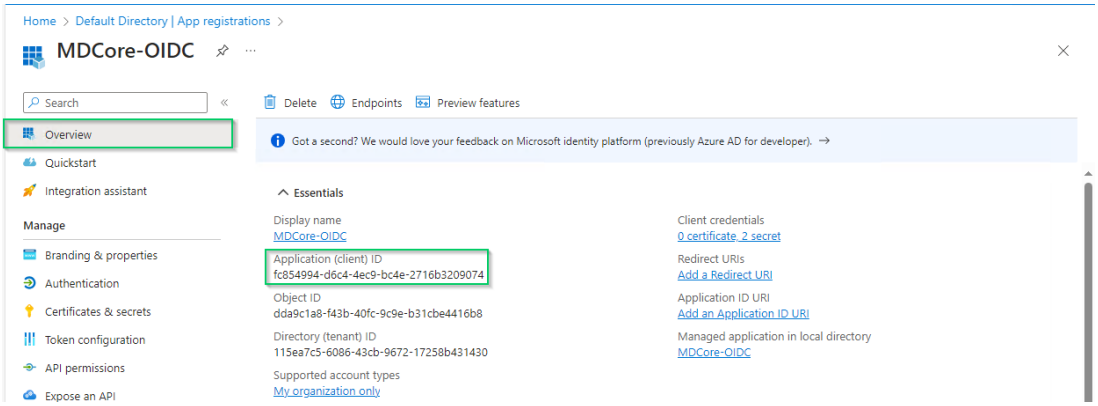
2. In `Authentication` page, click `Add a platform` and choose `Web` option in the right sidebar.



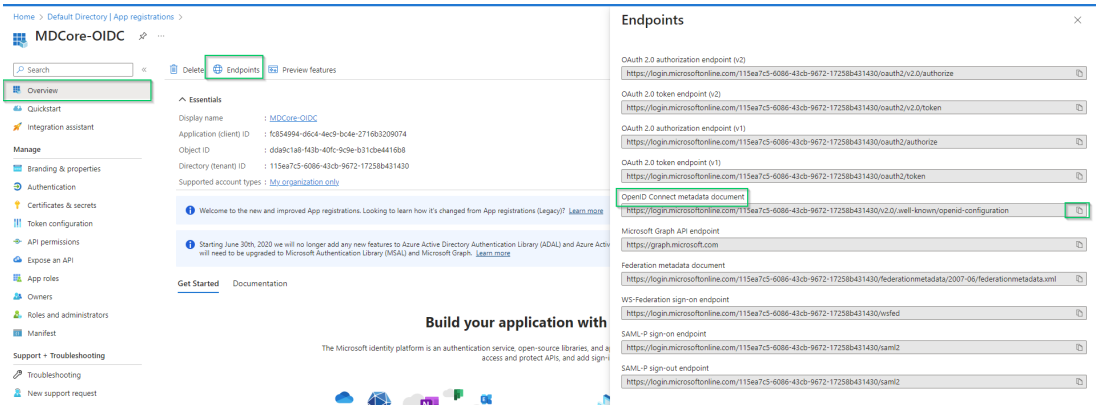
3. Fill in `Redirect URIs` field with the `Login URL` copied from MDCore, enable `Access tokens` option and click `Configure`.



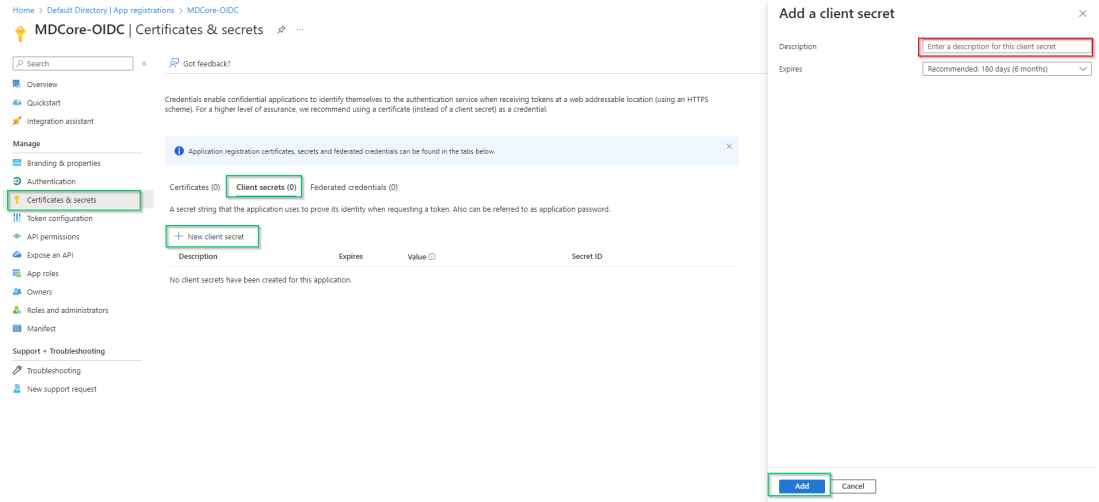
4. Select `Overview` in the left sidebar again, copy the string next to `Application (client) ID` and store as *client_id*.



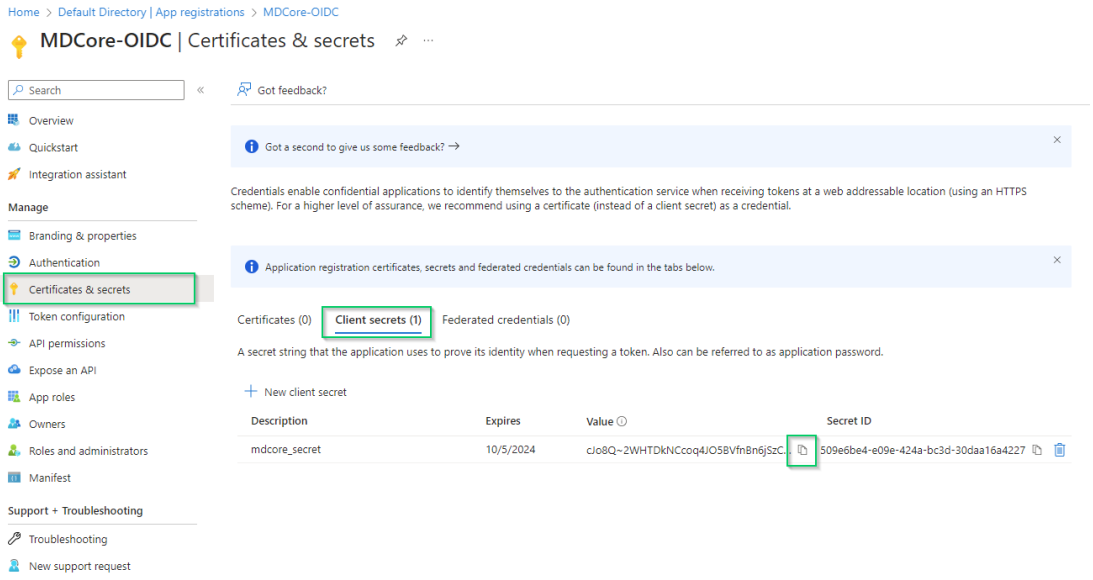
5. Click `Endpoints` in the top bar, copy the URI under `OpenID Connect metadata document` in the right sidebar and store it as *metadata_uri*.



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*.

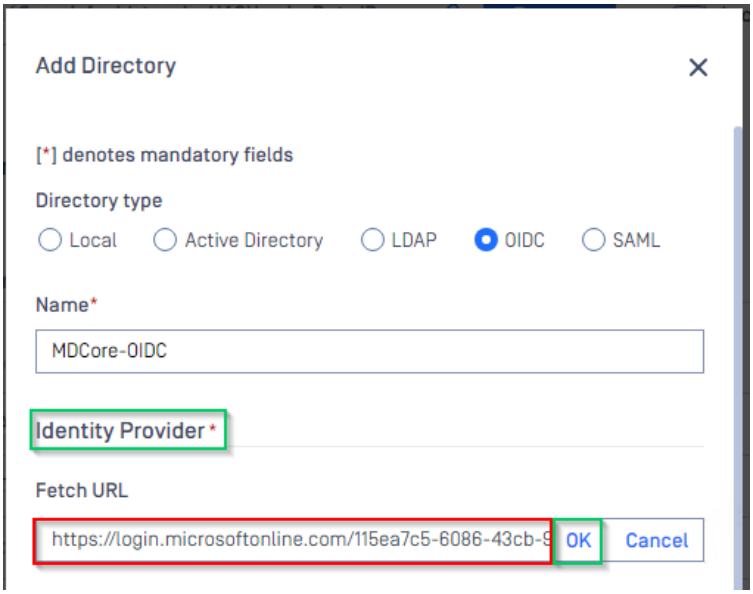


Info

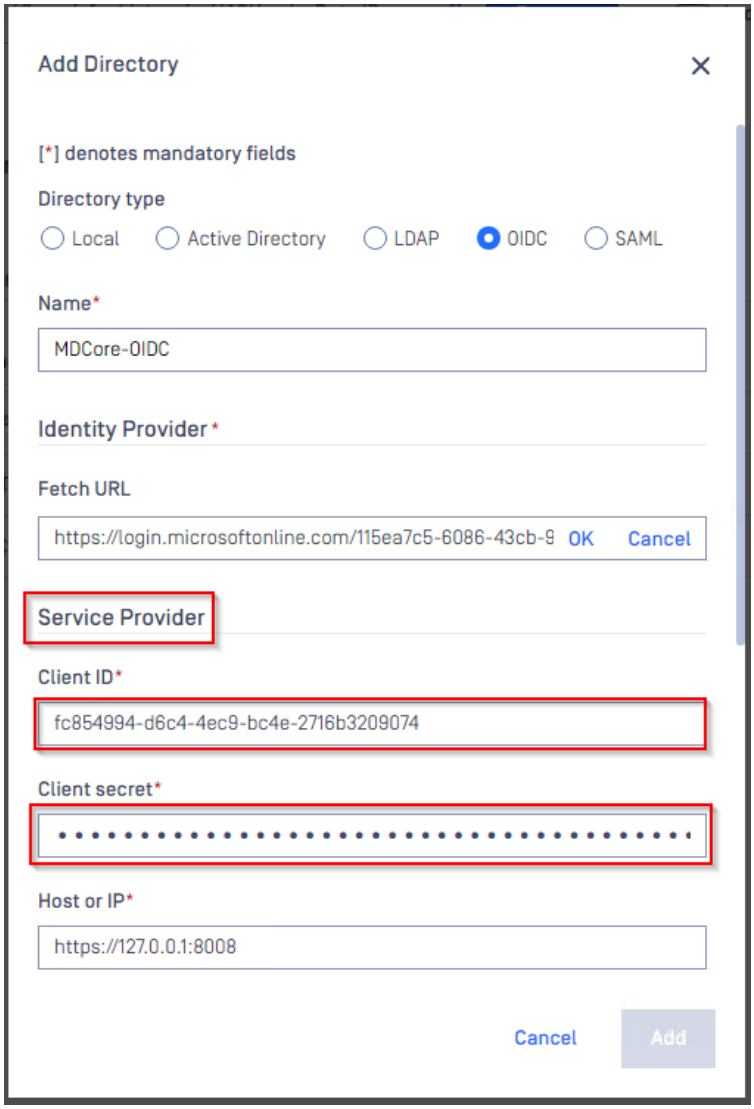
Entra ID will hide the client secret completely when page `Certificate & secrets` is left. Administrator is recommended to store the client secret in a safe place, otherwise they have to create a new one.

Complete configuration in MetaDefender Core

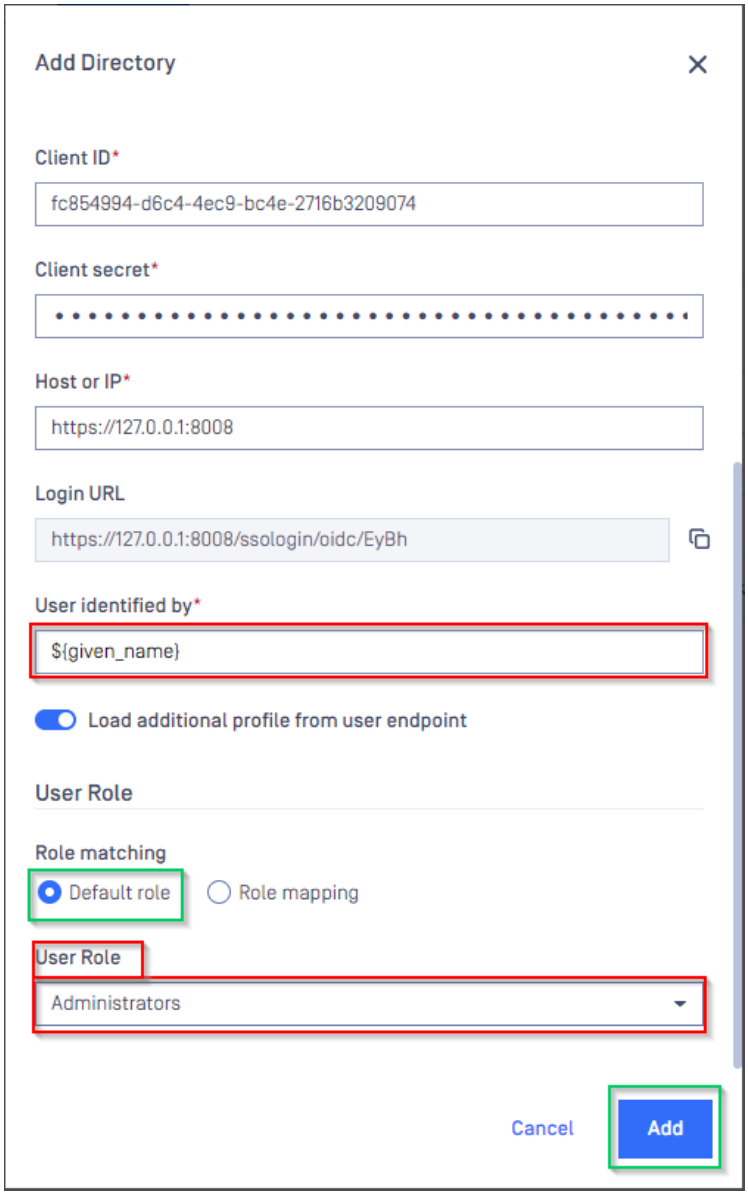
1. Switch back to MetaDefender Core. Under `Identity Provider` section, click `Fetch URL` and paste *metadata_uri* gathered in **Configure Azure** stage into the box under `Fetch URL`. Click `OK` and wait a moment for MetaDefender Core to set Microsoft Entra ID as its IDP.



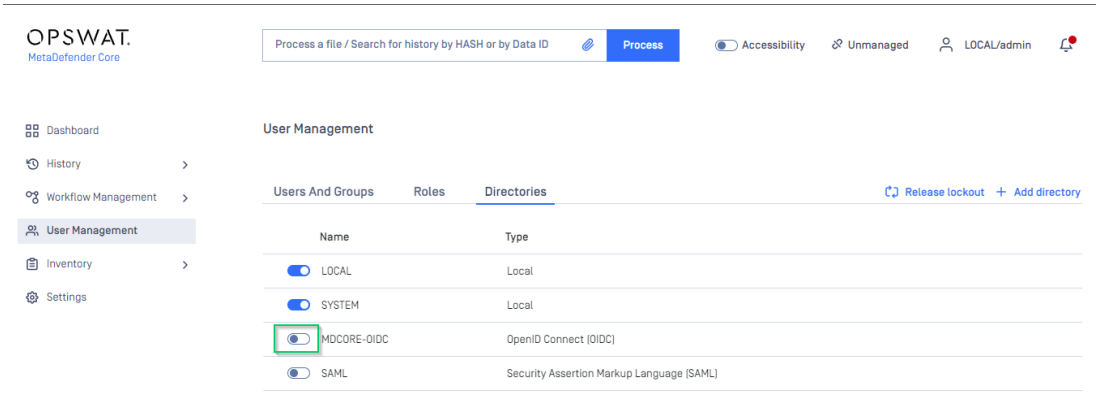
2. Under `Service Provider` section, paste *client_id* and *client_secret* gathered in **Configure Azure** stage to `Client ID` and `Client secret` respectively.



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.

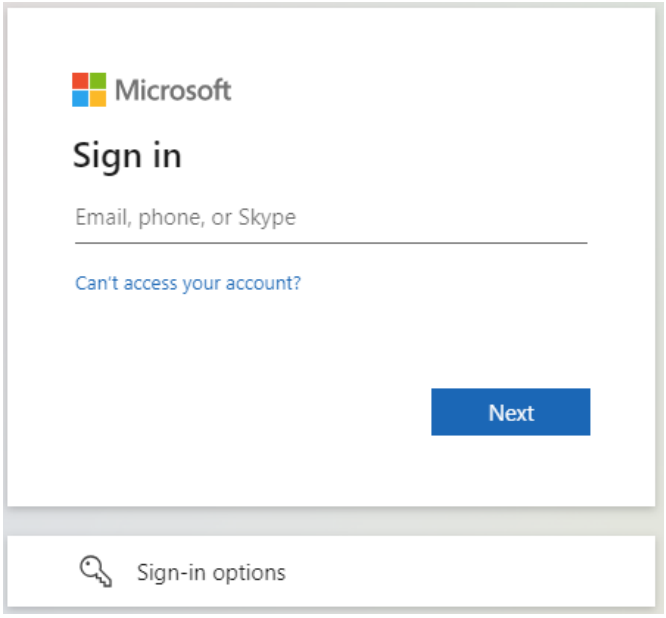


6. In `User Management` screen of MetaDefender Core, toggle MDCORE-OIDC directory. A dialog box appears to confirm the action. Once `Enable` is clicked , all sessions expire immediately.

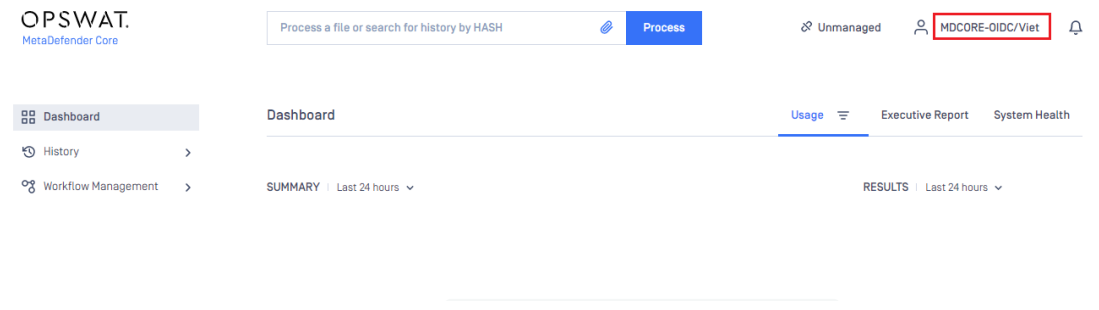


Test the integration

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

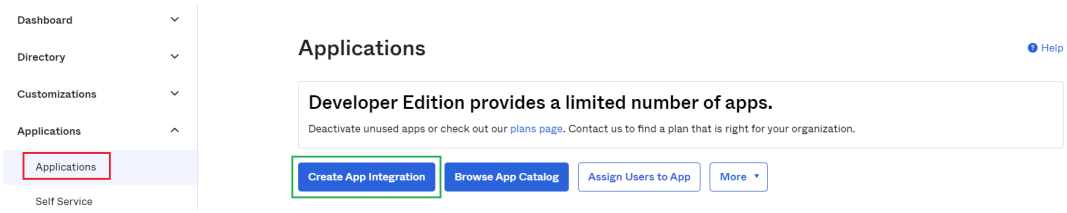
The instructions guide users on how to configure Okta SAML and OIDC Single Sign-on for MetaDefender Core.

Configuring OIDC Single Sign-on

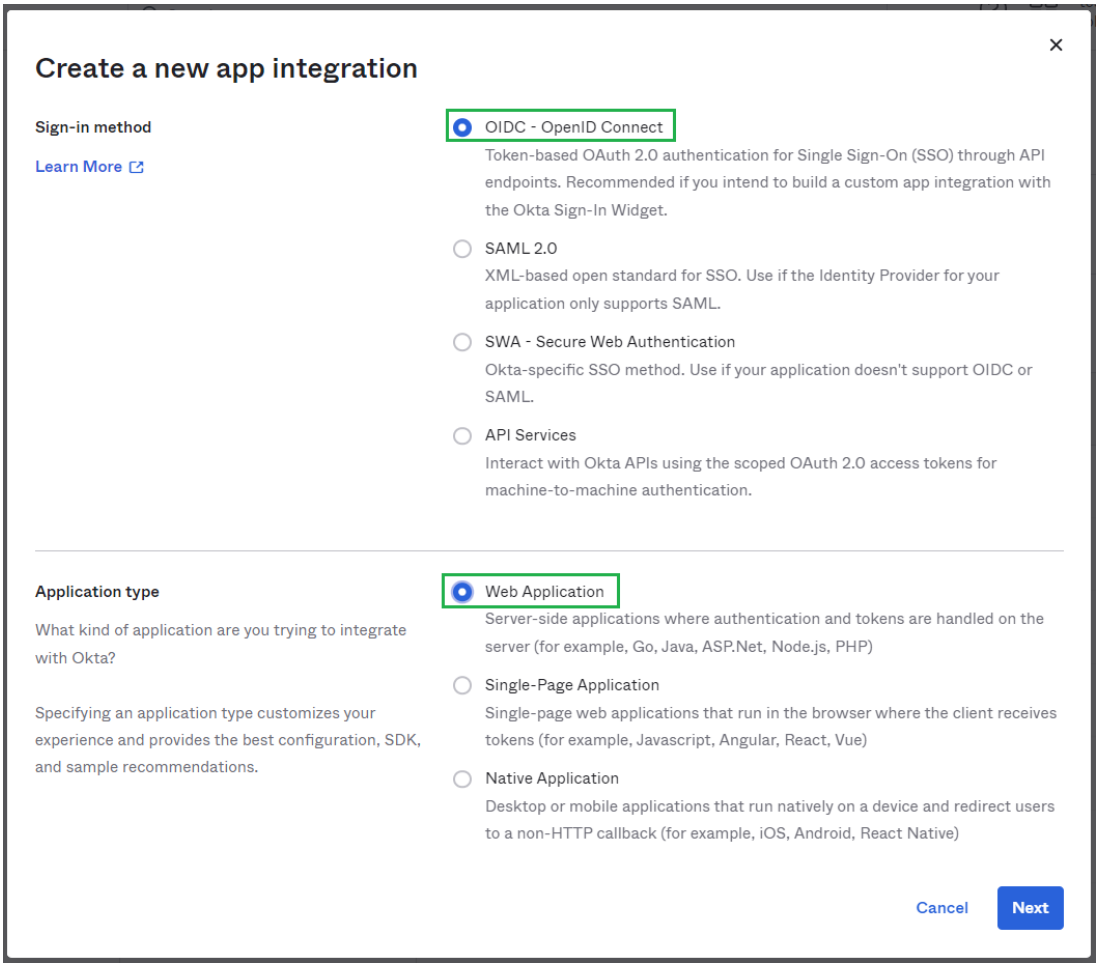
> **Warning**
> After the integration is successfully set up, any modifications on Okta application could lead to the regeneration of the client secret, causing Okta to reject requests from MetaDefender Core due to a mismatch in the client secret. In this instance, the customer should copy the new secret from Okta and update `Client secret` field within `Service Provider` of MetaDefender Core with the new one.

Create Okta application

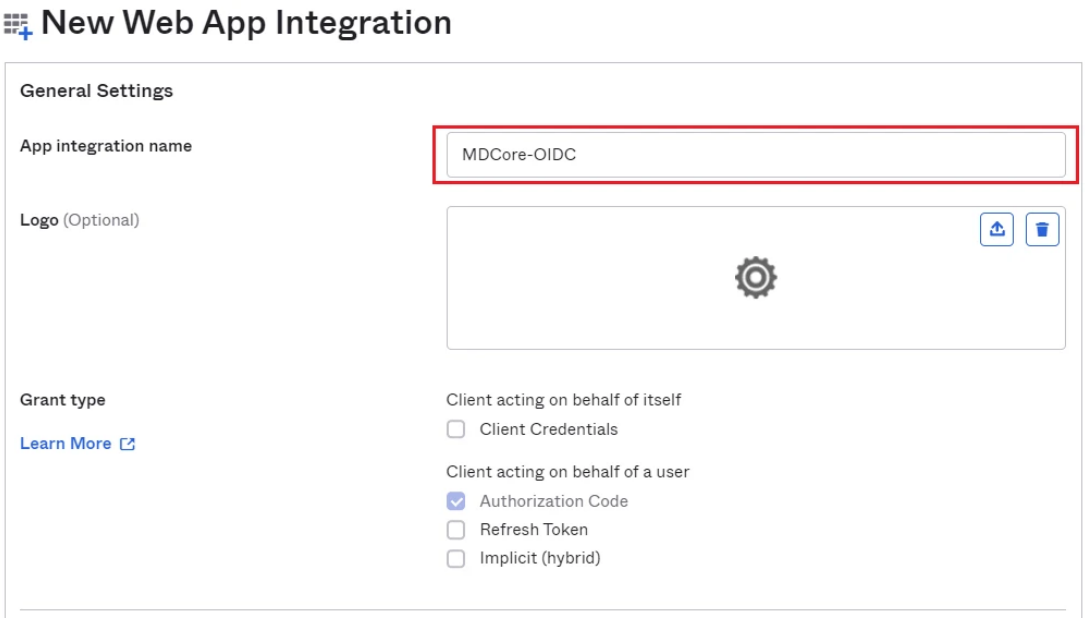
1. Access https://www.okta.com/login/ and sign in.
2. In the sidebar of dashboard, hit `Applications` to drop sub items down, then select `Applications`.
3. Click `Create App Integration`.



4. In `Sign-in method`, select `OIDC - OpenID Connect`. In `Application type` choose `Web Application` and click `Next`.

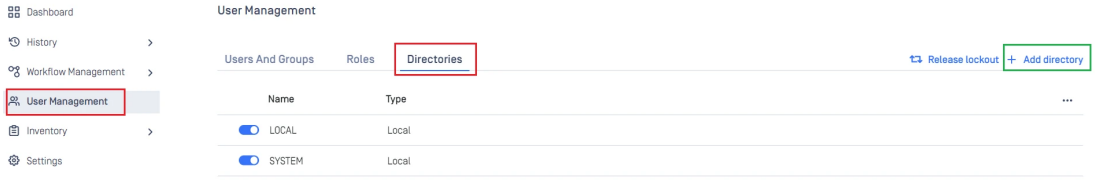


5. Fill `App name`, MDCore-OIDC for example and click `Next`.

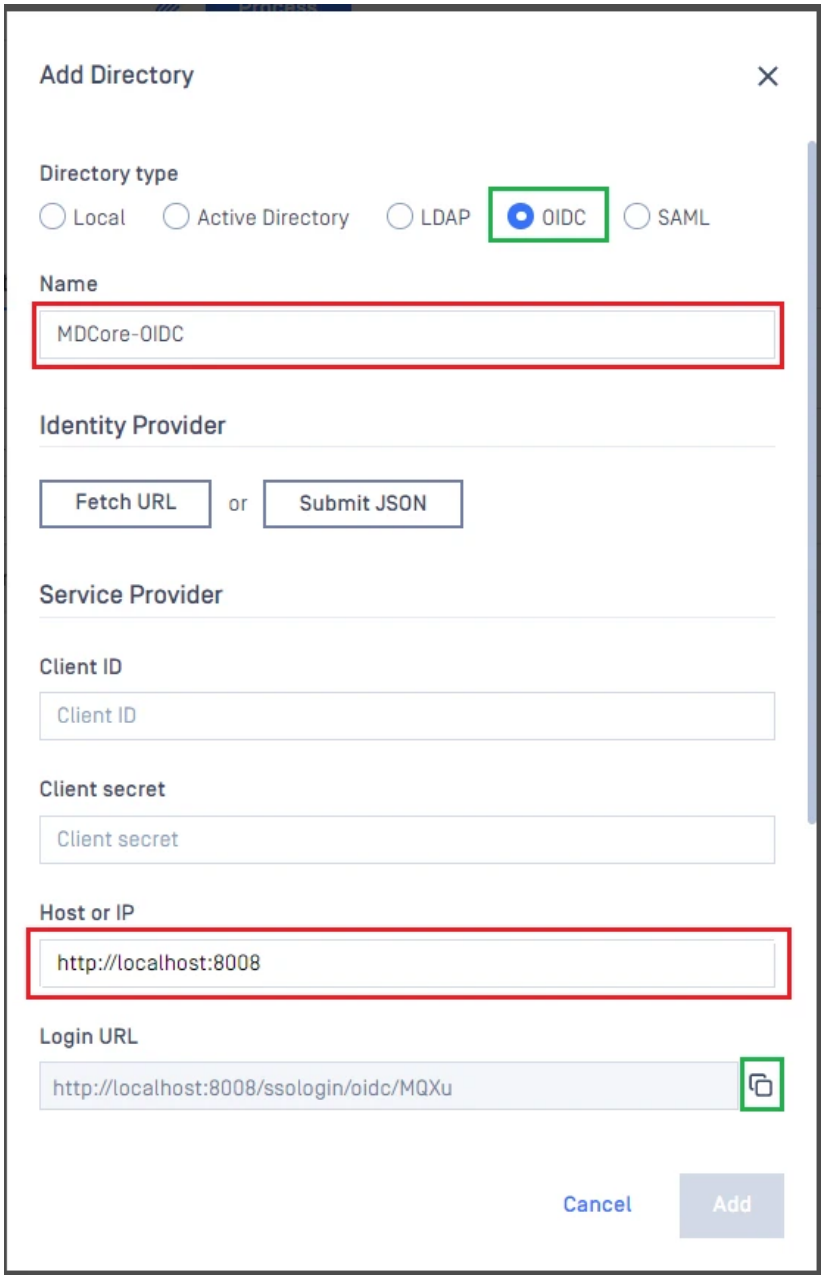


Create OIDC directory in MetaDefender Core

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.

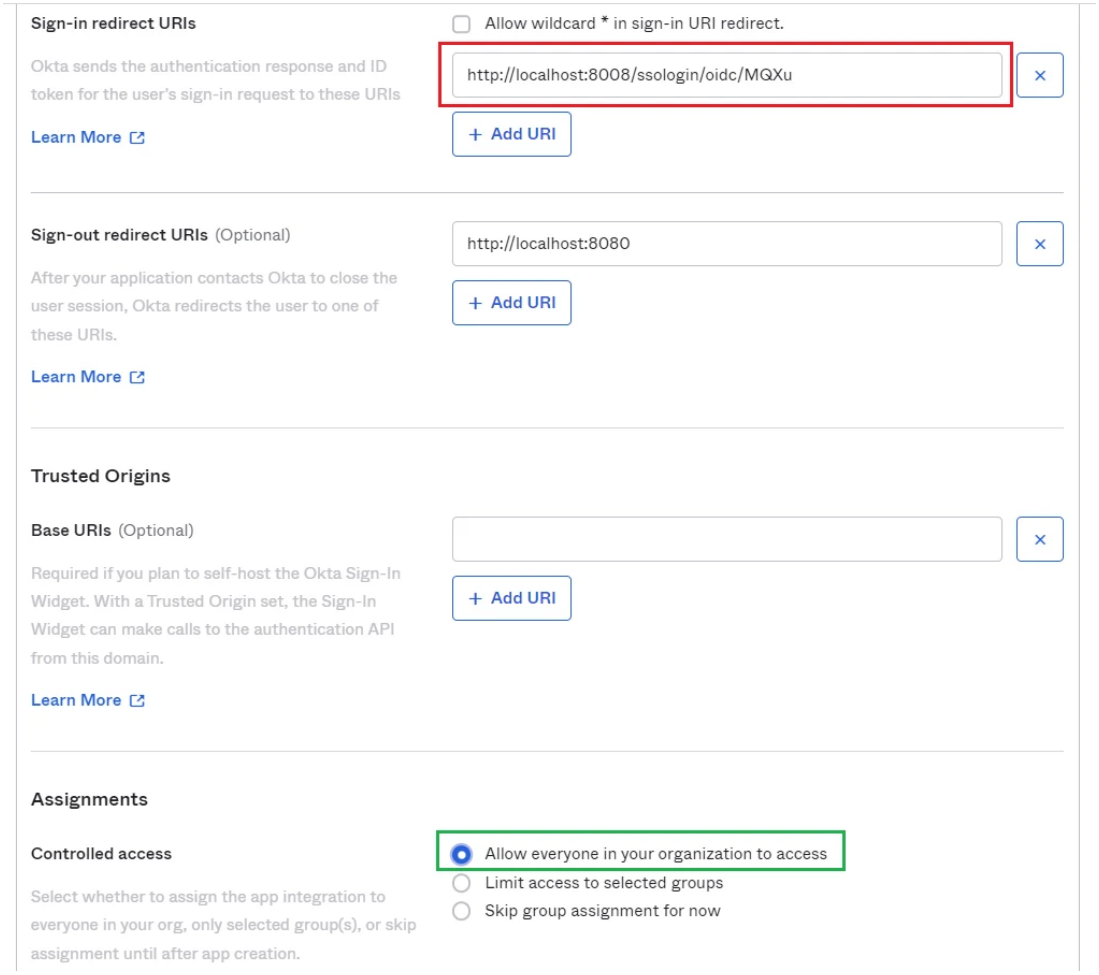


4. In `Add Directory` page, choose OIDC for `Directory Type`.
5. Fill `Name` of the new directory, MDCore-OIDC for example.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, *https://localhost:8008* for this example.

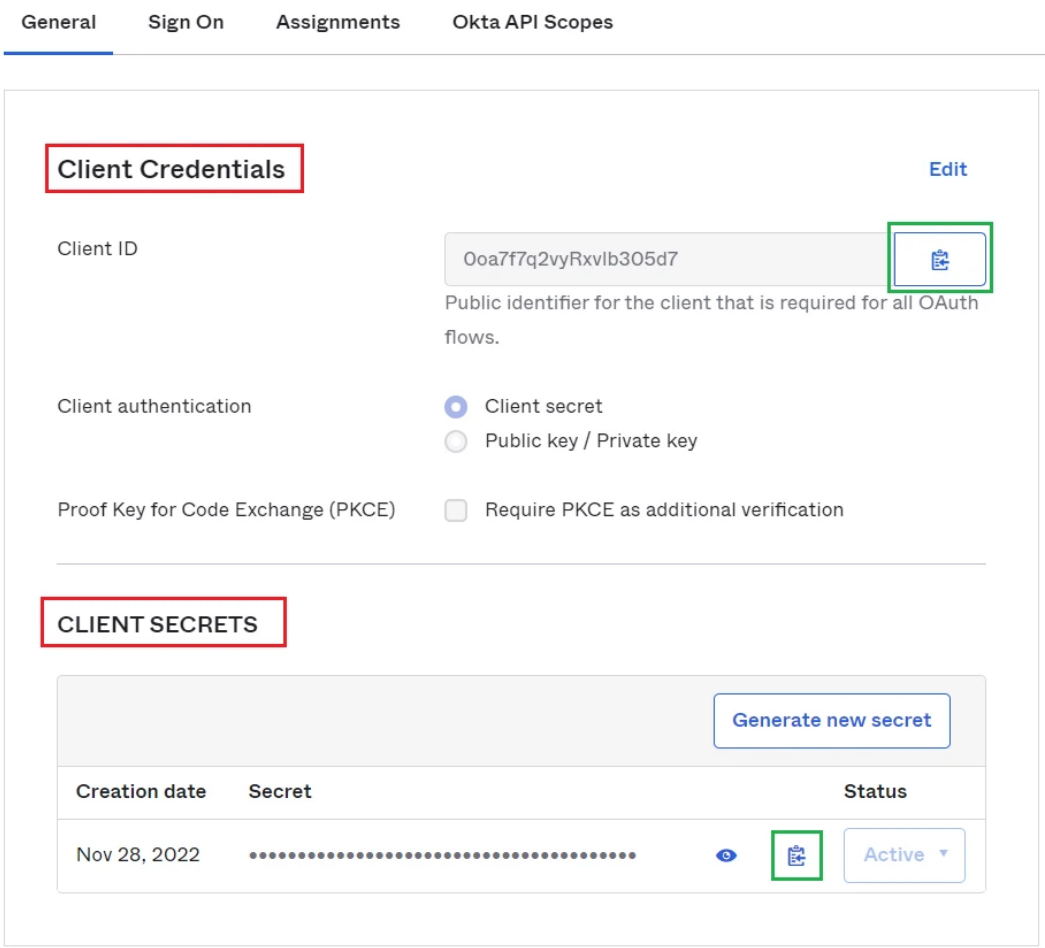


Complete configuration in Okta

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`.

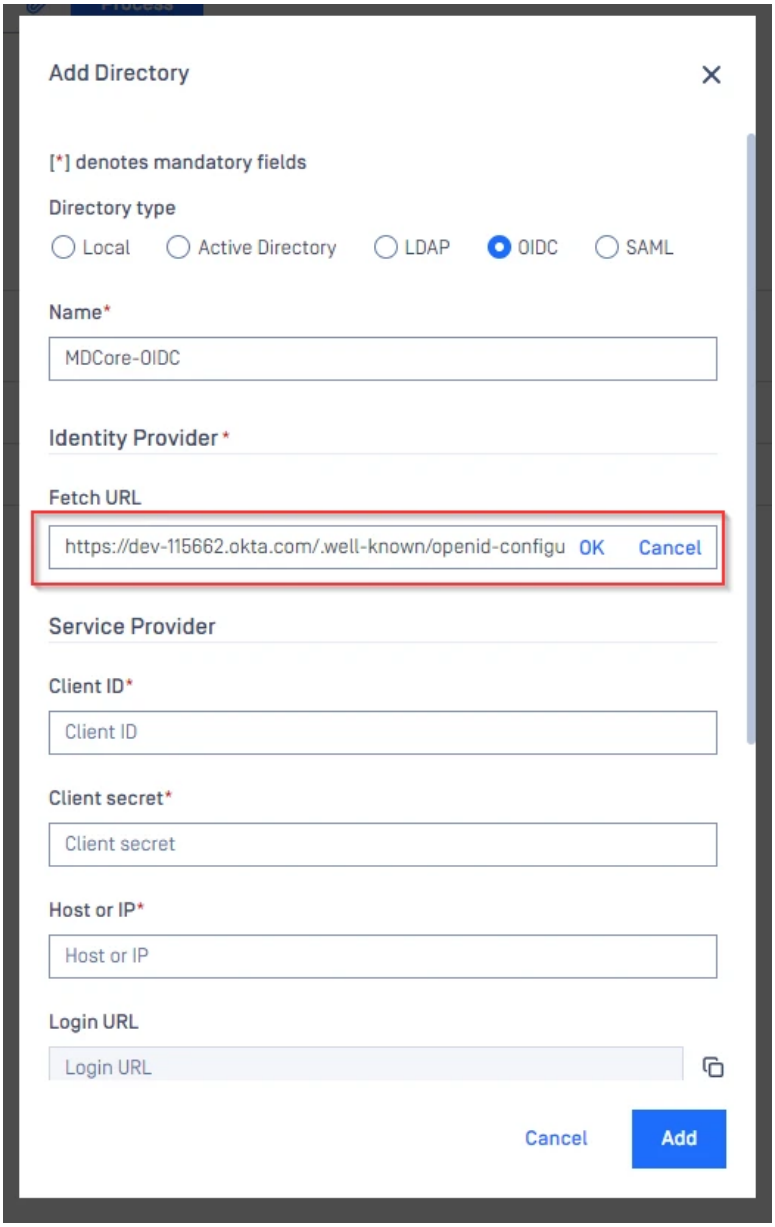


3. In `General` tab, copy `Client ID` and `Client Secret`.

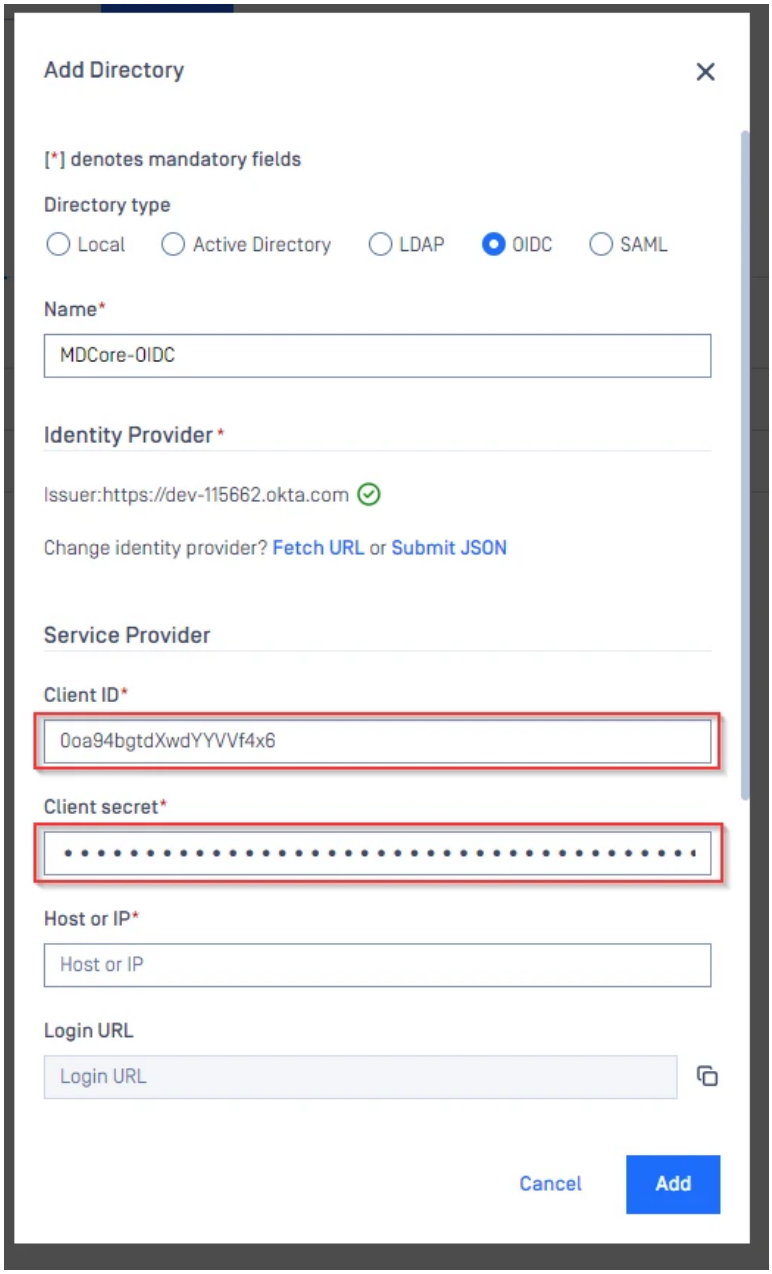


Complete configuration in MetaDefender Core

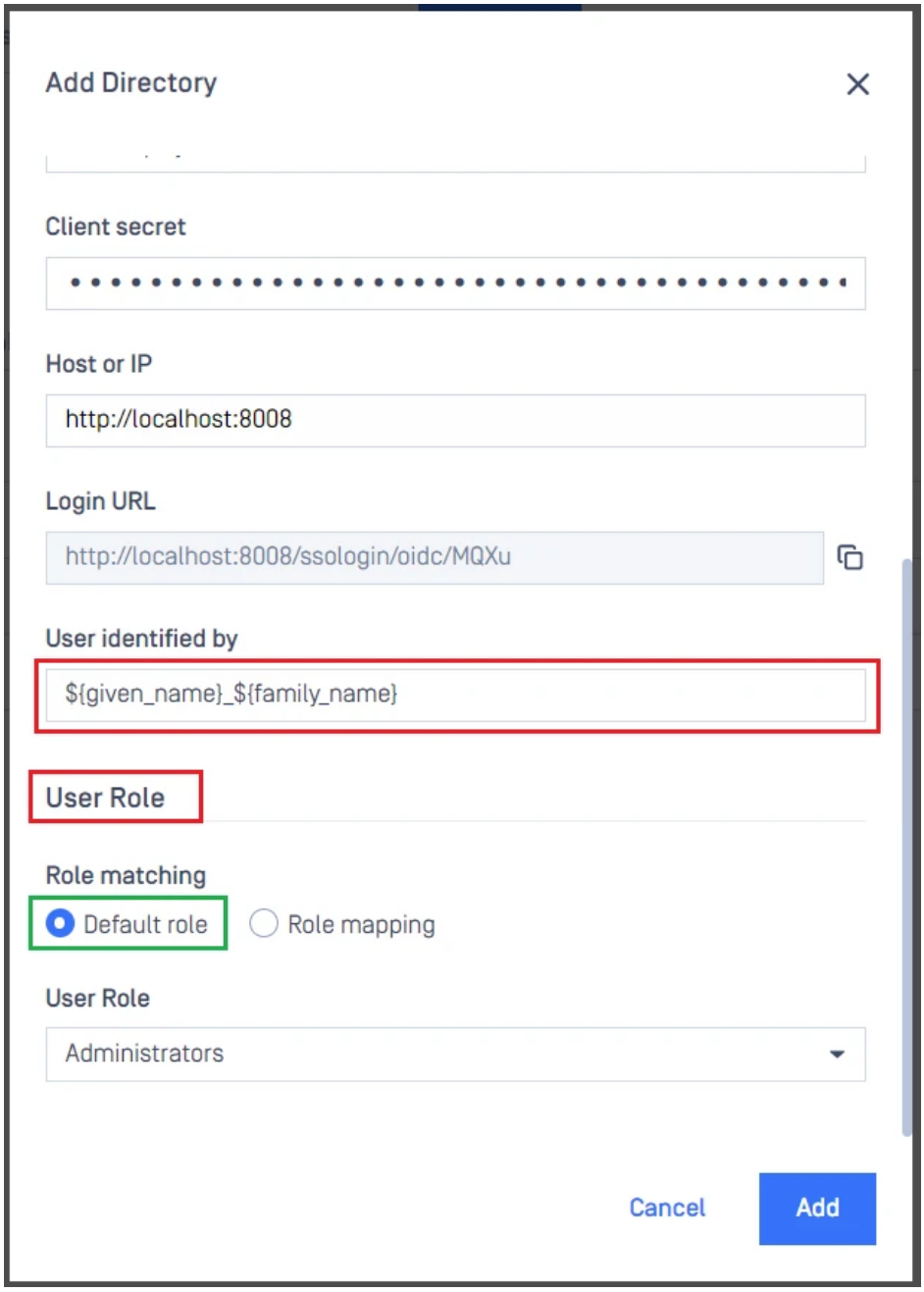
1. Switch to MetaDefender Core screen, under `Identity Provider`, click on `Fetch URL`.
2. OIDC link will be built by appending */.well-known/openid-configuration* to your Okta domain url (Referred Okta document). For example `dev-115662.okta.com/.well-known/openid-configuration`.
3. Paste OIDC configuration link above to the box under `Fetch URL` and hit `OK` to ensure MetaDefender Core can set Okta as its IdP.



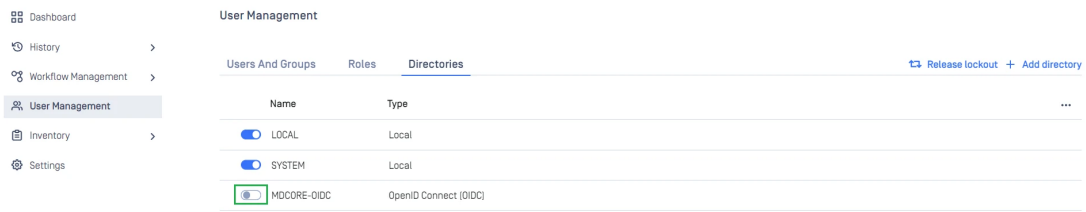
4. Under `Service Provider`, paste the value of `Client ID` and `Client secret` that you copied from Okta respectively.



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.

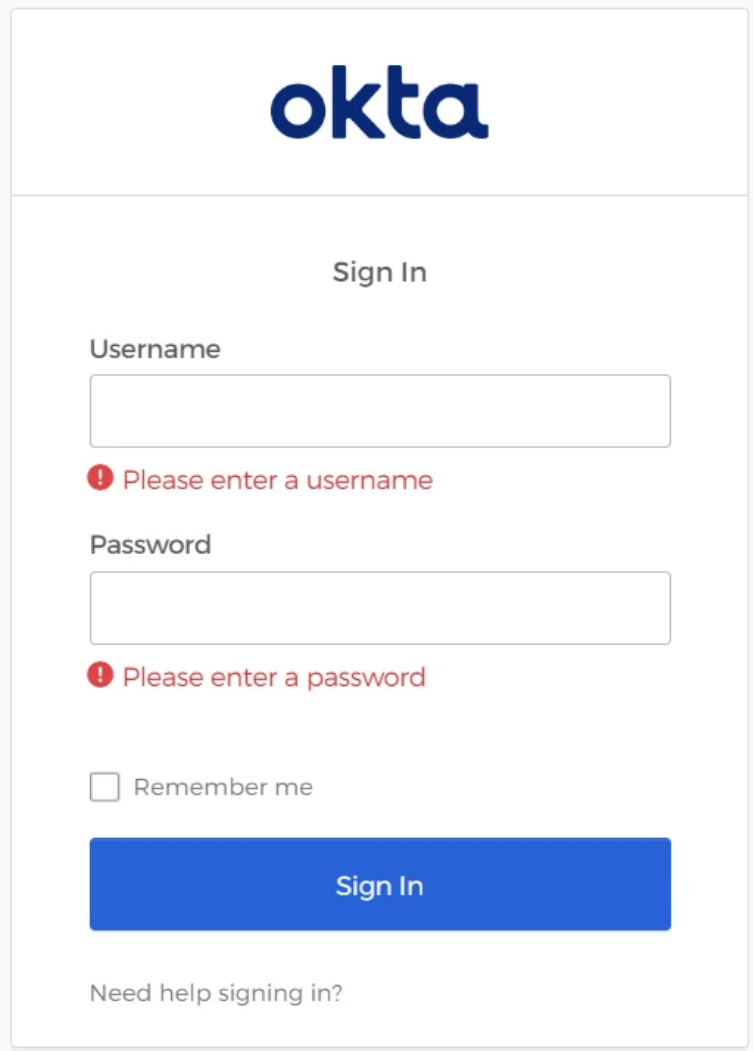


7. On `User Management` screen, toggle the new directory, MDCORE-OIDC in this example. A dialog box is shown to confirm the action. Once `Enable` is hit, all existing sessions will be expired immediately and Okta will be used to authenticate users going forward.

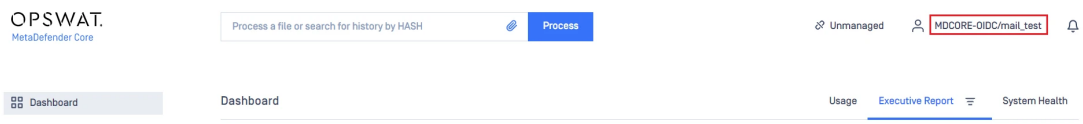


Test the integration

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`.

Configuring SAML Single Sign-on

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:

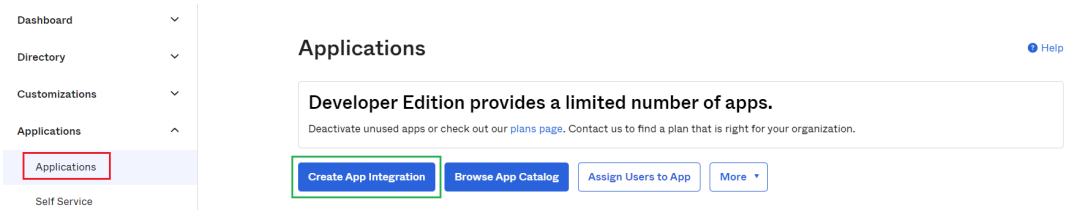
1. Sign to Okta console with your account.
2. In dashboard, expand `Applications` in the left sidebar and click `Applications`.
3. In your SSO integration, select `General` tab.
4. Navigate to `SAML Settings` and click `Edit`.
5. On `Edit SAML Integration` page, select `Configure SAML` tab.
6. Expand Advanced Settings, navigate to `Digest Algorithm` and select SHA256 or higher.

From MetaDefender Core,

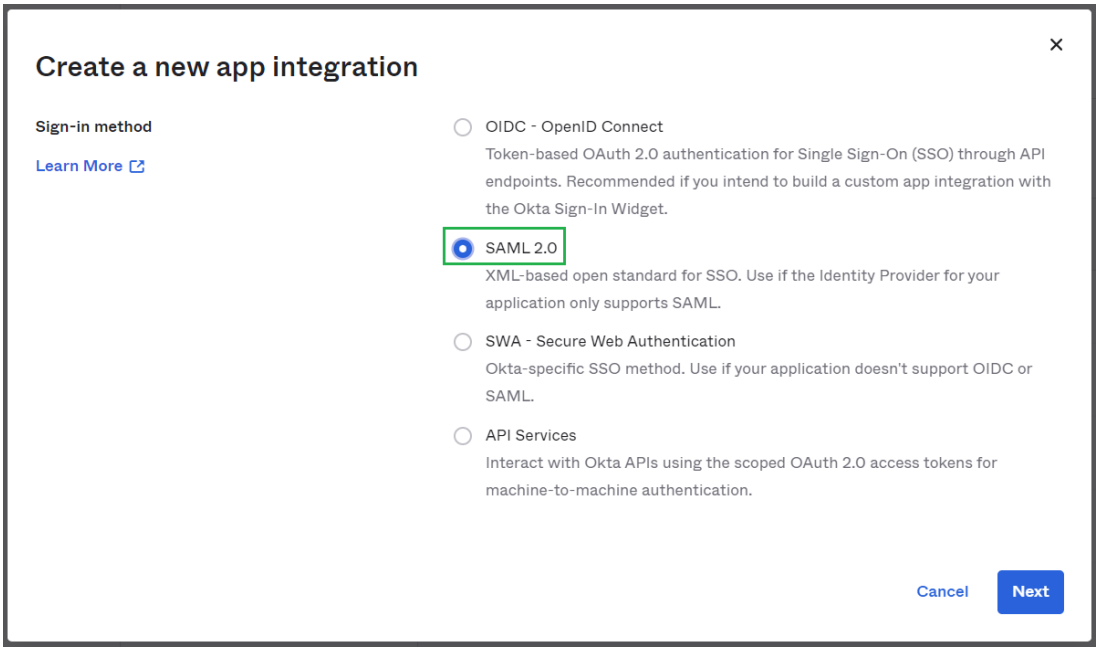
1. Sign in to MetaDefender Core with your account.
2. Select your SSO directory and click `Edit`.
3. Under `Identity Provider`, click `Fetch URL`.
4. Fill the metadata URL with SAML metadata link from Okta.
5. Click `Save Changes`.

Create Okta application

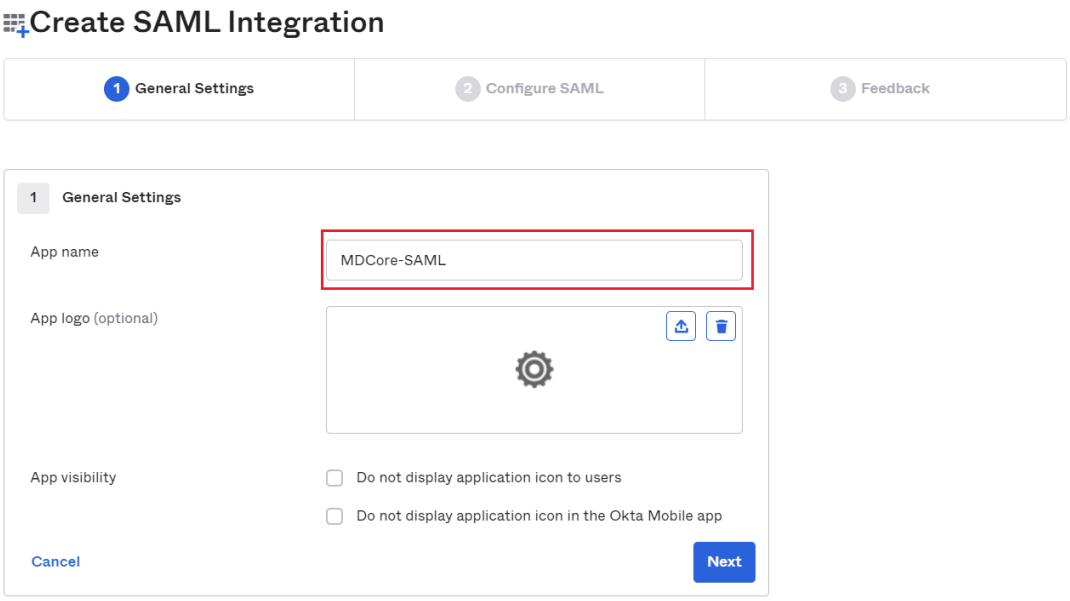
1. Access https://www.okta.com/login/ and sign in.
2. In sidebar of dashboard, hit `Applications` to drop sub items down, then select `Applications`.
3. Click `Create App Integration`.



4. In `Sign-in method`, choose `SAML 2.0` and click `Next`.

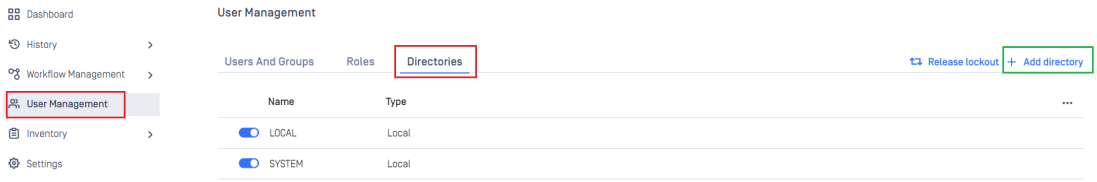


5. Fill `App name`, MDCore-SAML for example and click `Next`.

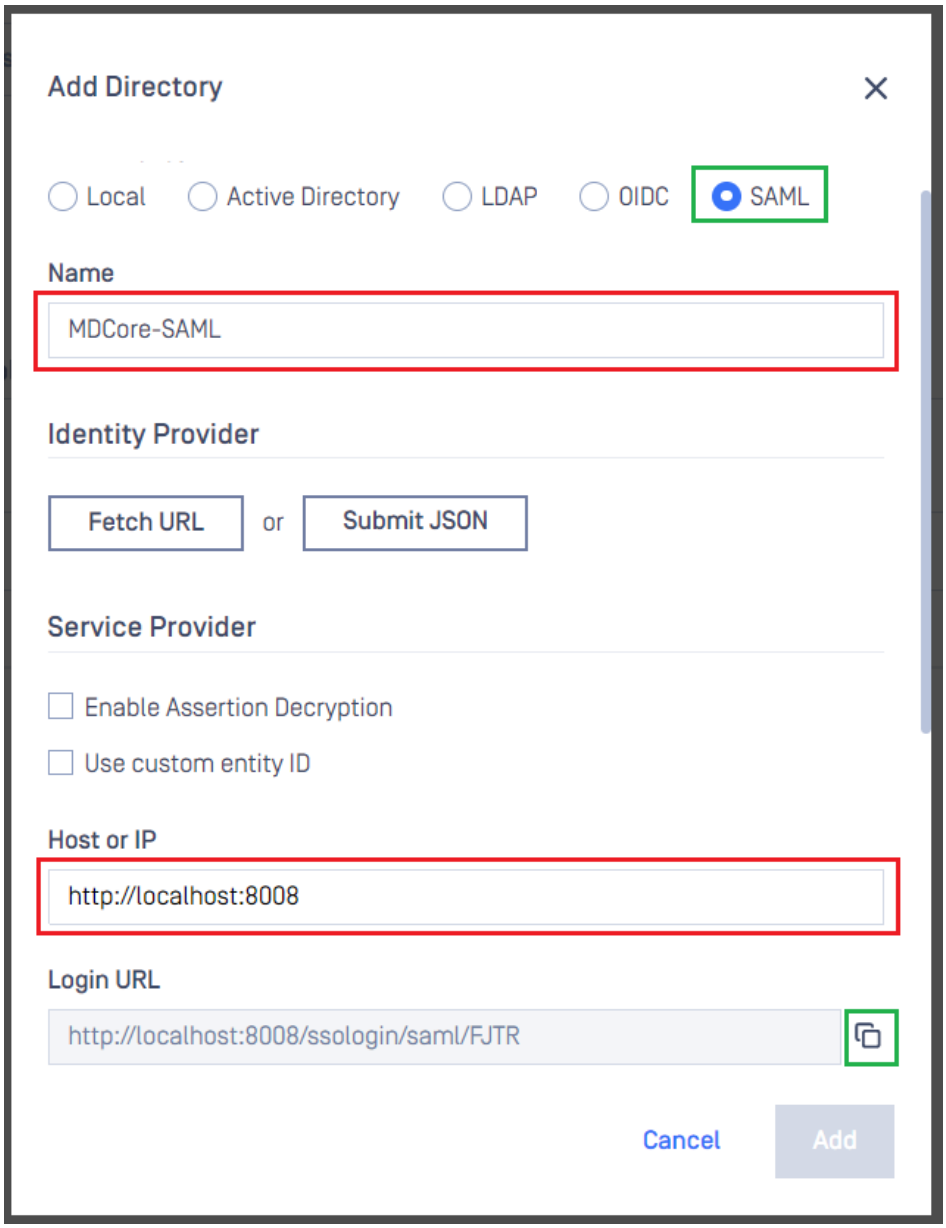


Create SAML directory in MetaDefender Core

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.

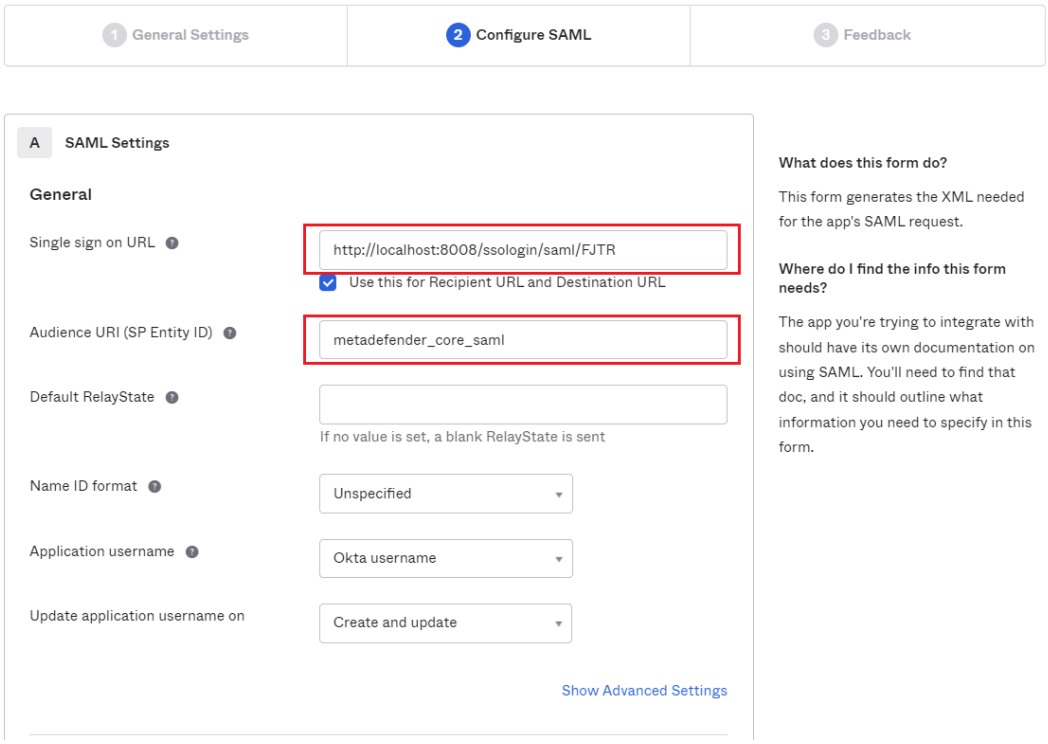


4. In `Add Directory` page, select SAML for `Directory Type`.
5. Fill `Name` of the new directory, MDCore-SAML for example.
6. Under `Service Provider`, fill in `Host or IP` where MetaDefender Core is being hosted, *https://localhost:8008* for this example.

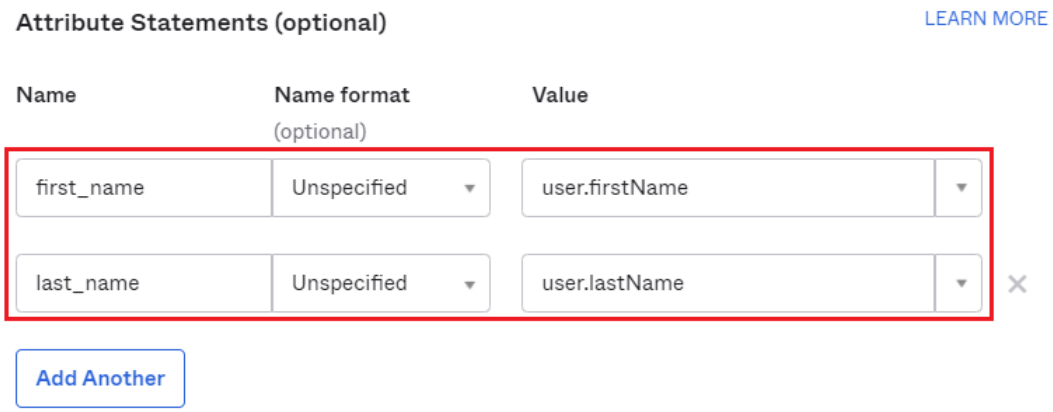


Configure Okta

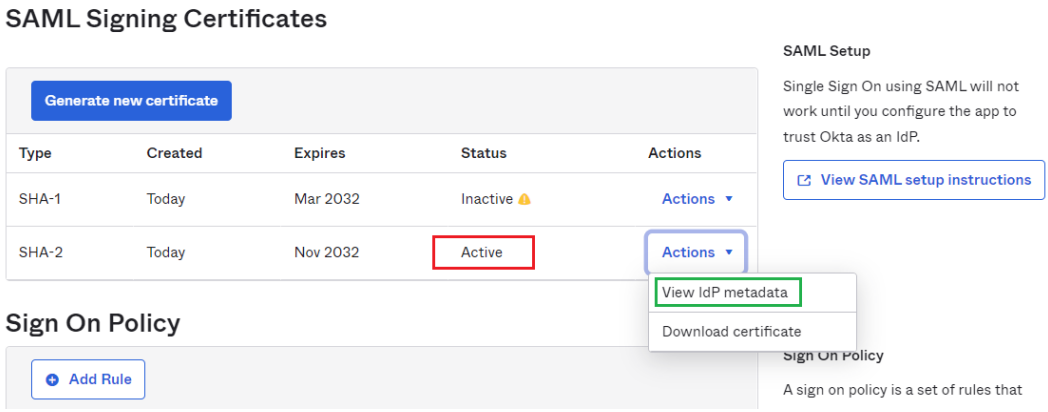
1. Switch to `Applications` screen in Okta home page.
2. Paste the `Login URL` from MetaDefender Core into `Single sign on URL` and set ID for `Audience URI`, metadefender_core-saml for example.



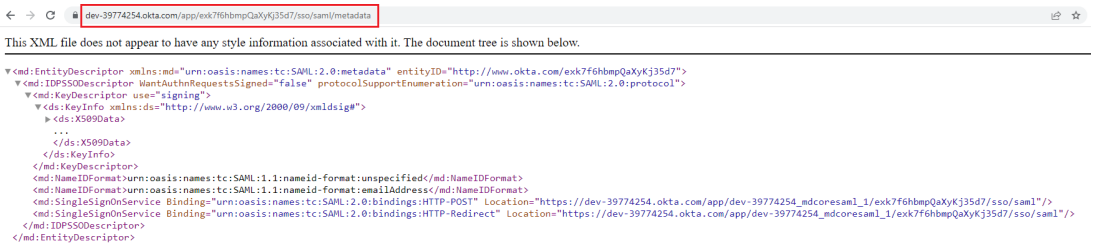
3. Scroll down, navigate to `Attribute Statements` and set mapping from Okta key name to the one set at `User identified by` in MetaDefender Core, then hit `Next`.



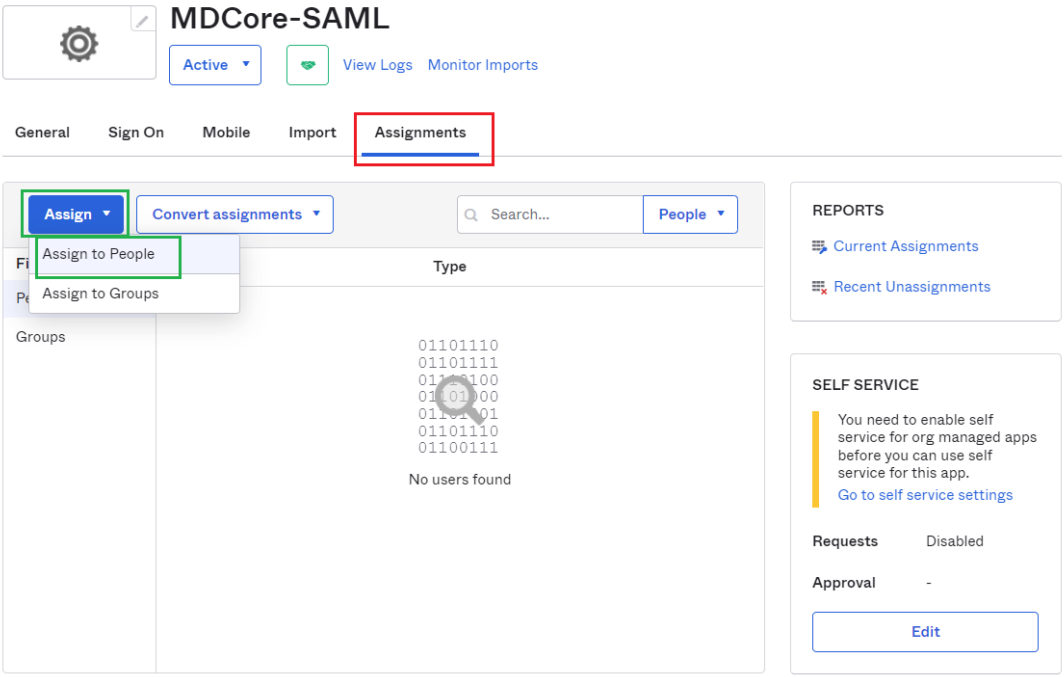
4. Choose your answer at `Feedback` screen and click `Finish`.
5. On the next screen, navigate to `SAML Signing Certificates`, choose the row that has value of column `Status` is "Active" then drop `Actions` down and click `View IdP metadata`.



6. Copy SAML Metadata link.

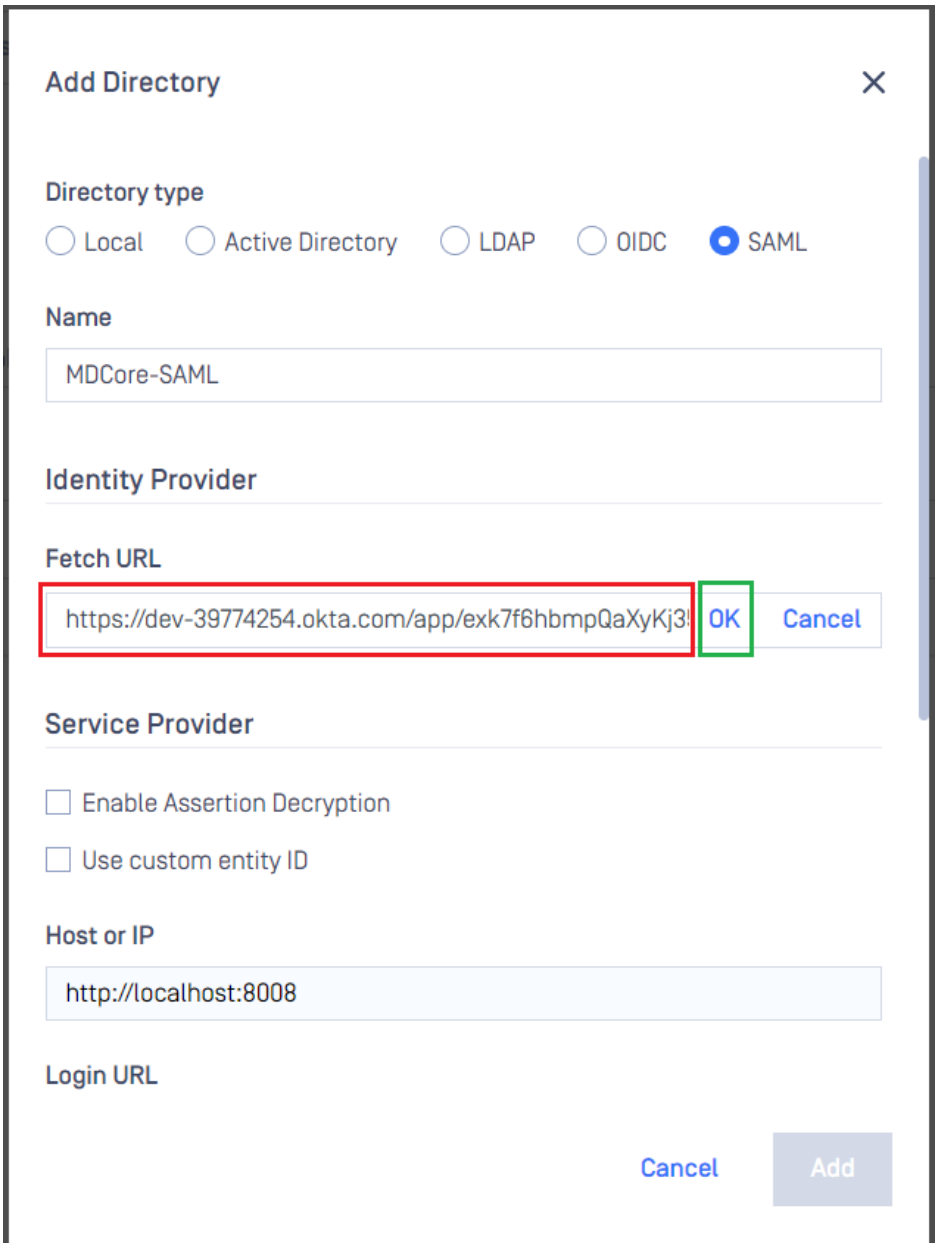


7. Switch to `Applications` screen in Okta, navigate to `Assignments` tab, and assign users for the application.

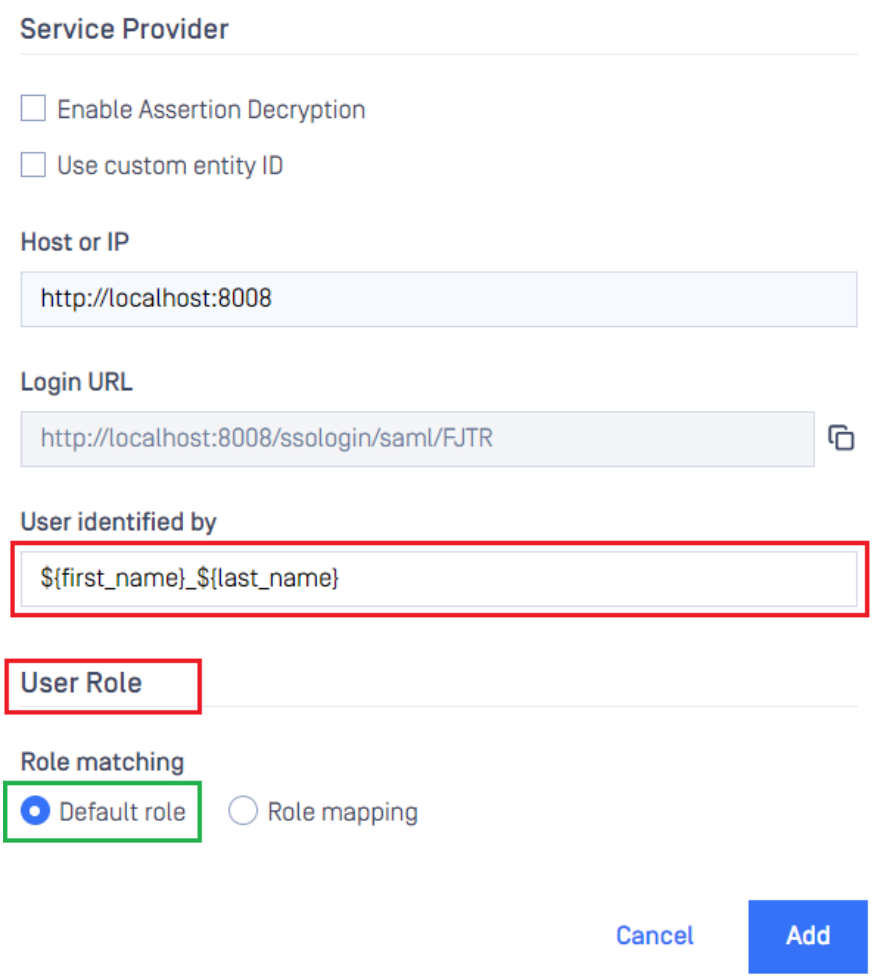


Complete configuration in MetaDefender Core

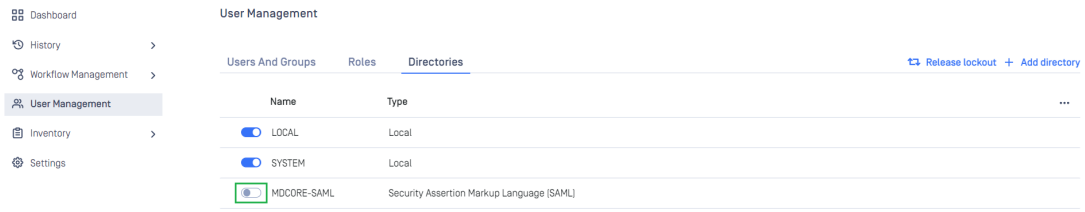
1. Switch to MetaDefender Core screen, under `Identity Provider`, click on `Fetch URL`.
2. Paste "SAML Metadata link" from Okta to the box under `Fetch URL` and click `OK` to ensure MetaDefender Core can set Okta as its IdP.



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.

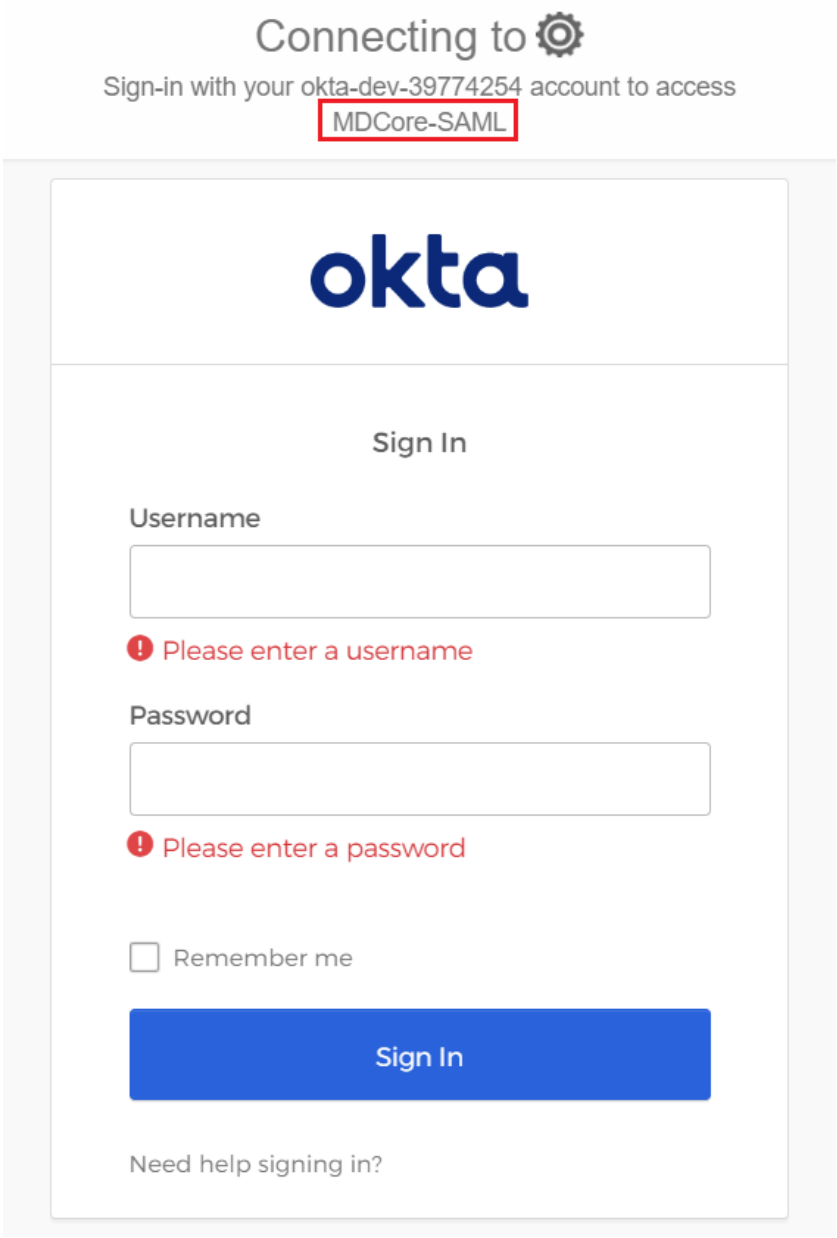


6. In `User Management` screen, toggle the new directory, MDCORE-SAML in this example. A dialog box is shown to confirm the action. Once `Enable` is hit, all existing sessions will be expired immediately and Okta will be used to authenticate users going forward.

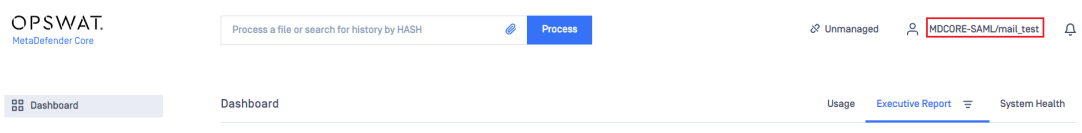


Test the integration

1. Browse MetaDefender Core, hit `Login`, user is redirected to Okta's sign-in page.



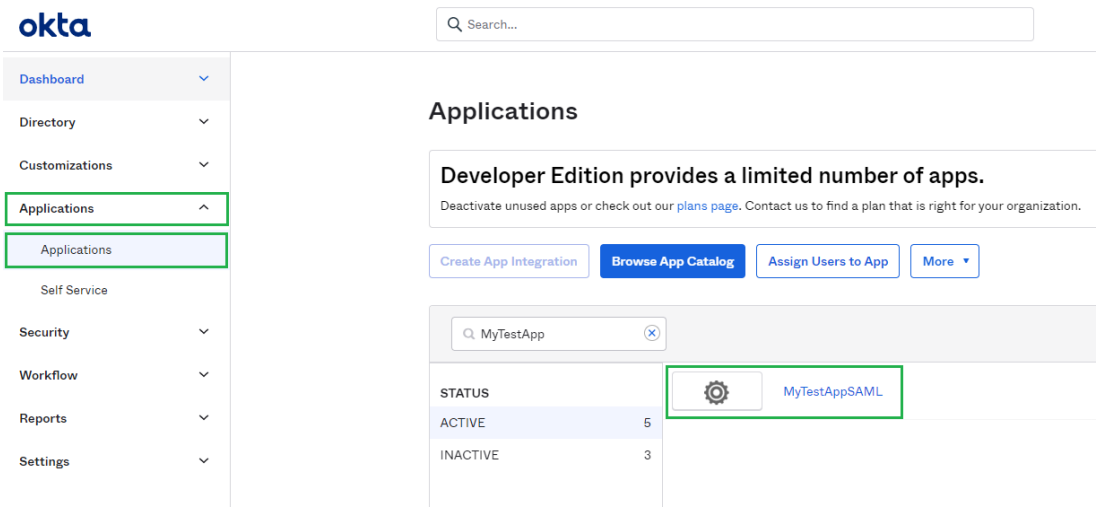
2. Sign in by the account registered in Okta.
3. If everything goes right, MetaDefender Core dashboard is shown with user identity set at the top right corner.



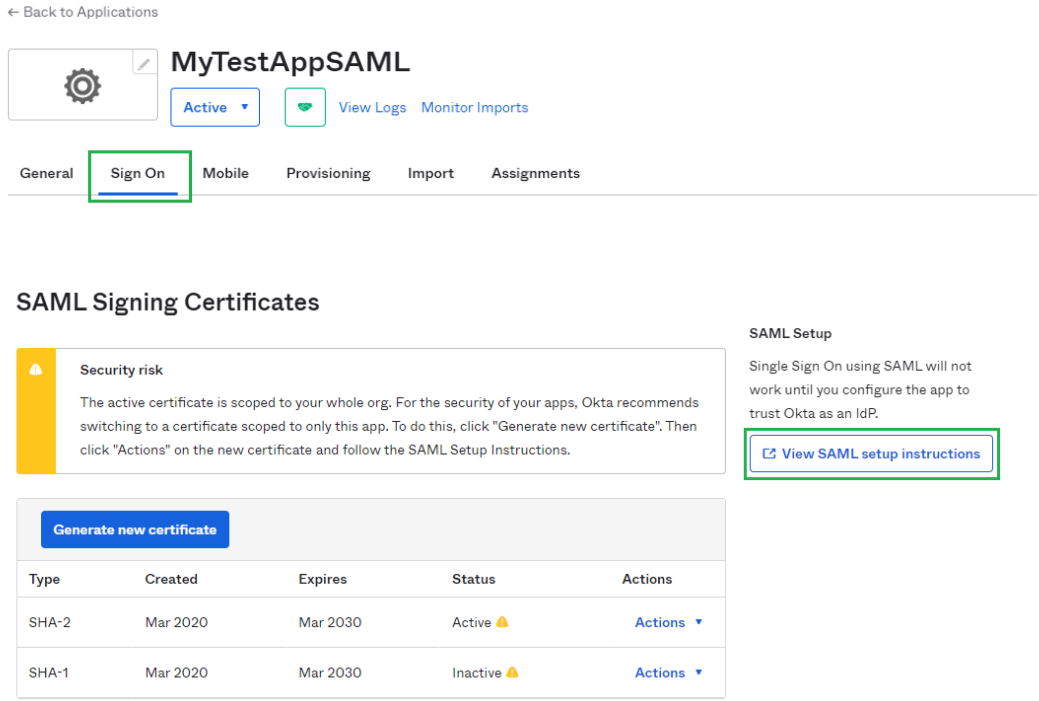
4. For troubleshooting, browse `<mdcore-host>#/public/backuplogin`.

Test IdP-initiated SSO

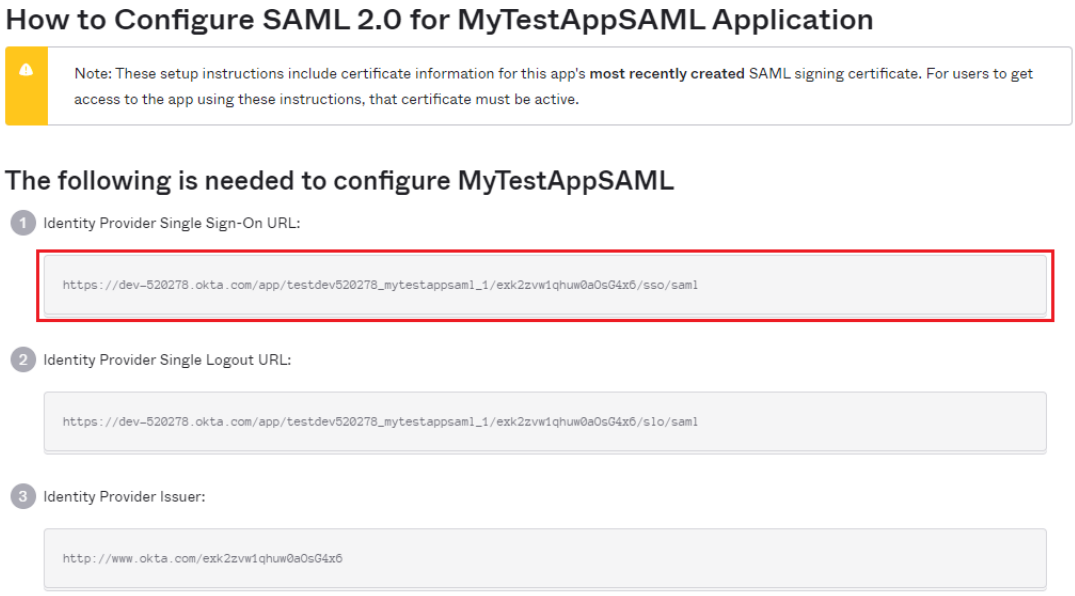
1. Sign in to Okta management page.
2. Navigate to `Applications` / `Applications` on the sidebar.
3. Select the active SAML application.



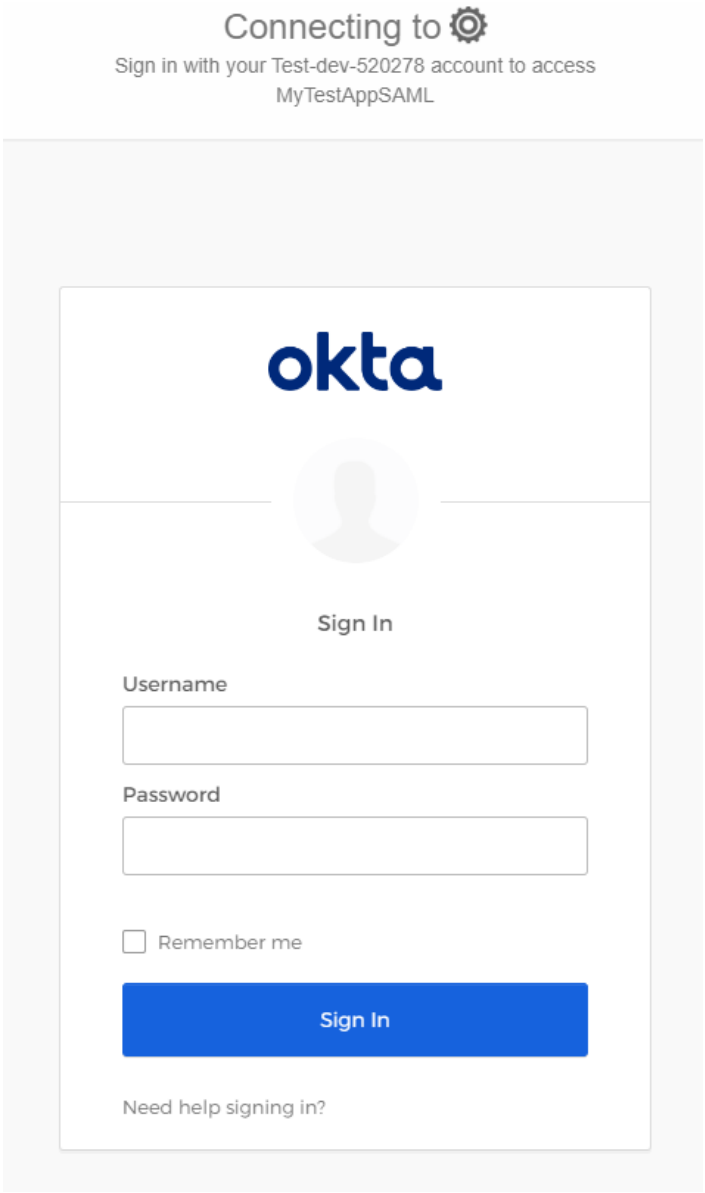
4. In `Sign On` tab, find and hit `View SAML setup instructions` button.



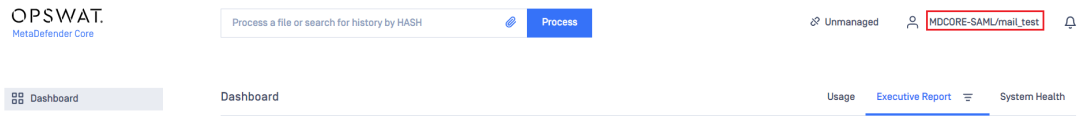
5. In the new page, copy the URL under `Identity Provider Single Sign-On URL` section.



6. Paste the copied URL to a web browser and sign in.



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`.

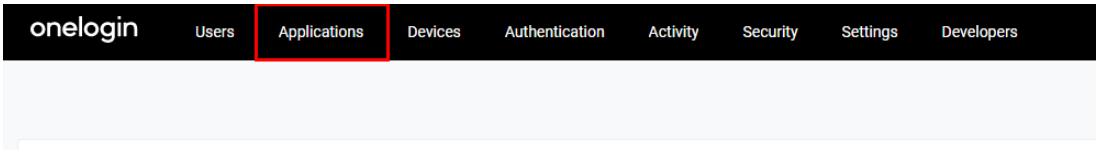
OneLogin

The instructions guide users on how to configure OneLogin SAML and OIDC Single Sign-on for MetaDefender Core.

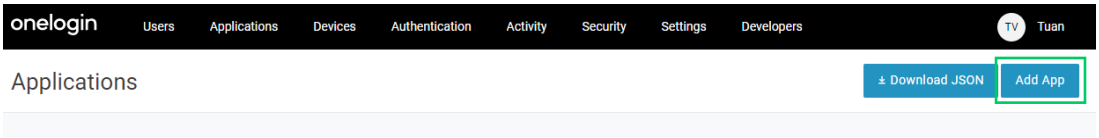
Configuring SAML Single Sign-on

Create OneLogin application

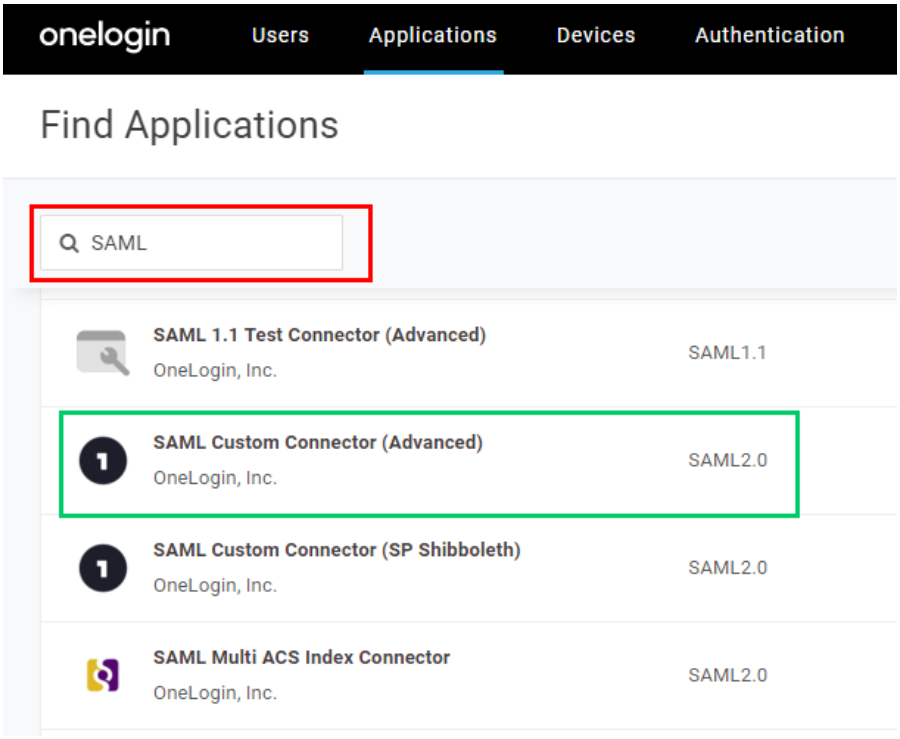
1. Access OneLogin and sign in.
2. On the navigation bar, click `Administration` to access administrator mode, then click on `Applications` to add a new application for integration.



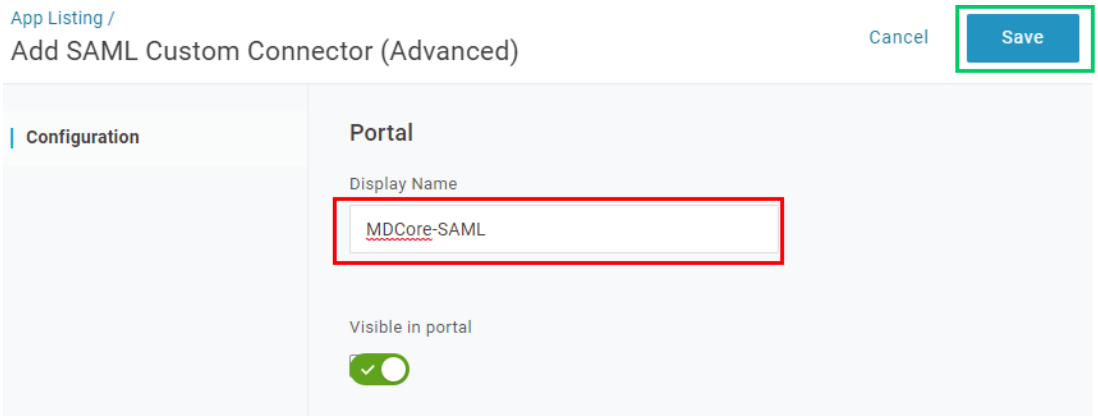
3. Click `Add App` to add a new application.



4. Enter "SAML" in the search box and select `SAML Custom Connector (Advanced)` version SAML2.0 in the results panel.

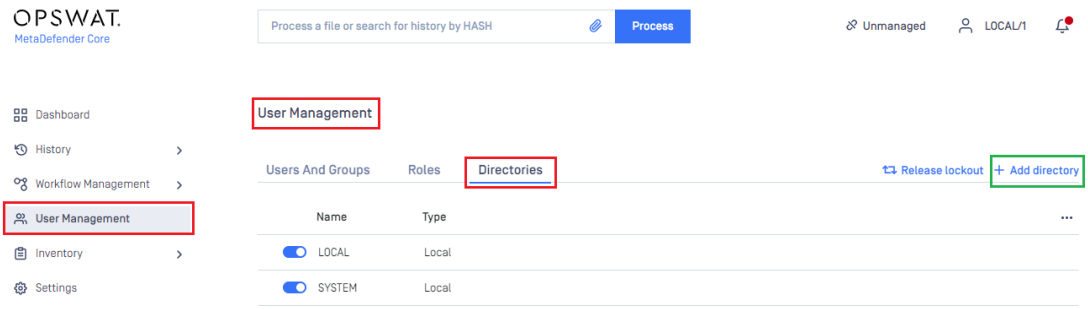


5. In the new page, fill in `Display Name` with MDCore-SAML, for example, and click `Save`.

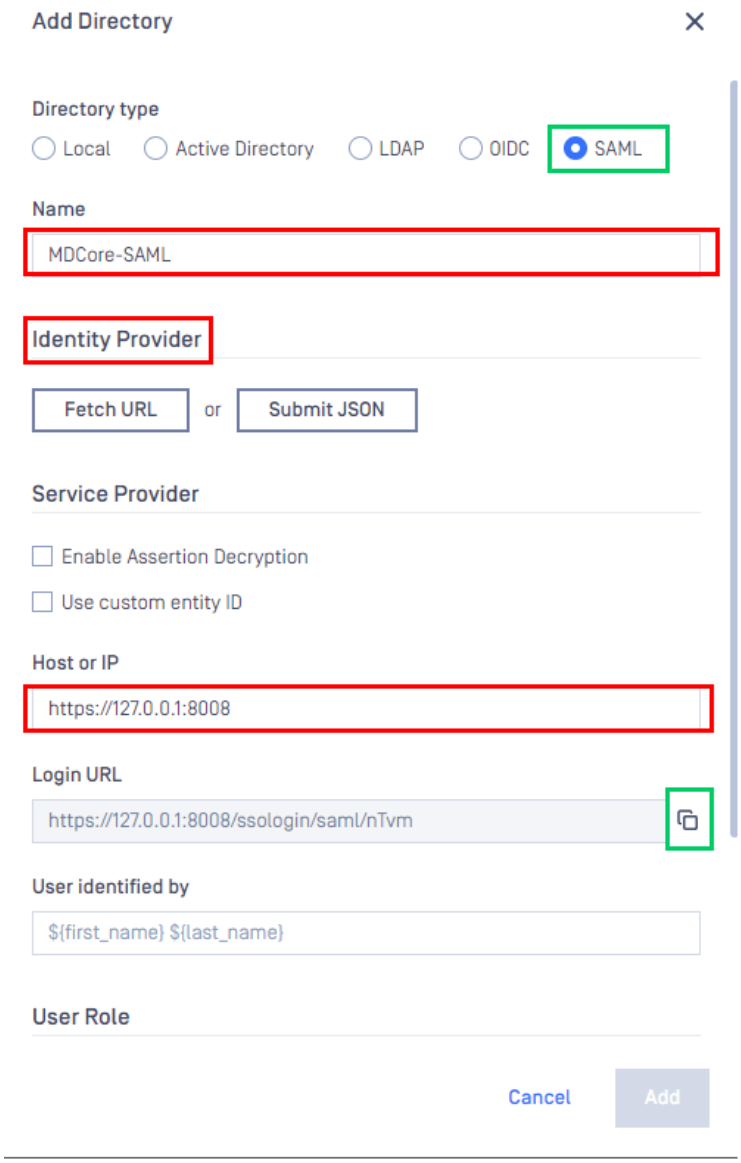


Create SAML directory in MetaDefender Core

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.

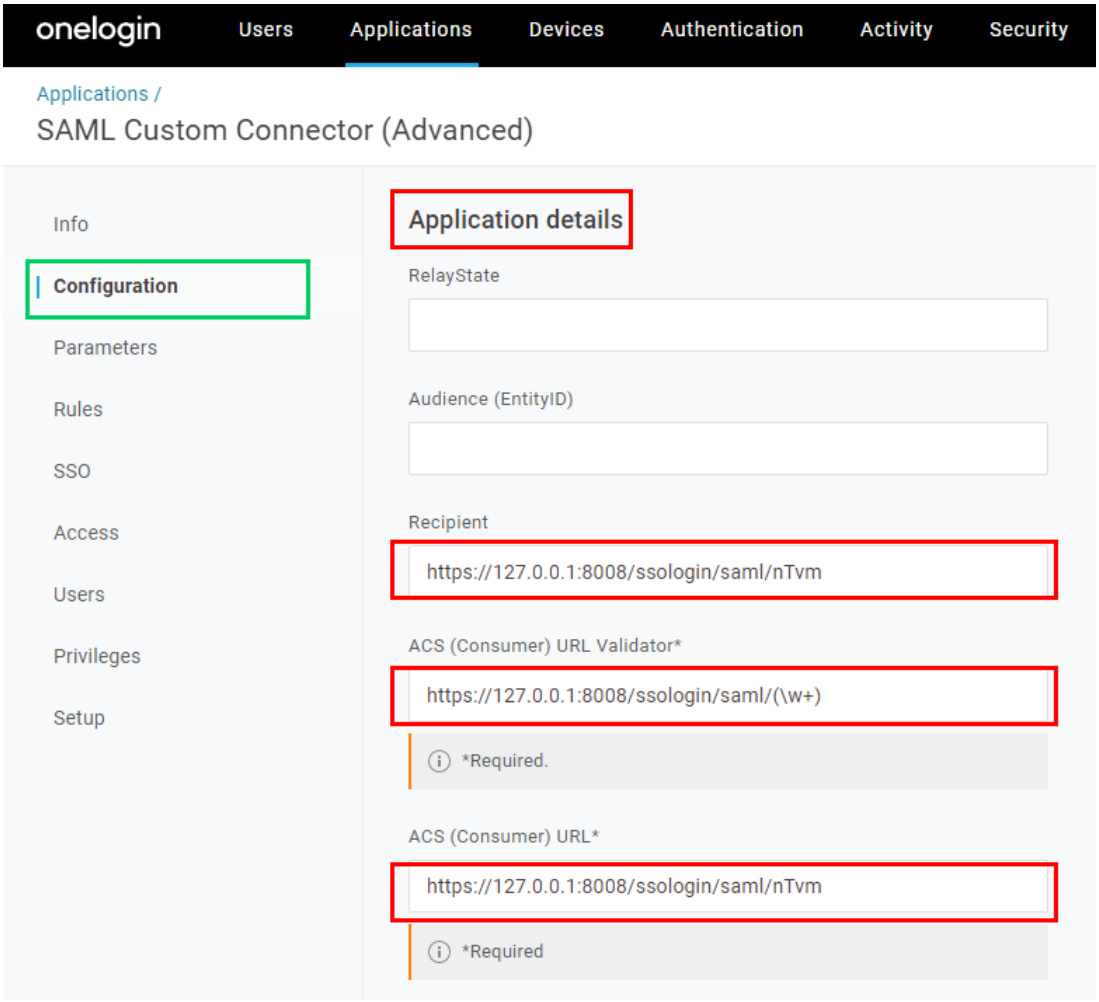


4. In `Add Directory` page, select SAML as `Directory type`, and enter a name for the new directory, such as MDCore-SAML.
5. In `Service Provider` section, enter `Host or IP` where MetaDefender Core is hosted, for this example use https://127.0.0.1:8008.
6. Copy the string generated under `Login URL` and store it as *reply_uri*.

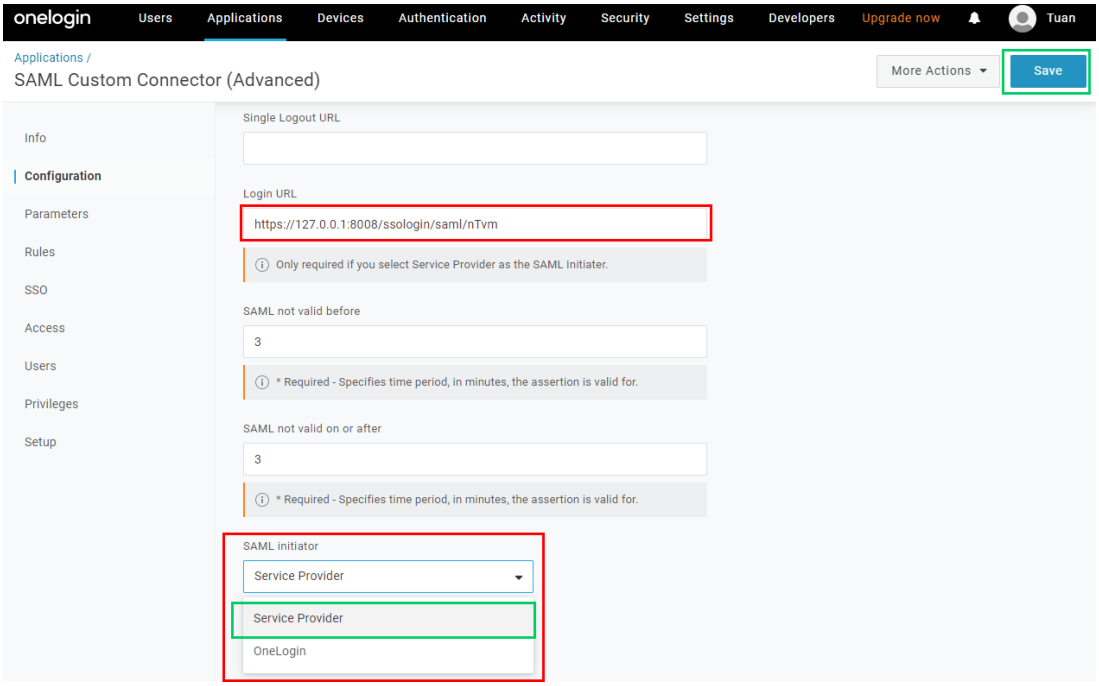


Complete configuration in OneLogin

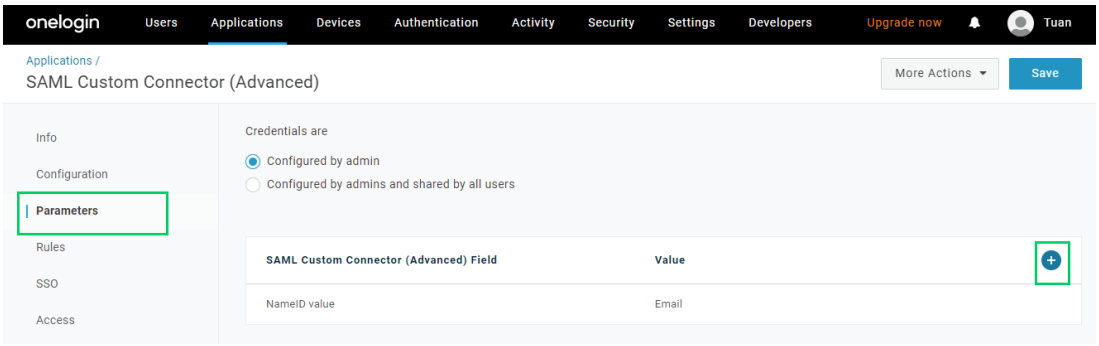
1. In `Applications` screen on OneLogin, select `Configuration` tab.
2. In `Application Details` section, fill in `Recipient` and `ACS (Consumer) URL*` with the `Login URL` copied from MetaDefender Core. Fill in `ACS (Consumer) URL Validator*` with a regular expression version of `Login URL`.



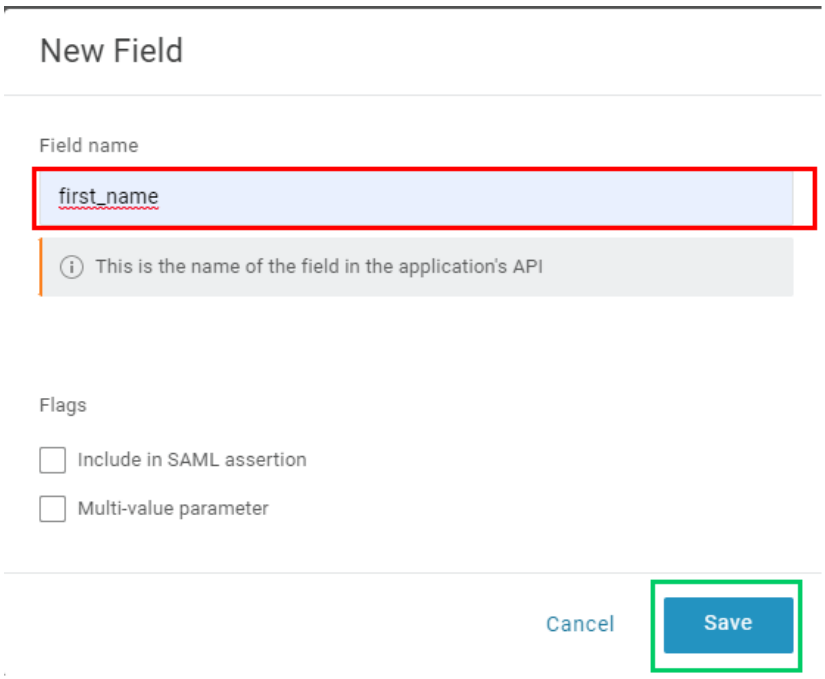
3. Expand `SAML Initiator` section, select `Service Provider`, enter the `Login URL` copied from MetaDefender Core and click `Save`.



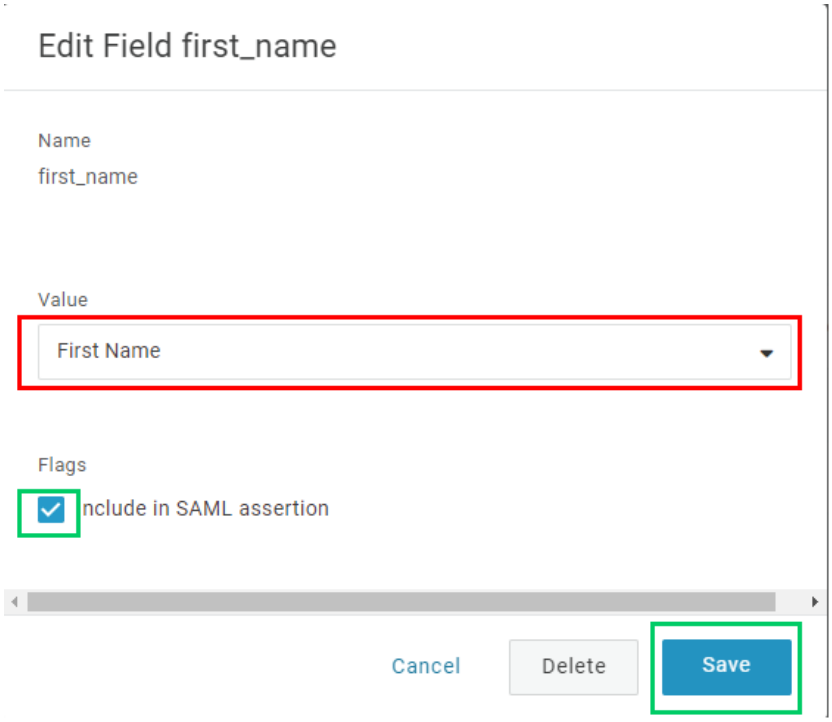
4. Select `Parameters` tab and click the plus button to add more fields that will be used to identify the signed-in users in MetaDefender Core.



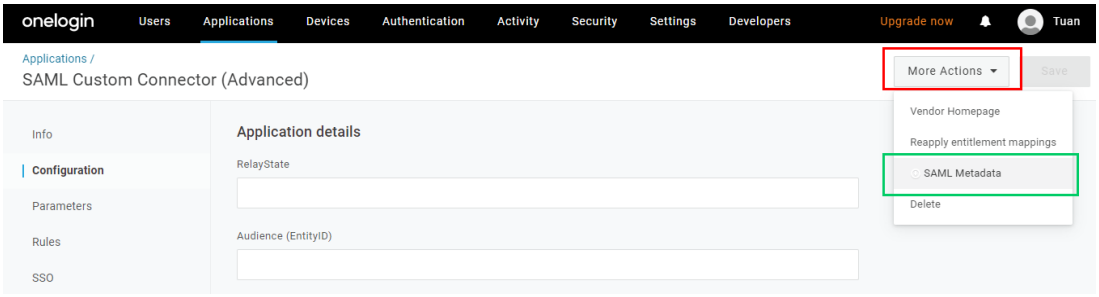
5. Fill in `Field name` with first_name and click `Save`.



6. Expand `Value` section, choose an appropriate attribute, First Name is an example. Then toggle `Include in SAML assertion` and click `Save`.

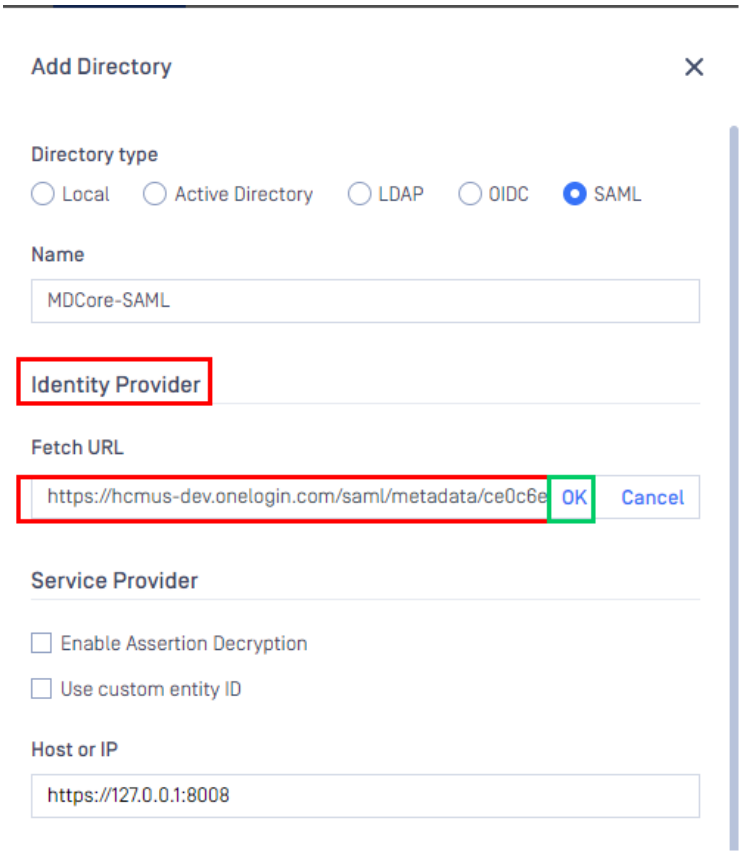


7. Click `Save` in the top right corner.
8. Drop `More Actions` down and click on `SAML Metadata` to copy *metadata_url*.

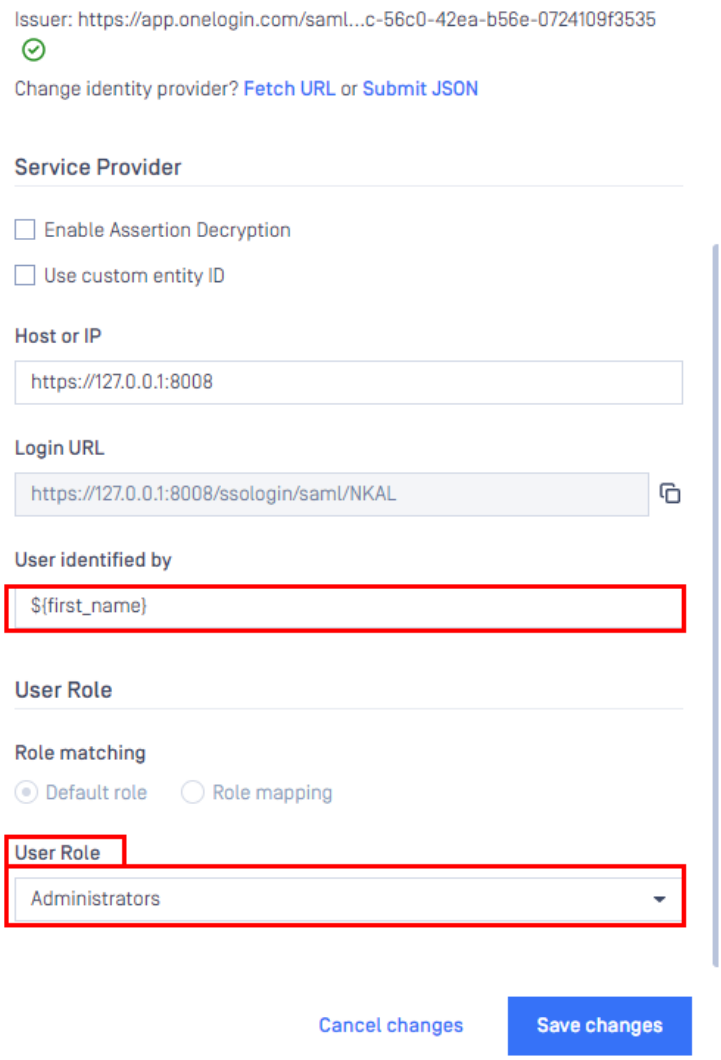


Complete configuration in MetaDefender Core

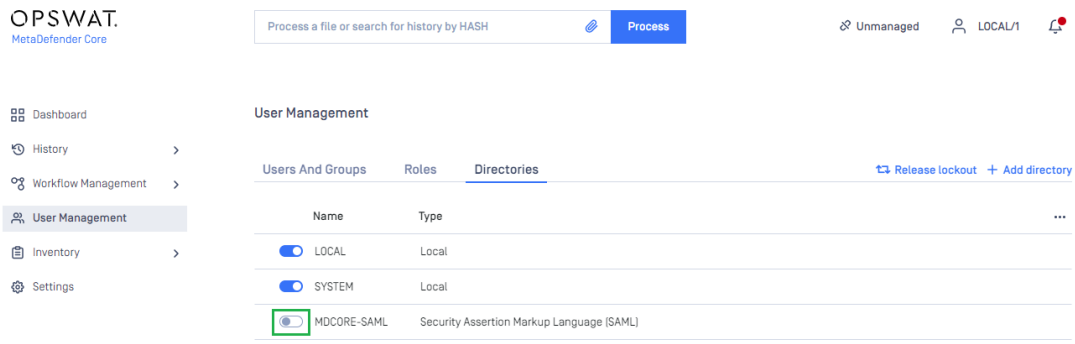
1. Switch back to MetaDefender Core. Under `Identity Provider`, click `Fetch URL`, paste the value of *metadata_url* into the box below, click `OK`, and wait a moment for MetaDefender Core to check and set OneLogin as its IdP.



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.

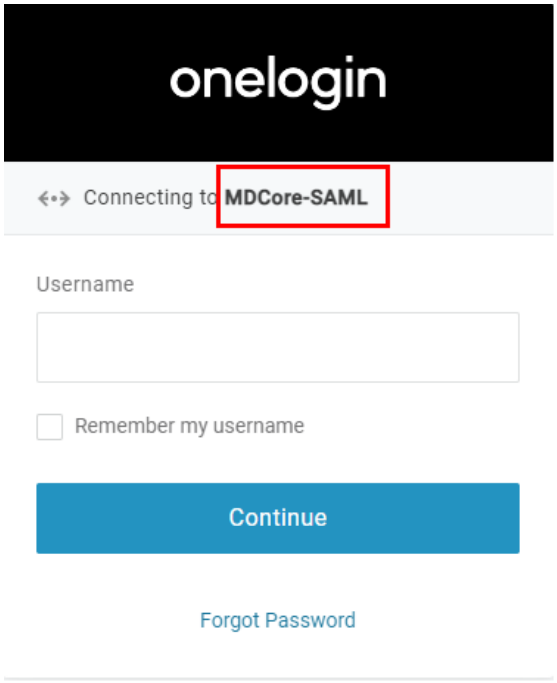


4. On `User Management` page, toggle the new directory, MDCORE-SAML in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked , all sessions will expire immediately.

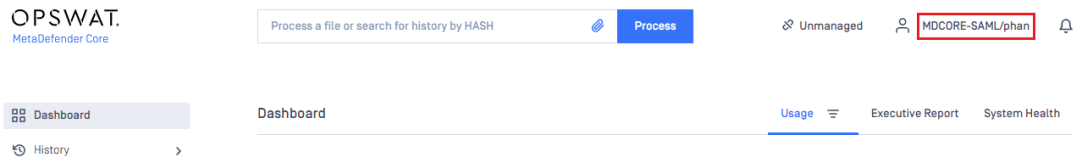


Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to OneLogin page.



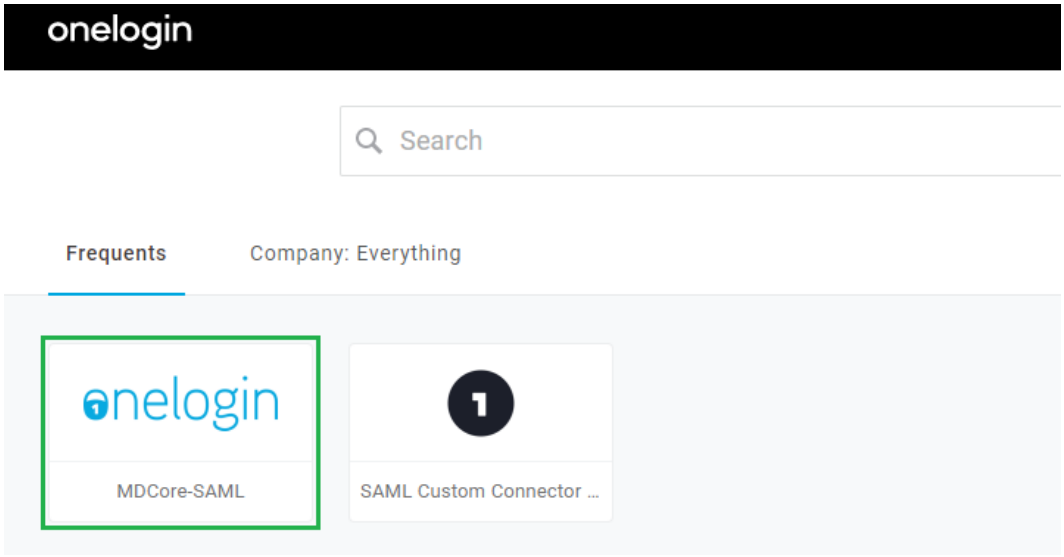
2. Sign in using the account registered with OneLogin.
3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.



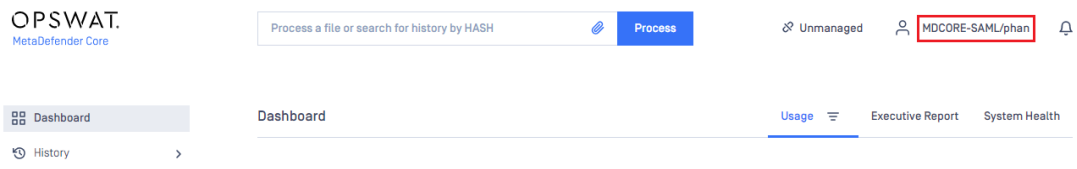
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

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.

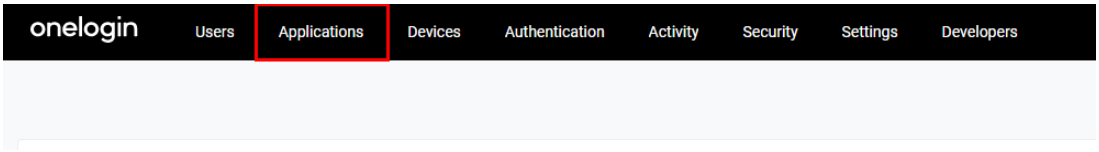


4. Otherwise, access backup login page at <mdcore-host>#/public/backuplogin for trouble shooting.

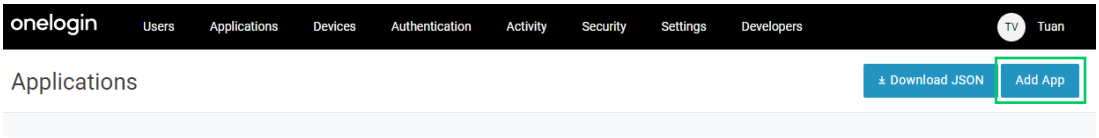
Configurating OIDC Single Sign-on

Create OneLogin application

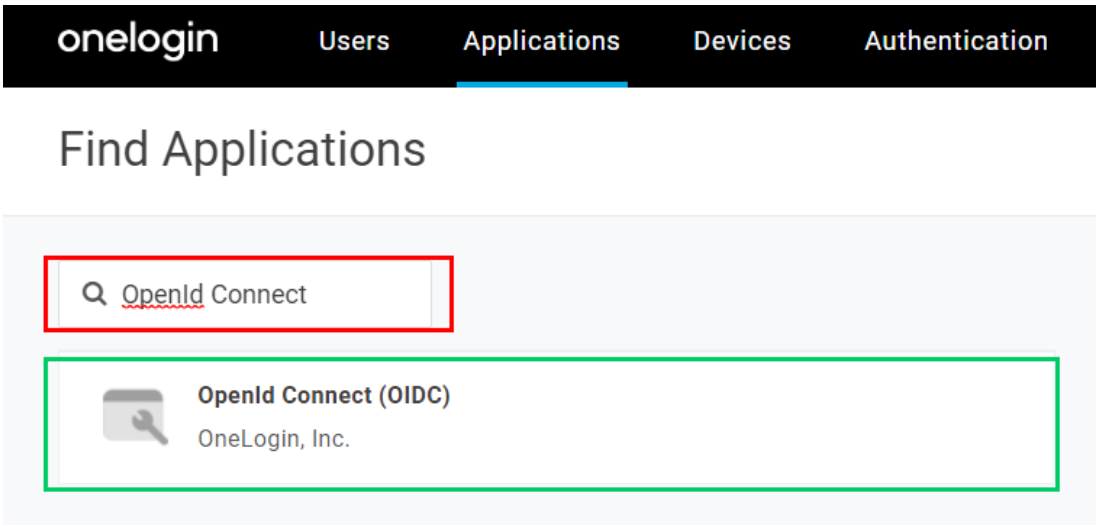
1. Access OneLogin and sign in.
2. On the navigation bar, click `Administration` to access administrator mode, then click on `Applications` to add a new application for integration.



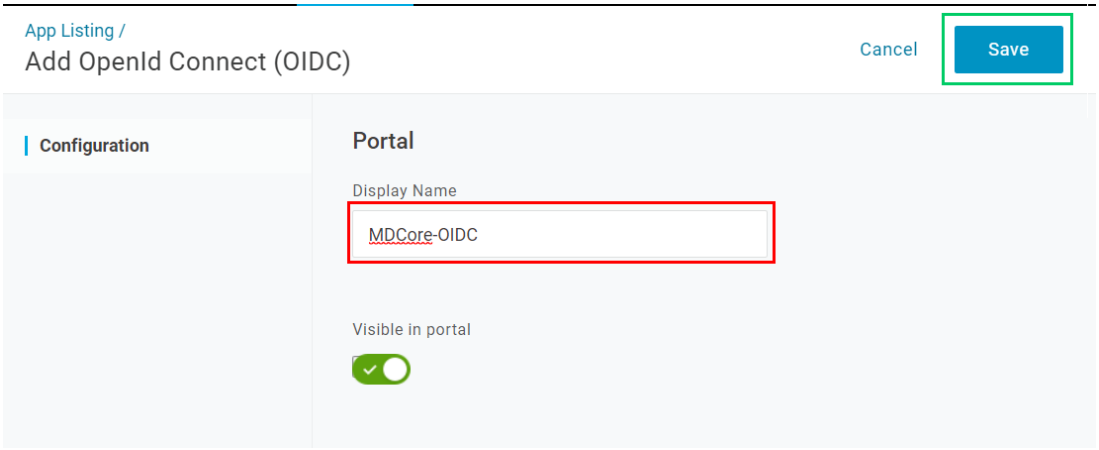
3. Click `Add App` to add a new application.



4. Enter "OpenId Connect" in the search box and select `OpenId Connect (OIDC)` in the results panel.

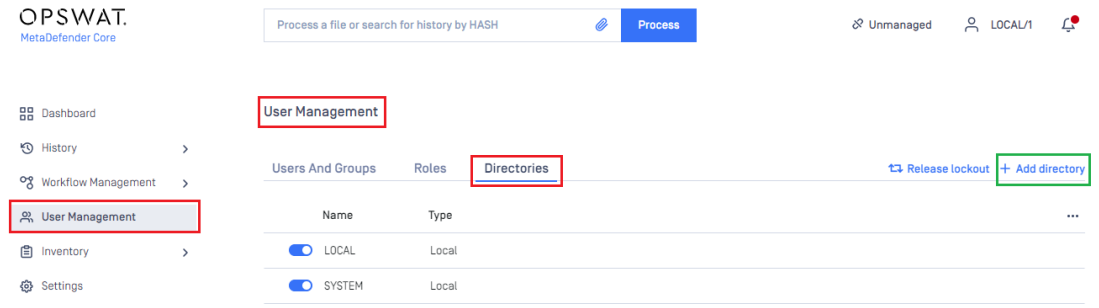


5. In the new page, fill in `Display Name` with MDCore-OIDC, for example, and click `Save`.

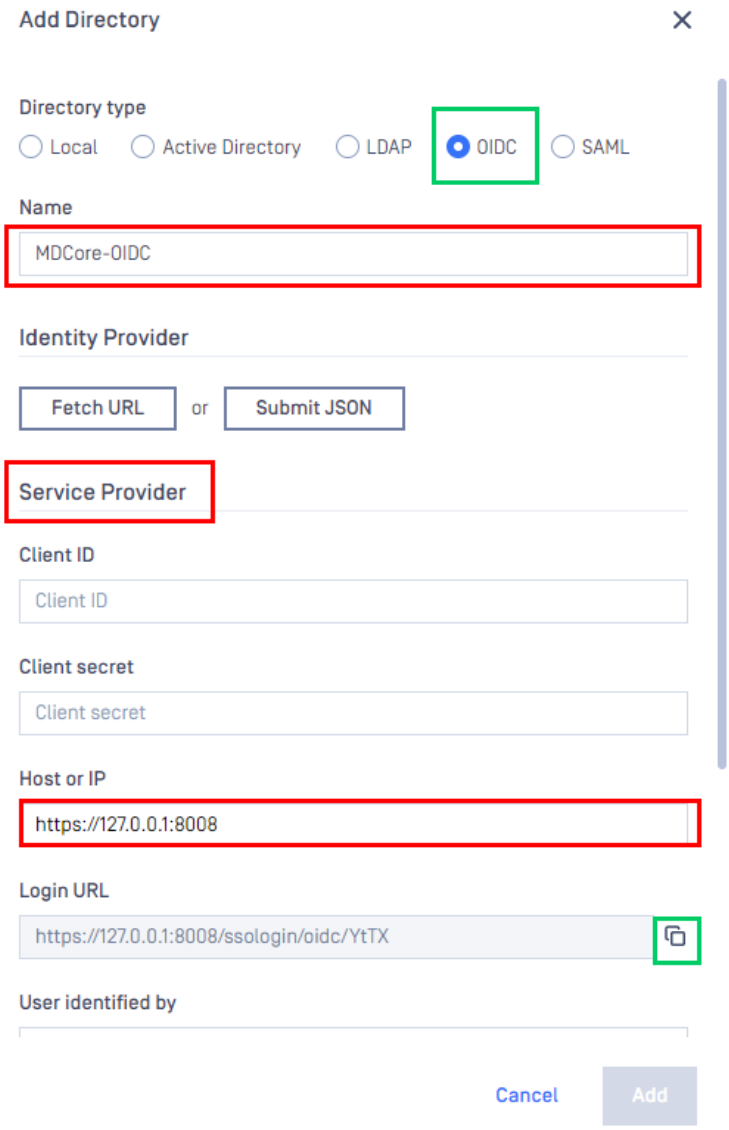


Create OIDC directory in MetaDefender Core

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.

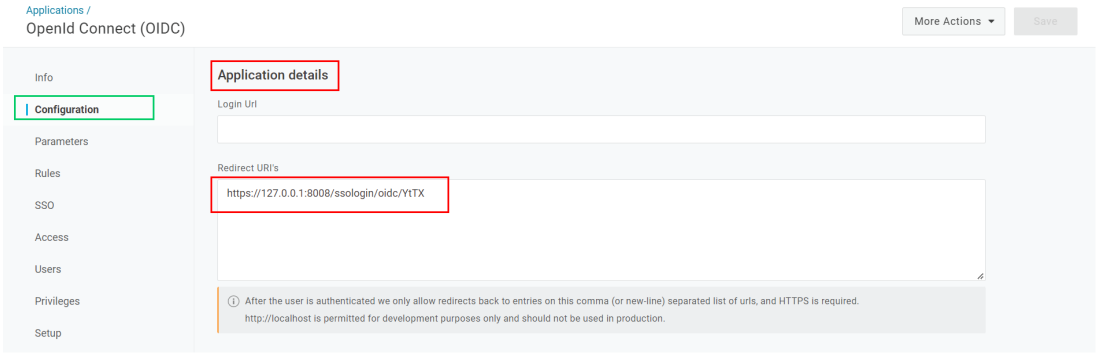


4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCore-OIDC.
5. Enter `Host or IP` where MetaDefender Core is hosted, for this example use https://127.0.0.1:8008.
6. Copy the string generated under `Login URL`.

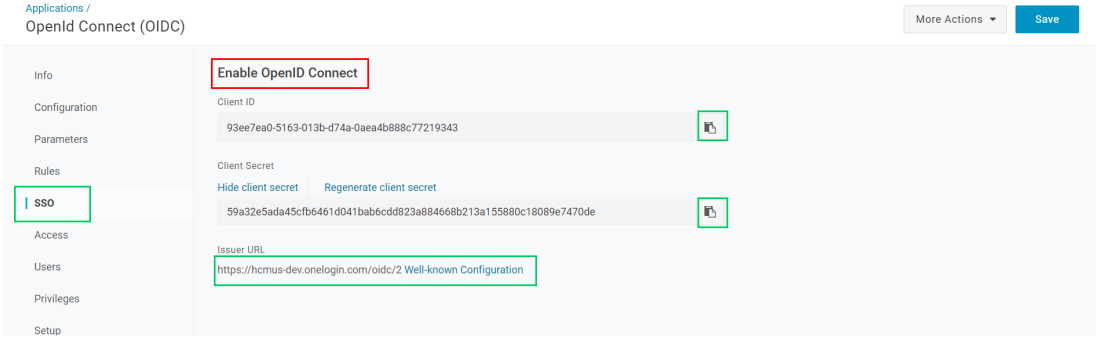


Complete configuration in OneLogin

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.



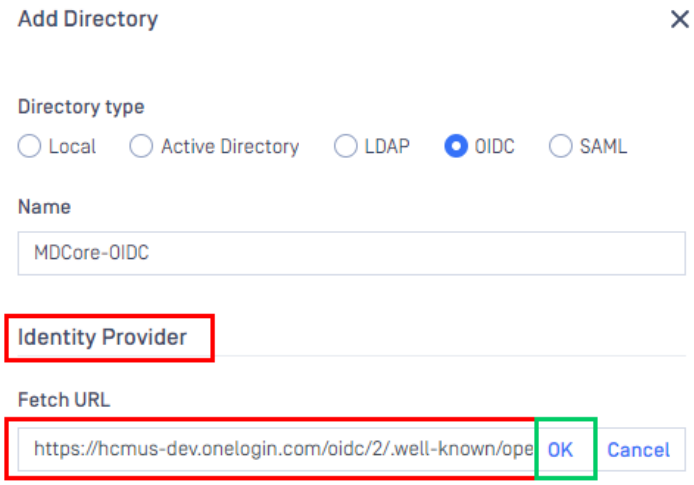
3. In `SSO` of the left side bar, under `Enable OpenID Connect` section, copy `Client ID`, `Client Secret`, and `Issuer URL`.



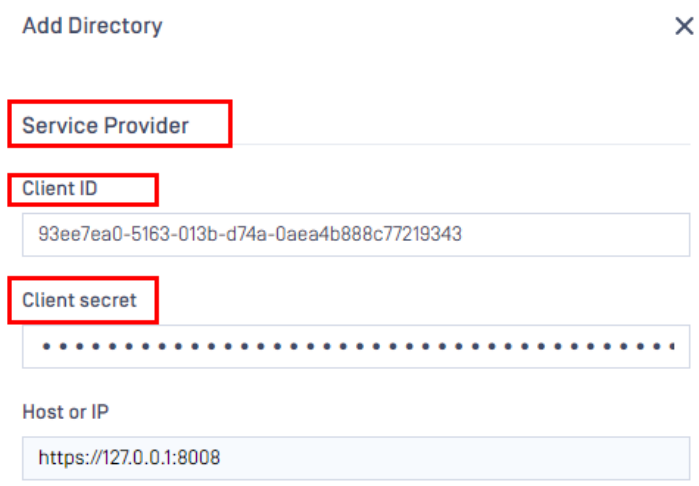
4. Click `Save` to complete.

Complete configuration in MetaDefender Core

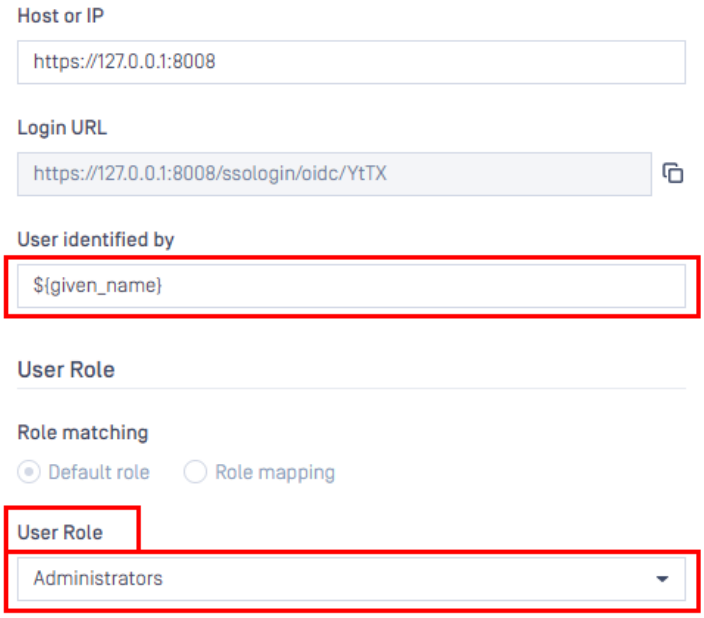
1. Switch back to MetaDefender Core. Under `Identity Provider`, click `Fetch URL`, paste value of `Issuer URL` from OneLogin into the box below, click `OK`, and wait a moment for MetaDefender Core to check and set OneLogin as its IdP.



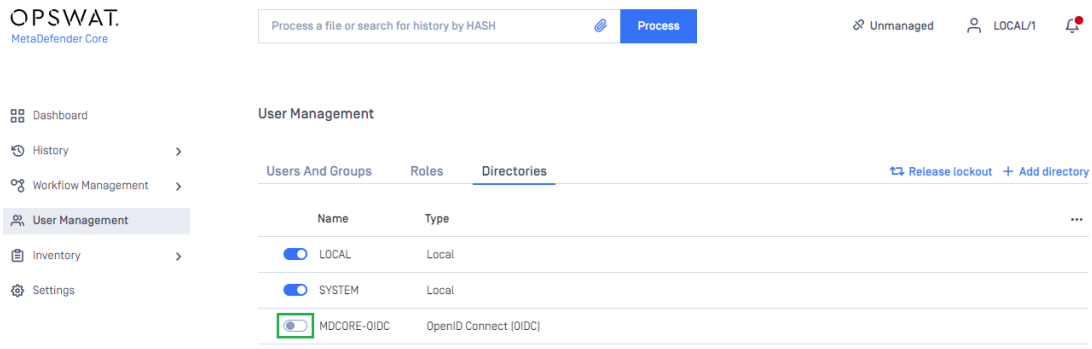
2. Under `Service Provider`, paste the values of `Client ID` and `Client Secret` from OneLogin into the boxes labeled `Client ID` and `Client secret`, respectively.



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.

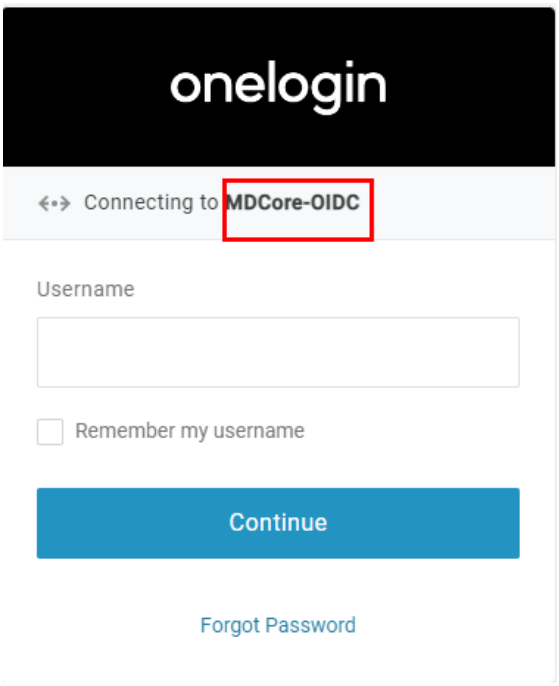


5. In `User Management` page, toggle the new directory, MDCORE-OIDC in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all sessions will expire immediately.

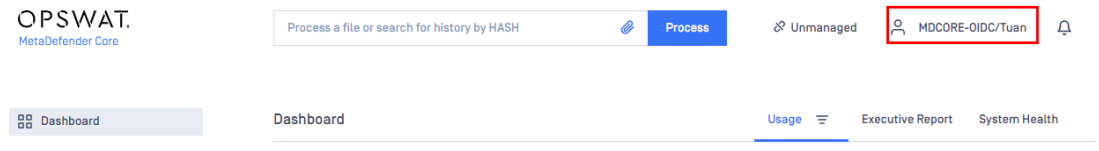


Test the integration

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

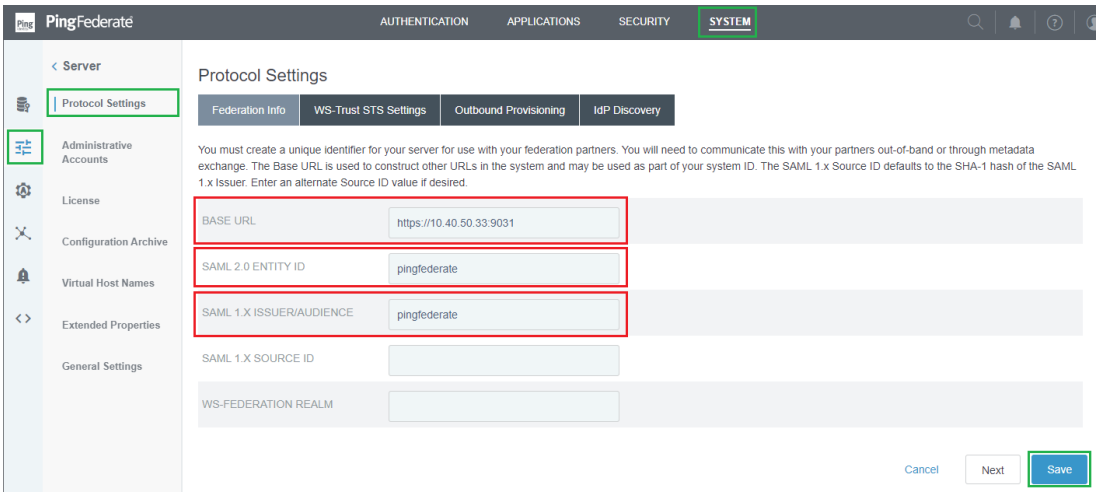
The instructions guide users on how to configure PingFederate SAML and OIDC Single Sign-on for MetaDefender Core.

Configuring OIDC Single Sign-on

Set up PingFederate

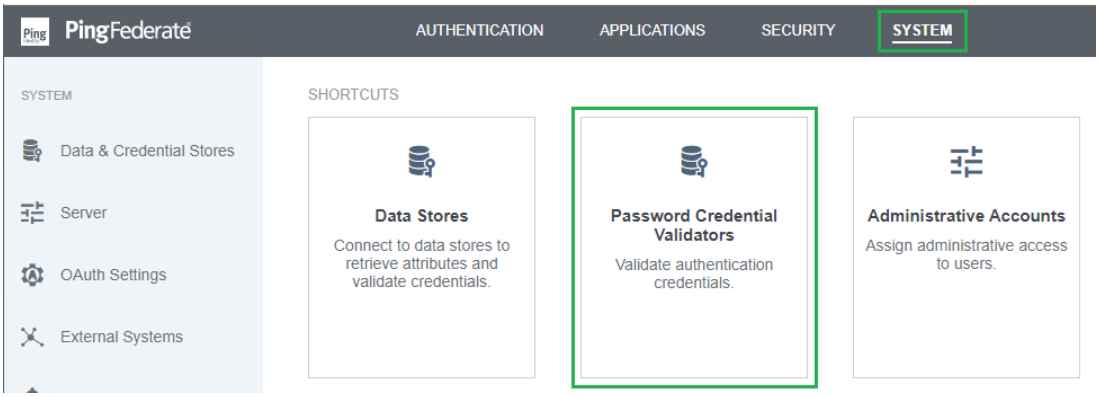
Configurate PingFederate

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.



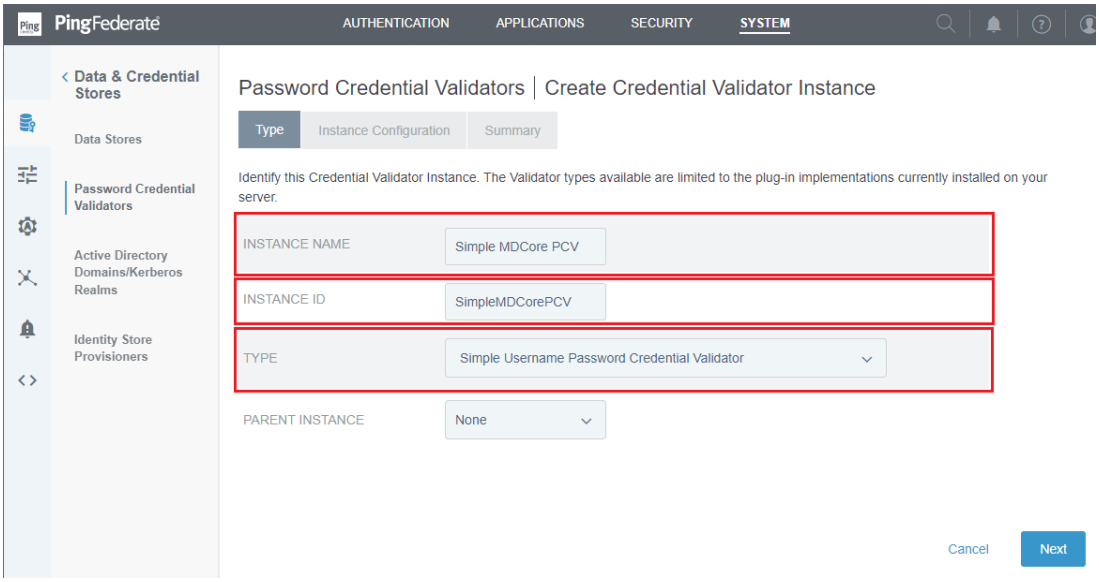
Set up simple Password Credential Validators

1. In homepage, navigate to `SYSTEM` tab and click on `Password Credential Validators`.

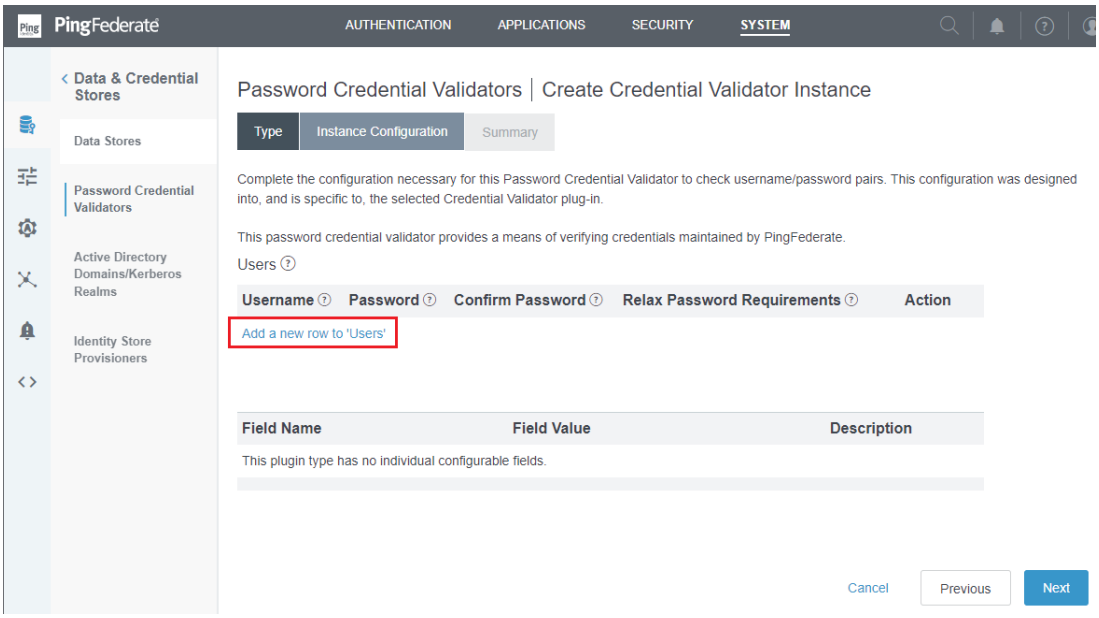


2. Click `Create New Instance`.

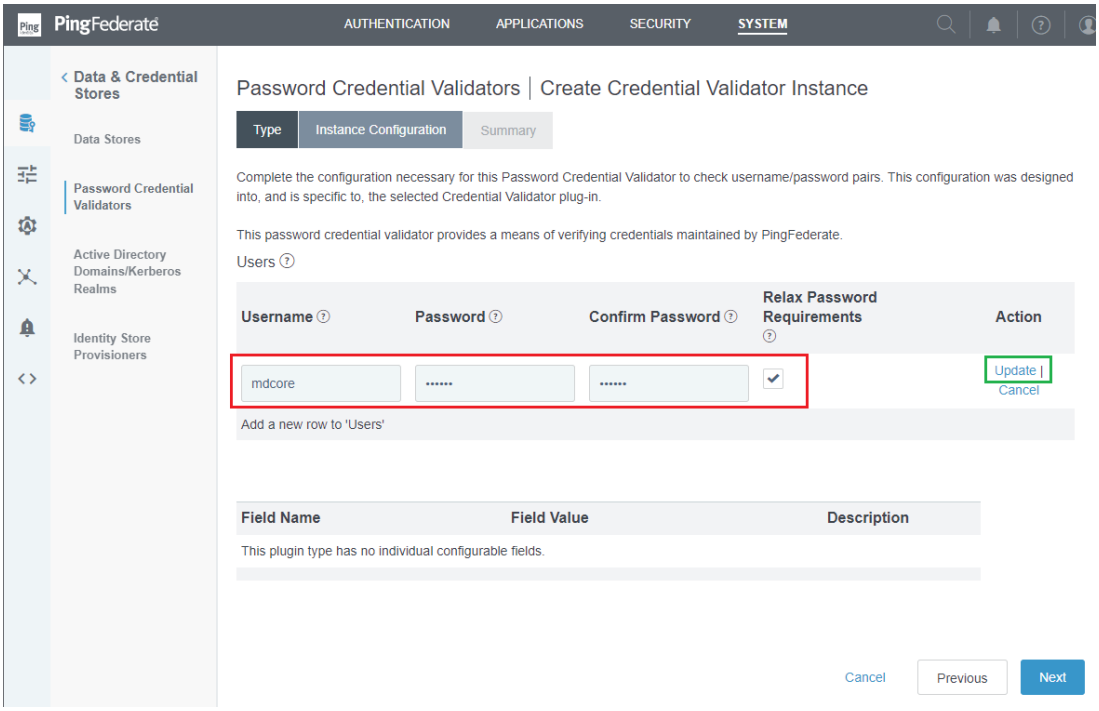
3. Enter `INSTANCE NAME` and `INSTANCE ID`, select `Simple Username Password Credential Validator` for `TYPE`, and click `Next`.



4. In `Create Credential Validator Instance` page, click `Add a new row to 'Users'.`



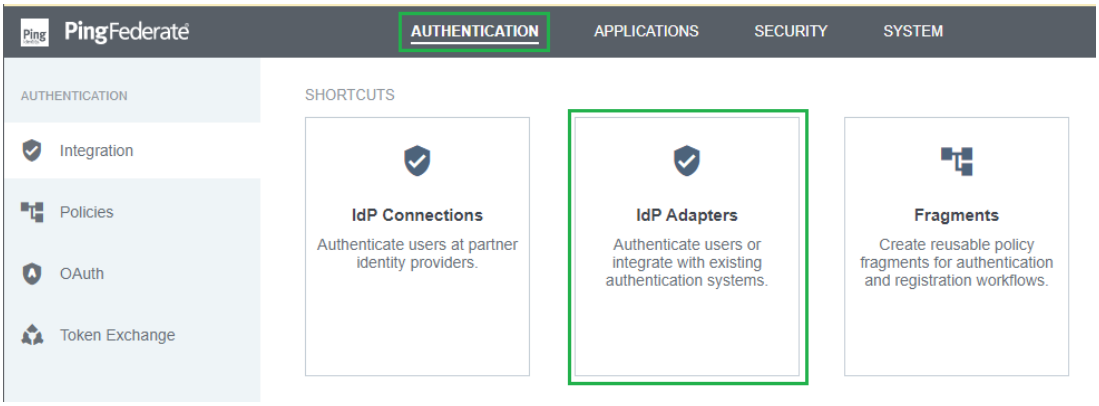
5. Enter `Username`, `Password`, and `Confirm Password`, then click `Update` and `Next`.



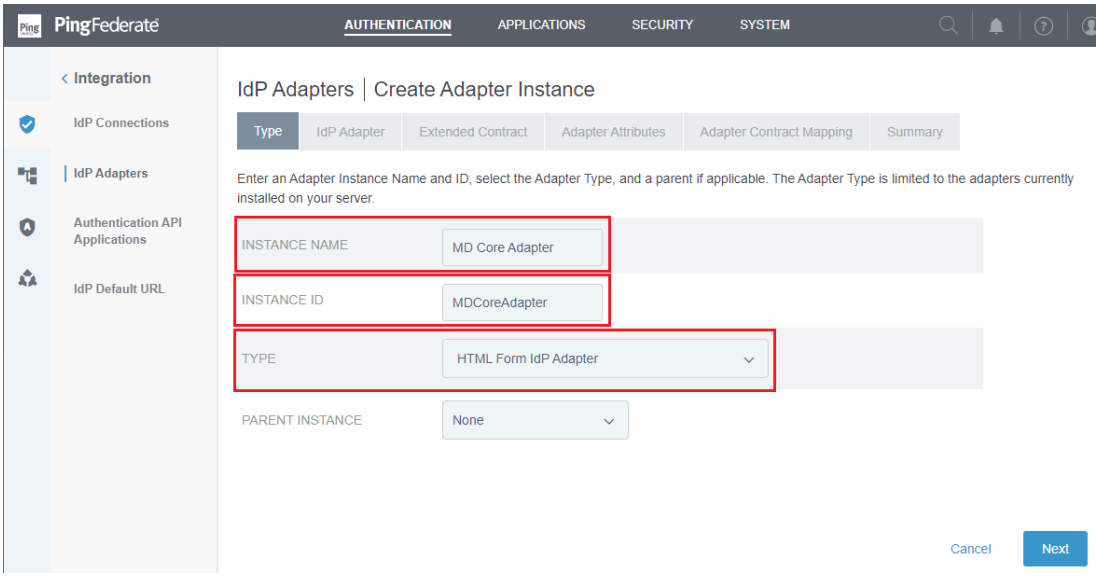
6. Review all selected options in `Summary` tab, and click `Save`.

Set up a simple IdP Adapter

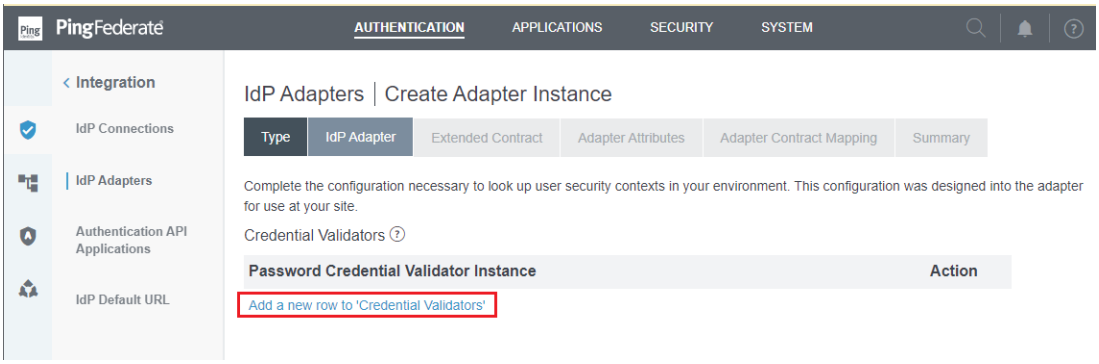
1. In homepage, navigate to `AUTHENTICATION` tab and click on `IdP Adapters`.



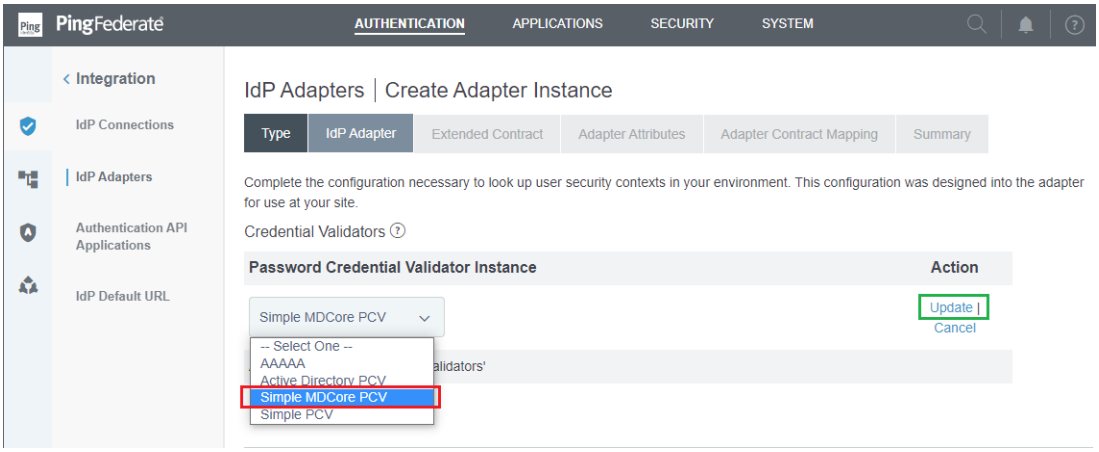
2. Click `Create New Instance`.
3. Enter `INSTANCE NAME`, `INSTANCE ID`, and select `HTML Form IdP Adapter` for `TYPE`. Click `Next`.



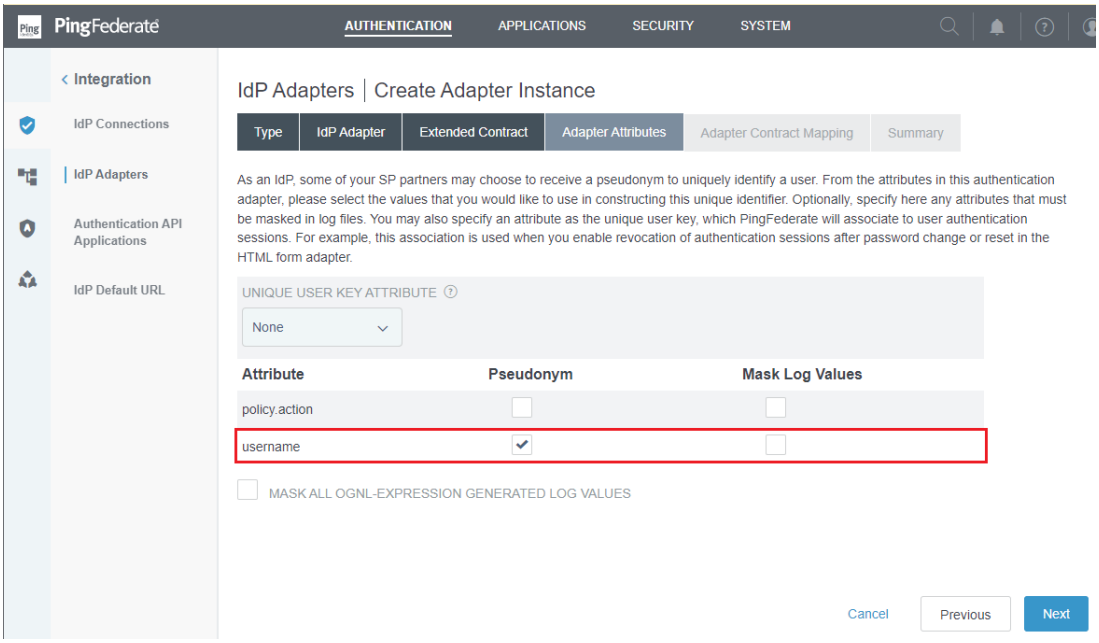
4. Click `Add a new row to 'Credential Validators'`.



5. Select the Password Credential Validator created in the previous stage, that is "Simple MDCore PCV", then click `Update` and `Next`.



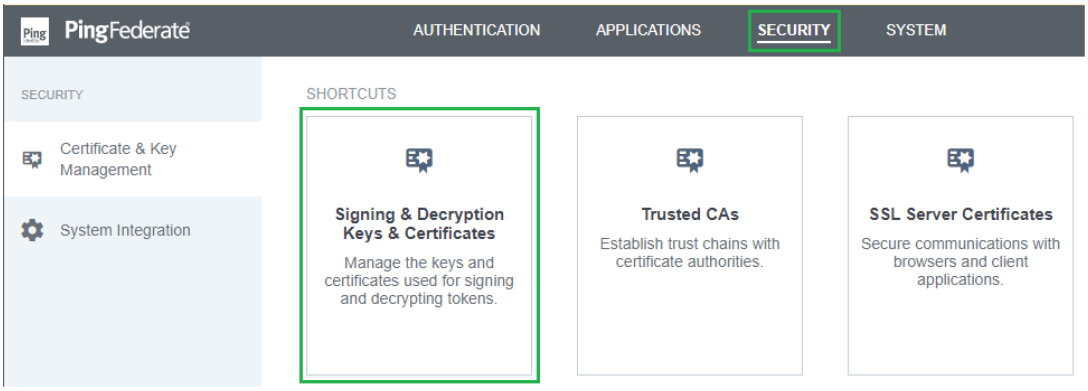
6. Click `Next` on `Extended Contract` tab.
7. In `Adapter Attributes` tab, toggle `username` attribute under `Pseudonym` column, and click `Next`.



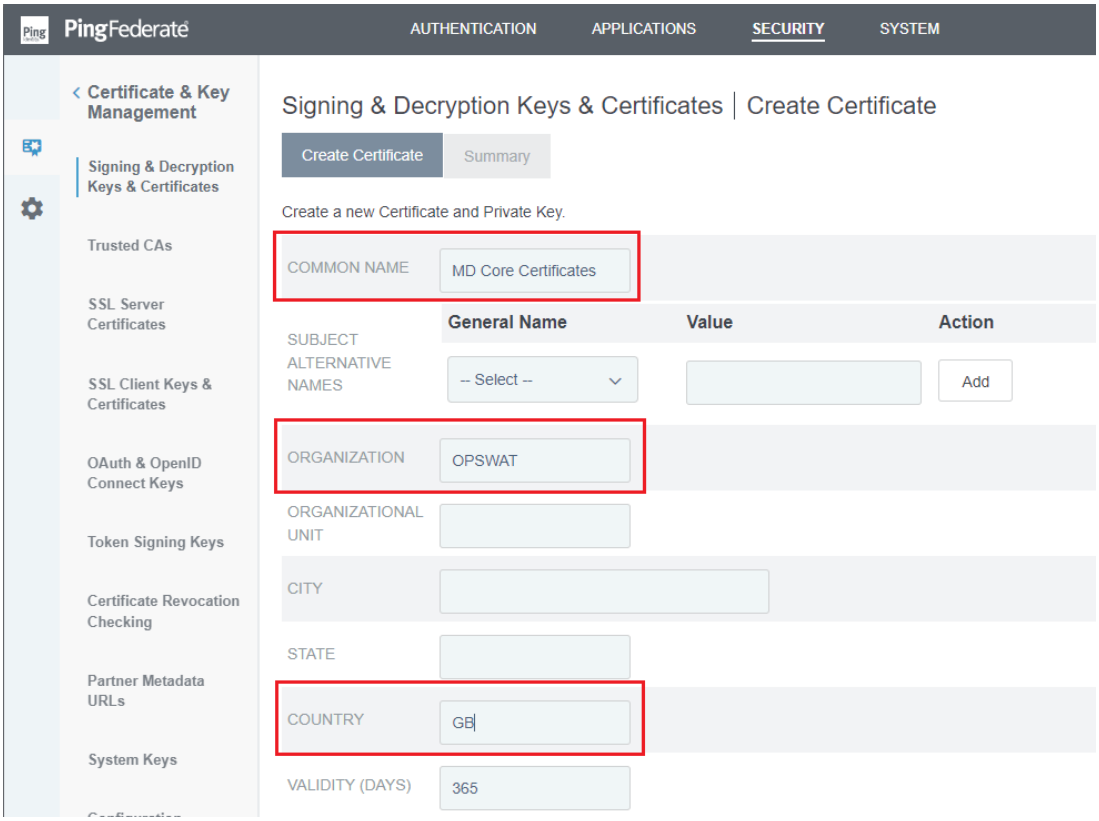
8. Click `Next` on `Adapter Contract Mapping` tab.
9. Review all selected options in `Summary` tab and click `Save` to complete.

Set up Signing, Decryption Keys and Certificates

1. In homepage, navigate to `SECURITY` tab and select `Signing & Decryption Keys & Certificates`.



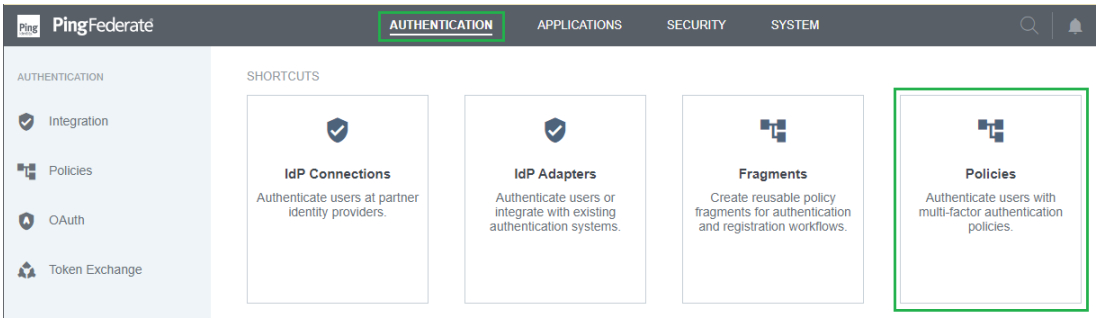
2. Click `Create New`.
3. Enter `COMMON NAME`, `ORGANIZATION`, and `COUNTRY`, along with any other necessary information, then click `Next`.



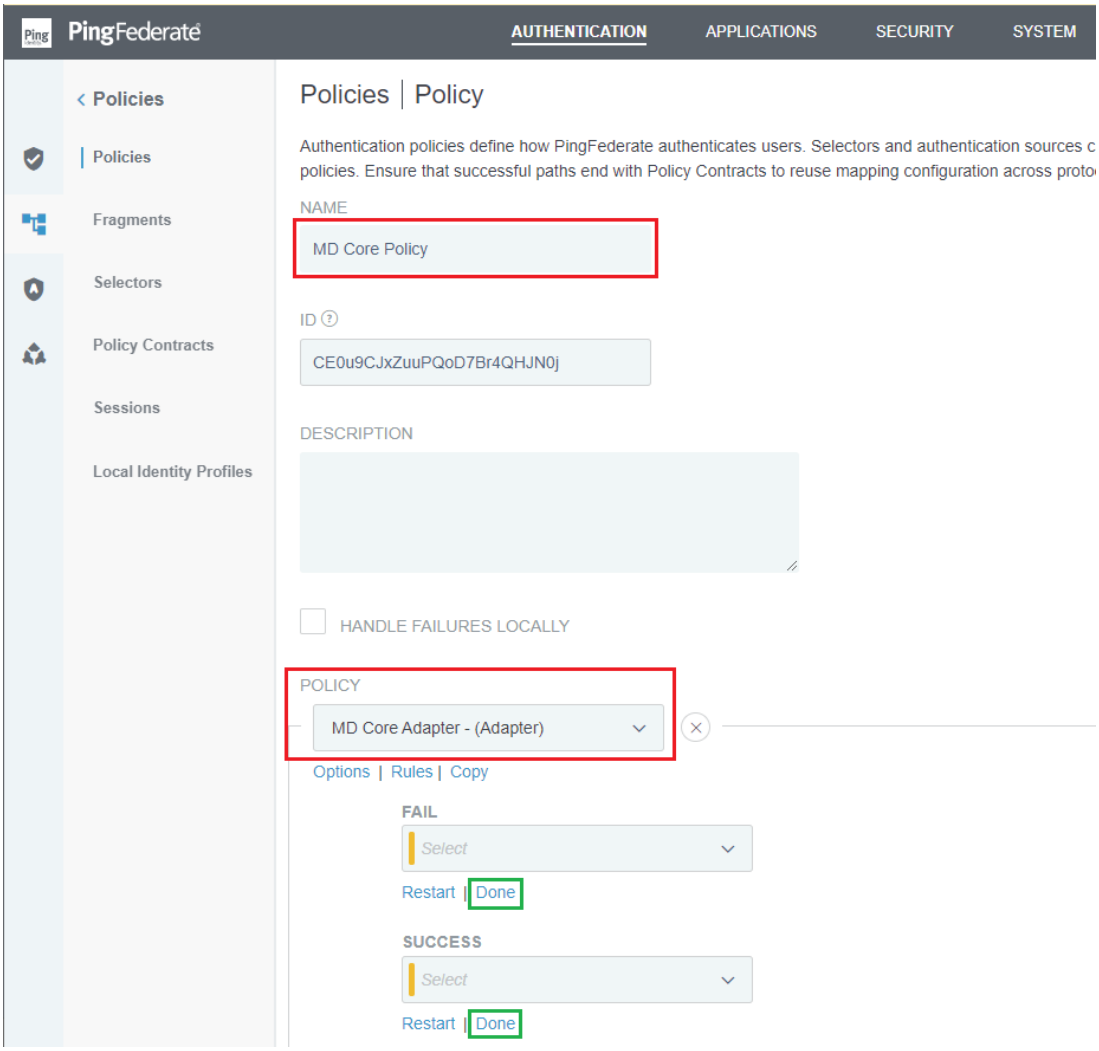
4. Review all selected options in `Summary` tab and click `Save` to complete.

Set up Authentication Policies

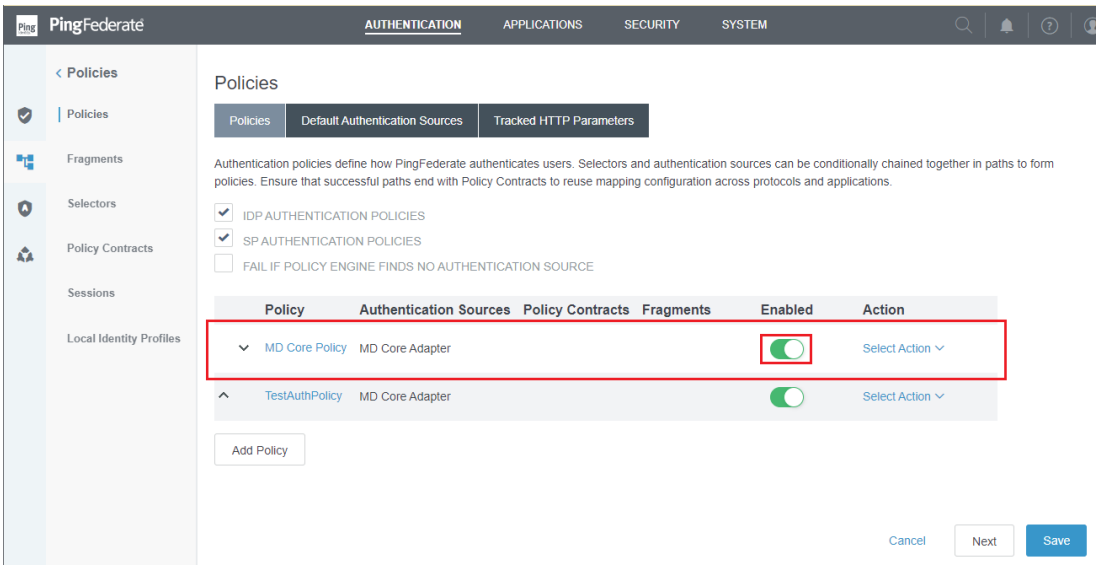
1. In homepage, navigate to `AUTHENTICATION` tab, and click `Policies`.



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`.

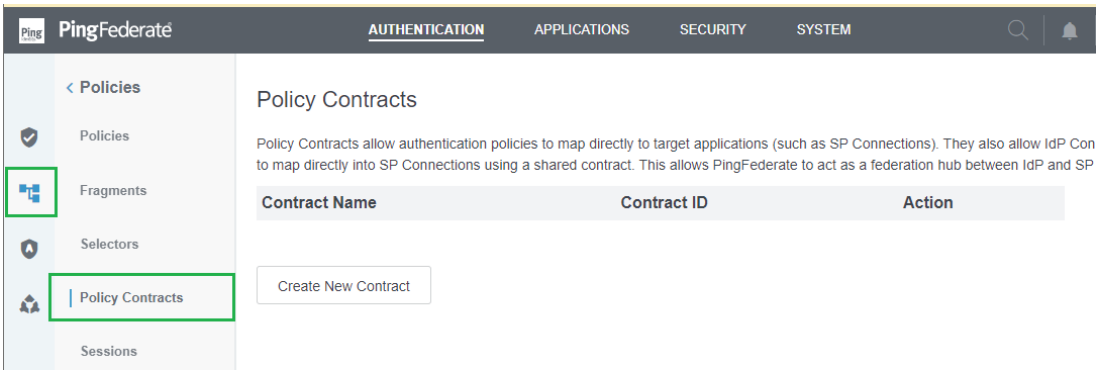


4. Ensure the new policy is enabled, then click `Save` to complete.

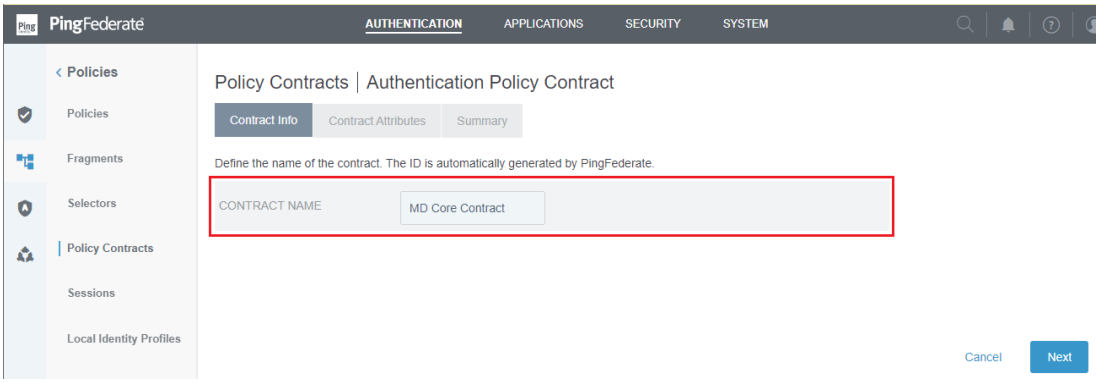


Set up Policy Contracts

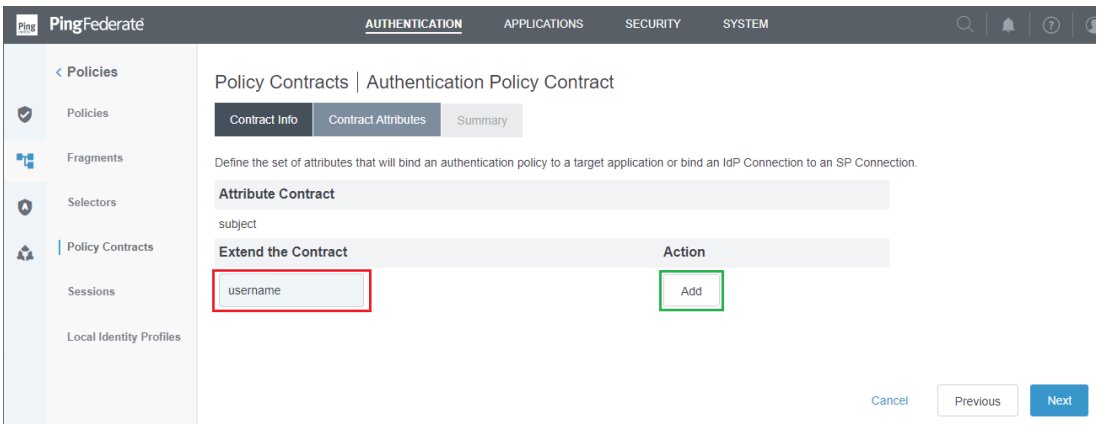
1. In homepage, navigate to `AUTHENTICATION` tab. Select `Policies` icon on the left sidebar, then click `Policy Contracts`.



2. Press `Create New Contract` on the next page.
3. In `Contract Info` step, enter `CONTRACT NAME` and click `Next`.



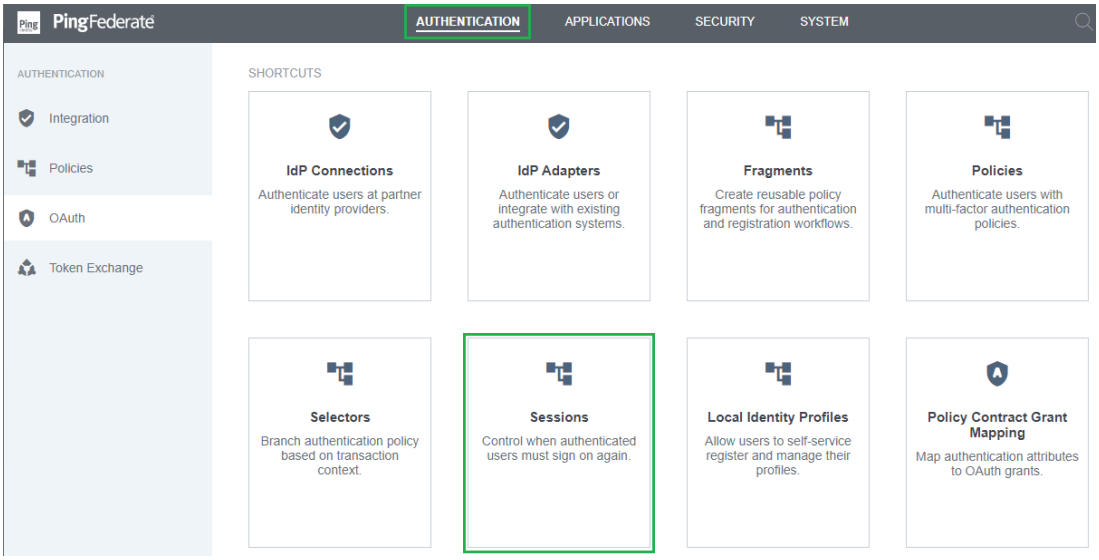
4. In `Contract Attributes` step, navigate to `Extend the Contract`, fill in claims that will be returned to MetaDefender Core, click `Add` to add the claim, and then `Next`.



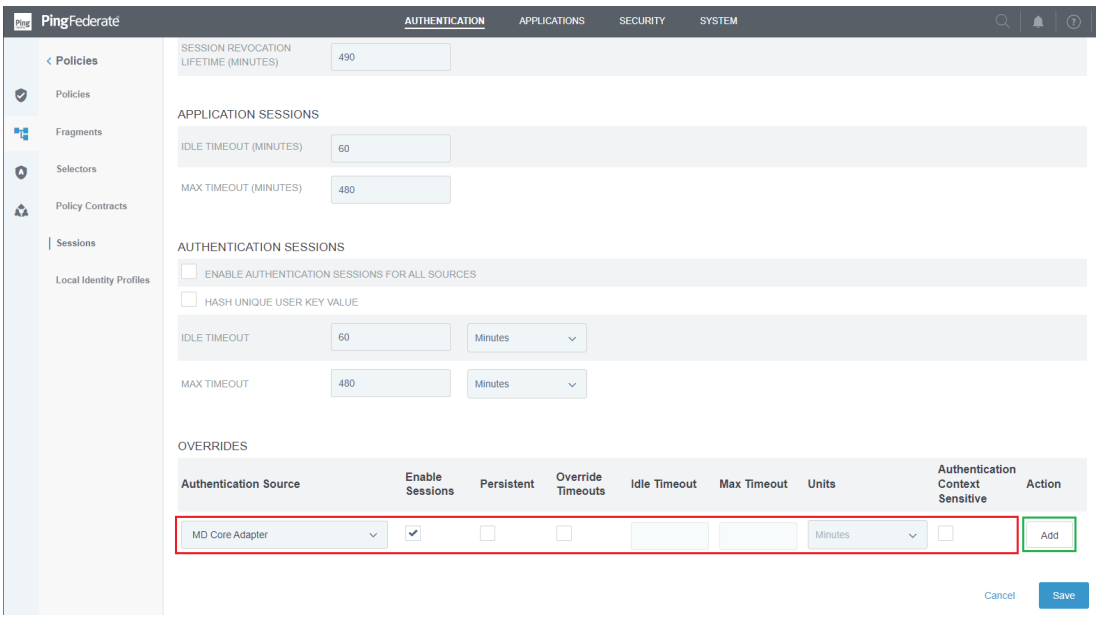
5. Review all selected options in `Summary` tab, then click `Save` to complete.

Set up Policy Sessions

1. In homepage, navigate to `AUTHENTICATION` tab and click `Sessions`.



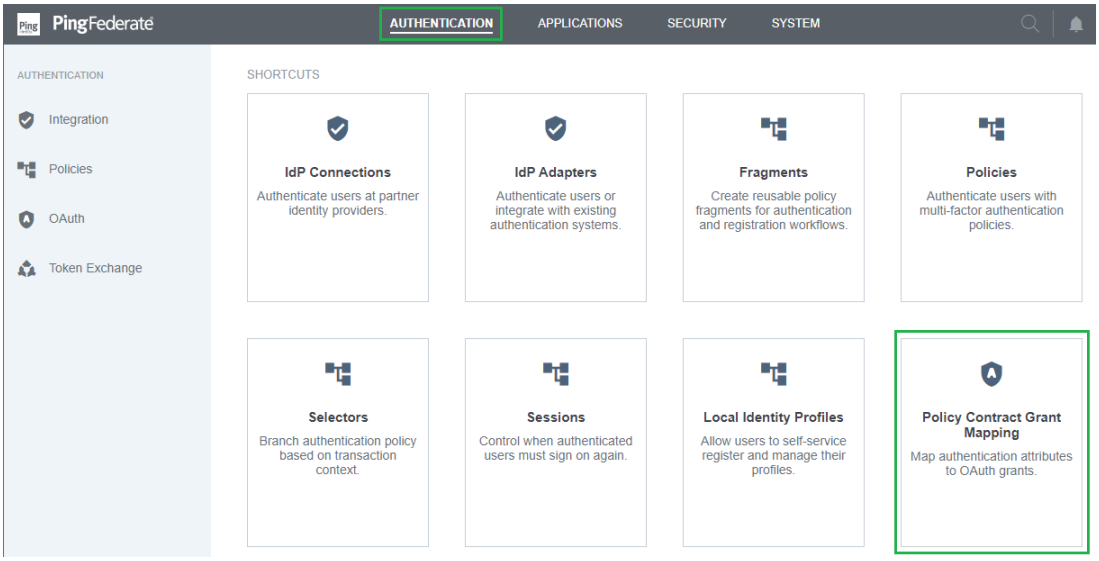
2. In `Sessions` menu, find `OVERRIDES` subtab.
3. Select "MD Core Adapter" for Authentication Source, tick `Enable Sessions`, and click `Add` to include the authentication source.



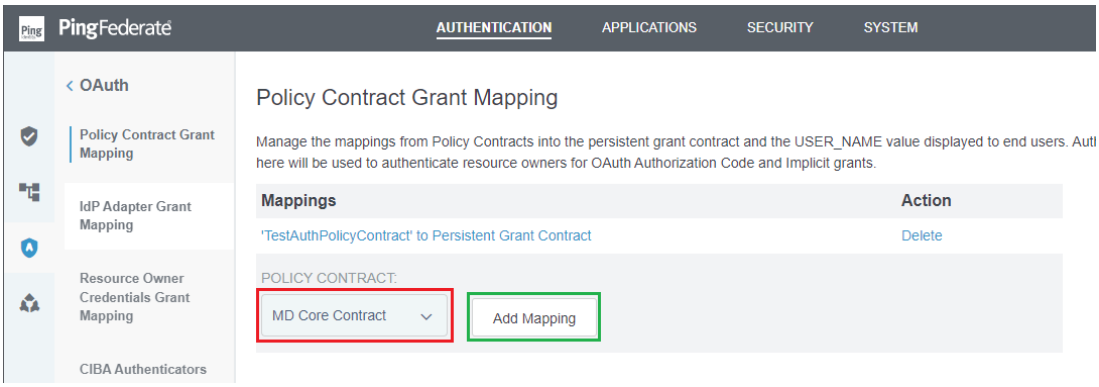
4. Click `Save` to complete.

Set up Policy Contract Grant Mapping

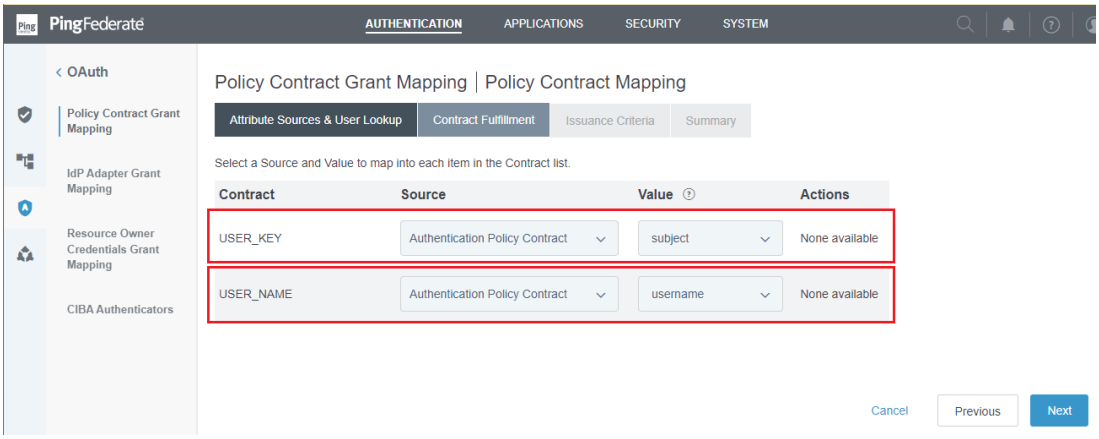
1. In homepage, navigate to `AUTHENTICATION` and click `Policy Contract Grant Mapping`.



2. Select the Policy contract in previous stage, then click `Add Mapping`.



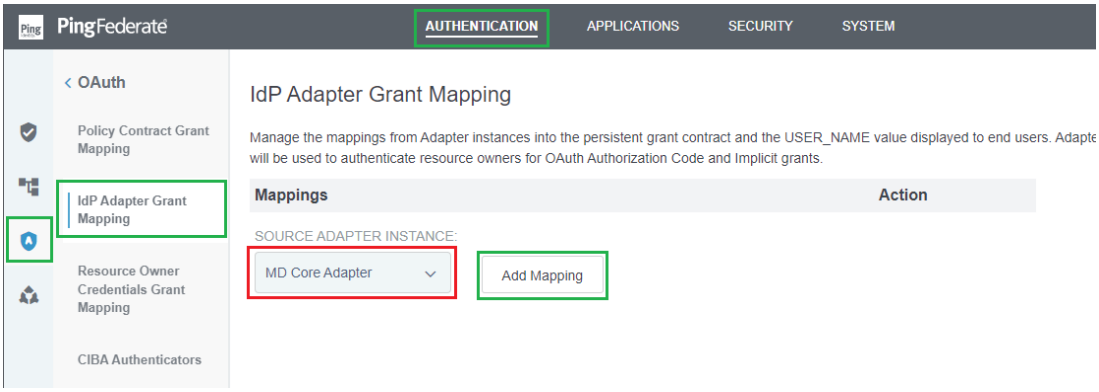
3. In `Attribute Sources & User Lookup` step, click `Next`.
4. In `Contract Fulfillment` step, select the appropriate mapping for `Source` and `Value` columns, then click `Next`.



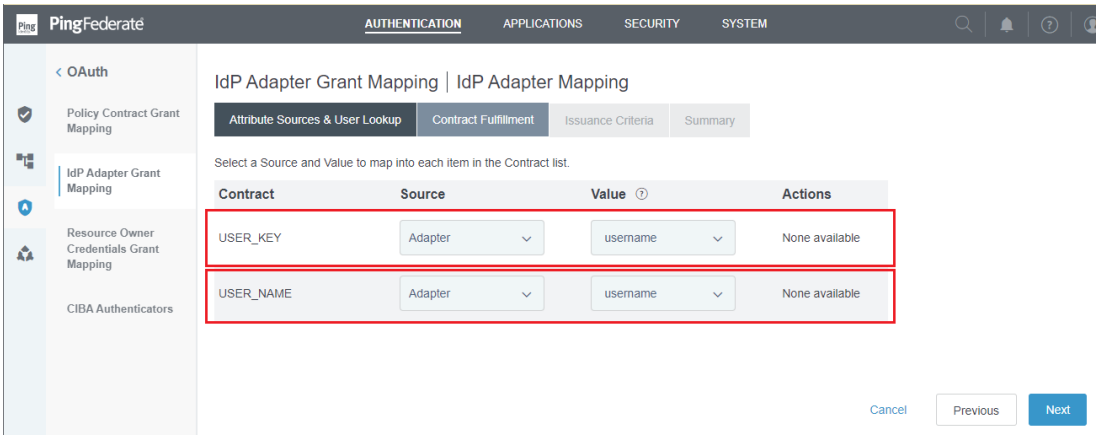
5. Click `Next` in `Issuance Criteria` step.
6. Review all selected options in `Summary` tab, then click `Save` to complete.

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`.



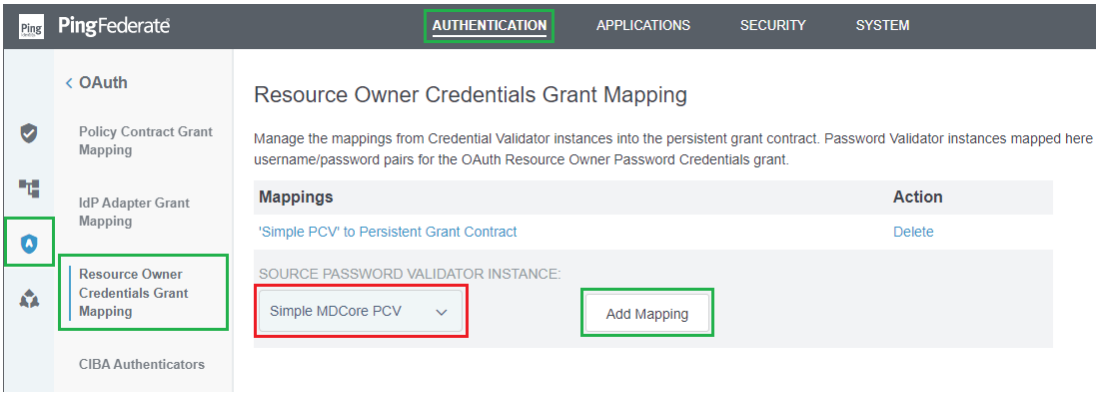
3. Click `Next` in `Attribute Sources & User Lookup` step.
4. In `IdP Adapter Mapping` step, select the appropriate mapping for `Source`, `Value`, and click `Next`.



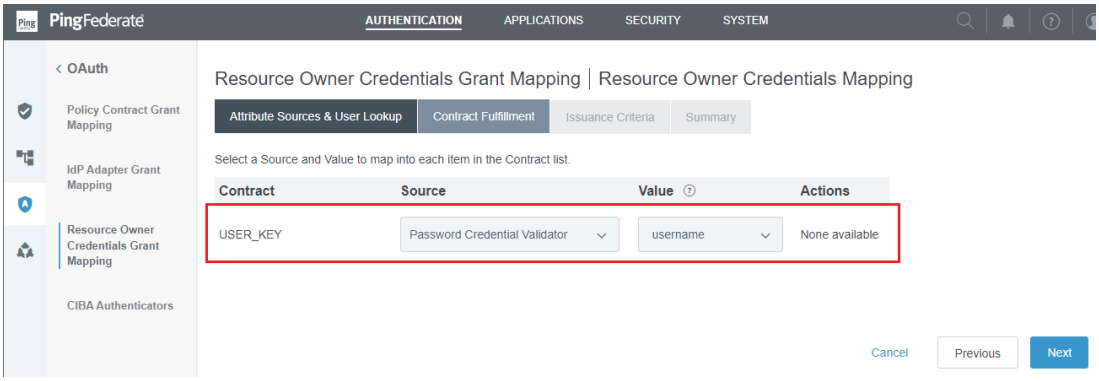
5. Click `Next` in `Issuance Criteria` step.
6. Review all selected options in `Summary` tab, then click `Save` to complete.

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`.



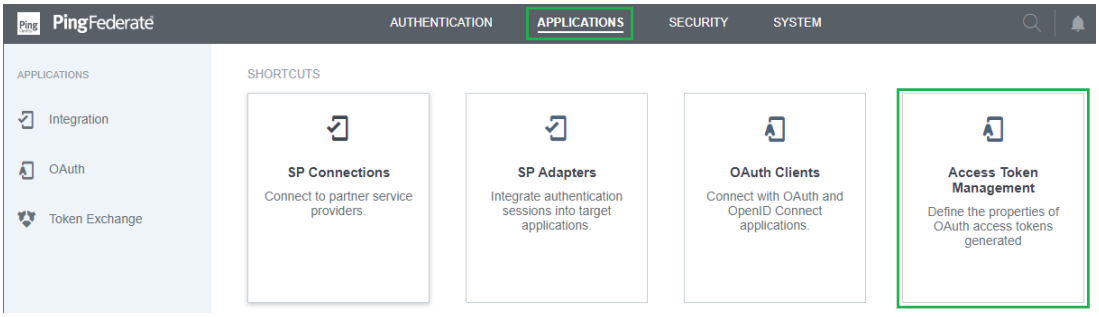
3. Click `Next` in `Attribute Sources & User Lookup` step.
4. In `Contract Fulfillment` step, choose the appropriate mapping for `Source` and `Value`, and then click `Next`.



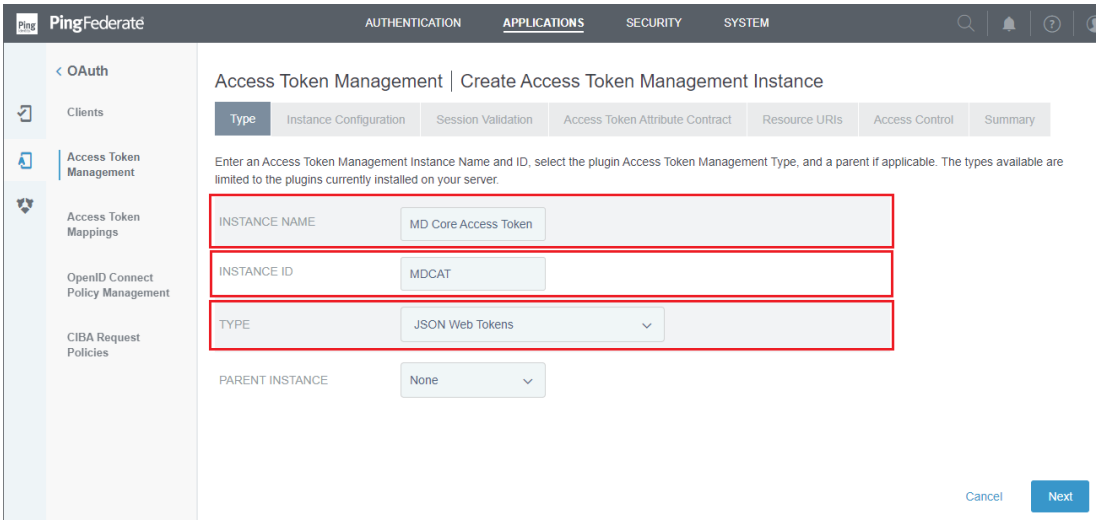
5. Click `Next` in `Issuance Criteria` step.
6. Review all selected options in `Summary` tab, then click `Save` to complete.

Set up Access Token Management

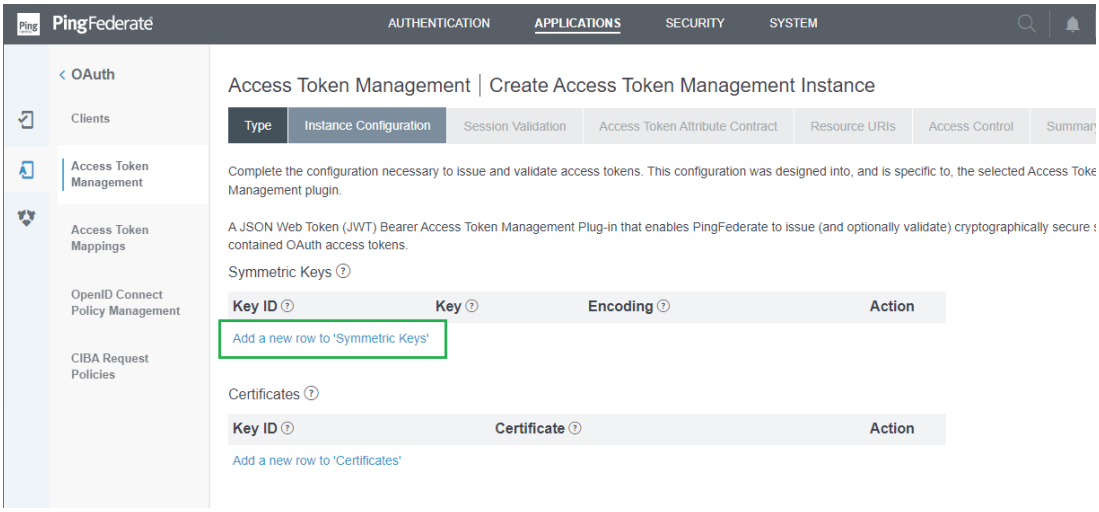
1. In homepage, navigate to `APPLICATIONS`, and click `Access Token Management`.



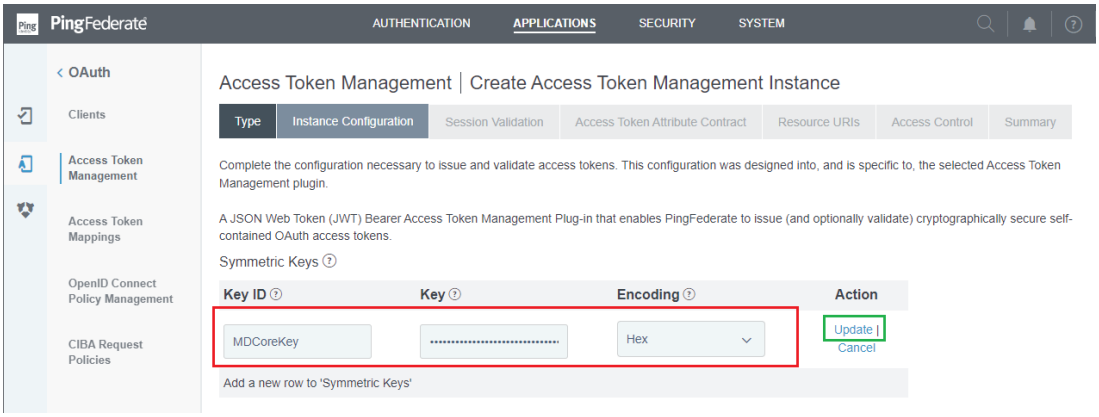
2. Click `Create New Instance`.
3. Enter `INSTANCE NAME`, `INSTANCE ID`, and select `JSON Web Tokens` for `TYPE`. Click `Next` to continue.



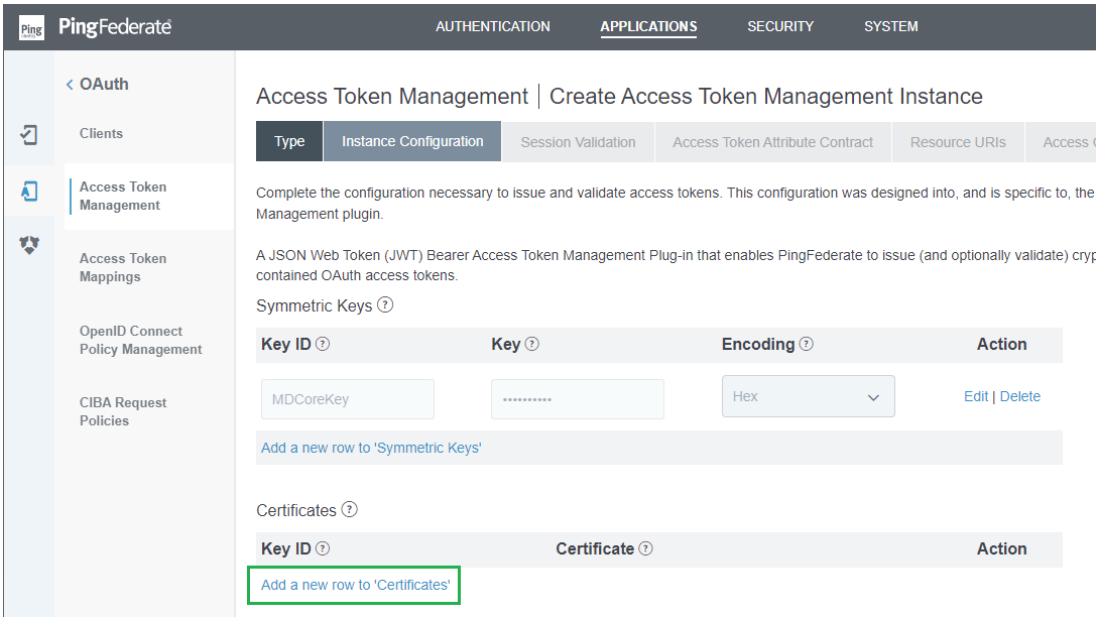
4. In `Instance Configuration` step, click `Add a new row to 'Symmetric Keys'` under `Symmetric Keys`.



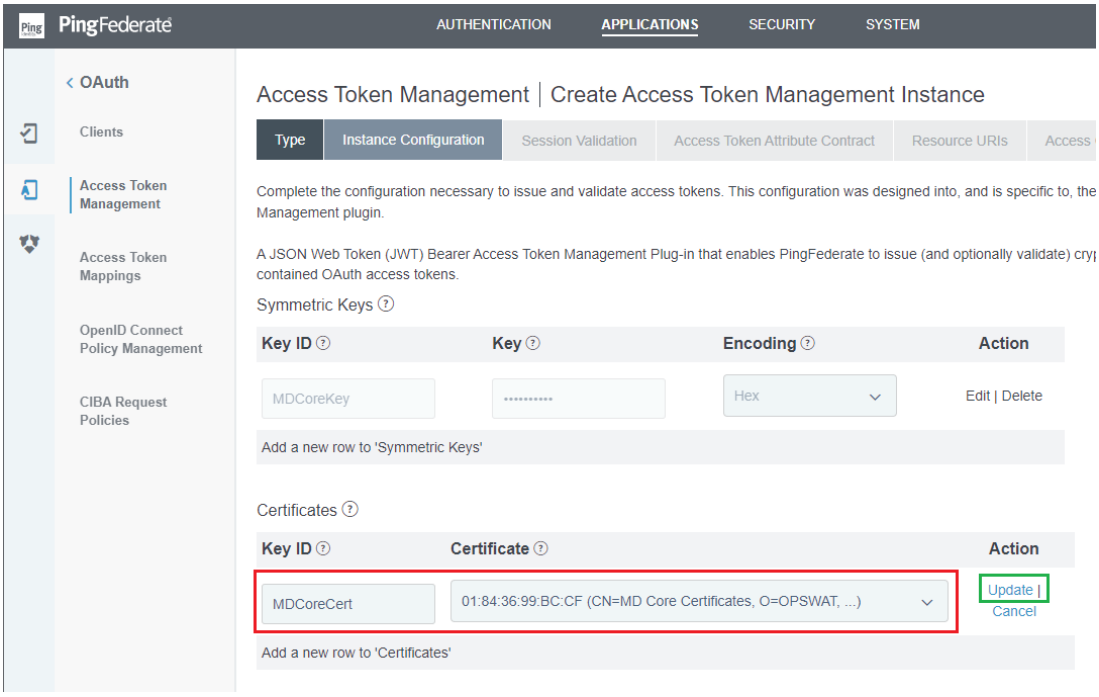
5. Enter `Key ID`, `Key`, and select the correct `Encoding`. Click `Update`.



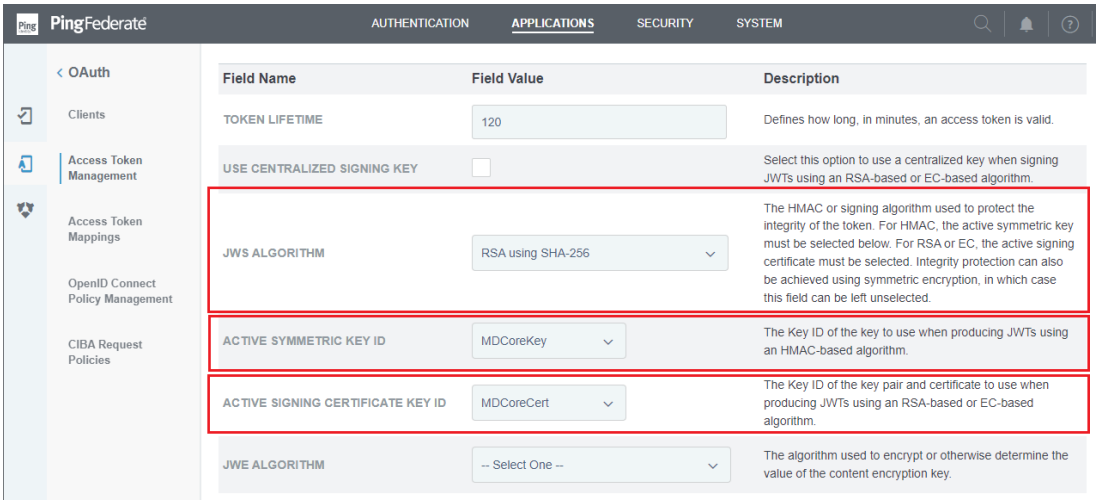
6. Back to `Instance Configuration` step, click `Add a new row to 'Certificates'` under `Certificates`.



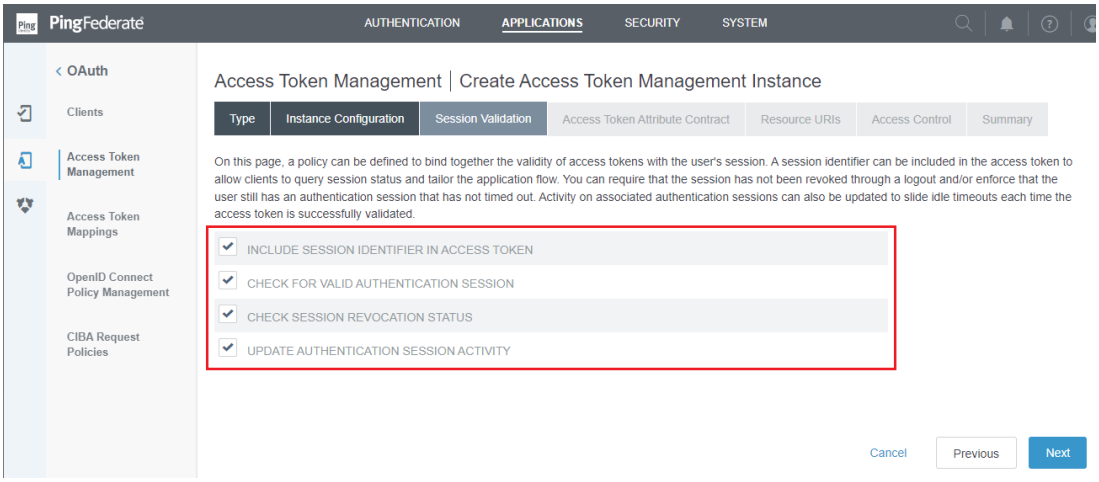
7. Enter `Key ID` and choose `Certificate` from the previous stage. Then click `Update`.



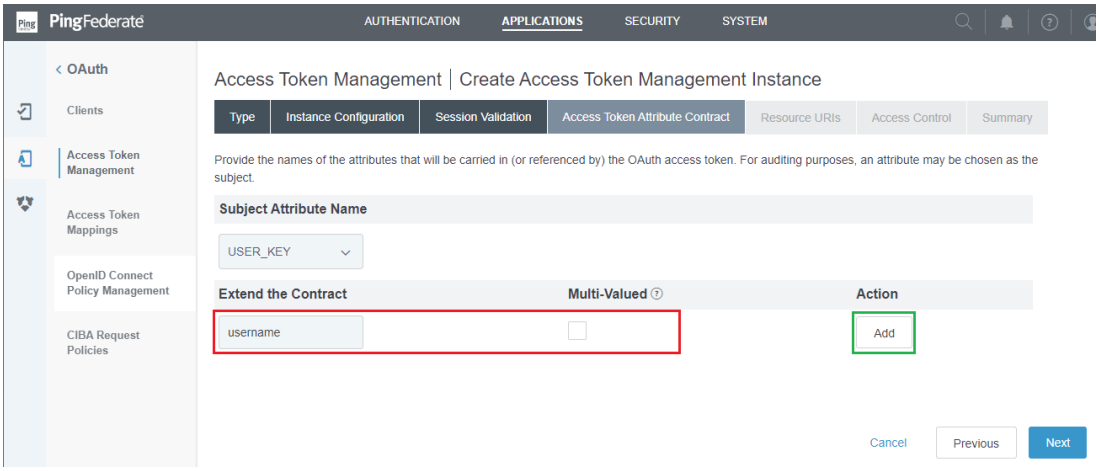
8. Back in `Instance Configuration` step, navigate to `JWS ALGORITHM` and select the appropriate algorithm. Proceed to `ACTIVE SYMMETRIC KEY ID` and `ACTIVE SIGNING CERTIFICATE KEY ID`, then select the key and certificate that were set up in steps 4 and 5 of this stage. Click `Next` when finished.



9. In `Session Validation` step, check all options and click `Next`.



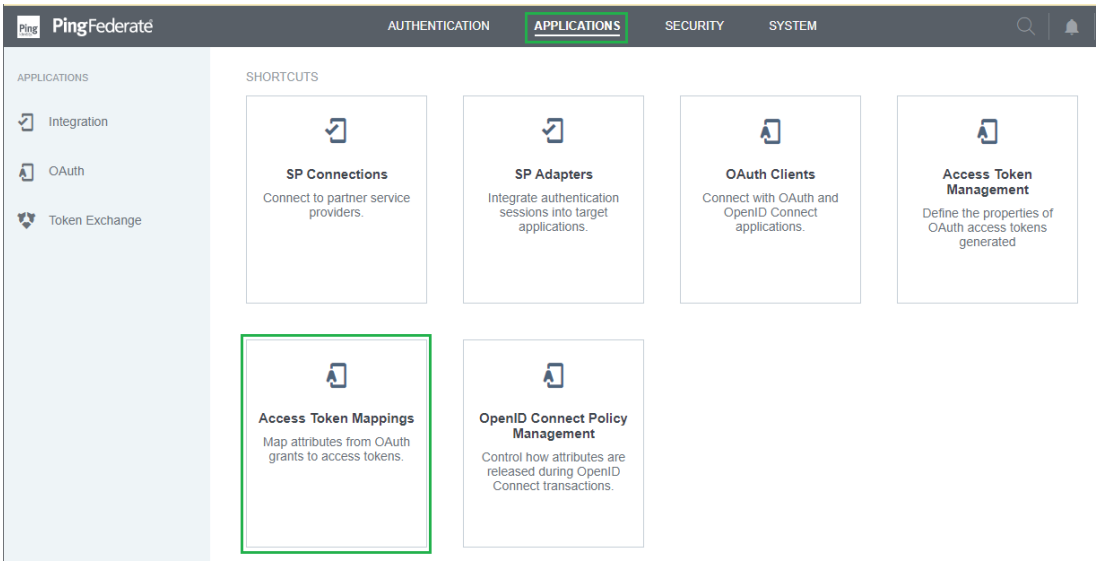
10. In `Access Token Attribute Contract` step, navigate to `Extend the Contract` and add the same claims that have already been set up in the stages: `Policy Contract Grant Mapping`, `IdP Adapter Grant Mapping`, and `Resource Owner Credentials Grant Mapping`. Click `Next`.



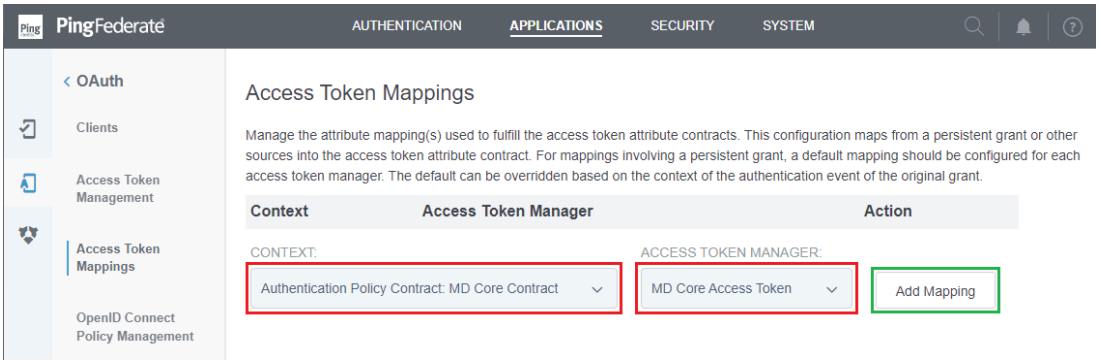
11. Click `Next` in `Resource URI`s and `Access Control` steps.
12. Review all selected options in `Summary` tab, then click `Save` to complete.

Set up Access Token Mappings

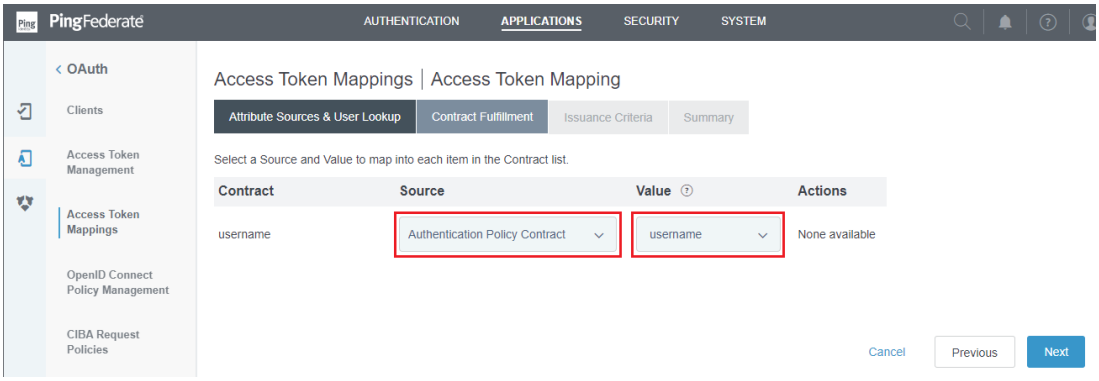
1. In homepage, navigate to `APPLICATIONS` and click on `Access Token Mappings`.



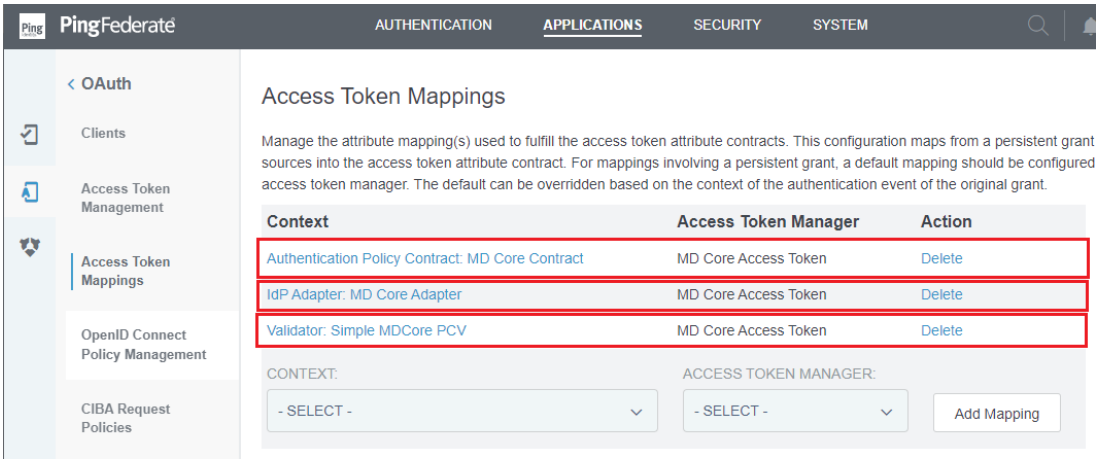
2. In `Access Token Mappings` page, select the Authentication Policy Contract from the previous step to map to `ACCESS TOKEN MANAGER`. Click `Add Mapping`.



3. Click `Next` in `Attribute Sources & User Lookup` step.
4. In `Contract Fulfillment` step, select the appropriate `Source` and `Value`, then click `Next`.

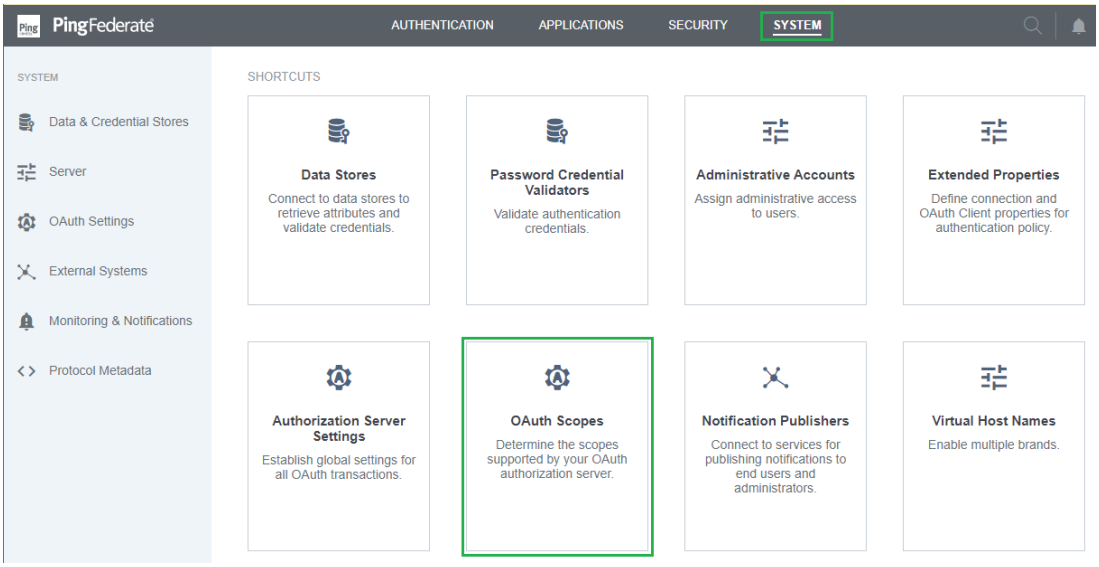


5. Click `Next` in `Issuance Criteria` step.
6. Review all selected options in `Summary` tab, then click `Save`.
7. Return to `Access Token Mappings` page and repeat the steps 2 to 6 for Validator and IdP Adapter.

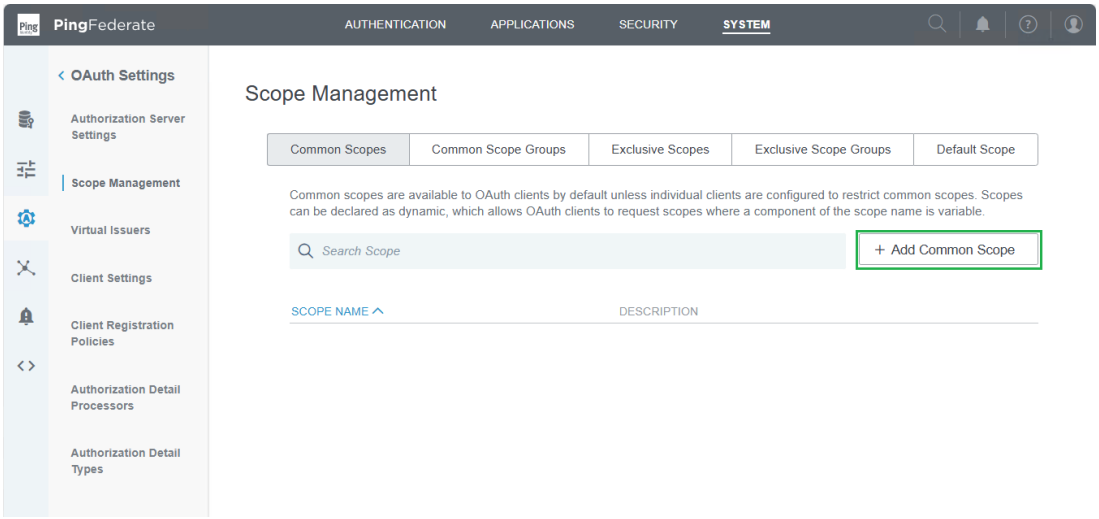


Set up Scope Management

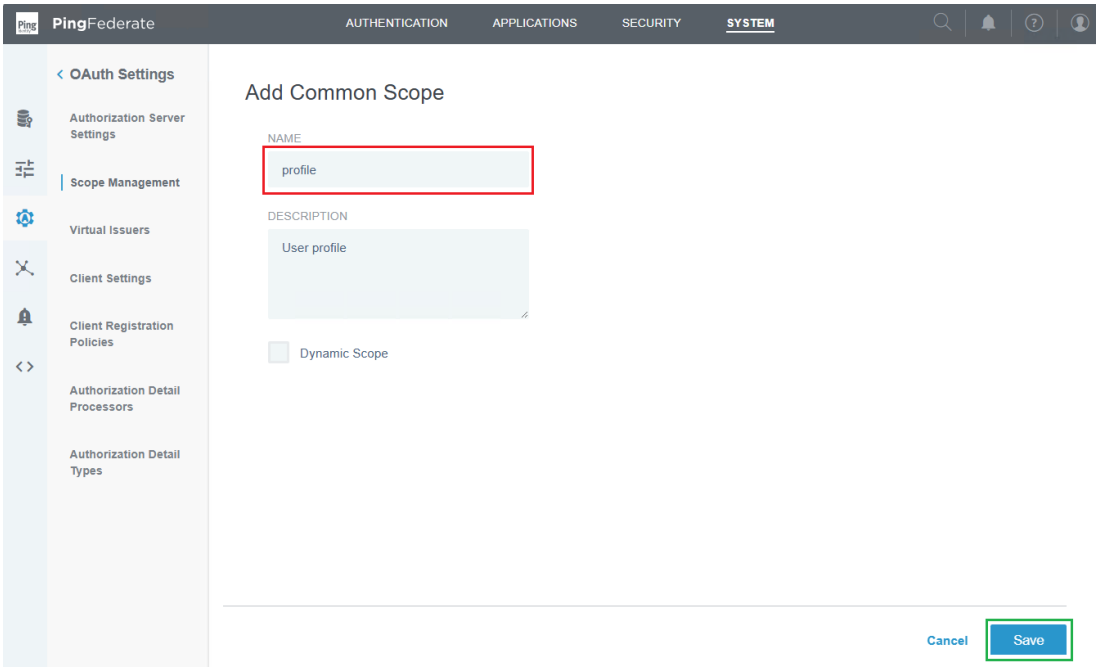
1. In homepage, navigate to `SYSTEM` and click `OAuth Scopes`.



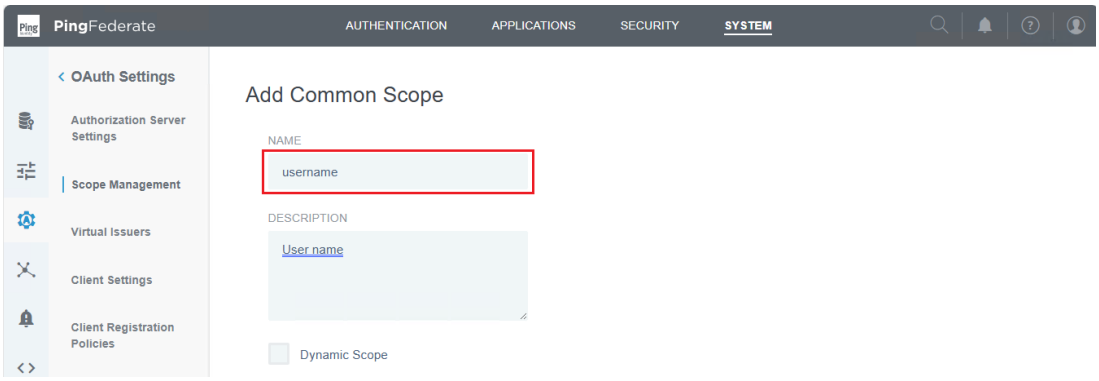
2. Under `Scope Management`, at `Common Scopes` tab, click `Add Common Scope`.



3. Due to MetaDefender Core requires **profile** scope, fill the scope name and click `Save`.



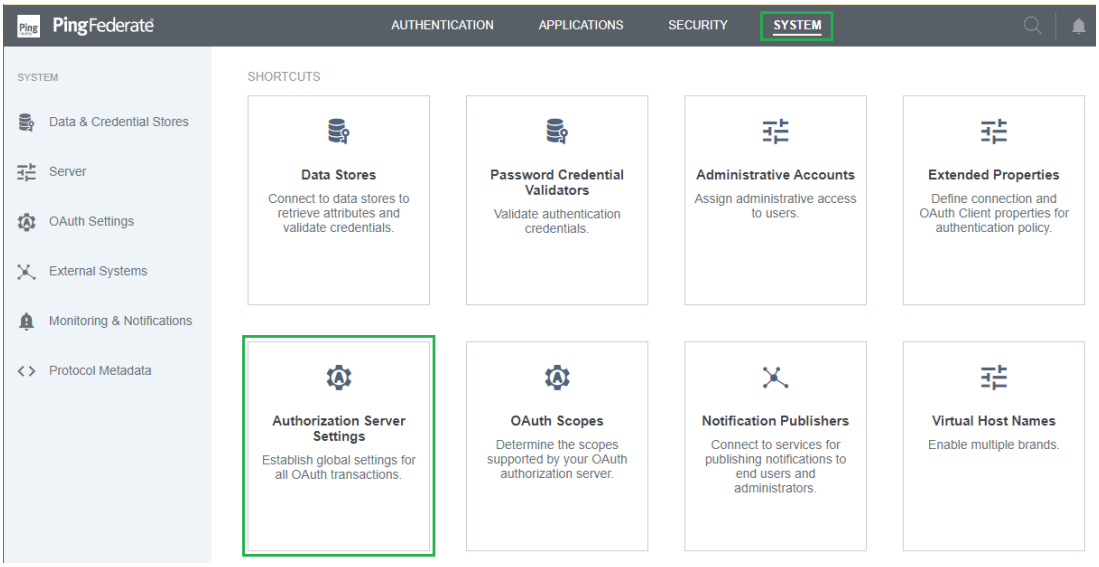
4. Repeat the above step to add `username` scope.



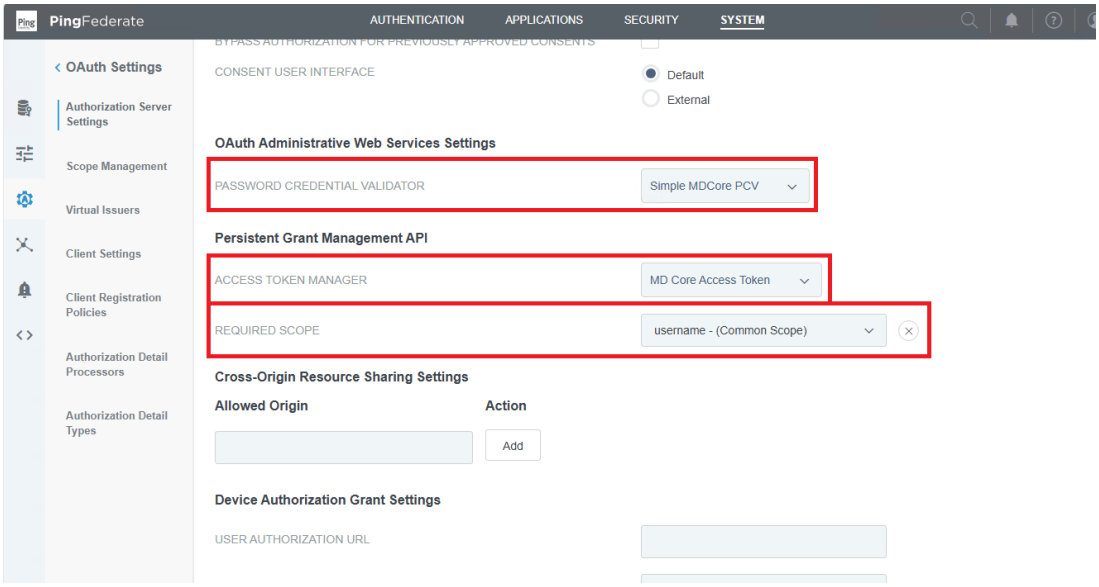
5. Click `Save` to complete.

Setup Authorization Server Settings

1. In homepage, navigate to `SYSTEM` and click `Authorization Server Settings`.



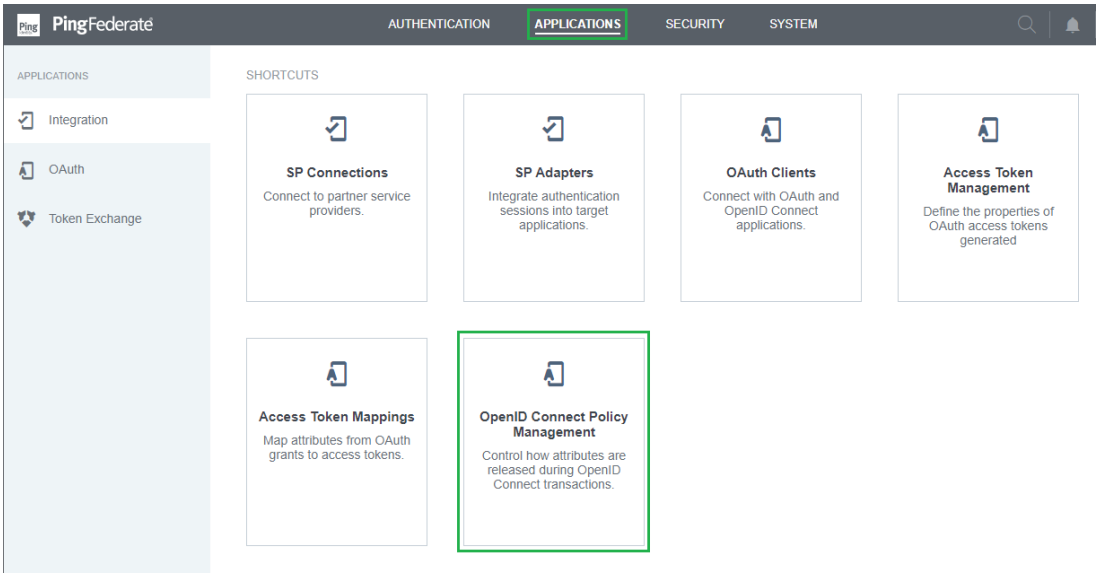
2. Navigate to `OAuth Administrative Web Services Settings` and select the password credential validator that was set up in previous stage for `PASSWORD CREDENTIAL VALIDATOR`.
3. Navigate to `Persistent Grant Management API`.
4. Select the access token manager and scope that were set up in the previous stage for `ACCESS TOKEN MANAGER` and `REQUIRED SCOPE`, respectively.



5. Click `Save` to complete.

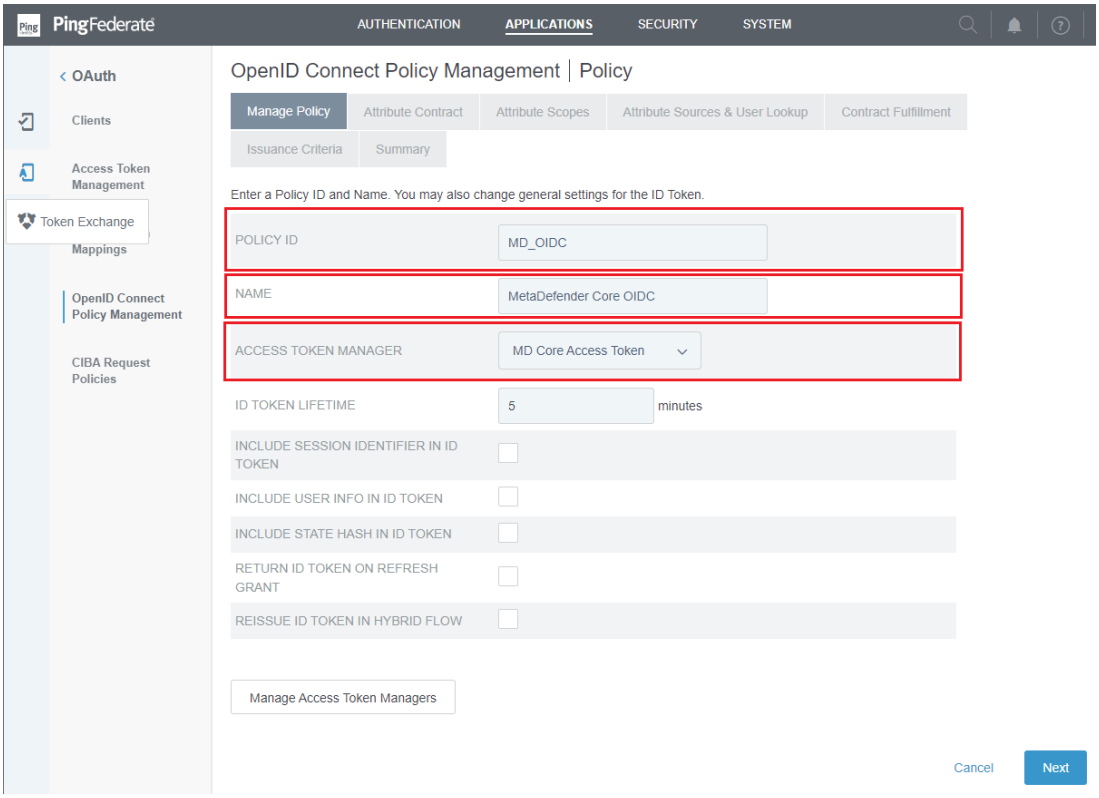
Set up OpenID Connect Policy Management

1. In homepage, navigate to `APPLICATIONS` and click `OpenID Connect Policy Management`.



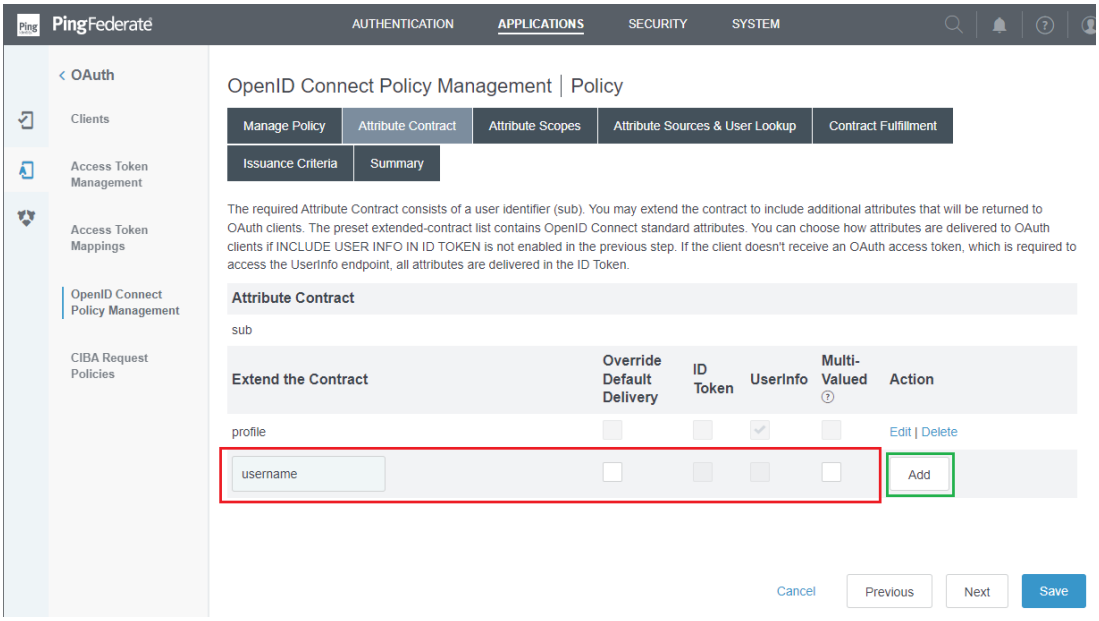
2. Click `Add Policy` in the next page.
3. In `Manage Policy` step, enter `POLICY ID`, `NAME` and select the manager that was set up in previous stage for `ACCESS TOKEN MANAGER`. Click `Next` on done.

Info
Store the value assigned to `POLICY ID` as *policy_id*. It will be used later on fetching metadata in MetaDefender Core.

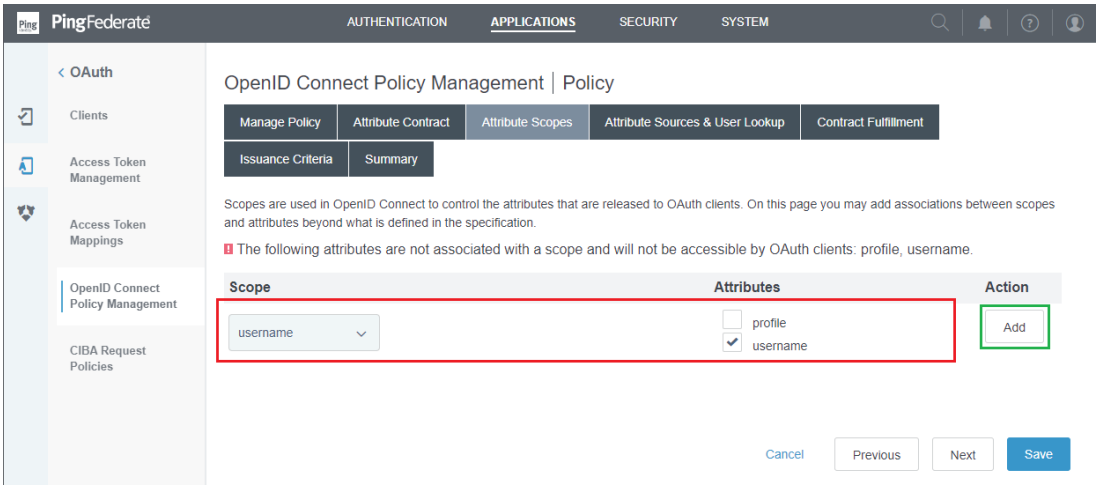


5. In `Attribute Contract` step, navigate to `Extend the Contract`, add the scopes requested by MetaDefender Core (i.e. username), and click `Next`.

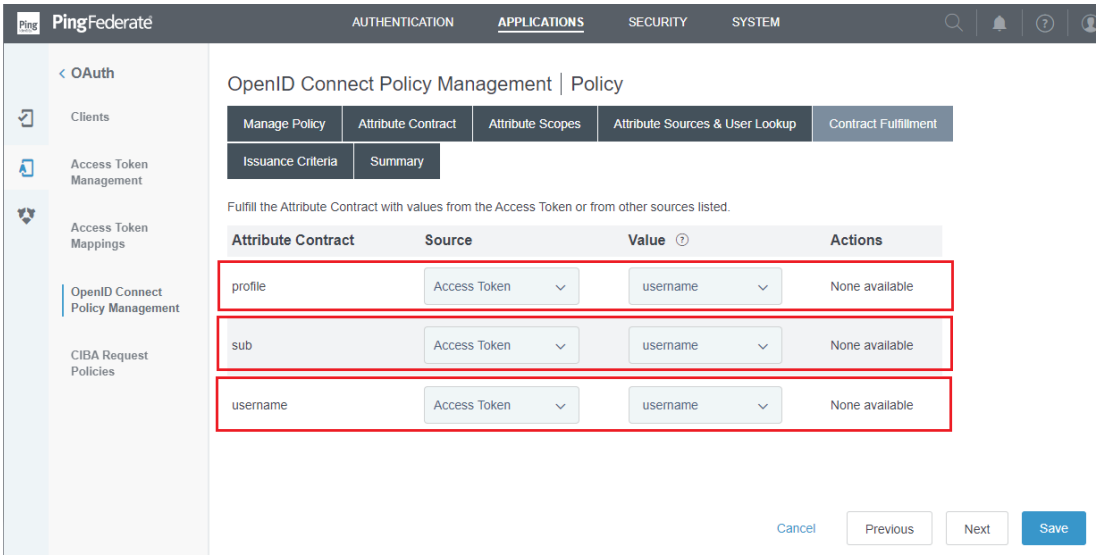
> **Info**
> PingFederate, by default, lists all extended contracts. Within the scope of the instructions, only `profile` contract is required, all the remaining ones must be deleted.



6. In `Attribute Scopes` step, match the scope with its attributes and click `Next`.



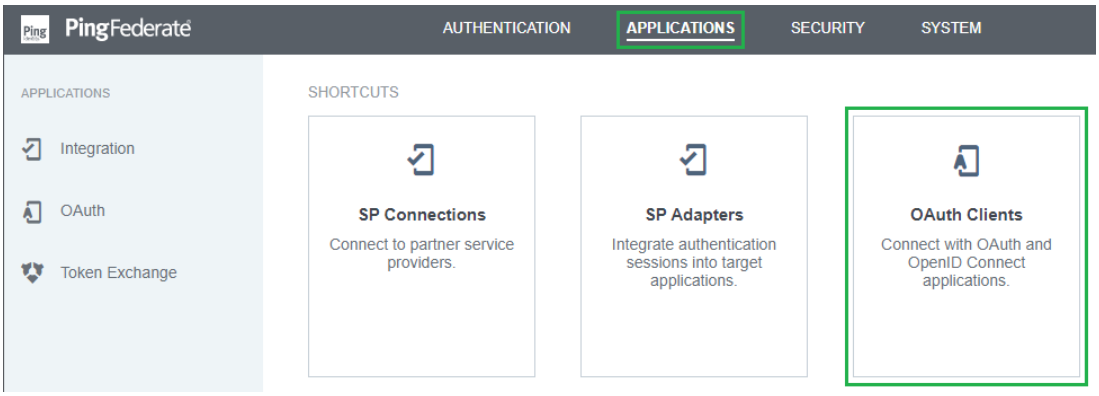
7. Click `Next` in `Attribute Sources & User Lookup` step.
8. In `Contract Fulfillment` step, select the appropriate `Source` and `Value`, and click `Next`.



9. Click `Next` in `Issuance Criteria` step.
10. Review all selected options in `Summary` tab, then click `Save`.

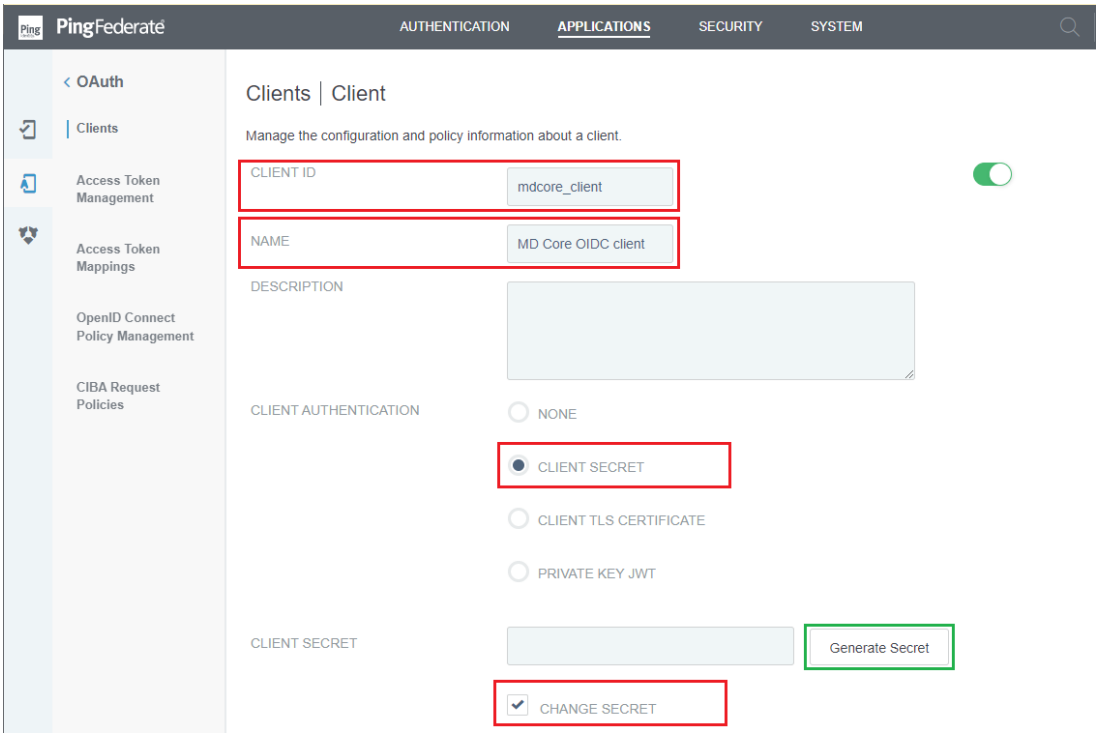
Add OAuth client

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*.

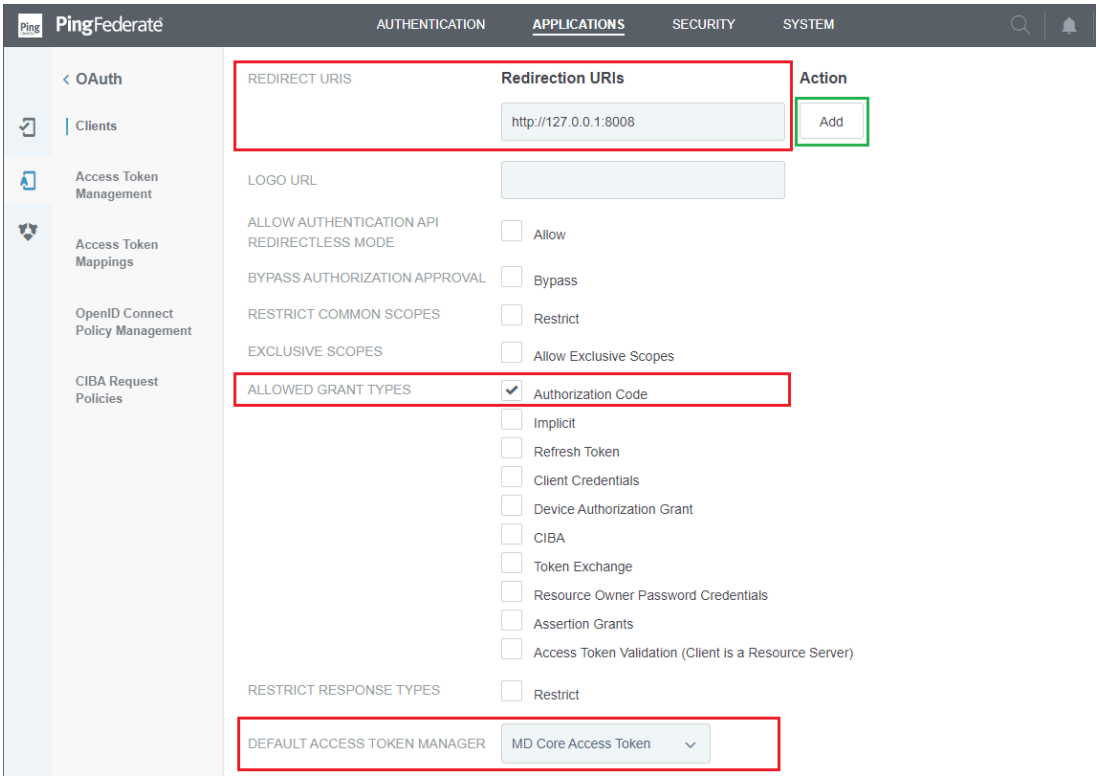
> **Info**
> *client_id* and *client_secret* will be used later on setting up MetaDefender Core.



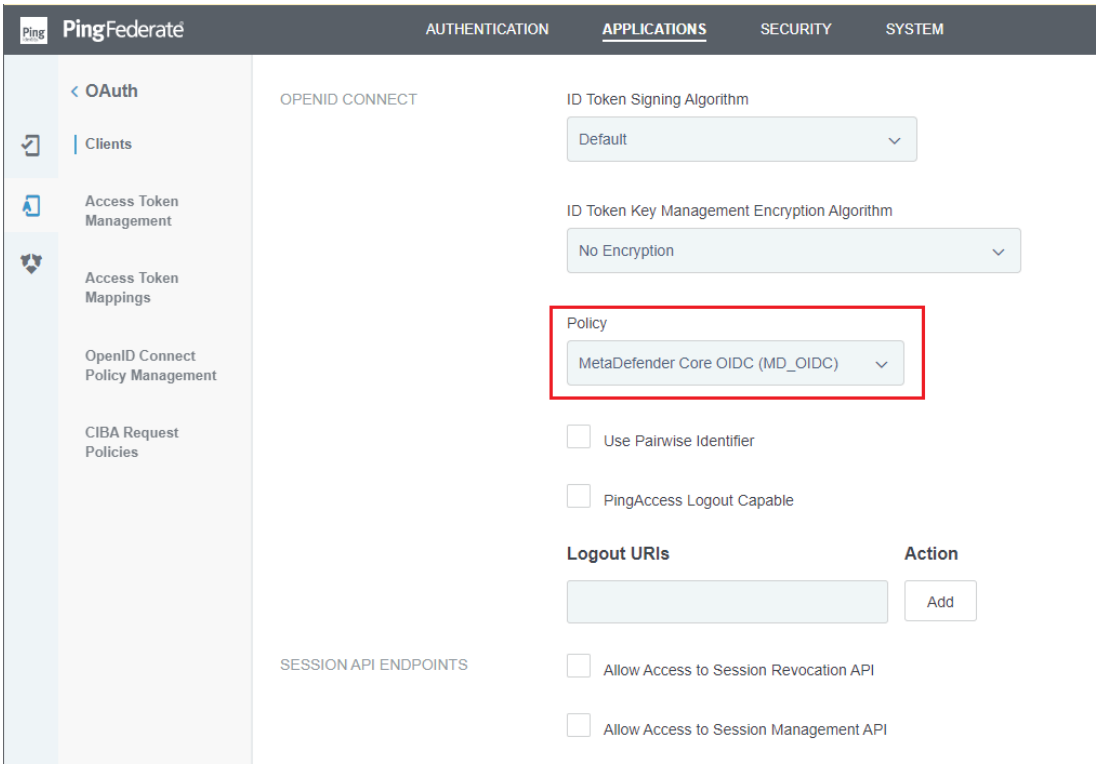
7. Navigate to `REDIRECT URIS`, fill in a temporary Redirection URI, and click `Add`.

> **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.

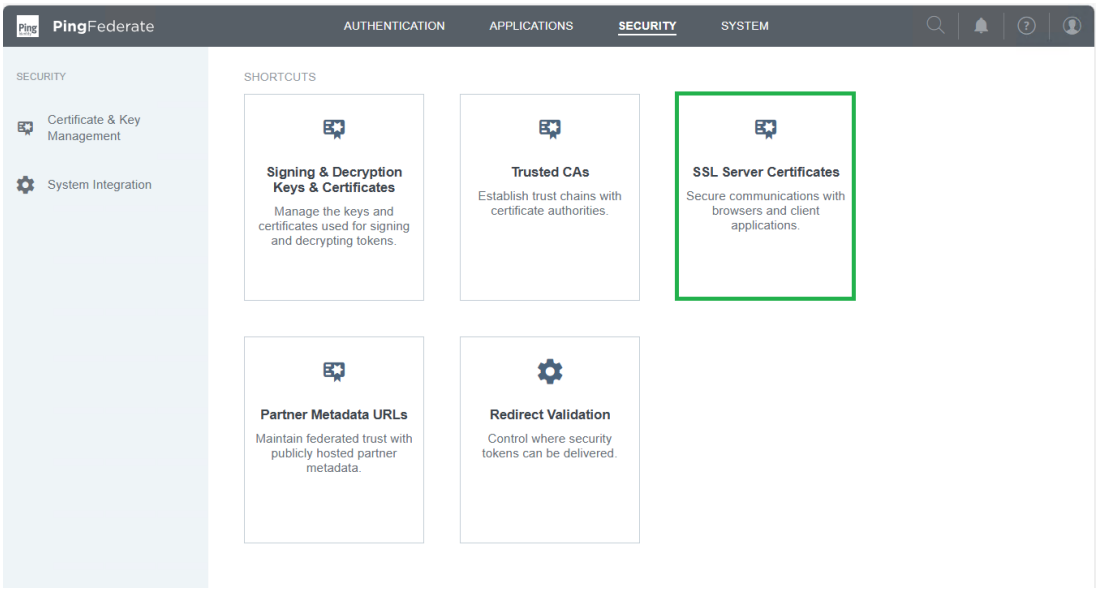


11. Click `Save` to complete.

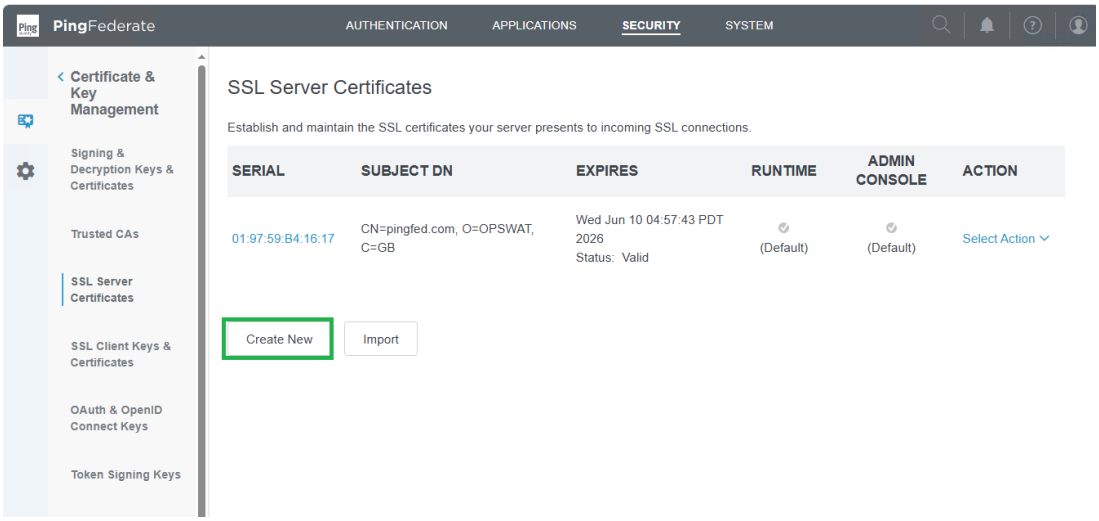
Set up MetaDefender Core

Install PingFederate certificate

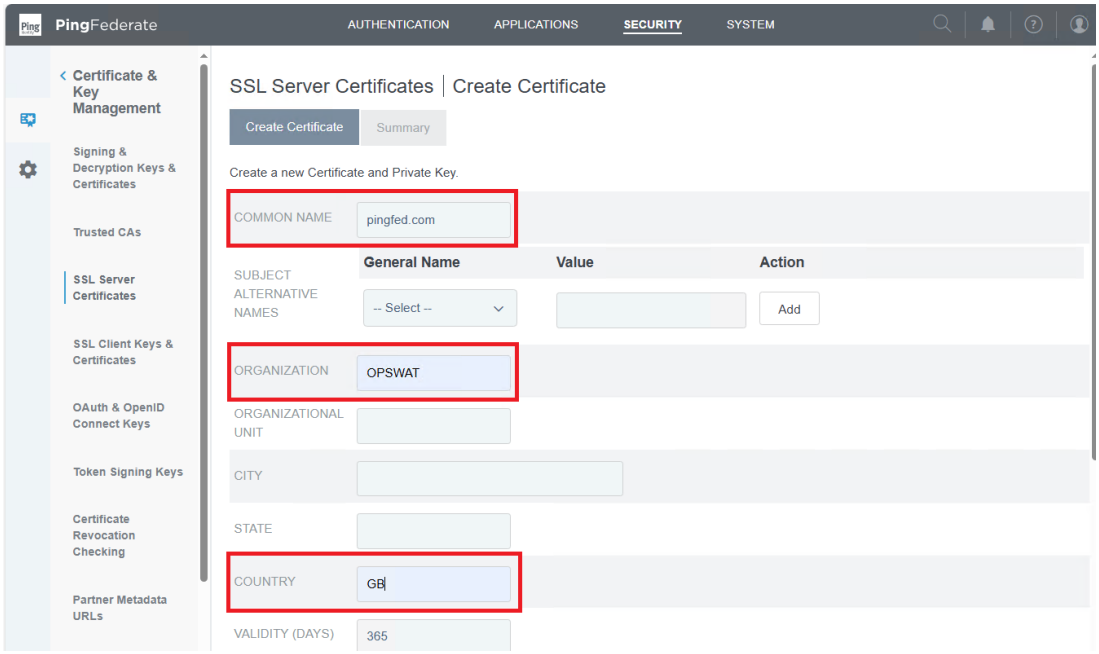
1. In homepage, go to `SECURITY` and click `SSL Server Certificates`.



2. Click `Create New` to generate a new certificate.



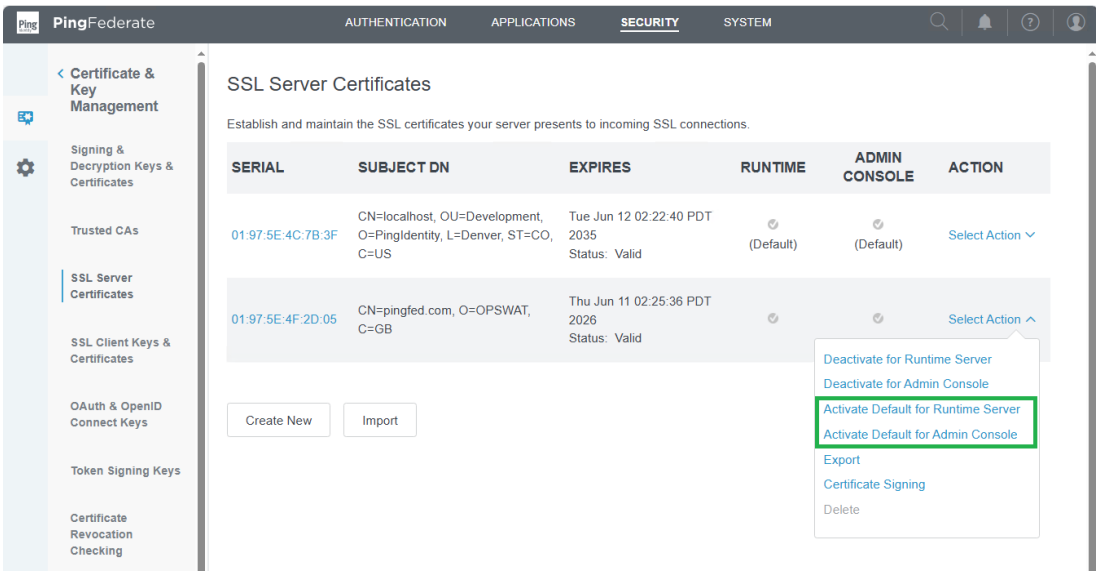
3. Complete the information for the new certificate, click `Next` and `Save`.



Info

`COMMON NAME` must refer to the domain or IP address hosting PingFederate.

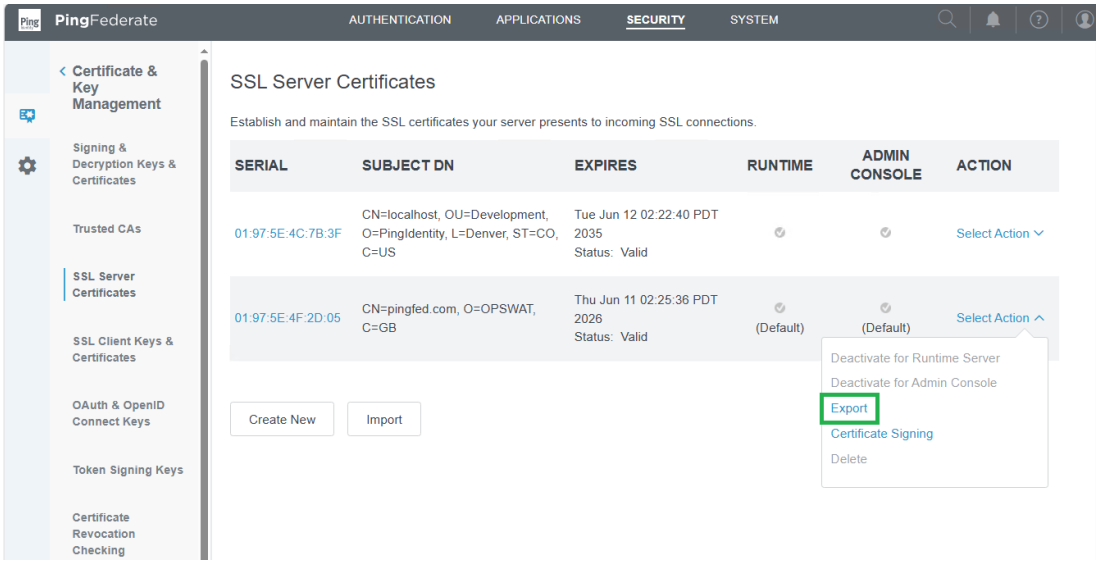
4. Activate the new certificate.



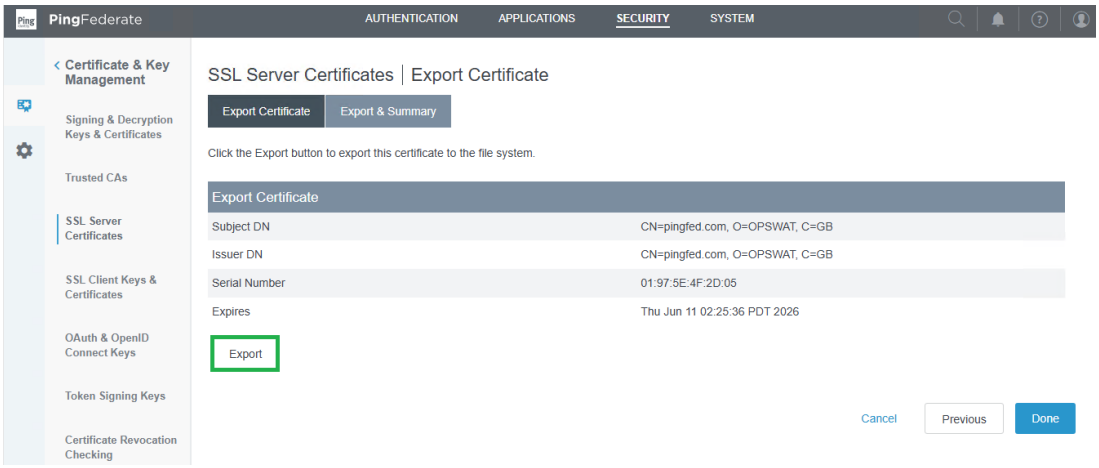
Warning

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`.



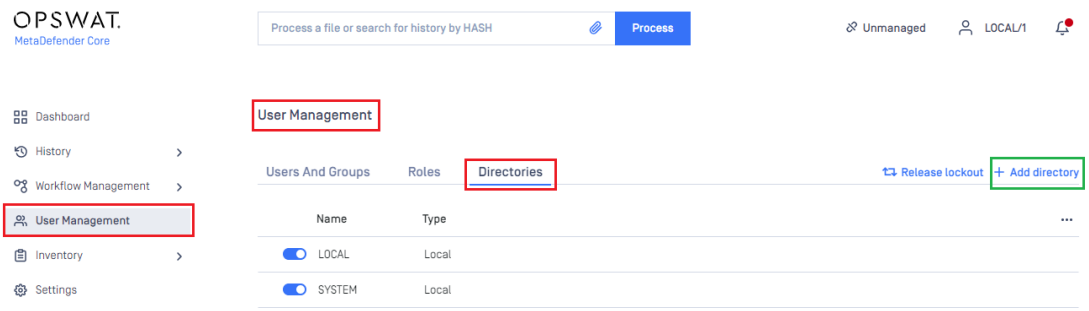
6. Retain the default settings and click `Next`.
7. Click `Export` to download the certificate.



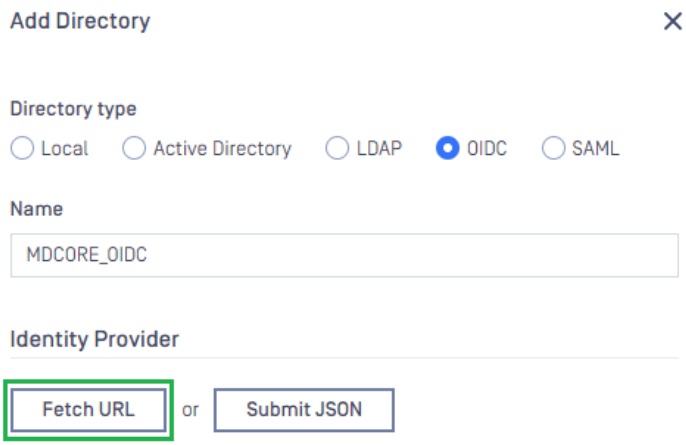
8. Install the certificate on the server running MetaDefender Core.
9. Restart MetaDefender Core for the changes to take effect.

Configure OIDC in MetaDefender Core

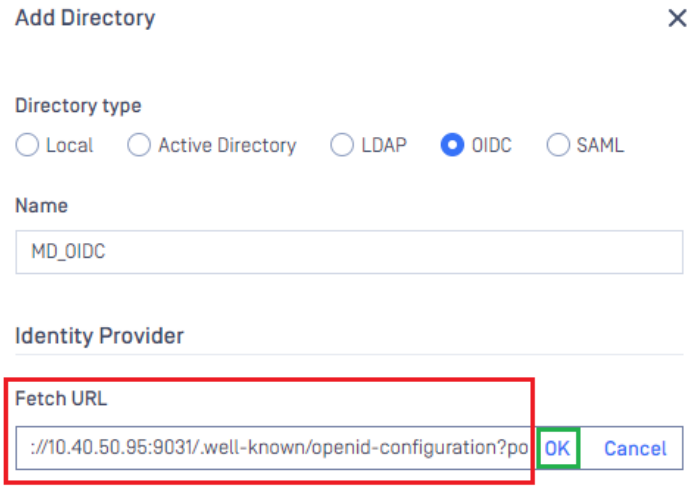
1. Sign in to MetaDefender Core.
2. In the dashboard, click on `User Management` in the left sidebar.
3. On `User Management` page, select `Directories` tab and click `Add Directory` in the top right corner.



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`.

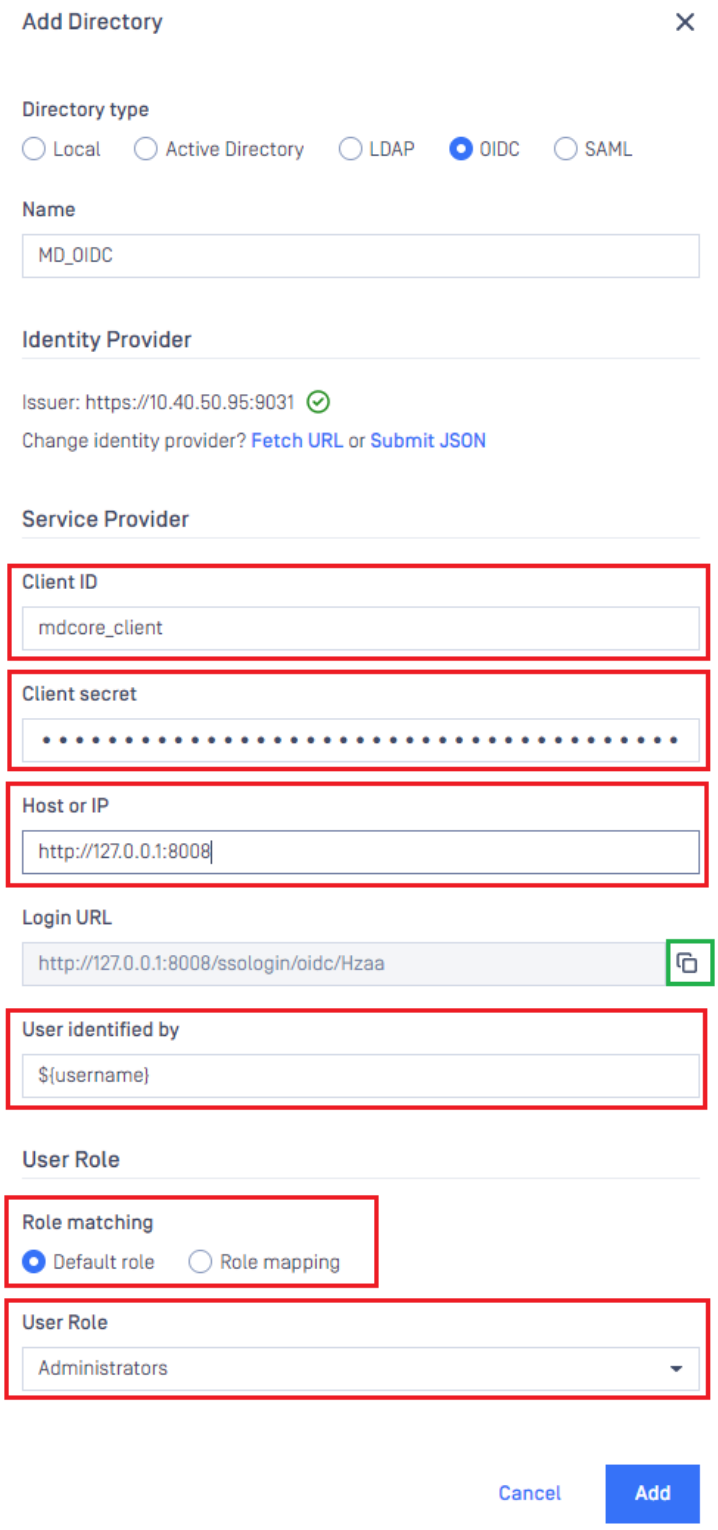


Info

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**.

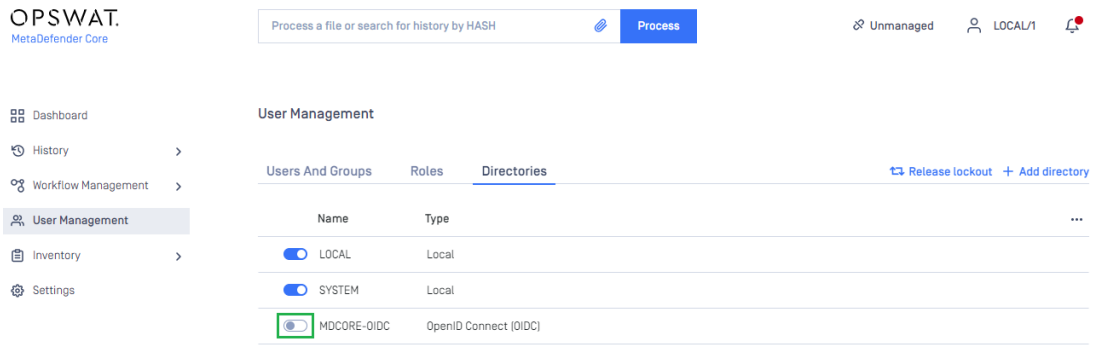
7. Under `Service Provider`, paste *client_id* and *client_secret* to boxes under `Client ID` and `Client secret` respectively.
8. Fill in `Host or IP` with the host or IP where MetaDefender Core is hosted, such as http://127.0.0.1:8008 for this example.



9. Copy `Login URL` and store it as *login_url*.

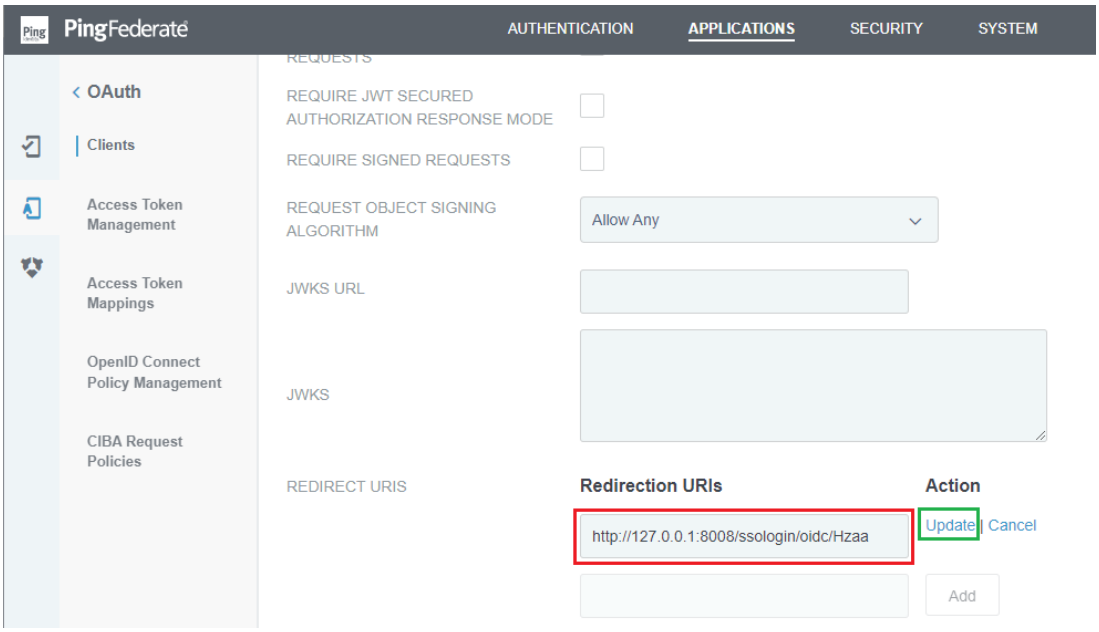
> **Info**
> *login_url* is used to update Redirection URI setting in PingFederate in the next stage.

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.



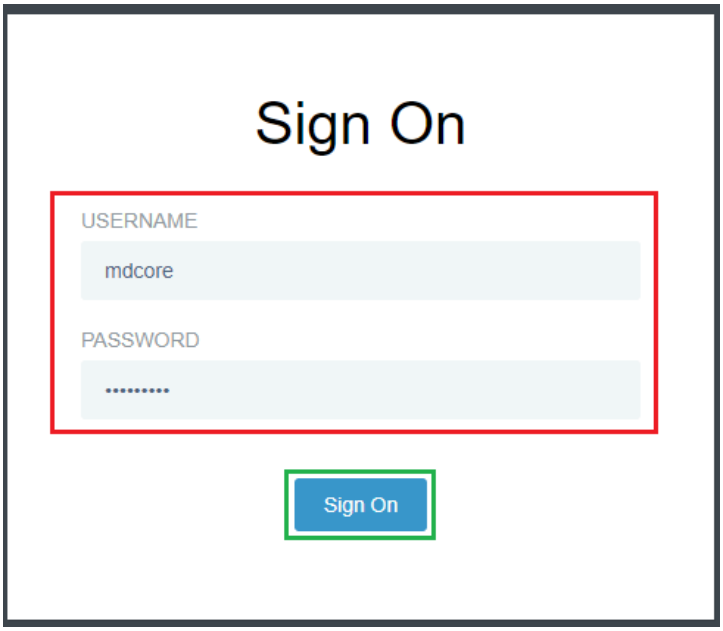
Complete configuration in PingFederate

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`.

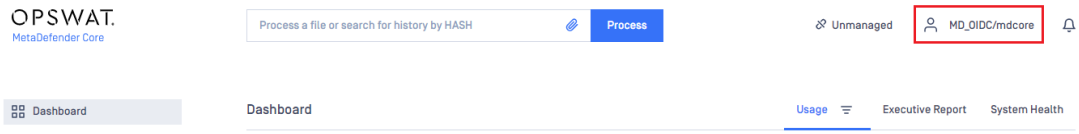


Test the integration

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.



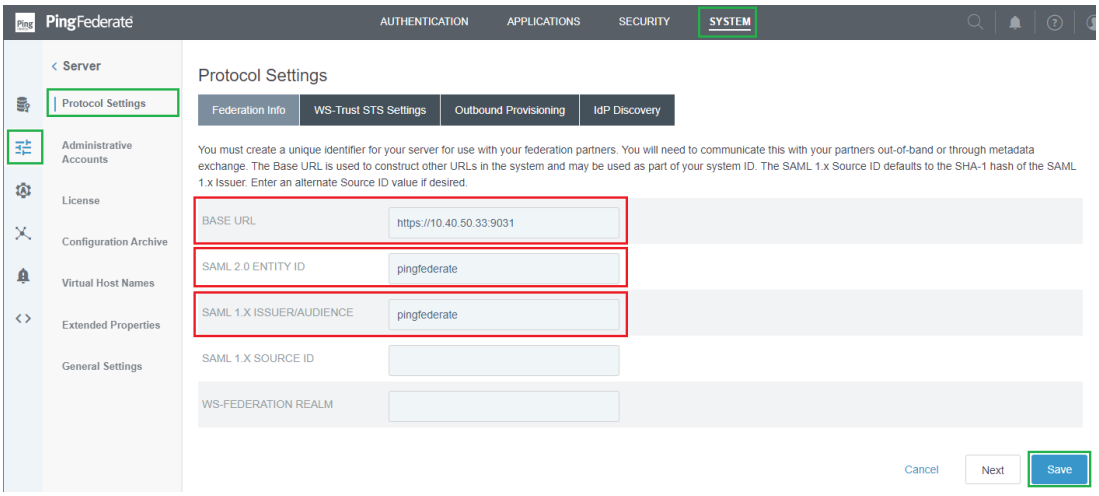
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Configuring SAML Single Sign-on

Set up PingFederate

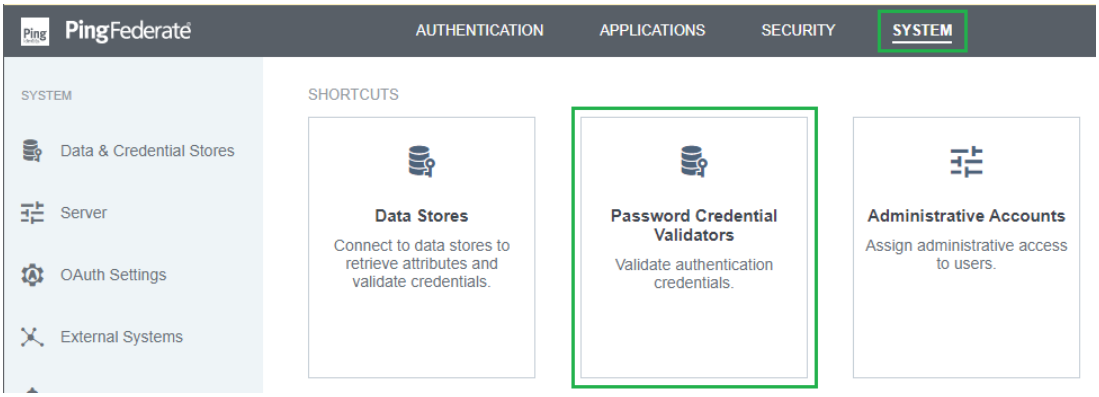
Configurate PingFederate

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.



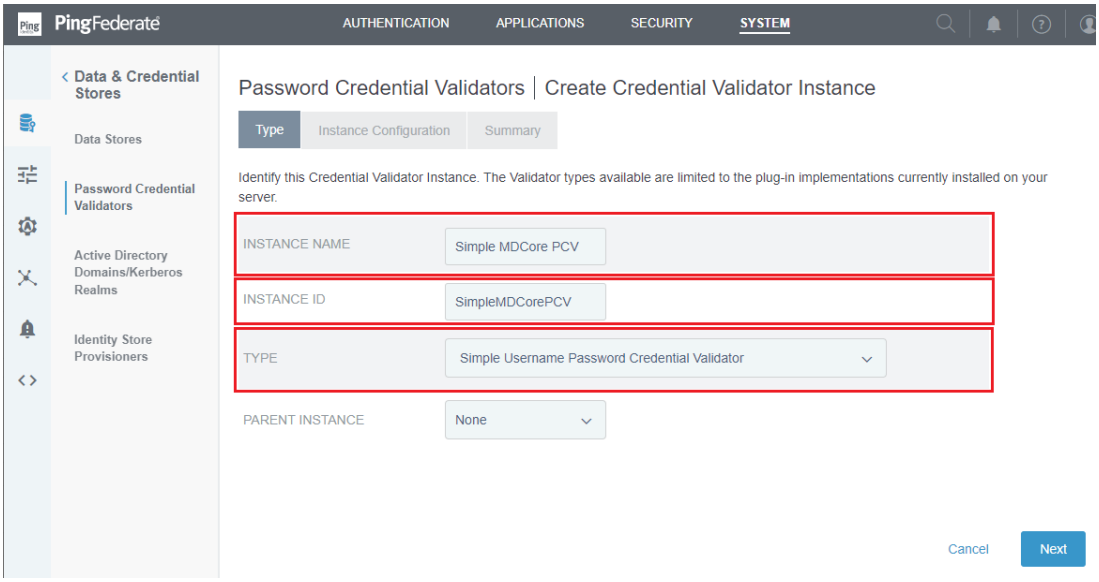
Set up simple Password Credential Validators

1. In homepage, navigate to `SYSTEM` tab and click on `Password Credential Validators`.

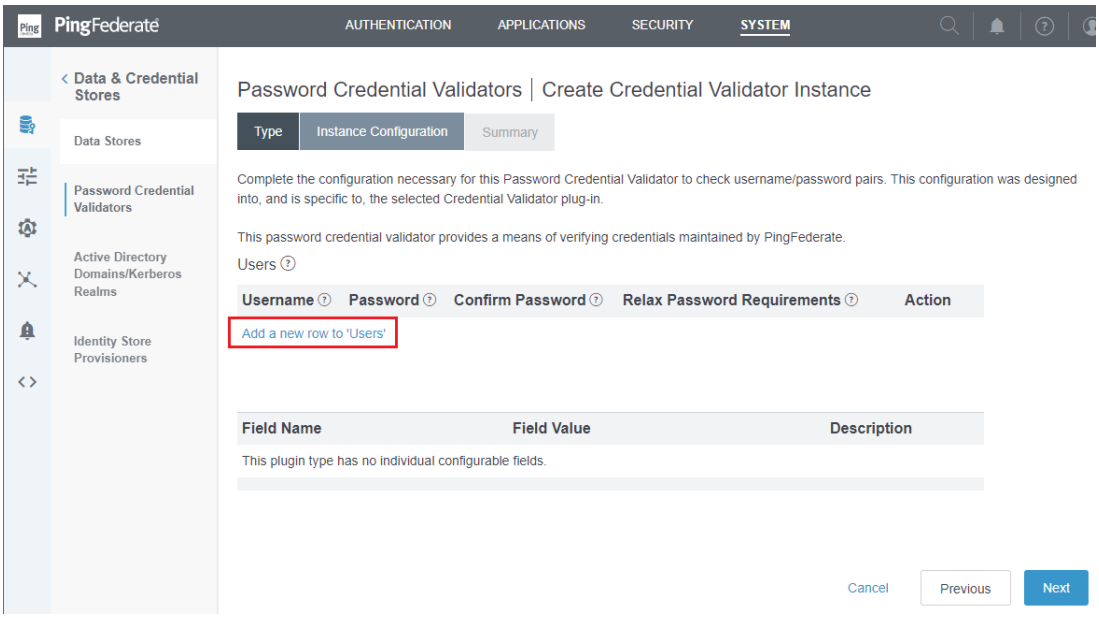


2. Click `Create New Instance`.

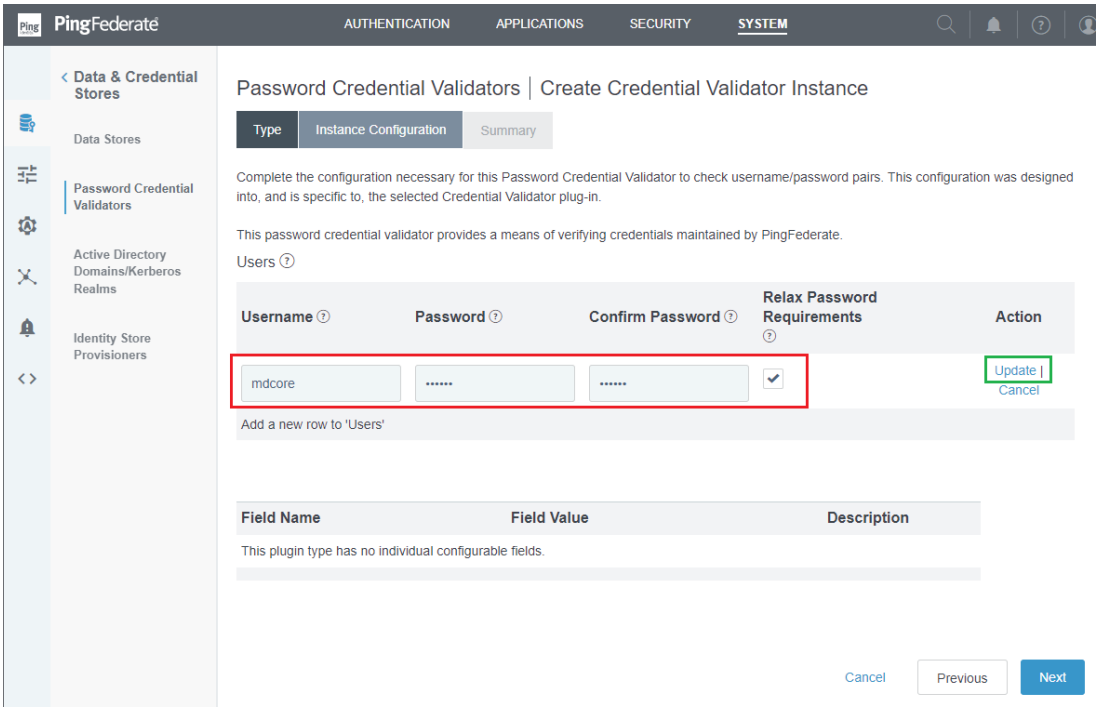
3. Enter `INSTANCE NAME` and `INSTANCE ID`, select `Simple Username Password Credential Validator` for `TYPE`, and click `Next`.



4. In `Create Credential Validator Instance` page, click `Add a new row to 'Users'.`



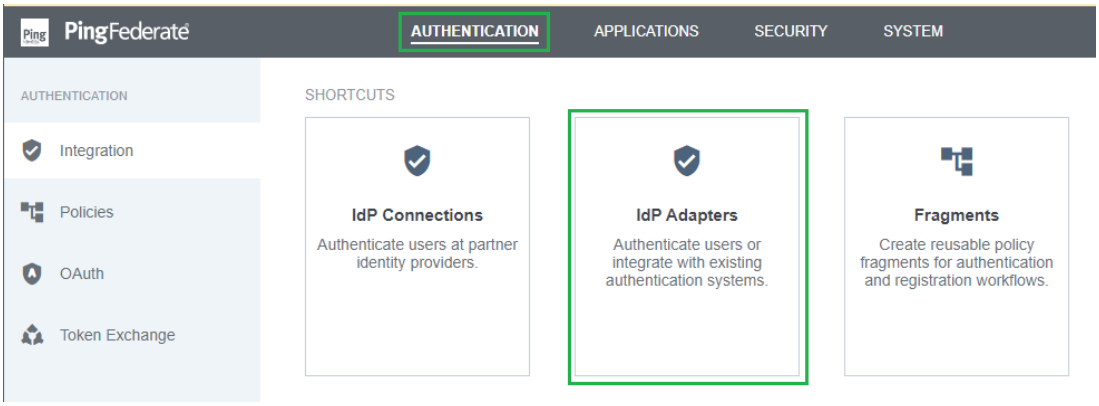
5. Enter `Username`, `Password`, and `Confirm Password`, then click `Update` and `Next`.



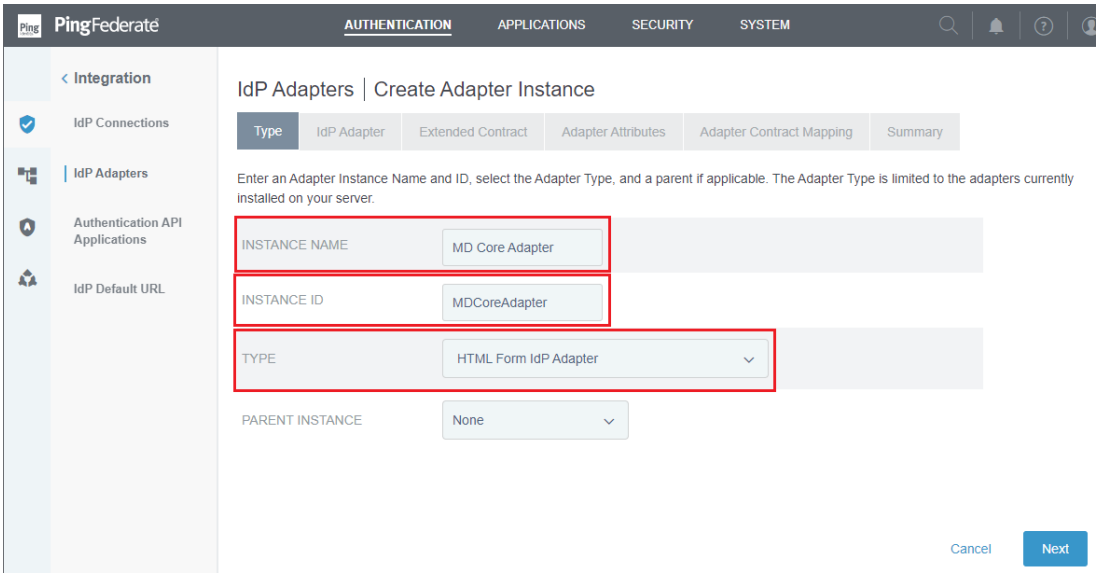
6. Review all selected options in `Summary` tab, and click `Save`.

Set up a simple IdP Adapter

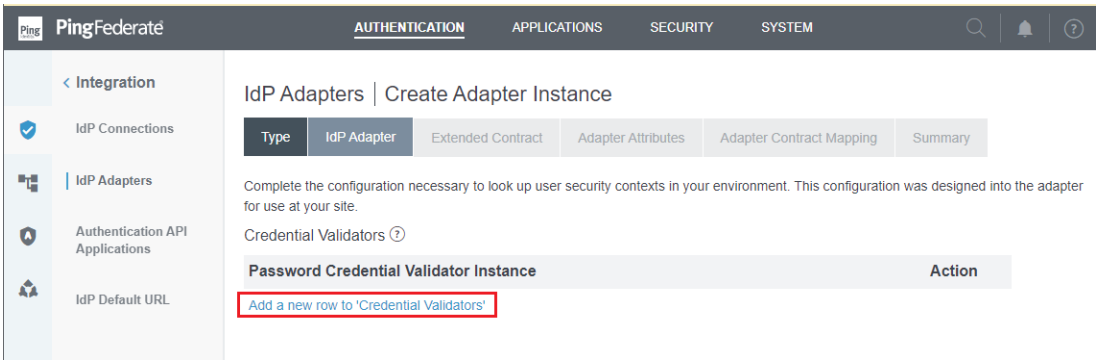
1. In homepage, navigate to `AUTHENTICATION` tab and click on `IdP Adapters`.



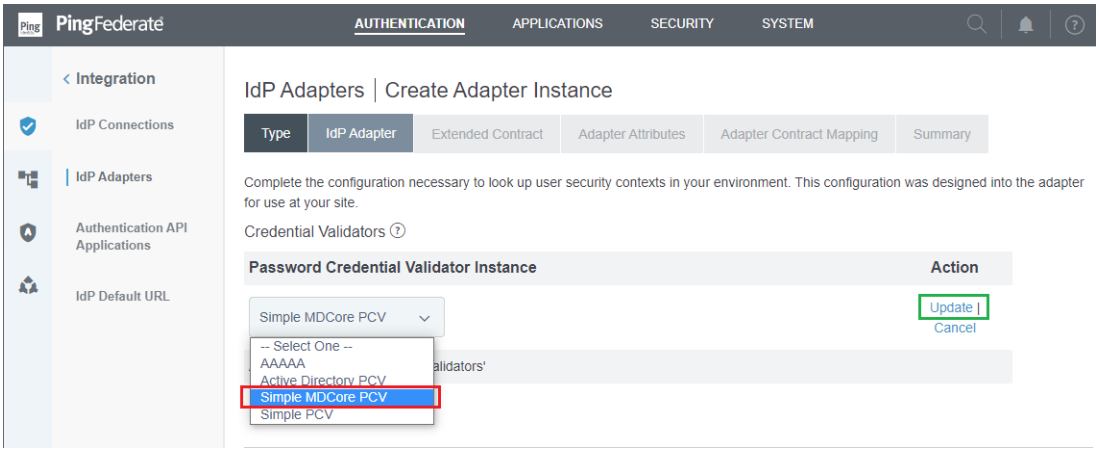
2. Click `Create New Instance`.
3. Enter `INSTANCE NAME`, `INSTANCE ID`, and select `HTML Form IdP Adapter` for `TYPE`. Click `Next`.



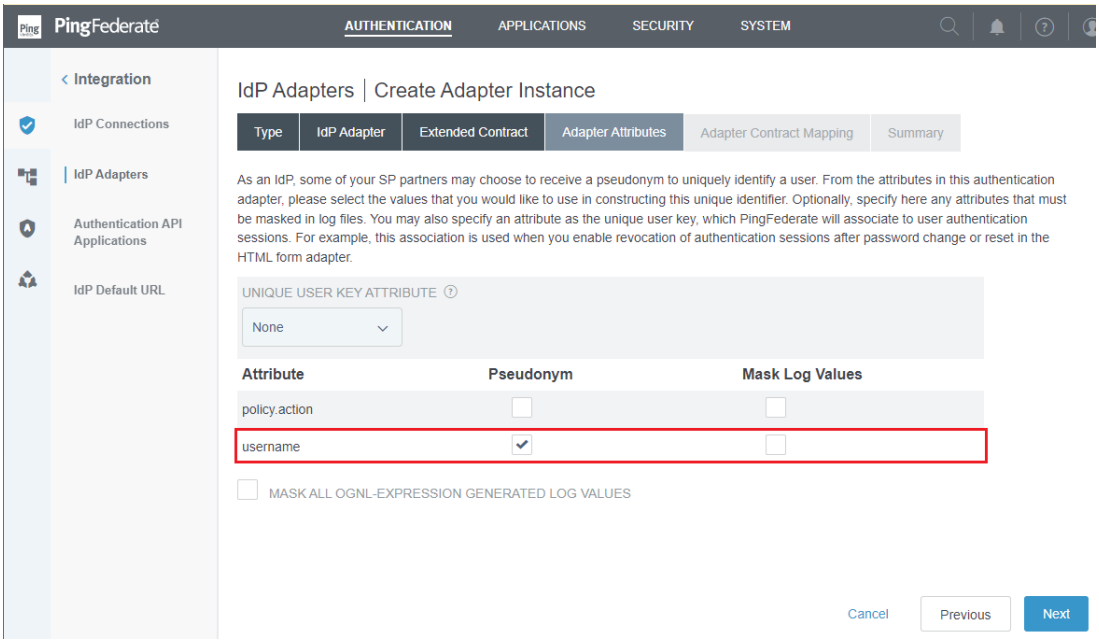
4. Click `Add a new row to 'Credential Validators'`.



5. Select the Password Credential Validator created in the previous stage, that is "Simple MDCore PCV", then click `Update` and `Next`.



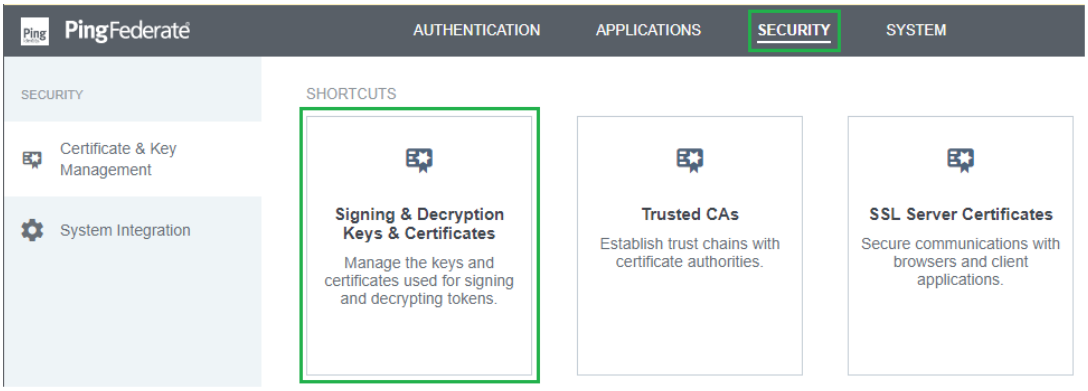
6. Click `Next` on `Extended Contract` tab.
7. In `Adapter Attributes` tab, toggle `username` attribute under `Pseudonym` column, and click `Next`.



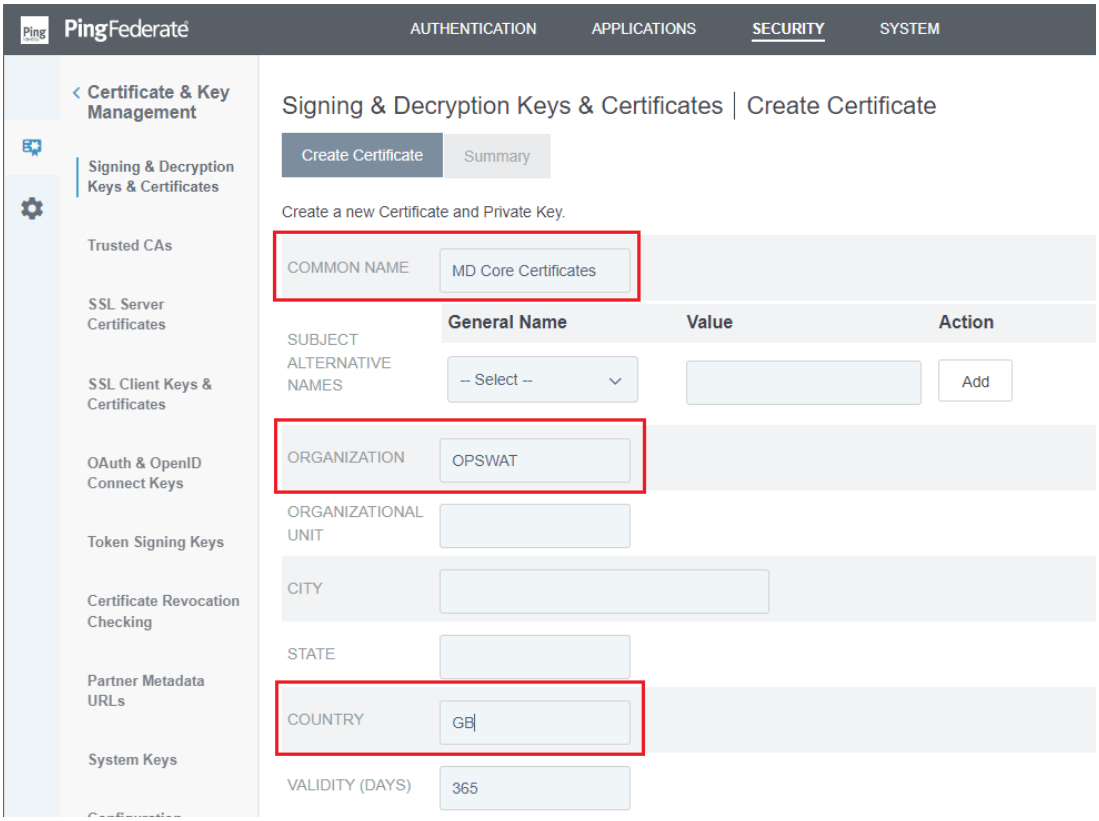
8. Click `Next` on `Adapter Contract Mapping` tab.
9. Review all selected options in `Summary` tab and click `Save` to complete.

Set up Signing, Decryption Keys and Certificates

1. In homepage, navigate to `SECURITY` tab and select `Signing & Decryption Keys & Certificates`.



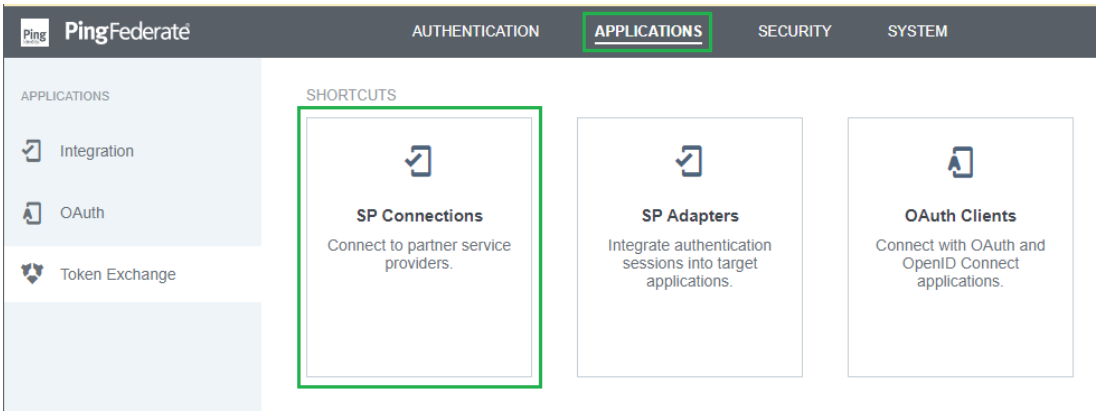
2. Click `Create New`.
3. Enter `COMMON NAME`, `ORGANIZATION`, and `COUNTRY`, along with any other necessary information, then click `Next`.



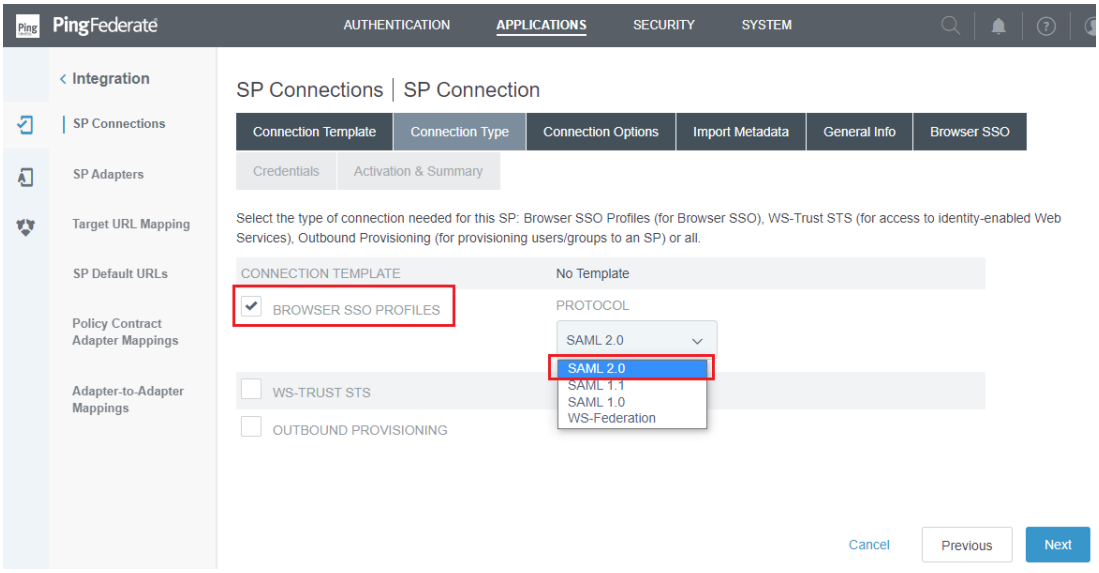
4. Review all selected options in `Summary` tab and click `Save` to complete.

Set up Service Provider connections

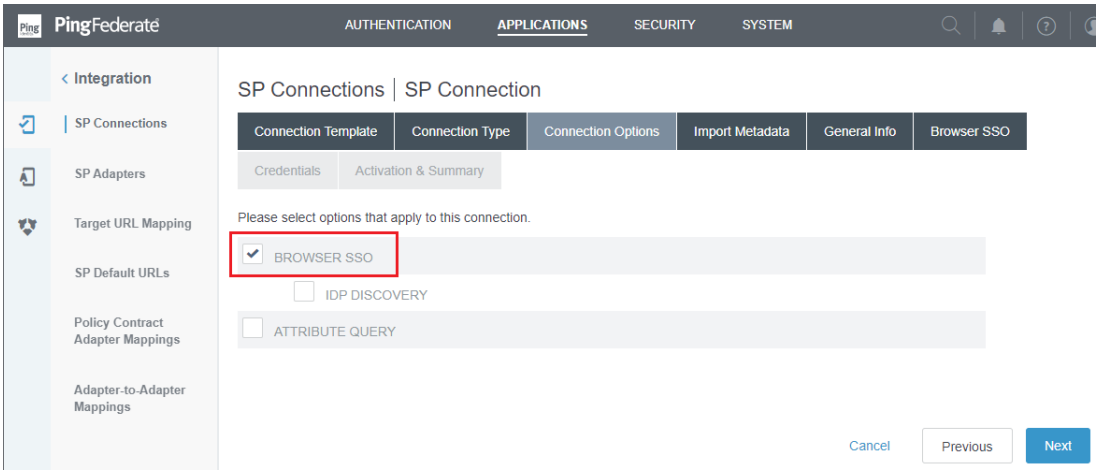
1. In homepage, go to `APPLICATIONS` tab and choose `SP Connections`.



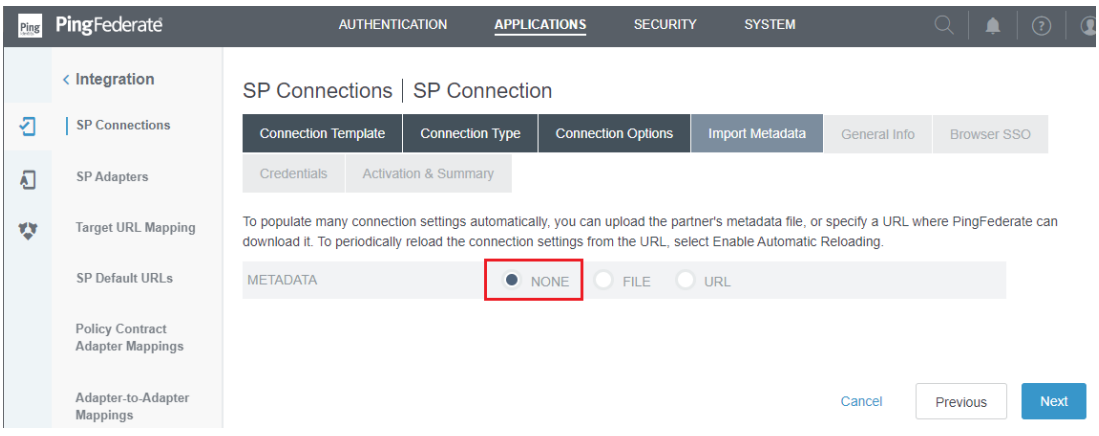
2. Select `Create Connection`.
3. Select `DO NOT USE A TEMPLATE FOR THIS CONNECTION` and then click `Next`.
4. In `Connection Type` phase, toggle `BROWSER SSO PROFILES` and choose `SAML 2.0` for `PROTOCOL`. Press `Next`.



5. In `Connection Options` phrase, select `BROWSER SSO` and then click `Next`.

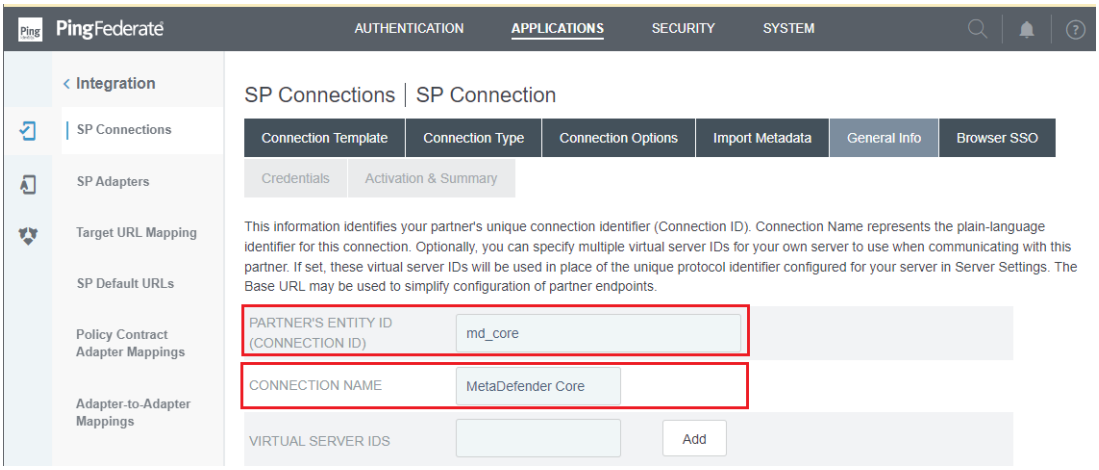


6. In `Import Metadata` phase, select `NONE` for `METADATA` subtab and press `Next`.

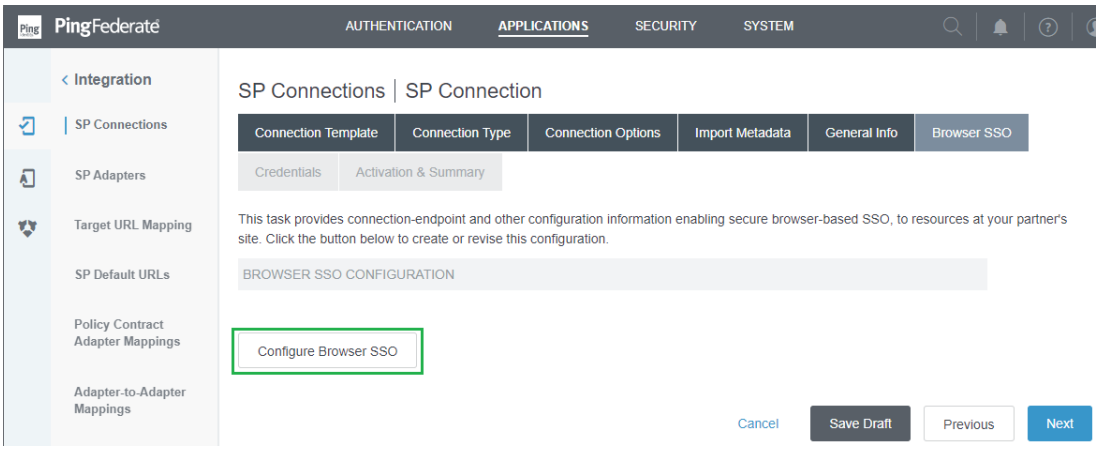


7. In `General Info` section, enter `PARTNER'S ENTITY ID (CONNECTION ID)` and `CONNECTION NAME`, then click `Next`.

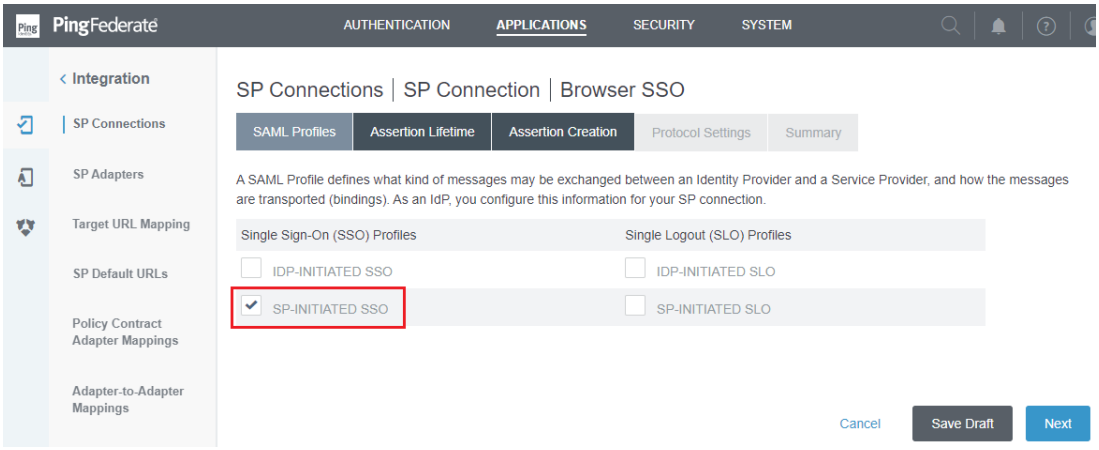
> **Info**
> `PARTNER'S ENTITY ID`, 'md_core' in this case, will later be used to configure `Custom entity ID` in MetaDefender Core.



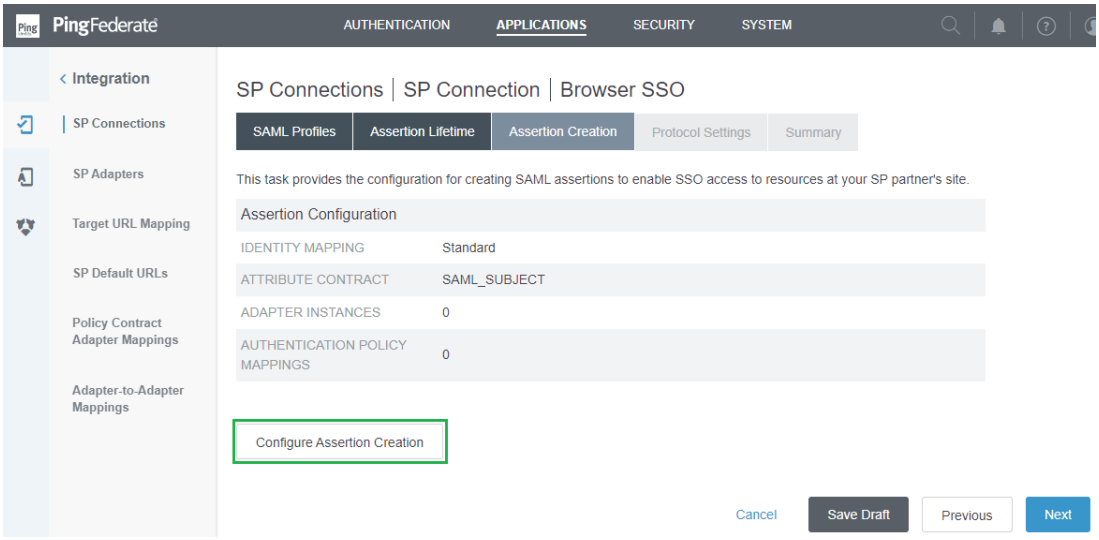
8. In `Browser SSO` phrase, select `Configure Browser SSO`.



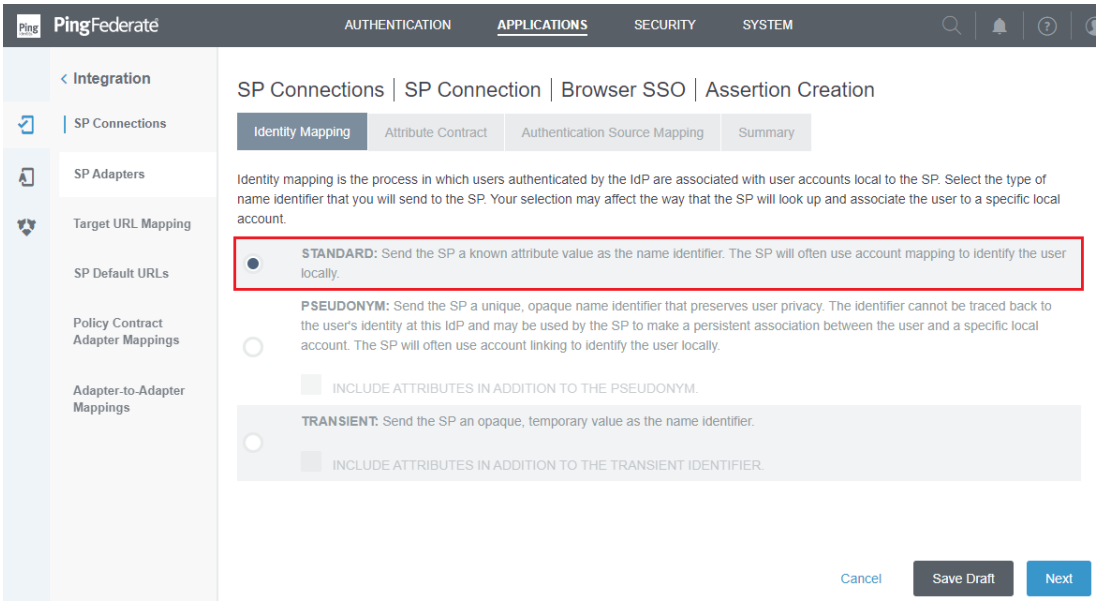
9. In `SAML Profiles` phrase, check `SP-INITIATED SSO` under `Single Sign-On (SSO) Profiles` column. Press `Next` once you are finished.



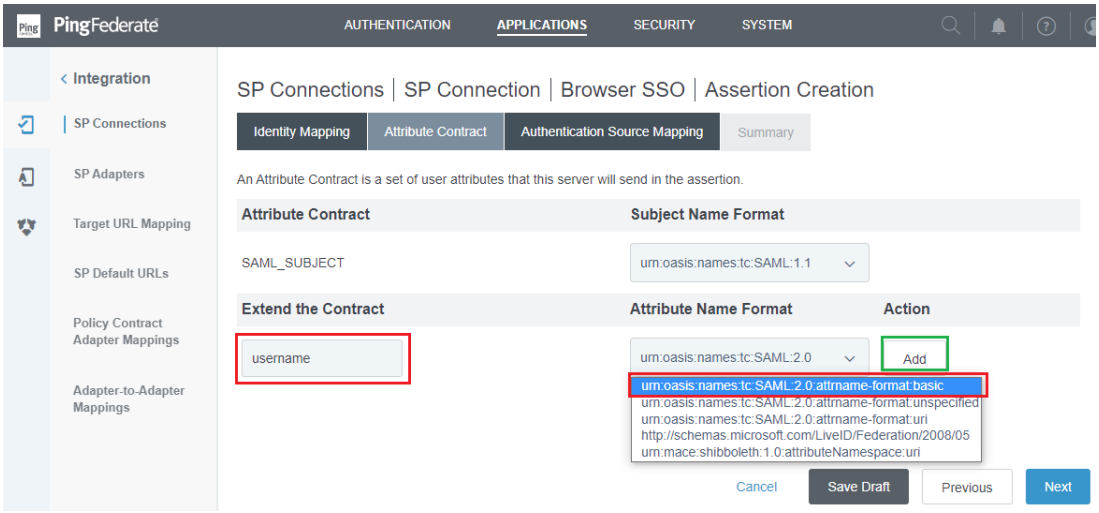
10. Select `Next` in `Assertion Lifetime` phase.
11. In `Assertion Creation` phase, select `Configure Assertion Creation`.



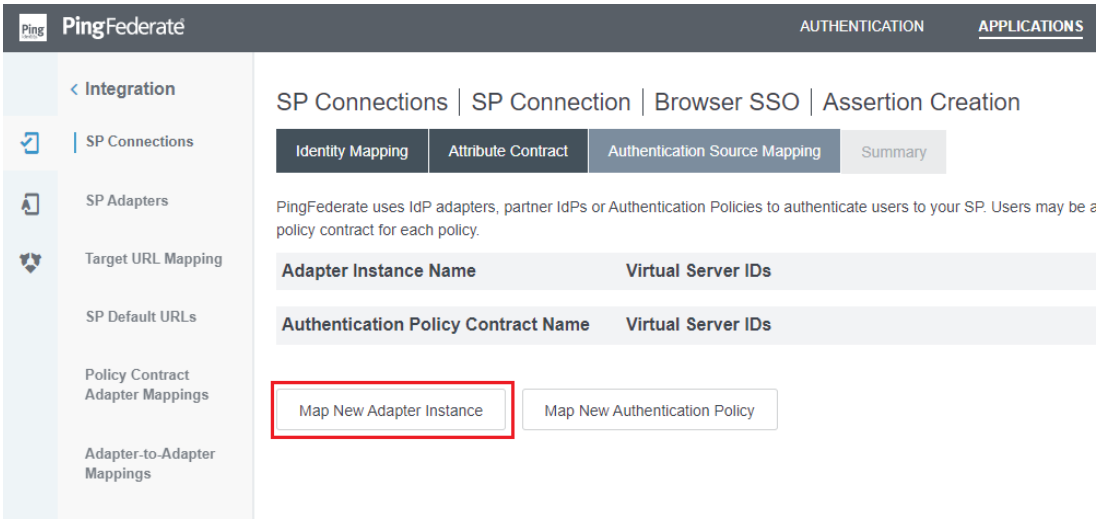
12. In `Identity Mapping` phrase, choose `STANDARD` and press `Next`.



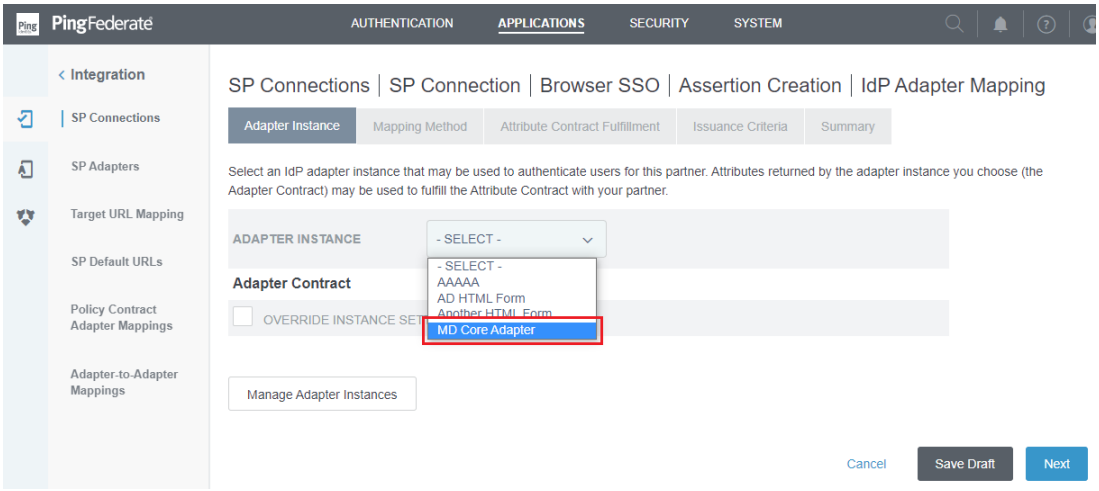
13. In `Attribute Contract` phase, navigate to `Extend the Contract`, complete the contract needed by MetaDefender Core, using 'username' as a sample. Select the basic format for `Attribute Name Format`, then click `Add` followed by `Next`.



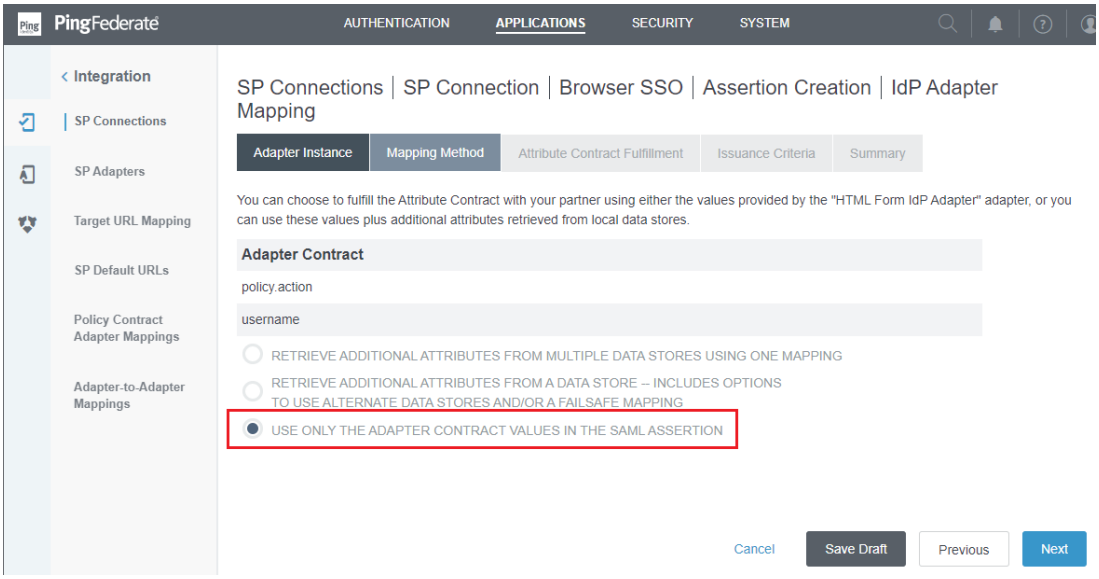
14. In `Authentication Source Mapping` phase, select `Map New Adapter Instance`.



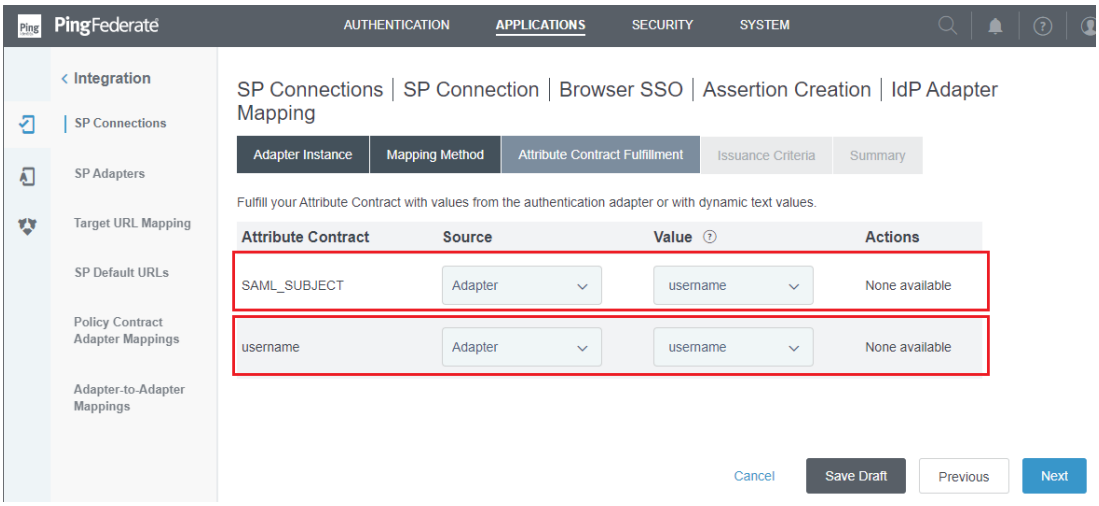
15. In `Adapter Instance` phase, choose the appropriate IdP Adapter for `ADAPTER INSTANCE` and then click `Next`.



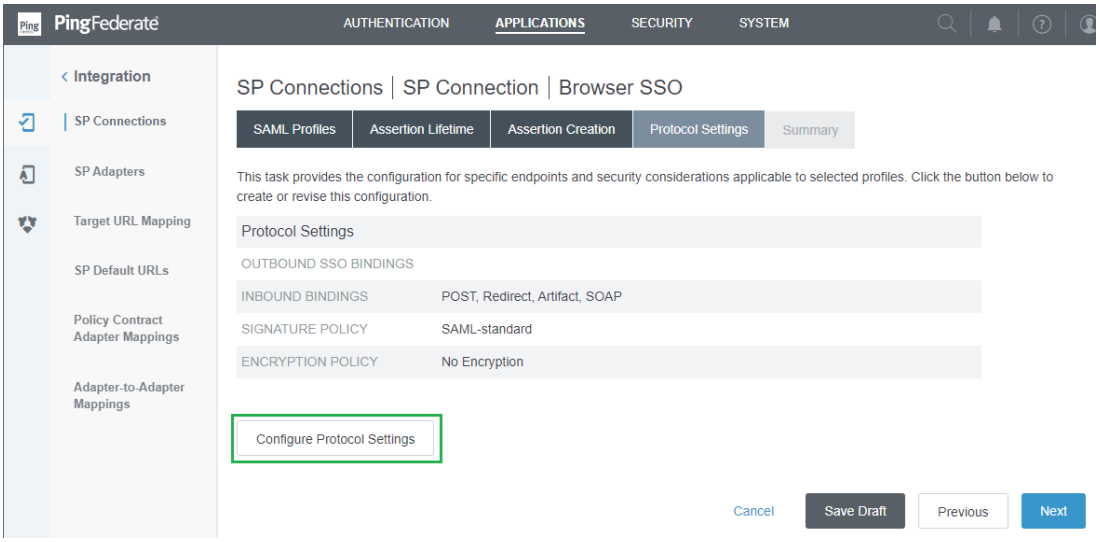
16. In `Mapping Method` phase, select `USE ONLY THE ADAPTER CONTRACT VALUES IN THE SAML ASSERTION`, then press `Next`.



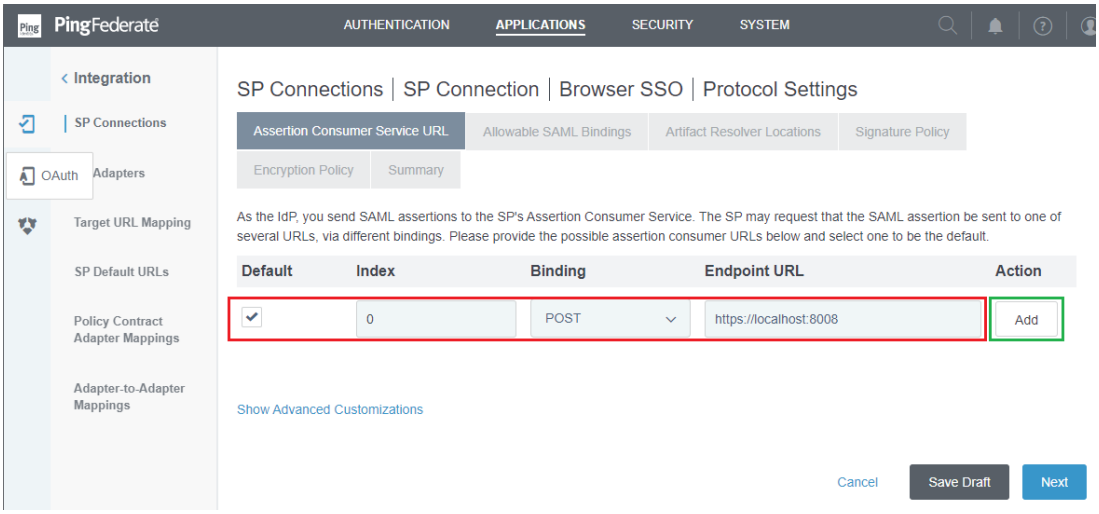
17. In `Attribute Contract Fulfillment` phase, choose 'Adapter' for `Source` column and 'username' for `Value` column, then click `Next`.



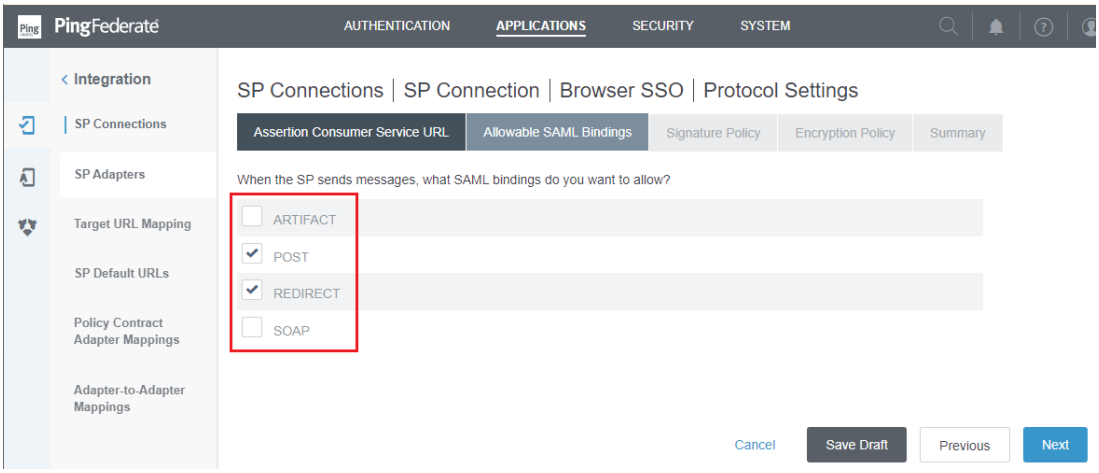
18. Press `Next` on `Issuance Criteria` phrase.
19. Examine all chosen options in `Summary`, then select `Done` and `Next`.
20. Return to `Assertion Creation` phrase, examine all chosen options in `Summary`, and then click `Done` and `Next`.
21. In `Protocol Settings` phase, select `Configure Protocol Settings`.



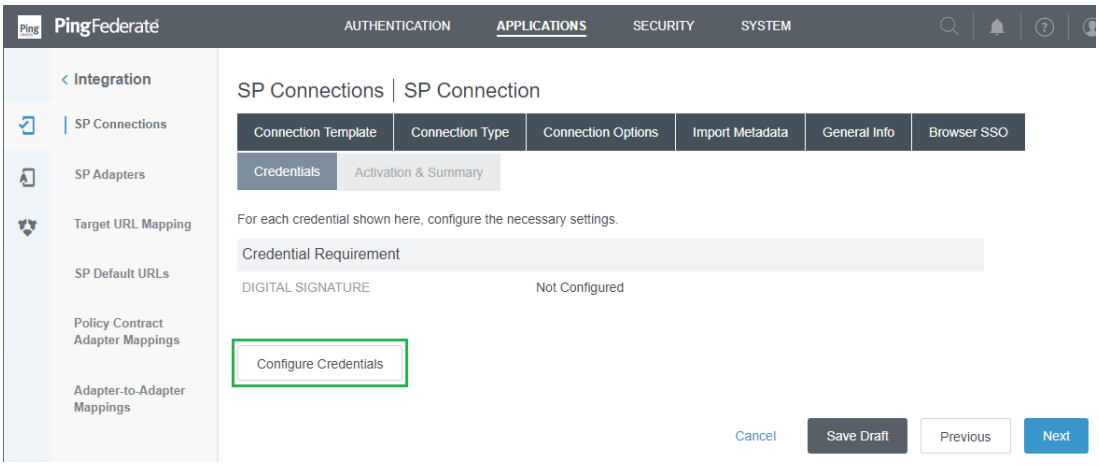
22. Check the box under `Default` column, enter **0** in `Index` column, select POST for `Binding`, and input the address of the machine hosting MetaDefender Core in `Endpoint URL` column, then click `Add` and `Next`.



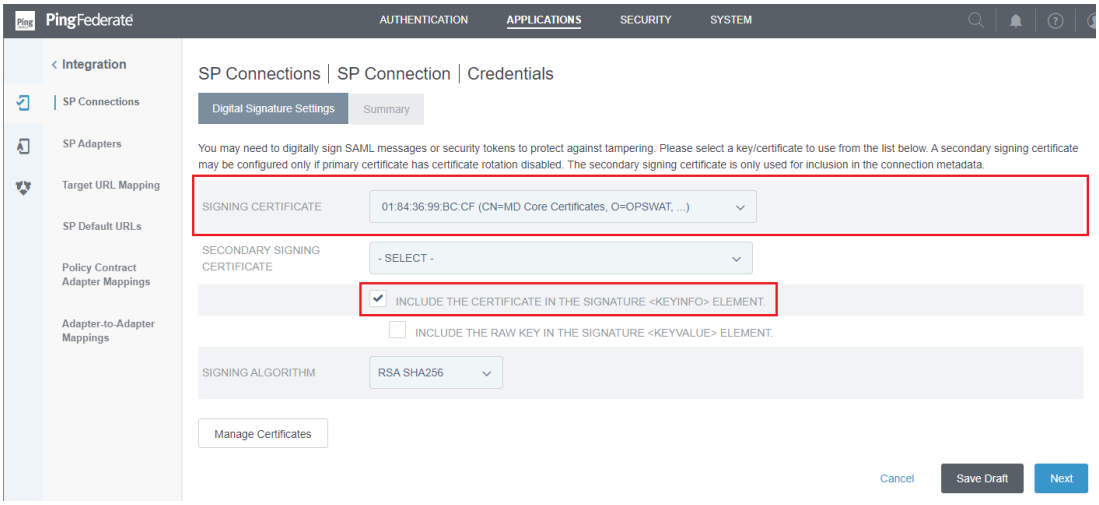
23. In `Acceptable SAML Bindings` phase, select only 'POST' and 'REDIRECT' choices, then click `Next`.



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`.

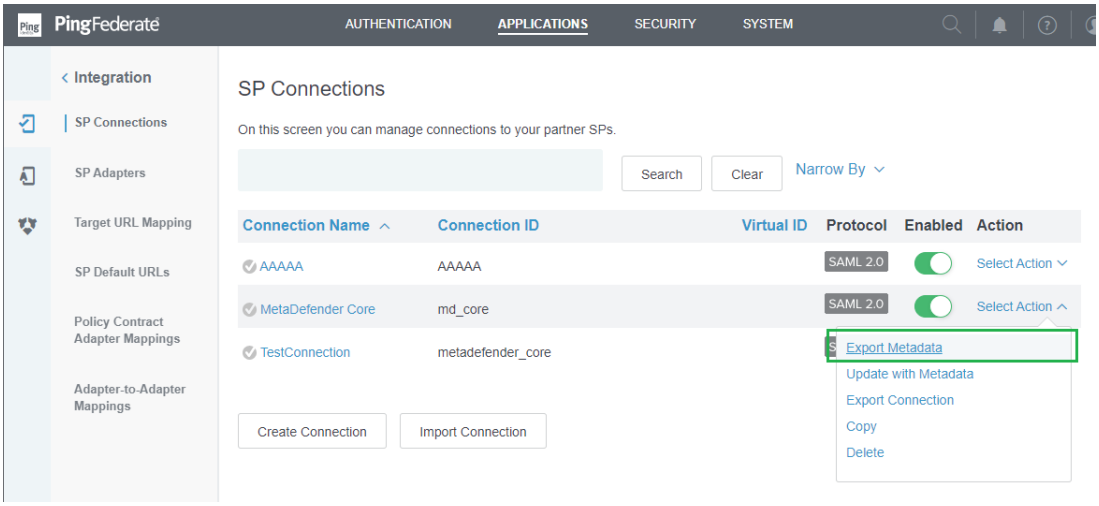


30. Examine all chosen options in `Summary` phrase, then press `Done`.
31. Return to `Credentials` tab and select `Next`.
32. Examine all chosen options in `Activation & Summary` phrase, verify that the connection is active, and then click `Save` to finish.

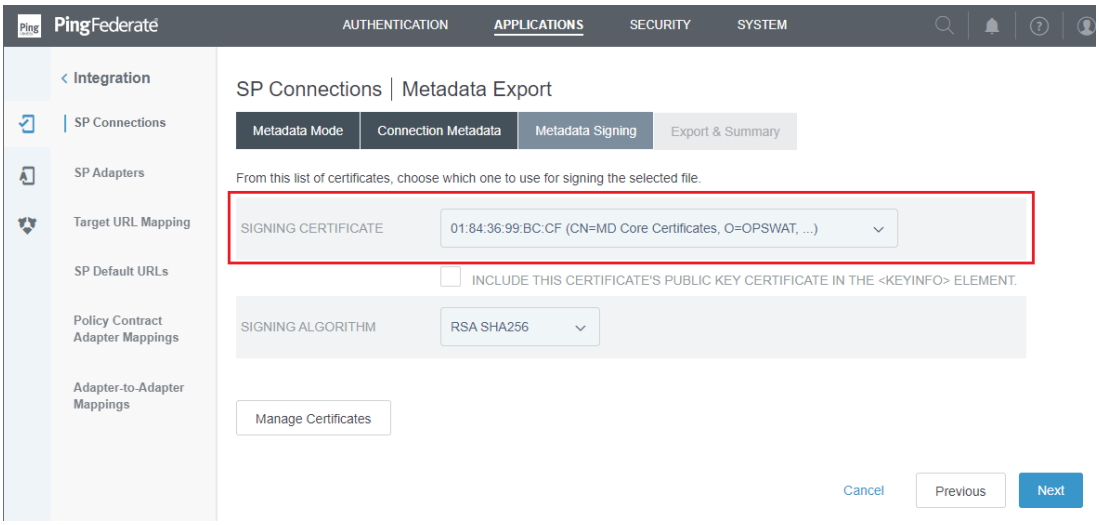


Export Identity Provider metadata

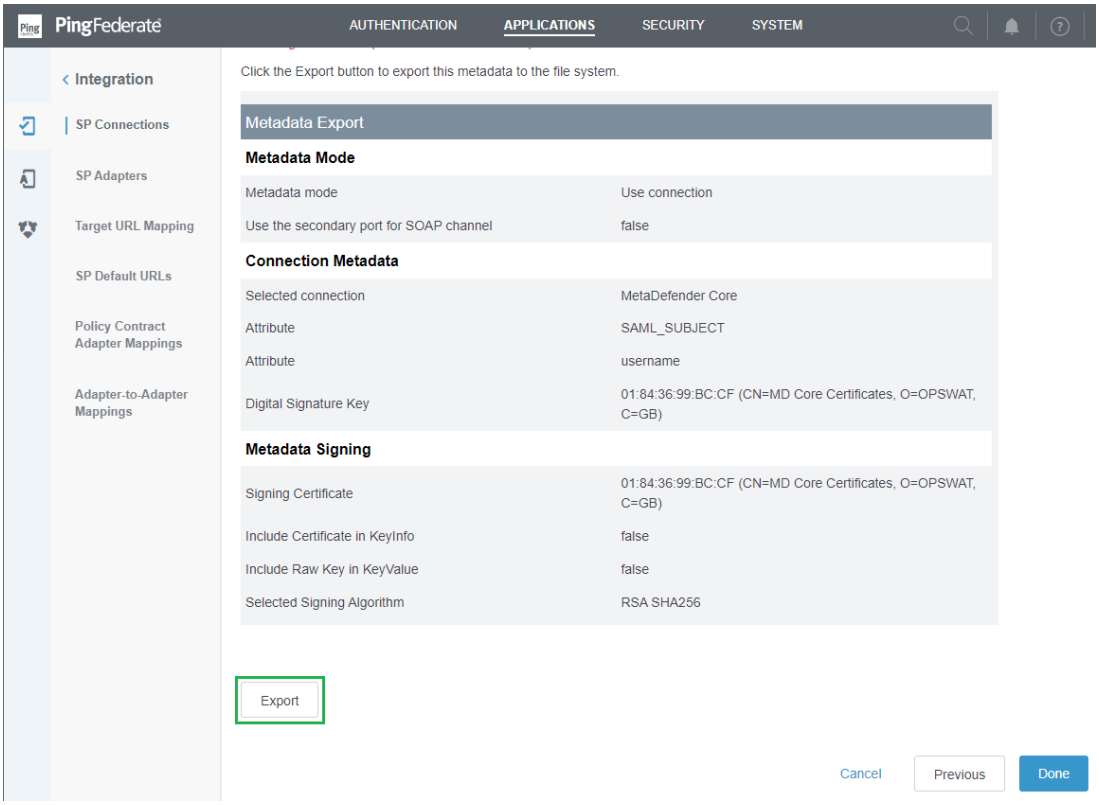
1. Go to `APPLICATIONS` and click on `SP Connections` on the left sidebar.
2. Go to the connection established in the prior step, click `Select Action` link located in `Action` column, and choose `Export Metadata`.



3. Choose the prior signing certificate, then press `Next`.



4. Examine all chosen options in `Summary` and select `Export`.

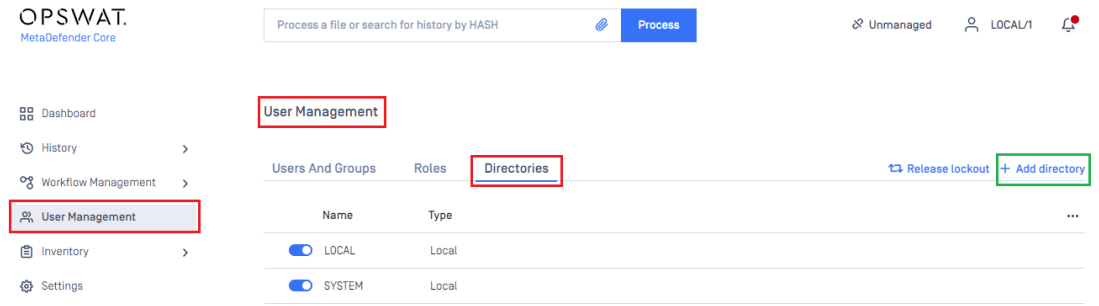


5. Save the exported file as 'metadata.xml', and then press `Done`.

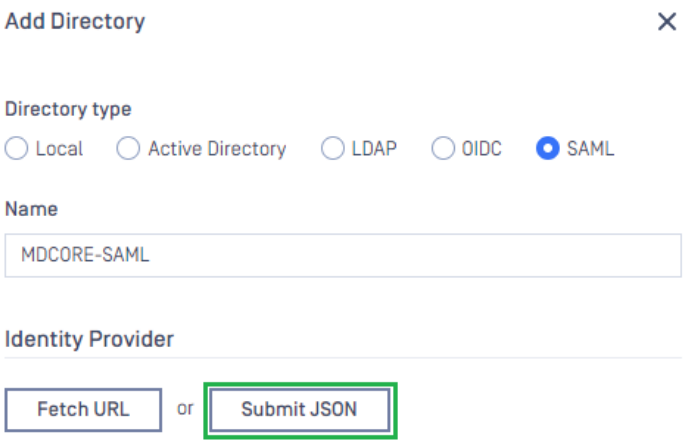
> **Info**
> The 'metadata.xml' file will be used in the upcoming phase to set up `Identity Provider` in MetaDefender Core.

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.



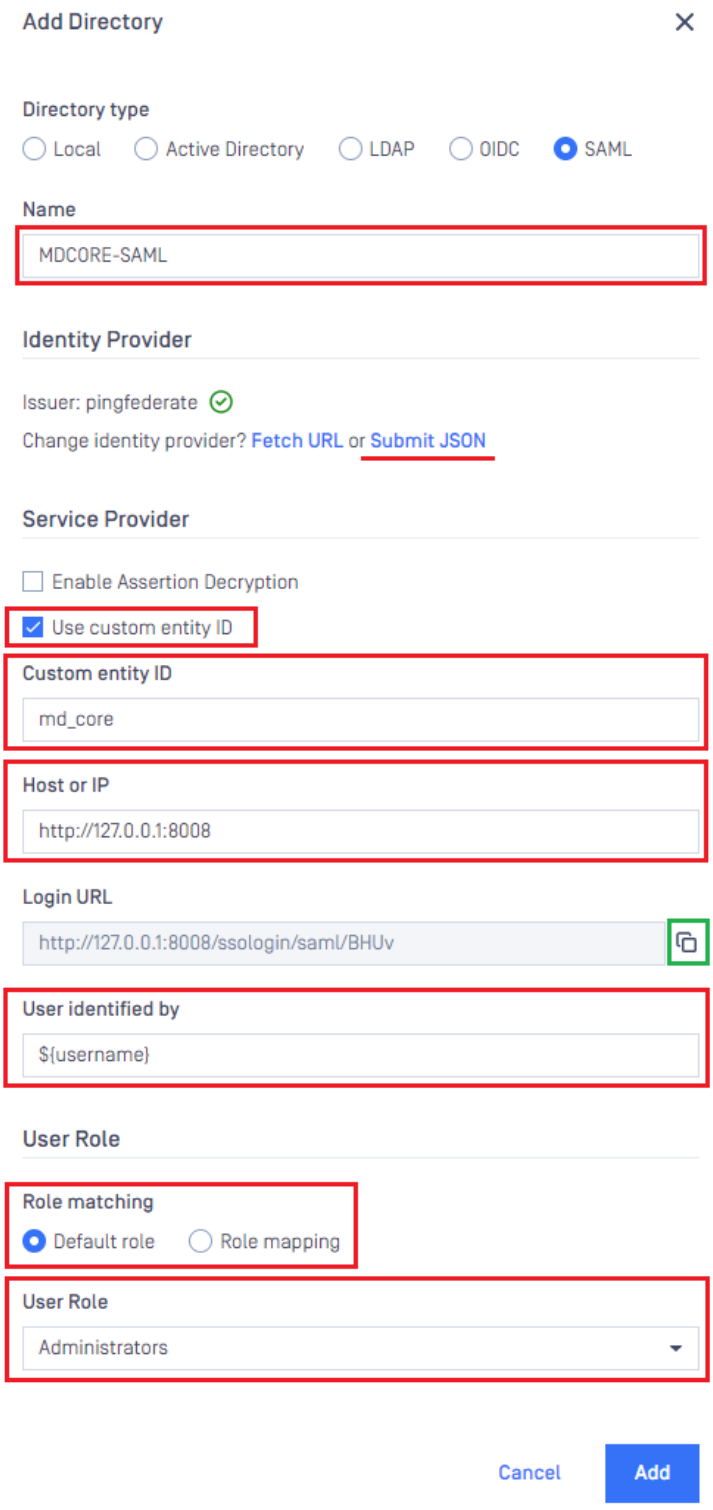
4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCORE-SAML.
5. Click `Submit JSON` and upload the 'metadata.xml' file exported in the earlier stage.



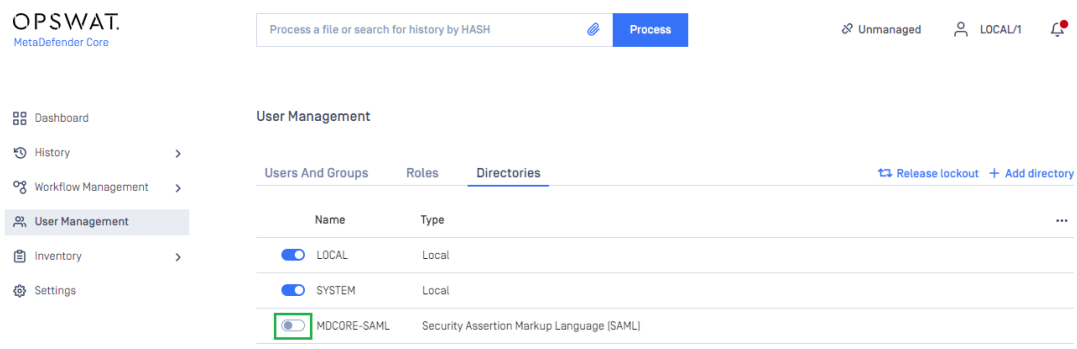
6. Navigate to `Service Provider`, toggle `Use custom entity ID` and fill in `Custom entity ID` with the partner's entity ID configured in PingFederate. That is 'md_core' for this case.
7. Enter value or `Host or IP`, click `Copy` and save it in *redirect_url*.

> **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.

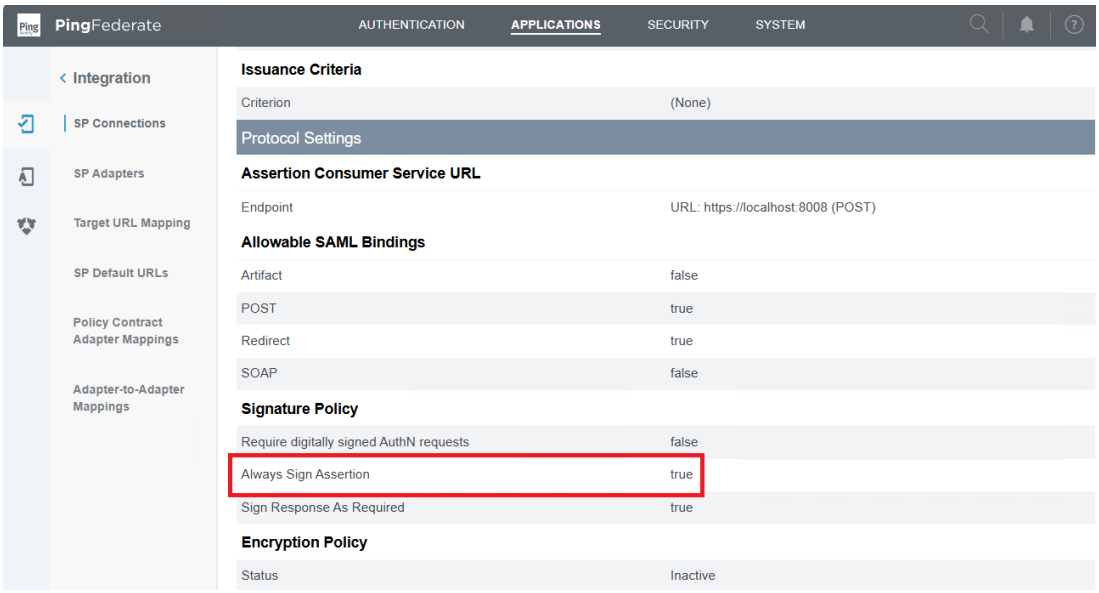


11. In `User Management` page, toggle the new directory, MDCORE-SAML, in this instance. A confirmation dialog box will pop up to verify the action. After clicking `Enable`, all sessions will end instantly.

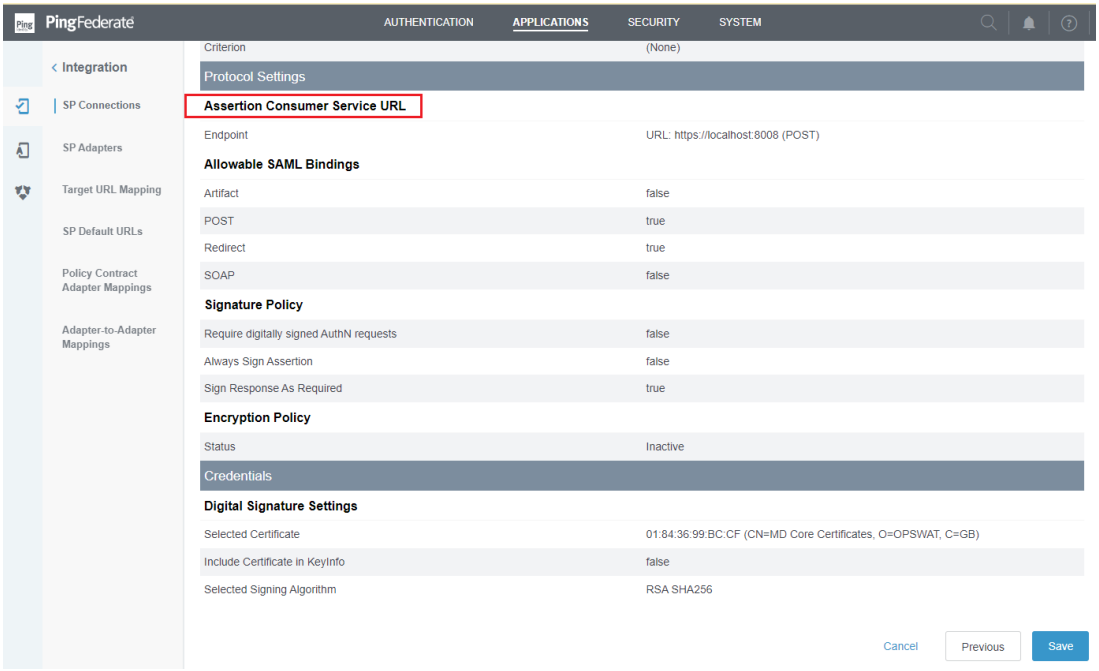


Complete configuration in PingFederate

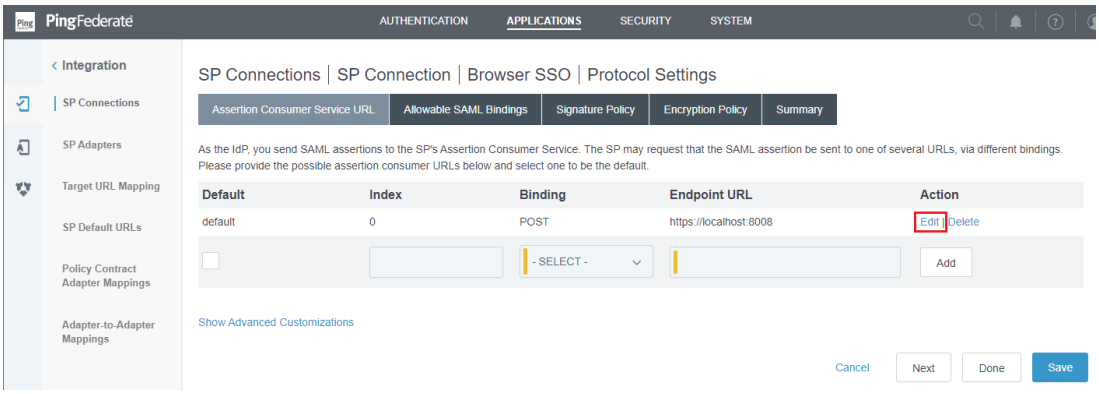
1. Access the PingFederate management console.
2. Go to `APPLICATION` and choose `SP Connections` from the left sidebar.
3. In `Signature Policy`, set `Always Sign Assertion` to 'true'.



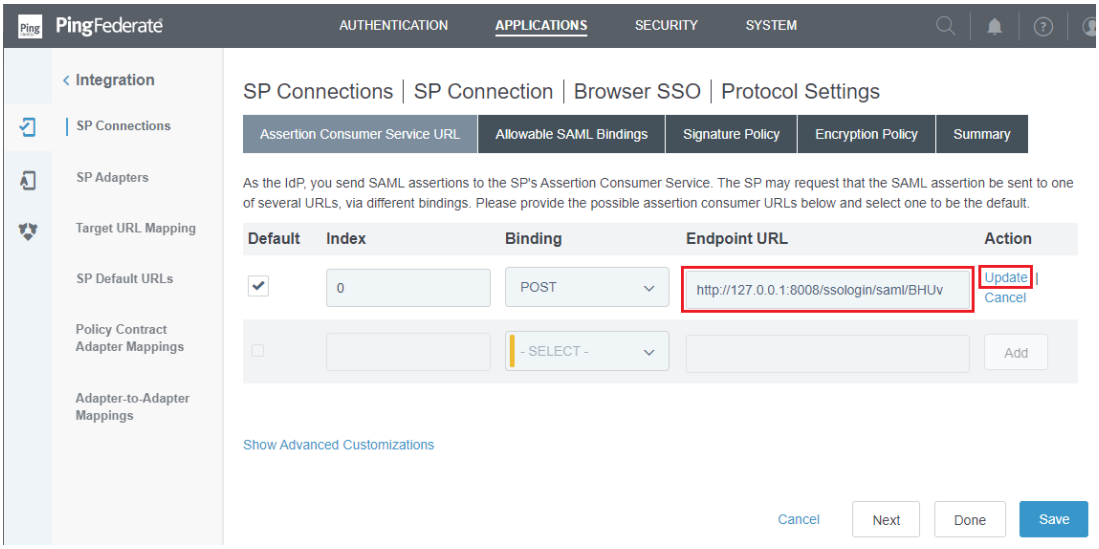
4. Go to `Assertion Consumer Service URL` and select the link next to `Endpoint`.



5. Click `Edit`.

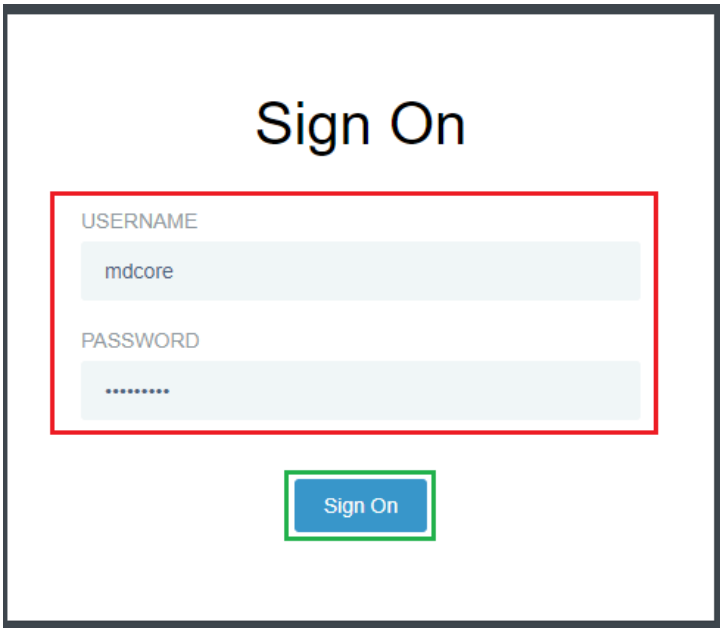


6. Modify the URL located below `Endpoint URL` to the value found in *redirect_url*, then click `Update` and `Save`.

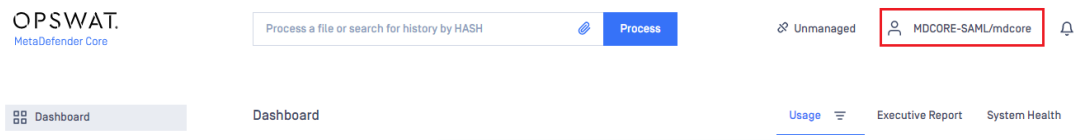


Test the integration

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.

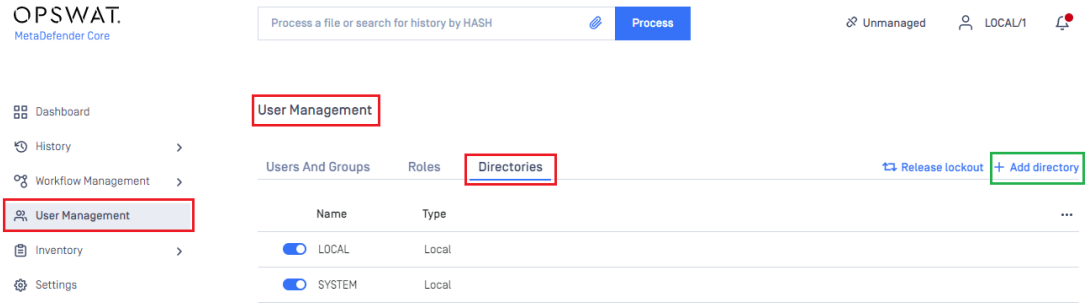
PingOne

The instructions guide users on how to configure PingOne SAML and OIDC Single Sign-on for MetaDefender Core.

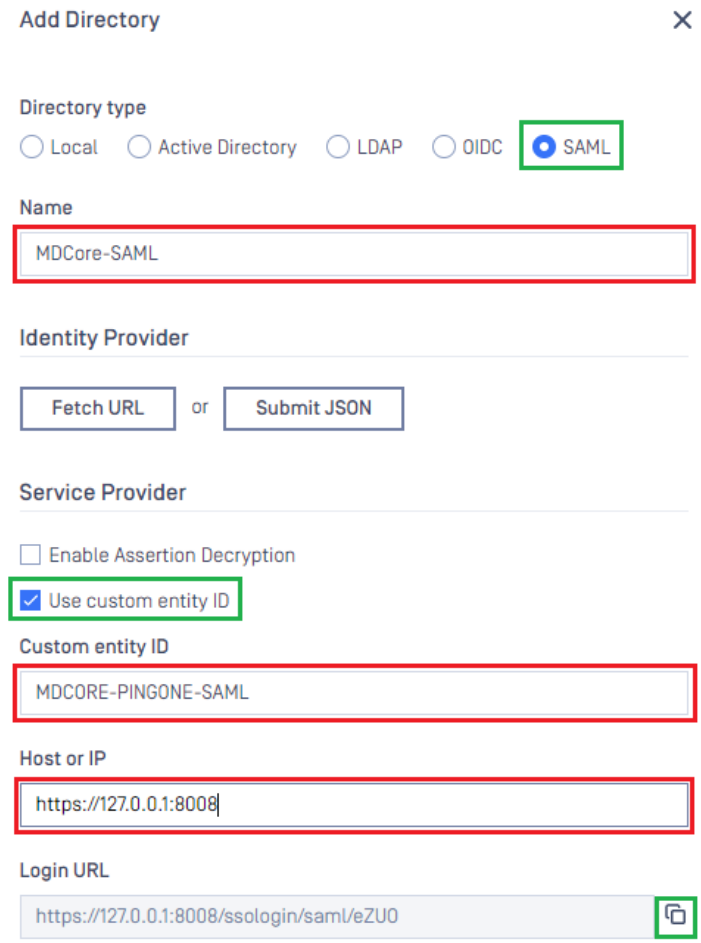
Configuring SAML Single Sign-on

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.

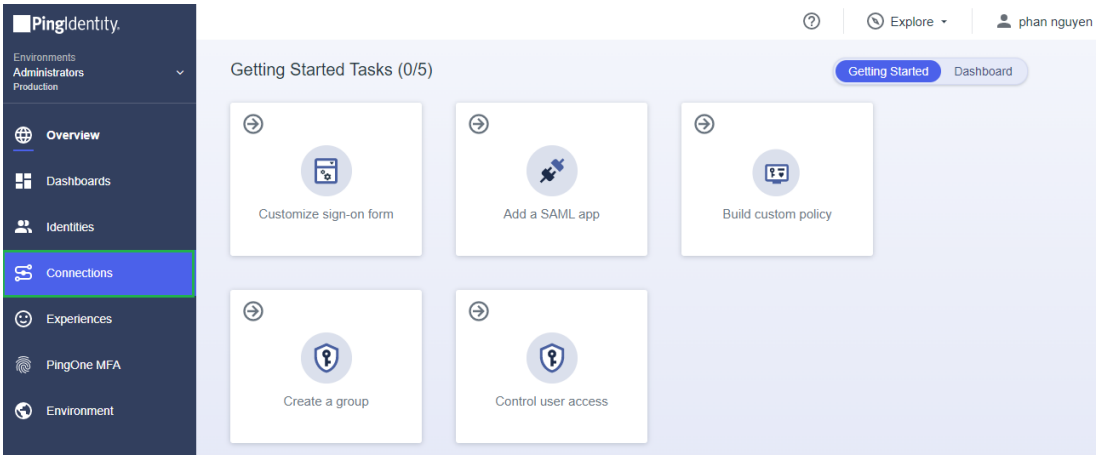


4. In `Add Directory` page, select `SAML` as `Directory type`, and enter a name for the new directory, such as MDCore-SAML.
5. In `Service Provider` section, enable `Use custom entity ID` option and enter the ID, such as MDCORE-PINGONE-SAML, then store it in *entity_id*.
6. Enter `Host or IP` where MDCore is hosted, for this example use https://127.0.0.1:8008.
7. Copy the string generated under `Login URL` and store it as *reply_uri*.

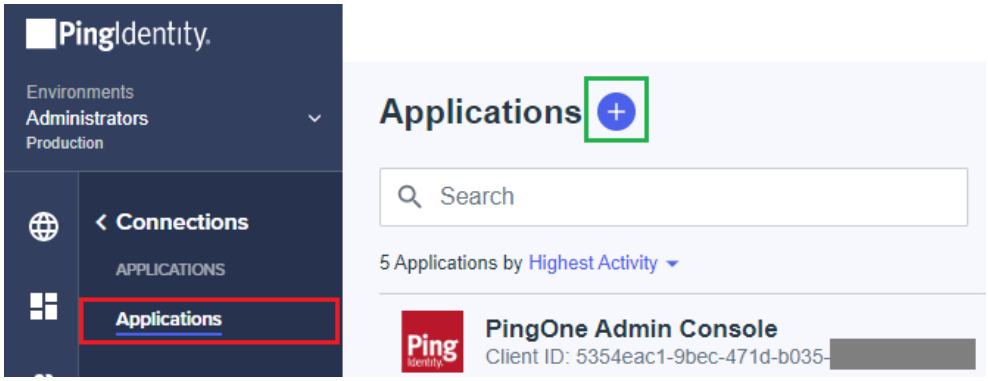


Create and configure SAML application in PingOne

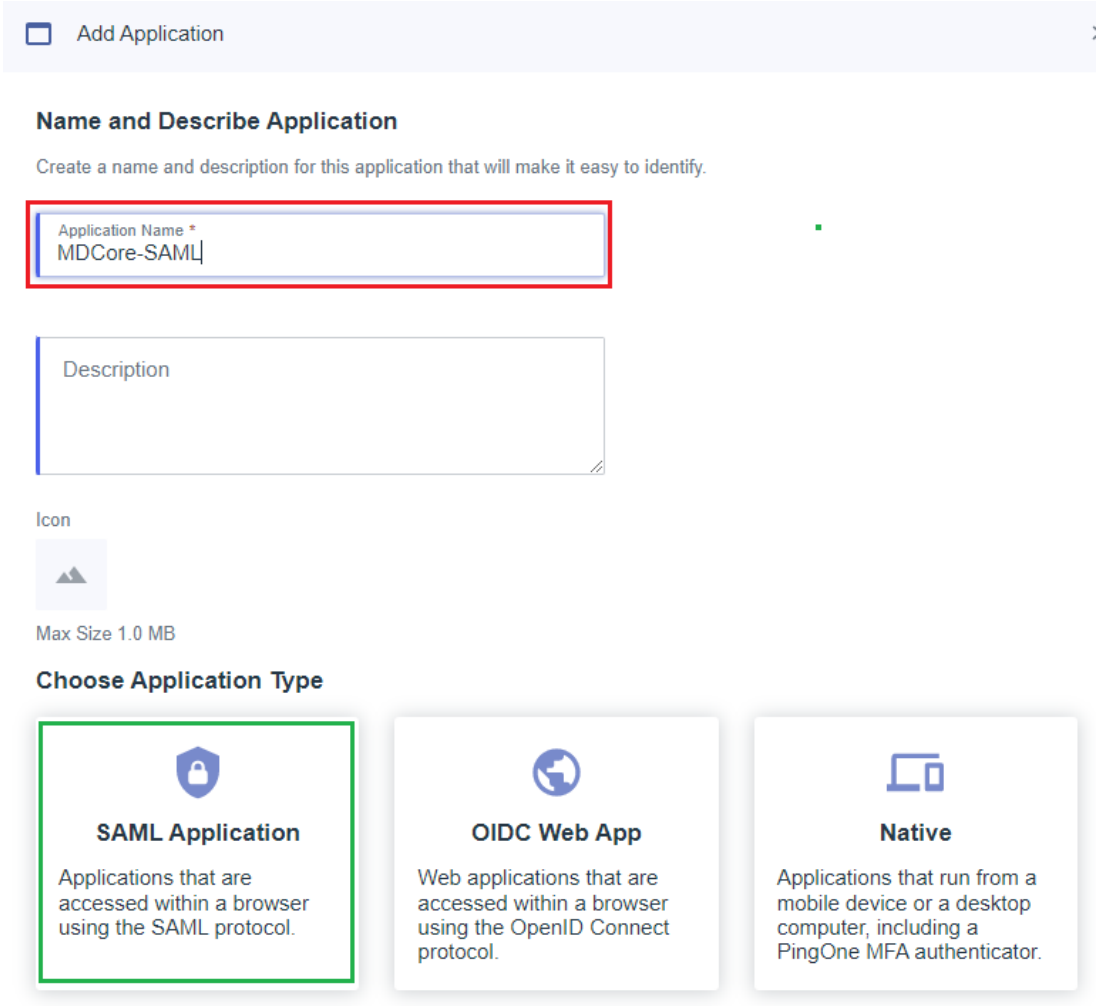
1. In the PingOne Overview page, navigate to `Connections` on the sidebar.



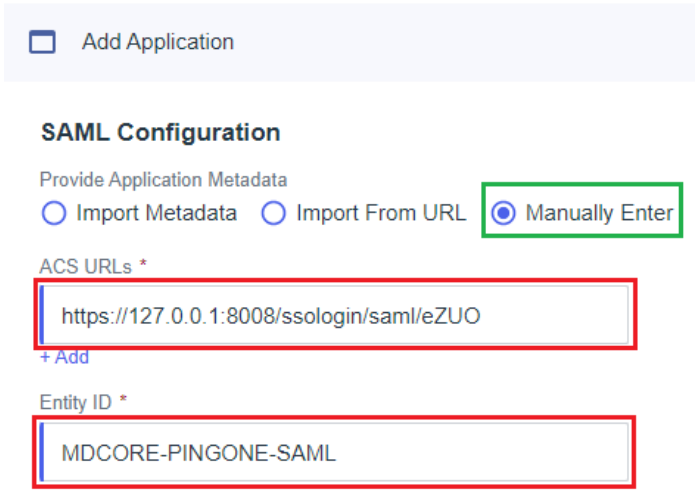
2. Click the plus button to add a new application.



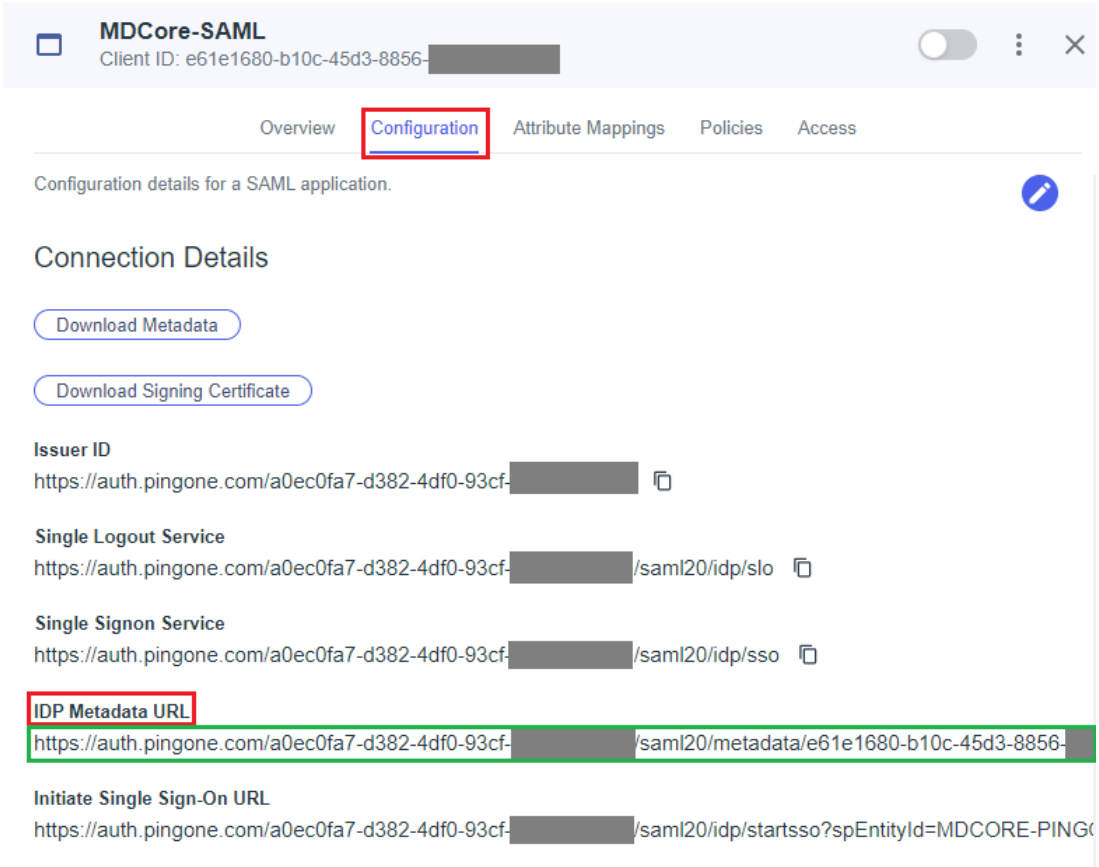
3. Enter the application name, MDCore-SAML as an example, choose `SAML Application` for `Application Type`, and click `Configure`.



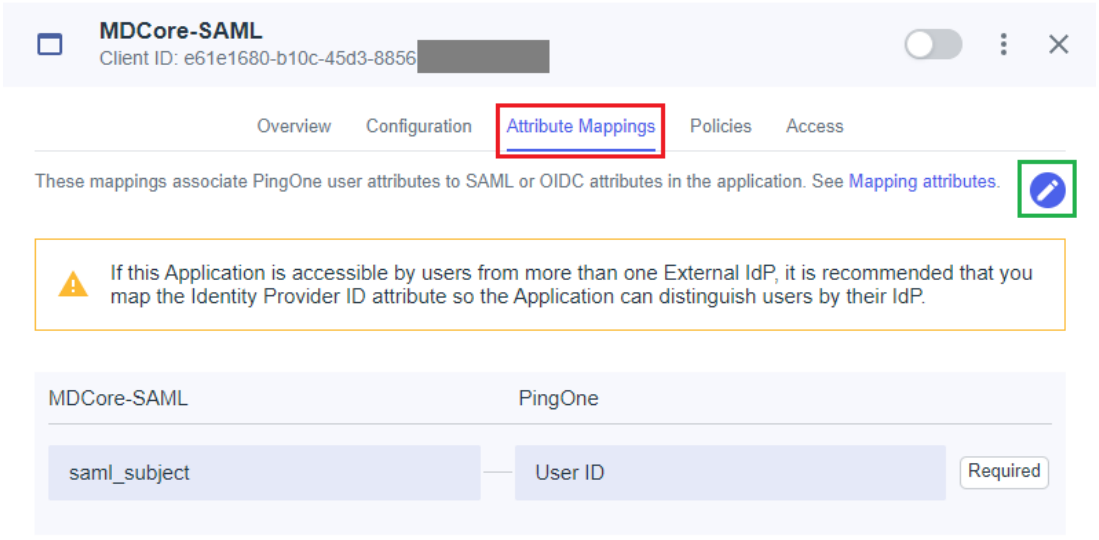
4. Select `Manually Enter` for SAML Configuration, fill in `ACS URLs` with *reply_uri*, `Entity ID` with *entity_id*, and click `Save`.



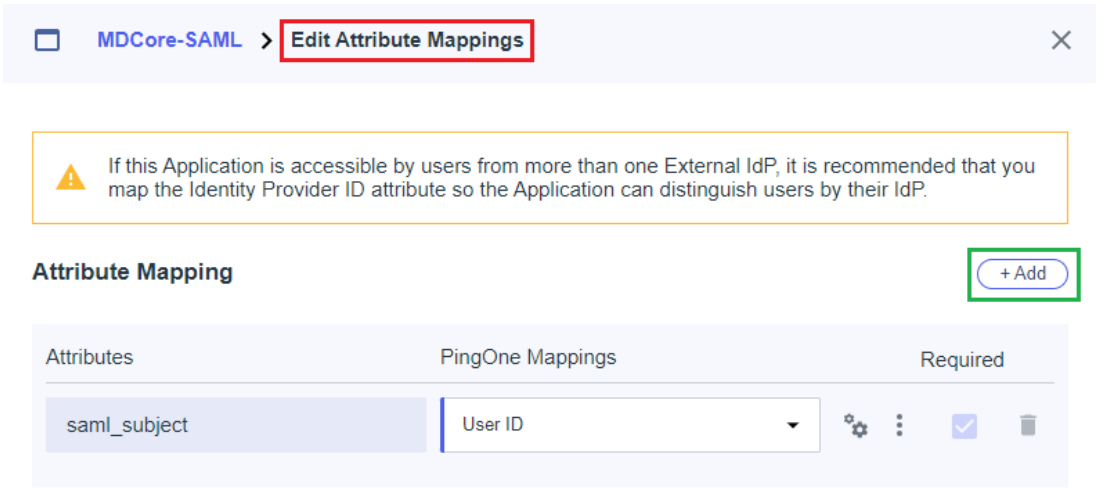
5. Select `Configuration` tab, navigate to `IDP Metadata URL`, copy the link below, and store it in *metadata_url*.



6. Select `Attribute Mappings` tab and click Edit button.

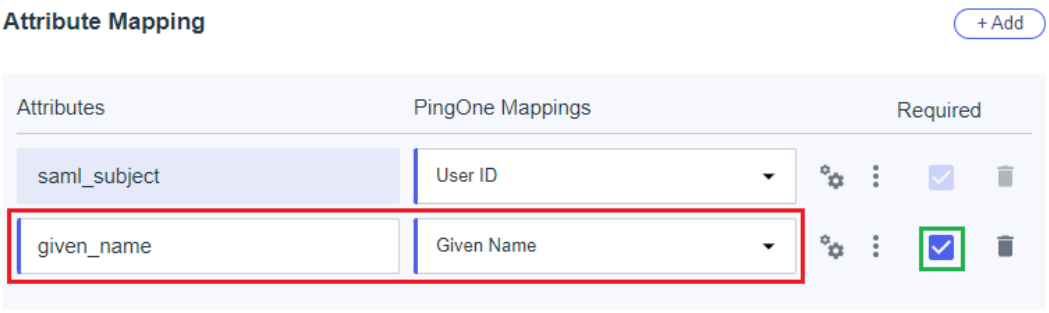


7. On `Edit Attribute Mappings` page, click `Add` to create a new mapping.

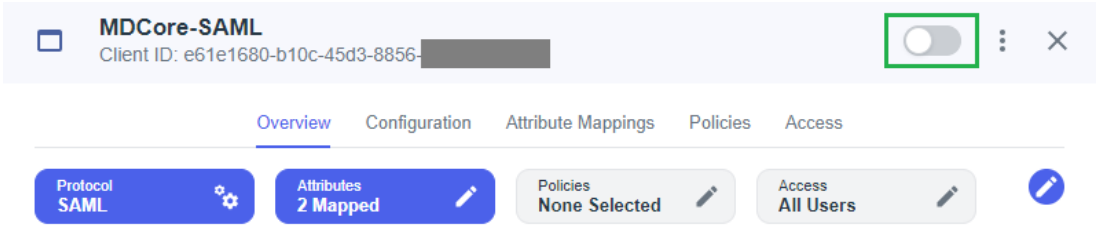


> **Info**
> In this step, we create a mapping from the key name used by MetaDefender Core to the one exported by PingOne. For example, we set `given_name` mapped to `Given Name` from PingOne. Later, `given_name` is utilized by MetaDefender Core to identify the logged-in user..

8. Enter the attribute name, `given_name` in this example, select the item `Given Name` in `PingOne Mappings`, enable `Required` option, and click `Save` to complete.

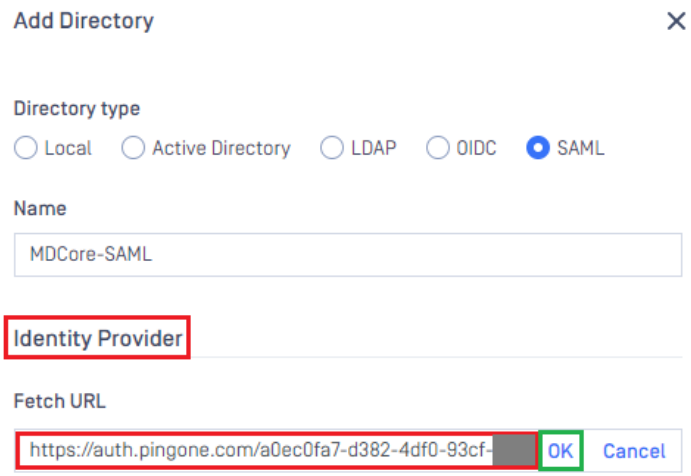


9. Enable the new application in PingOne.

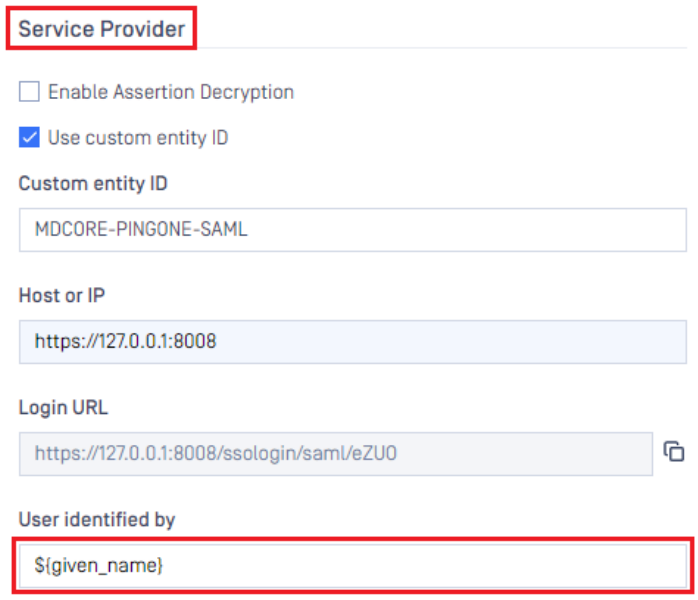


Complete configuration in MetaDefender Core

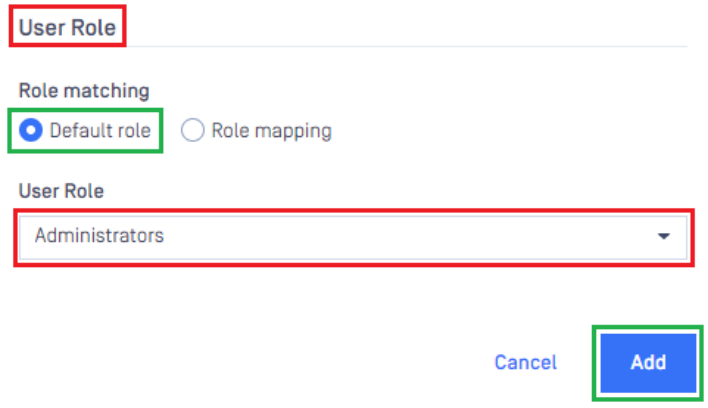
1. Switch back to MetaDefender Core. Under `Identity Provider`, click `Fetch URL`, paste the value of *metadata_url* into the box below, click `OK`, and wait a moment for MetaDefender Core to check and set PingOne as its IdP.



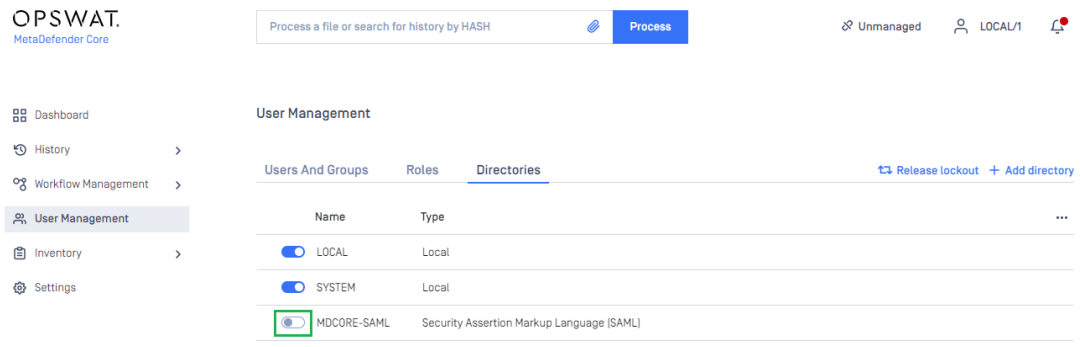
2. Under `Service Provider`, fill in ${given_name} in the box labeled `User identified by`.



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.

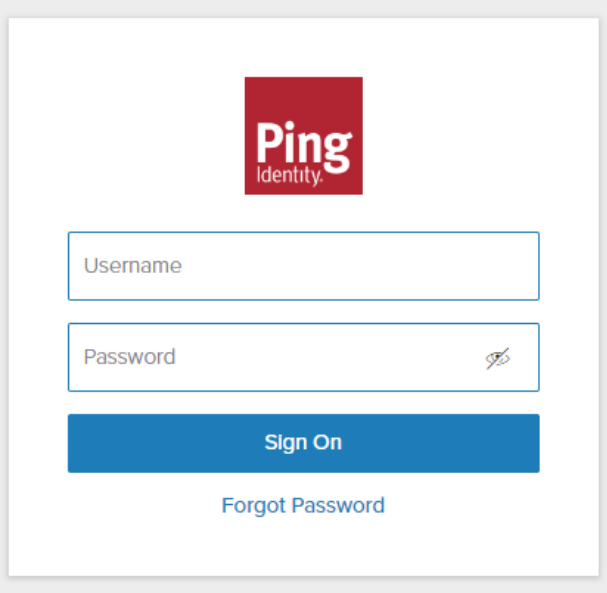


4. On `User Management` page, toggle the new directory, MDCORE-SAML in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked , all sessions will expire immediately.

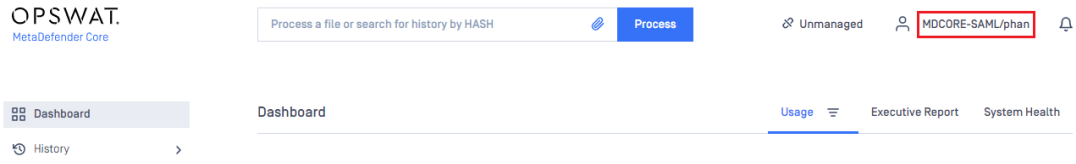


Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to PingIdentity login page.



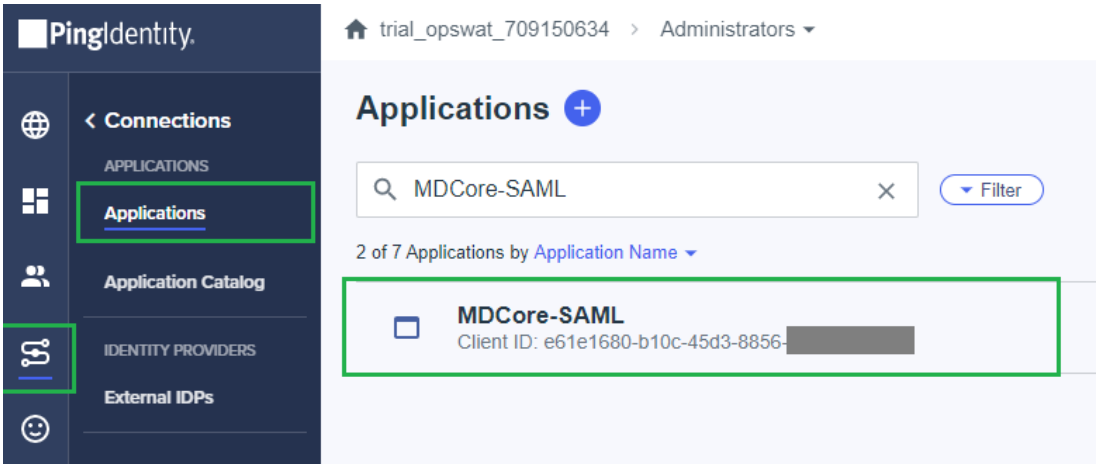
2. Sign in using the account registered with PingIdentity.
3. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.



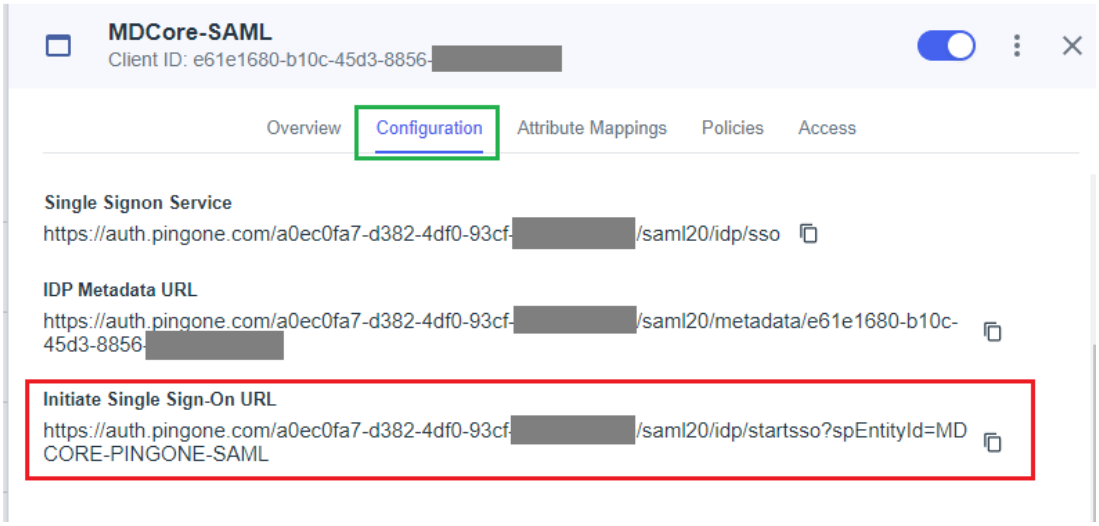
4. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

Test IdP-initiated login

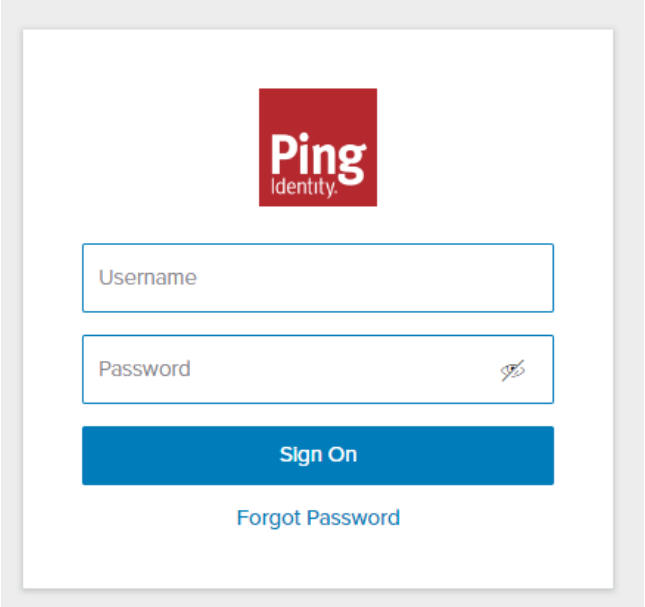
1. Sign in to PingOne management page.
2. In the left sidebar, click on `Connections`, then `Applications`.
3. Select the SSO directory from which IdP-initiated login is enabled, such as MDCore-SAML in this example.



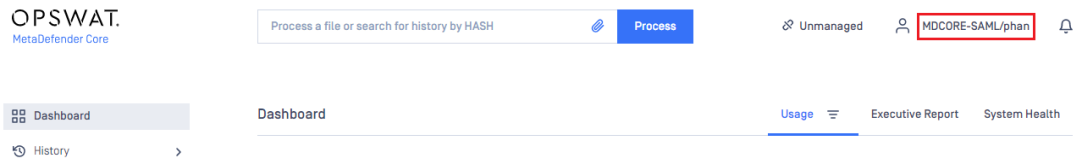
4. In the right sidebar, click on `Configuration` tab. Copy the URL under `Initiate Single Sign-On URL`.



5. Paste the URL into your browser and try to sign in.



6. If everything goes well, MetaDefender Core dashboard is displayed with the user identity in the top right corner.

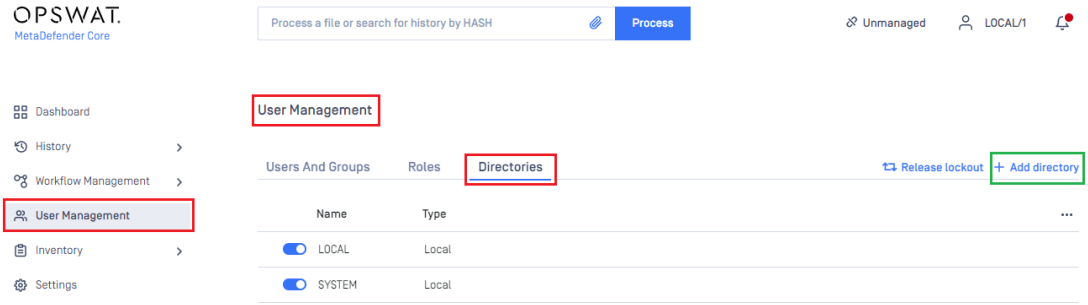


7. Otherwise, access backup login page at `<mdcore-host>#/public/backuplogin` for trouble shooting.

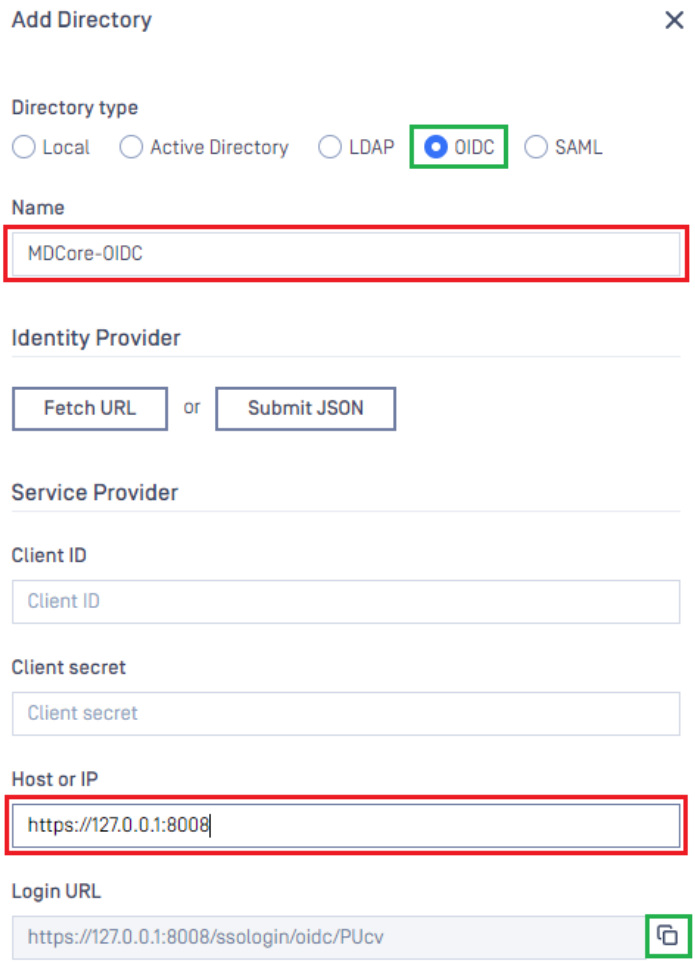
Configuring OIDC Single Sign-on

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.

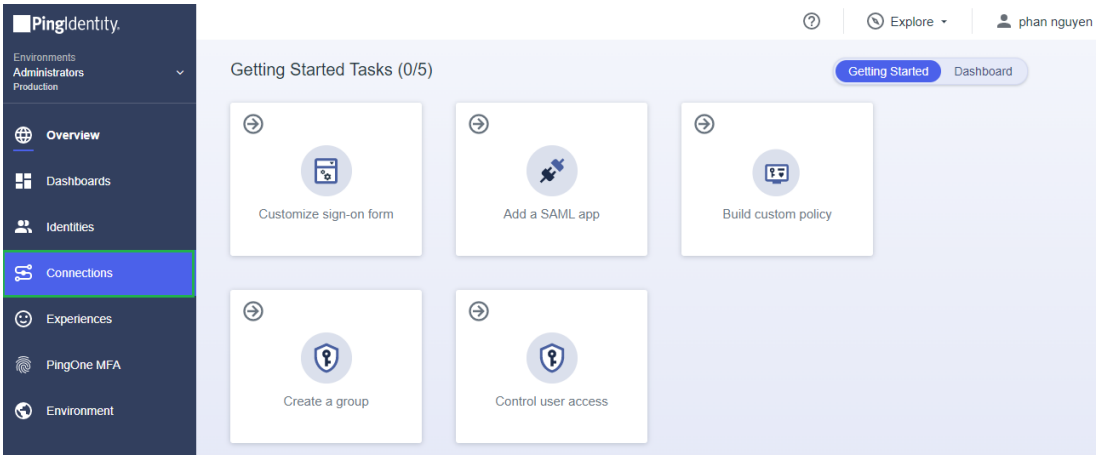


4. In `Add Directory` page, select OIDC as `Directory type`, and enter a name for the new directory, such as MDCore-OIDC.
5. Enter `Host or IP` where MetaDefender Core is hosted, for this example use https://127.0.0.1:8008.
6. Copy the string generated under `Login URL` and store it as *reply_uri*.

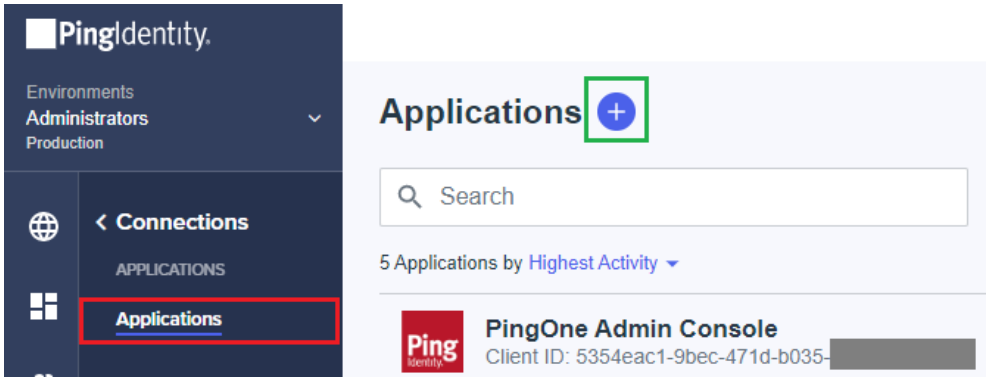


Create and configure OIDC application in PingOne

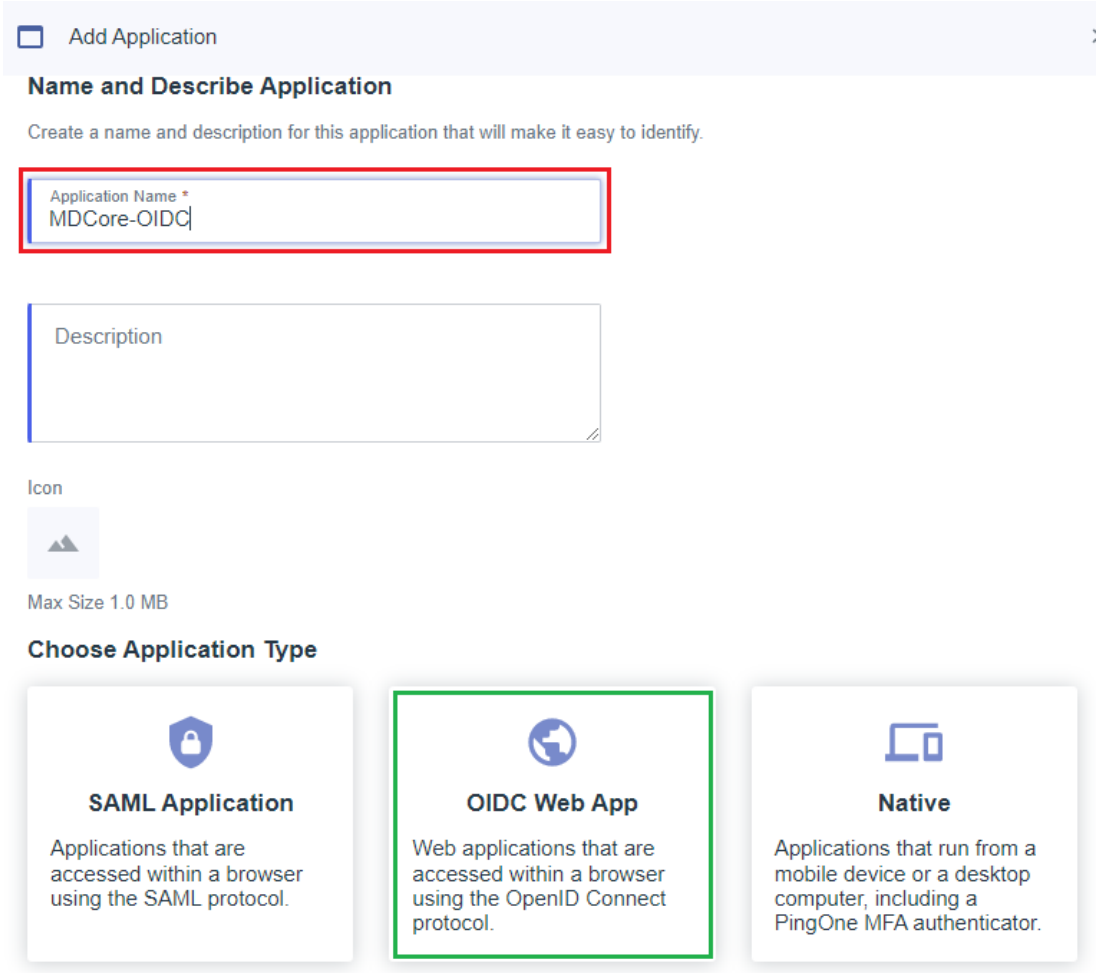
1. In the PingOne Overview page, navigate to `Connections` on the sidebar.



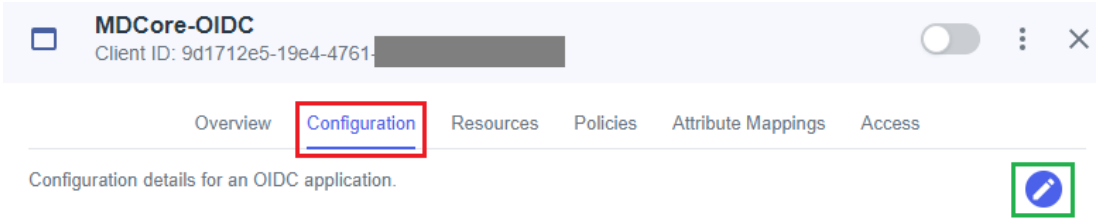
2. Click the plus button to add a new application.



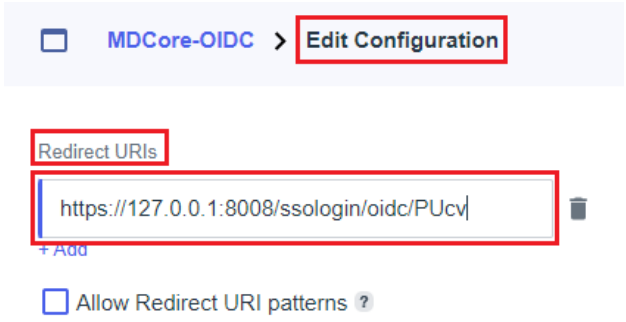
3. Enter the application name, MDCore-OIDC as an example, choose `OIDC Web App` for `Application Type`, and click `Save`.



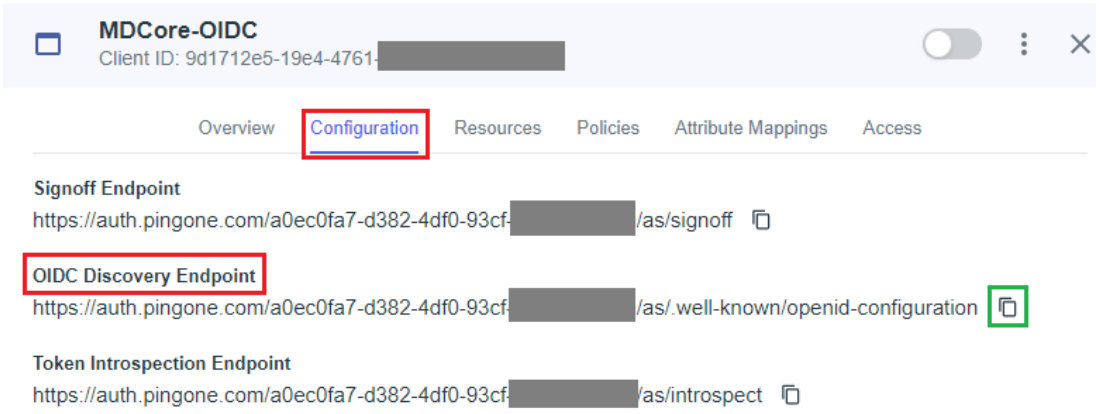
4. Select `Configuration` tab, then click Edit button.



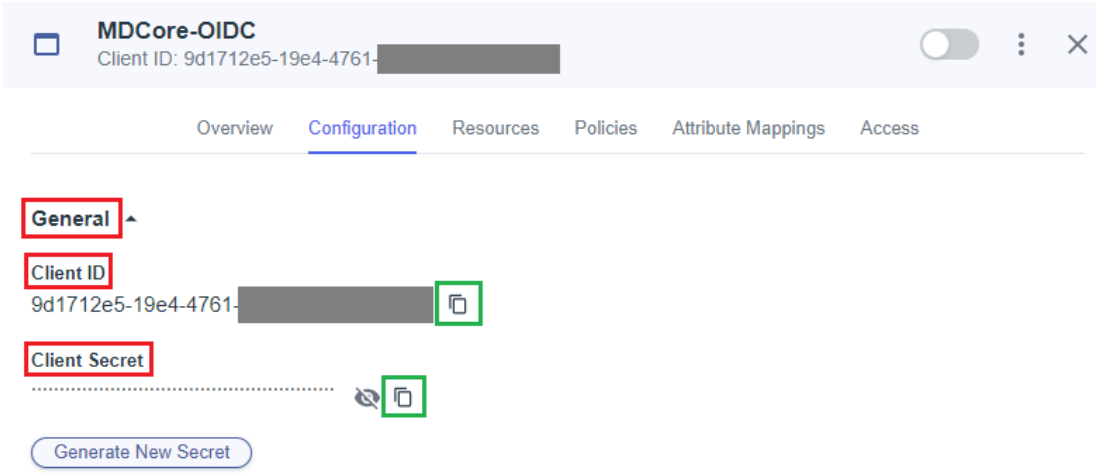
5. In `Edit Configuration` page, navigate to `Redirect URIs`, paste *redirect_uri* into the box below, and then click `Save`.



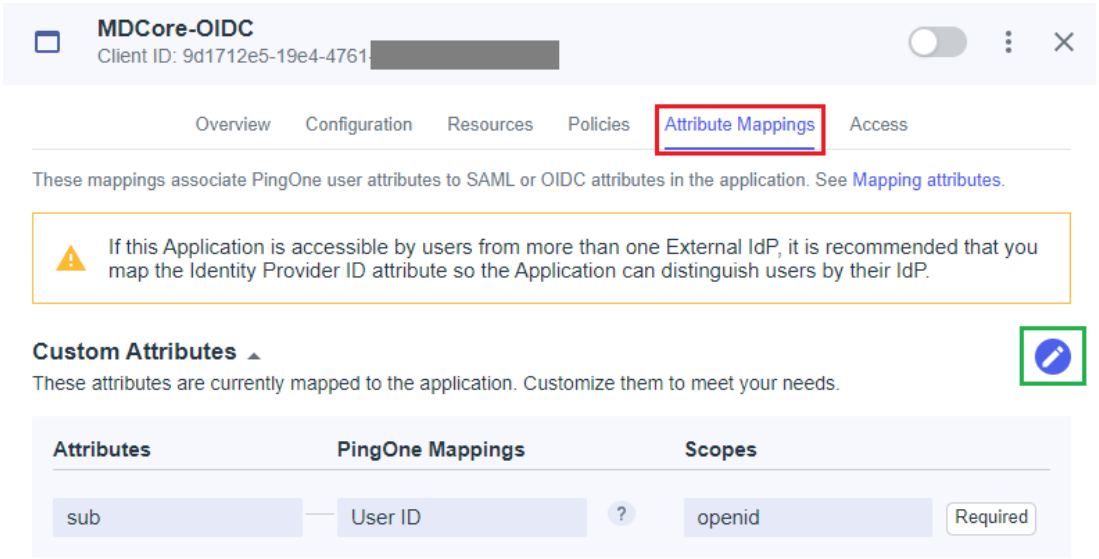
6. Back in `Configuration` tab, navigate to `OIDC Discovery Endpoint`, copy the link below, and save it as *metadata_url*.



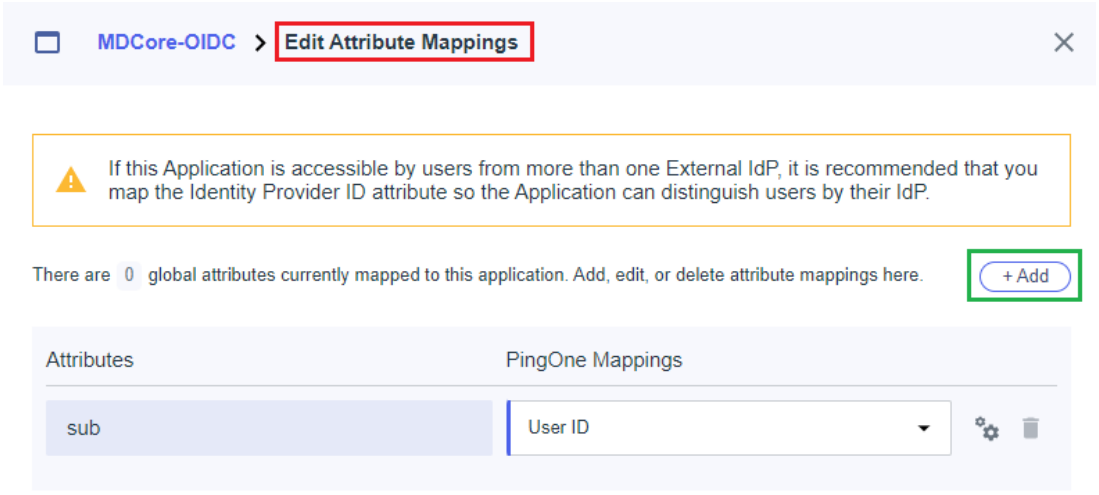
7. Expand `General` section, navigate to `Client ID` and `Client Secret`, and copy and save them as *client_id* and *client_secret*, respectively.



8. Select `Attribute Mappings` tab and click Edit button.



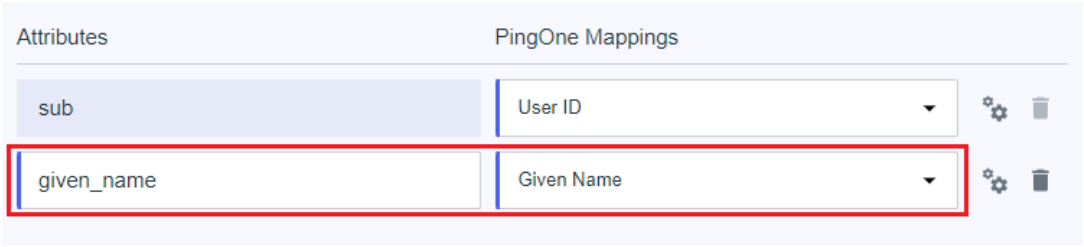
9. In `Edit Attribute Mappings` page, click `Add` to create a new mapping.



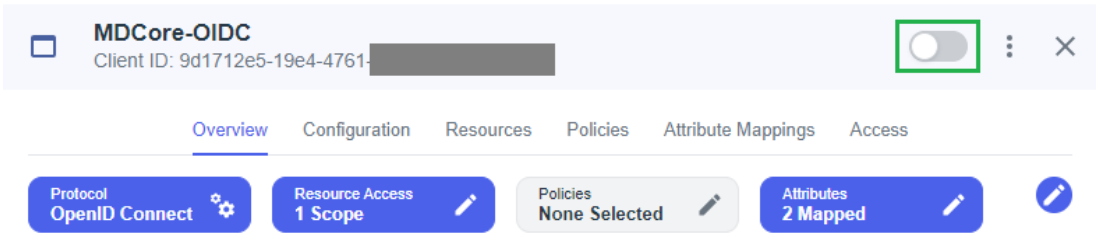
Info

In this step, we create a mapping from the key name used by MetaDefender Core to the one exported by PingOne. For example, we set `given_name` mapped to `Given Name` from PingOne. Later, `given_name` is utilized by MetaDefender Core to identify the logged-in user.

10. Enter the attribute name, `given_name` in this example, select the item `Given Name` in `PingOne Mappings`, enable `Required` option , and click `Save` to complete.



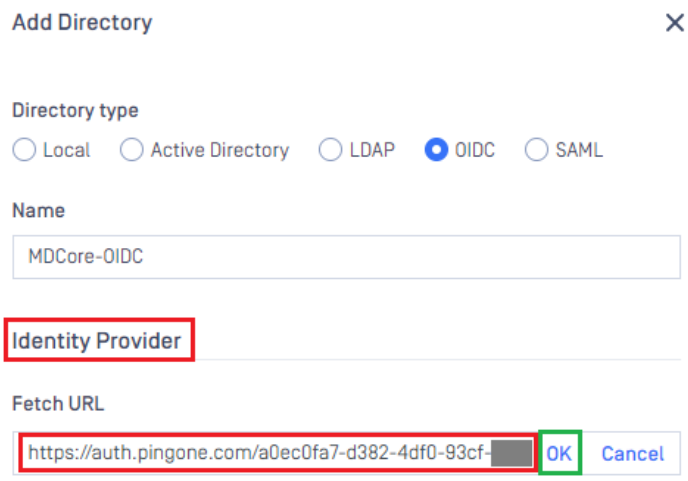
11. Enable the new application in PingOne.



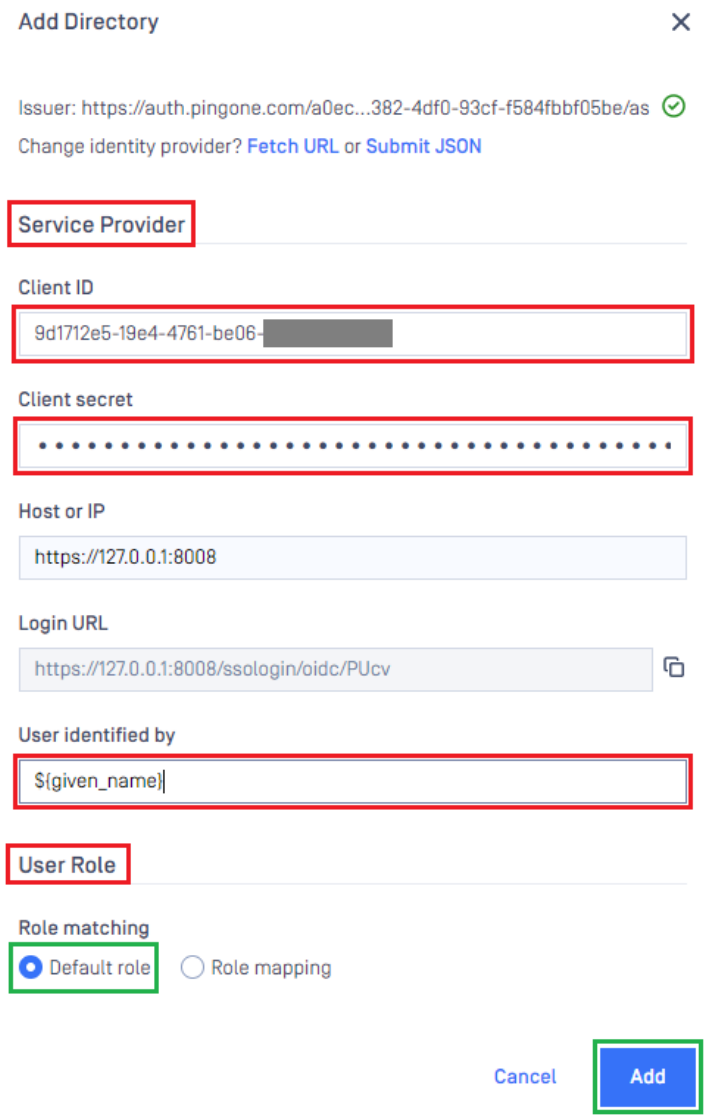
Complete configuration in MetaDefender Core

1. Switch back to MetaDefender Core. Under `Identity Provider`, click `Fetch URL`, paste value of *metadata_url* into the box below, click `OK`, and wait a moment for MetaDefender Core to check and set

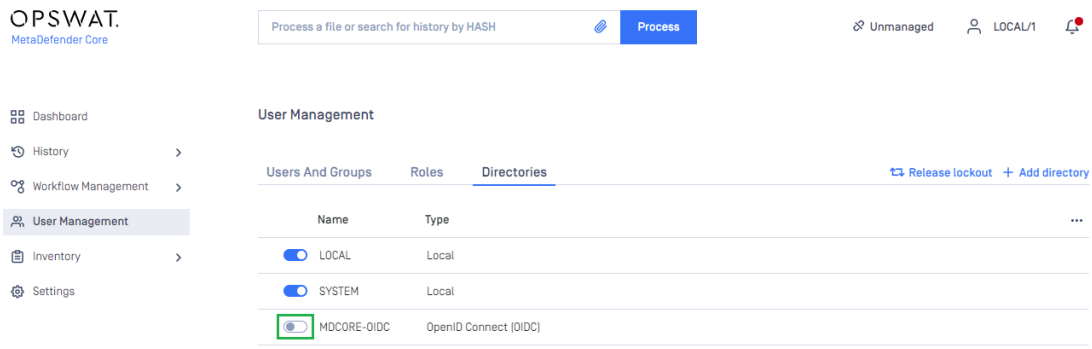
PingOne as its IdP.



2. Under `Service Provider`, paste the values of *client_id* and *client_secret* into the boxes labeled `Client ID` and `Client Secret`, respectively.
3. Fill in ${given_name} in the box under `User Identified By`.
4. 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.

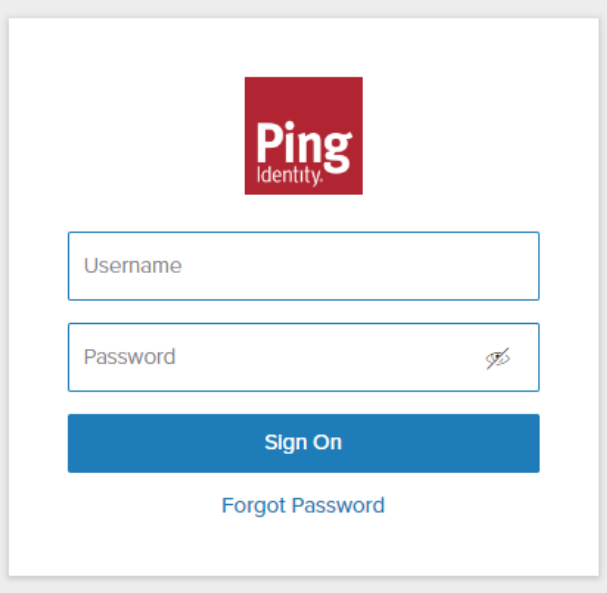


5. On `User Management` page, toggle the new directory, MDCORE-OIDC in this example. A dialog box will appear to confirm the action. Once `Enable` is clicked, all sessions will expire immediately.

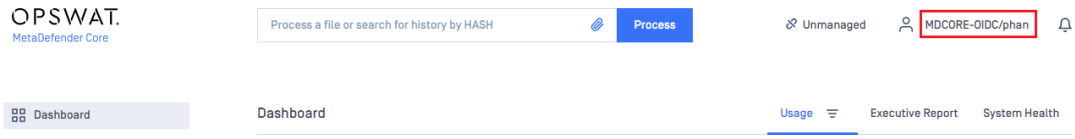


Test the integration

1. On the home screen of MetaDefender Core, click `Login`; the user is redirected to PingIdentity login page.



2. Sign in using the account registered with PingIdentity.
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.

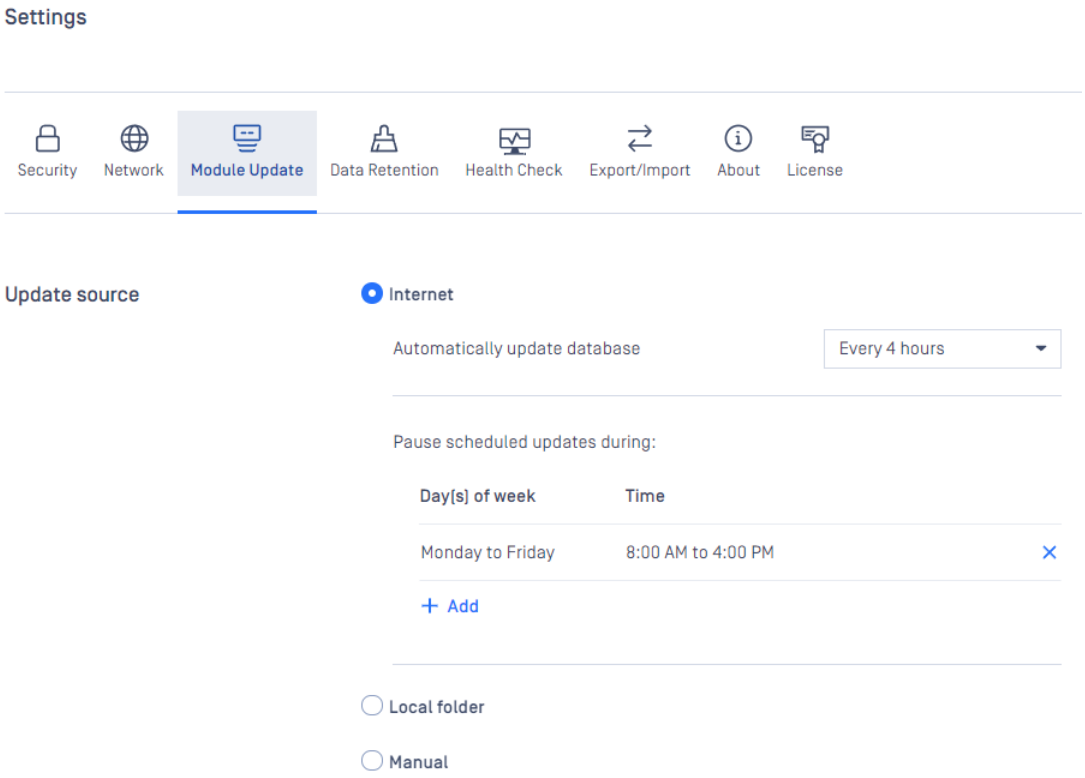
Engines Update Configuration

Update settings are accessible under **Settings > Module 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



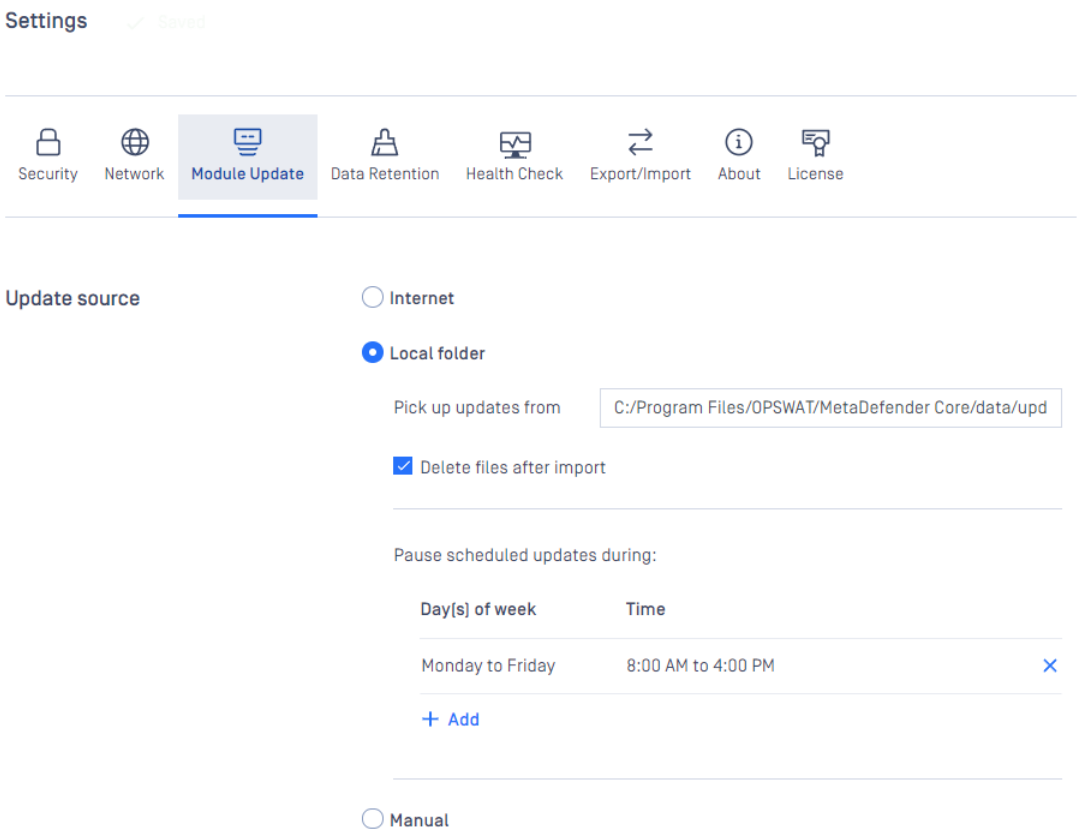
Choosing the Internet method means the product will typically perform automatic update downloads from the internet. In cases where the Core is managed by My OPSWAT / Central Management, the product will perform automatic update downloads from My OPSWAT / Central Management.

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 **Inventory > Modules**.

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.

> **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 **Inventory > Technologies.**

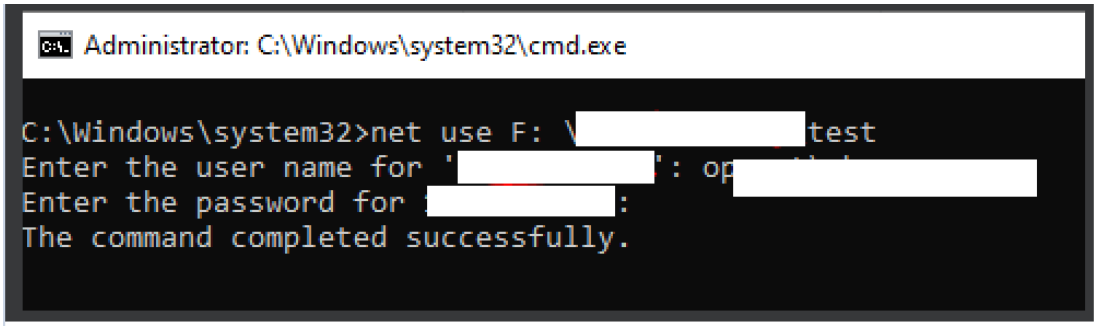
Advanced configuration for UNC path

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

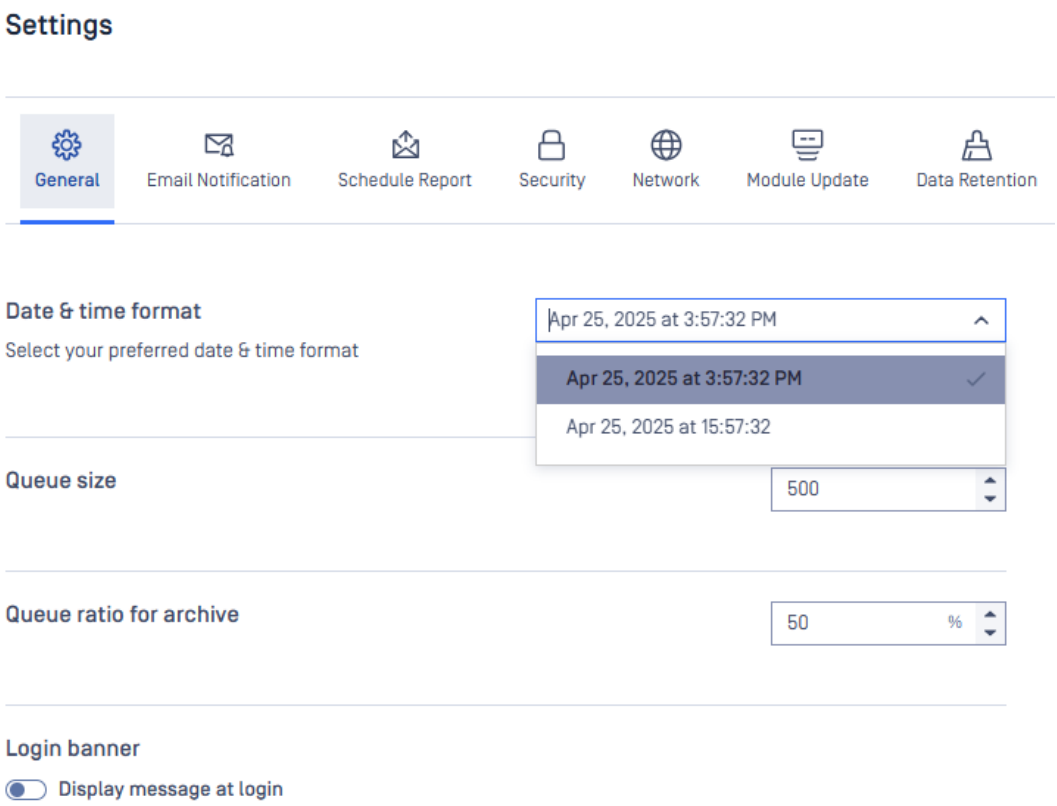
Info

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

General configuration

Date & time format, Queue size, Queue ratio for archive and Login banner are accessible under **Settings > General** after successful login.



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.

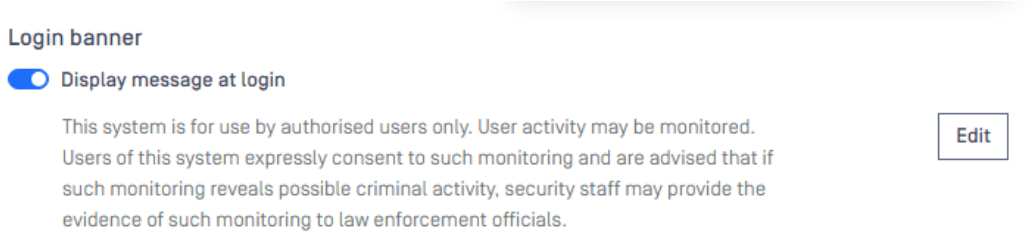
Queue ratio for archive

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.

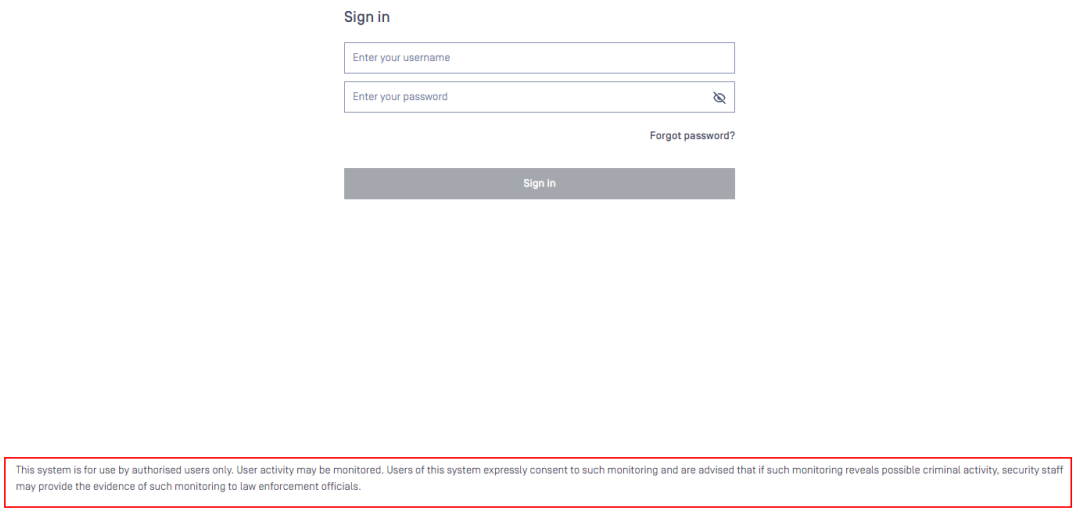
> **Warning**
> The higher the value of queue ratio set for archive, the fewer slots are available for processing new client requests, which may cause Metadefender Core to easily respond with HTTP code 503 and message `Server is too busy. Try again later.`

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.



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): (~!@#$%^&*_-+=`|(){}[]:;'"<>,.?/)

Password policies

Enforce password history

Number of unique new passwords associated with an account before an old password can be reused.

Passwords remembered (0-24) 12

Password must meet complexity requirements

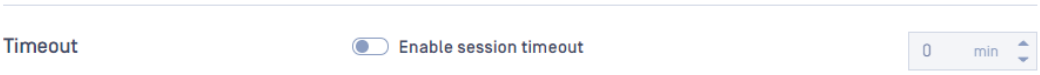
At least 4 characters in length
At least 1 uppercase letter of European languages (A through Z)
At least 1 lowercase letter of European languages (a through z)
At least 1 base 10 digits (0 through 9)
At least 1 non-alphanumeric characters: (~!@#$%^&*_-+=`|(){}[]:;"'<>,.?/\)

Enforce min password length

Min password length (0-30) 4

Session Timeout

User sessions can be terminated regardless of user activity. Under the **Settings** menu, on the **Security** page, a timeout value can be given in minutes, in order to limit the length of a user session. After the given time elapsed, the user session will be terminated even if it is during an operation.



REST API for session timeout
Learn more at: PUT - Session settings API

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:

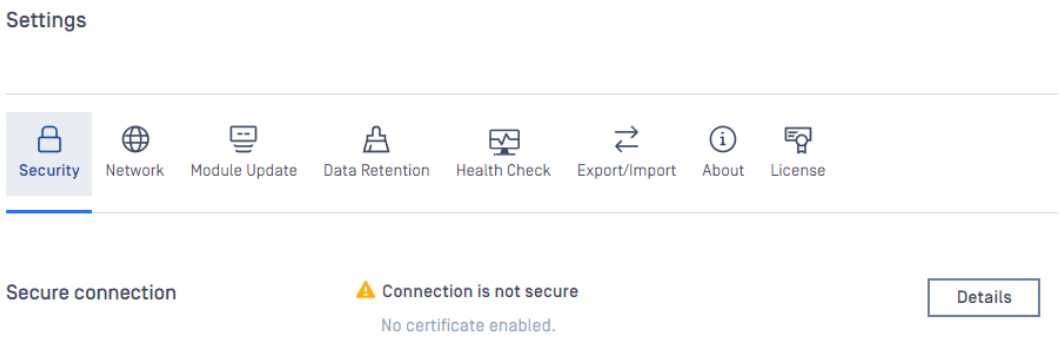
- via Management Console or
- modifying MetaDefender Core server configuration via configuration files.

Info
TLS v1.2 and v1.3 are supported HTTPS connection.

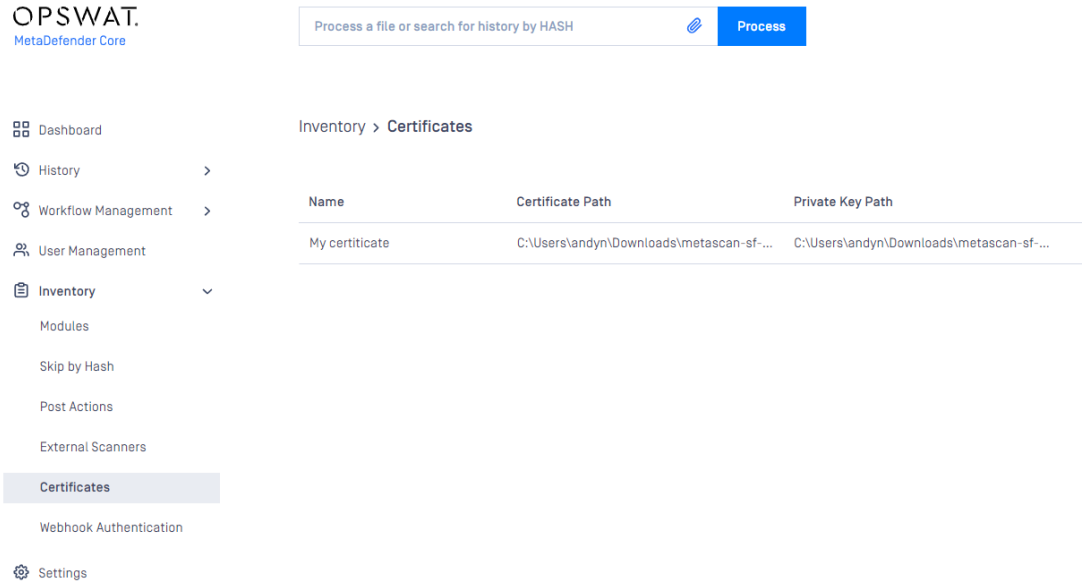
Warning
If HTTPS is configured via both ways, only the settings made on Management Console will take effect. It is highly recommended not to use both configuration files and user interface for HTTPS settings at the same time.

Enabling HTTPS via Management Console

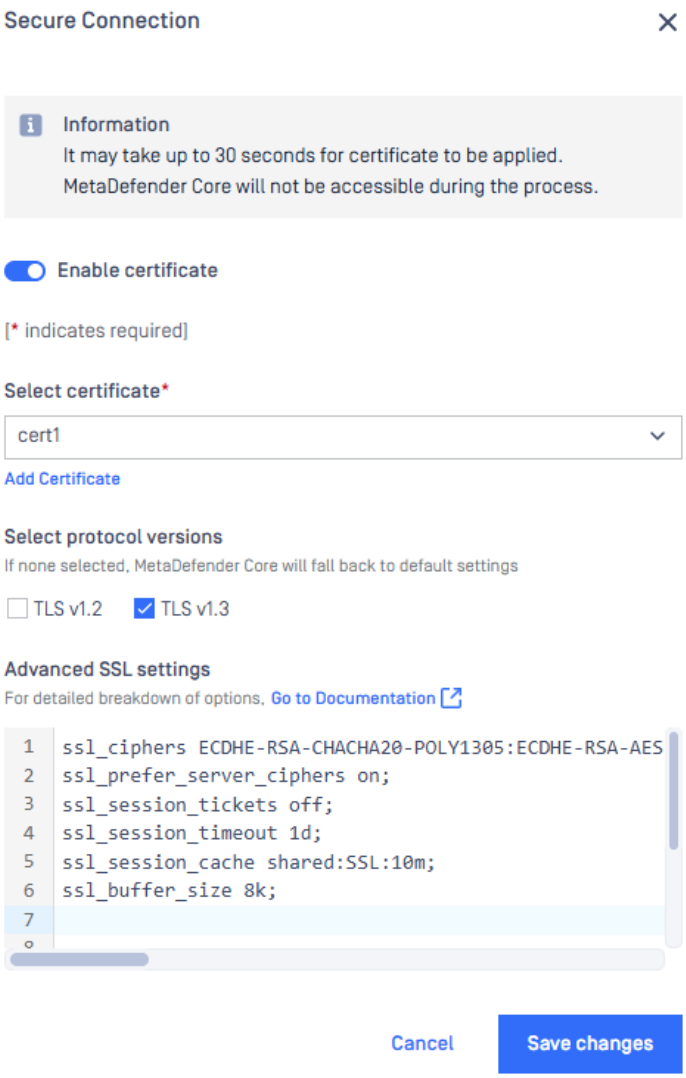
1. Go to **Settings→Security** page



2. If there's no certificate-key pair added to the inventory, please go to **Inventory→**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.

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.

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.

First create your certificate and key files in convenient directory. Let us take paths as an example `/etc/ometascan/nginx.d/your.crt` and `/etc/ometascan/nginx.d/your.key` for Linux and `C:/Program Files/OPSWAT/Metadefender Core/nginx/your.crt` and `C:/Program Files/OPSWAT/Metadefender Core/nginx/your.key` for Windows accordingly.

On Linux

1. 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.
2. Create file ssl.conf.mdcore (MetaDefender Core only accept this name) in the directory /etc/ometascan/nginx.d
3. Enter SSL-configuration according to Nginx. To allow simple SSL one needs to add the following lines only:

ssl.conf.mdcore bash

```
ssl on;
ssl_certificate /etc/ometascan/nginx.d/your.crt;
ssl_certificate_key /etc/ometascan/nginx.d/your.key;
```

3. Service restart is required to take these changes into effect.

On Windows

1. Create file ssl.conf.mdcore (MetaDefender Core only accept this name) in the directory <Installation Directory>\nginx.
2. Enter SSL-configuration according to Nginx. To allow simple SSL one needs to add the following lines only (note the forward "/" slashes)

ssl.conf.mdcore bash

```
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";
```

3. A restart of the “OPSWAT Metadefender Core” service is required.

Info

Advanced web server configurations (applicable to both Linux and Windows platform)

1.) Explicitly allow specific TLS versions, optionally with preferred ciphers. For example:

```
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_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/

```
ssl on;
ssl_certificate "C:/Program Files/OPSWAT/Metadefender Core/nginx/cert.pem";
ssl_certificate_key "/etc/keys/secretkey.pass";
ssl_certificate_key "C:/Program Files/OPSWAT/Metadefender
Core/nginx/your_encrypted.key";
ssl_password_file "/etc/keys/private.pass";
ssl_protocols tlsv1.2;
ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256;
```

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.

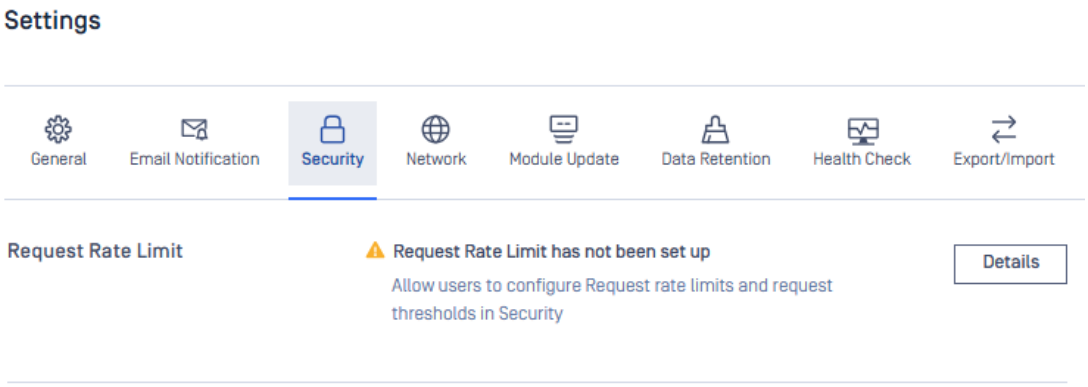
Note: When choosing location for cert and key files, make sure the files are in a location which is readable to the service user.

Request Rate Limit

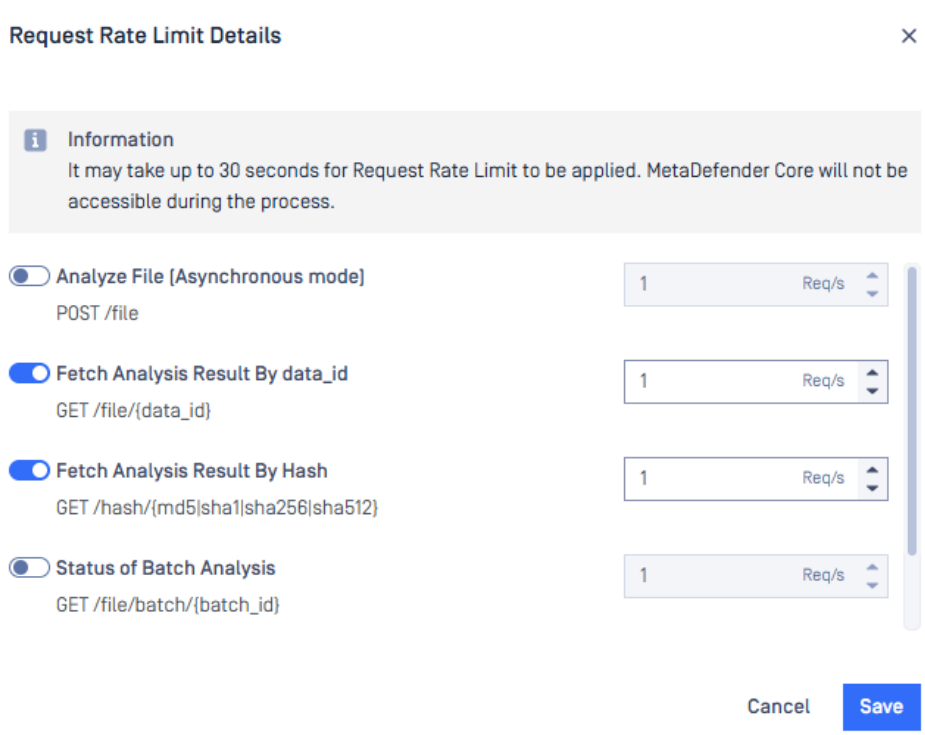
Since version 5.13.0, MetaDefender Core supports rate limiting for more API calls. This feature is not enabled by default.

To enable Request Rate Limit,

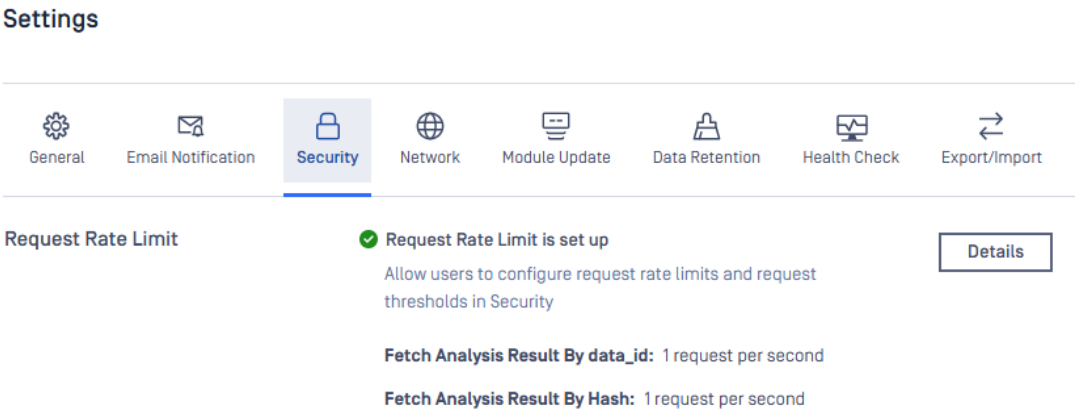
1. Go to **Settings** -> **Security** page



2. Hit **Details** to setup



3. Hit **Save** to complete. It takes MetaDefender Core a few seconds to apply the new settings.

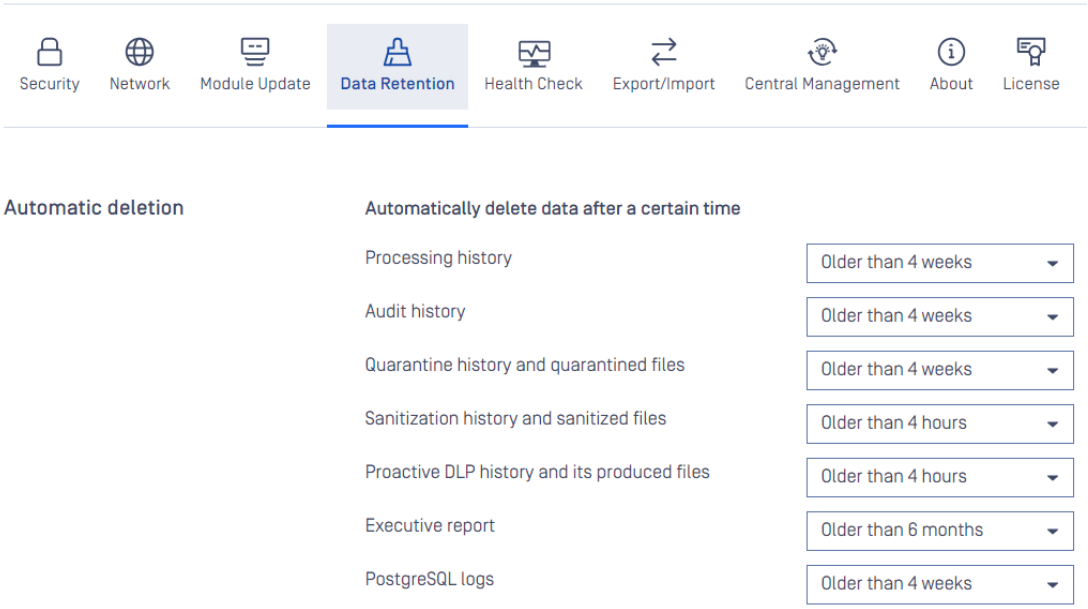


Warning
It may take MetaDefender Core up to 30 seconds to apply the settings. During this time, MetaDefender Core may not respond.

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 targetted for the data retention.

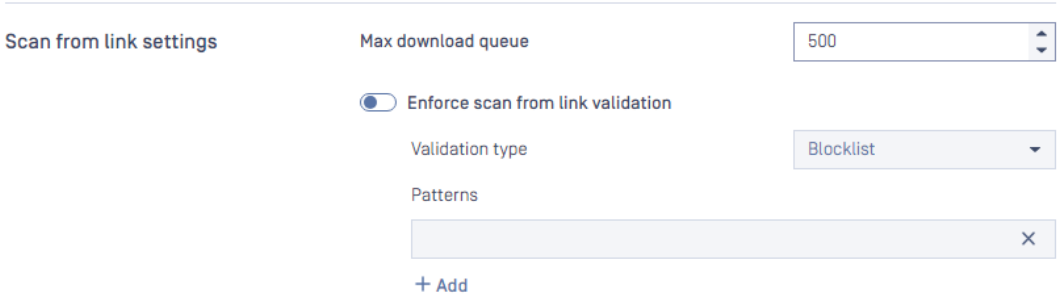


Stored scan results, quarantined files, audit log, statistics records and sanitized files that are older than the value set on this page, are permanently deleted from the server. In case you do not want to enable automatic clean up, set the value to off. This will prevent automatic removal of the scan history.

> **Warning**
> Setting the clean up value to off can have performance penalty.

Scan from link settings

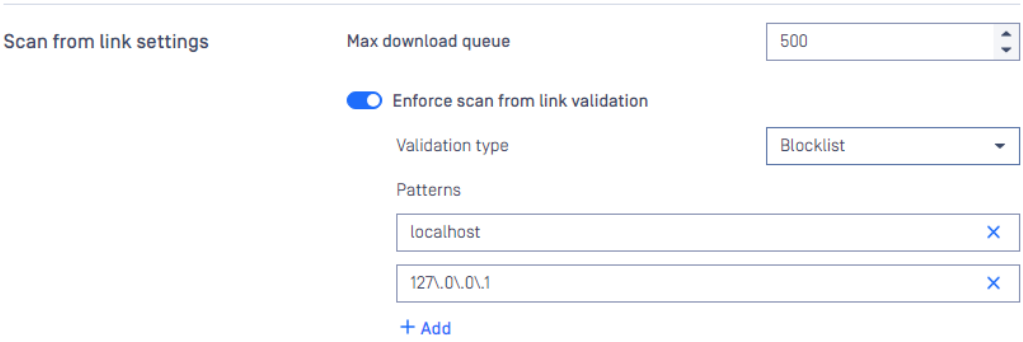
Scan from link settings are accessible under **Settings > Security** tab .



This settings is apply 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, restrict calling for the url that contains "localhost" and "127.0.0.1" we should use Blocklist and these patterns


Scan from link settings
Max download queue
500
Enforce scan from link validation
Validation type
Blocklist
Patterns
localhost
127\.0\.0\.1
+ Add

Nginx configurations

> **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

none

```
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.

none

```
## 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

none

```
## Deny certain Referers ###
if ( $http_referer ~*
(babes|forsale|girl|jewelry|love|nudit|organic|poker|porn|sex|teen) ) {
        return 403;
}
##
```

Block particular APIs

none

```
## 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

none

```
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

none

```
# 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.

SELinux Secured Policy

By default, SELinux (Linux security system based on role access, available on RedHat) does not protect the Nginx web server. The following instruction will help you setup and turn on the protection.

Step 1: First, install required SELinux compile-time support:

none

```
yum -y install selinux-policy-targeted selinux-policy-devel
```

Step 2: The download targeted SELinux policies to harden the Nginx web server on Linux servers from the

selinuxnginx project page:

none

```
cd /opt
wget 'http://downloads.sourceforge.net/project/selinuxnginx/se-
ngix_1_0_10.tar.gz?use_mirror=nchc'
```

Step 3: Untar the same:

none

```
tar -zxvf se-ngix_1_0_10.tar.gz
```

Step 4: Compile the same

none

```
cd se-ngix_1_0_10/nginx
make
```

Sample output:

none

```
Compiling targeted nginx module
/usr/bin/checkmodule:  loading policy configuration from tmp/nginx.tmp
/usr/bin/checkmodule:  policy configuration loaded
/usr/bin/checkmodule:  writing binary representation (version 6) to tmp/nginx.mod
Creating targeted nginx.pp policy package
rm tmp/nginx.mod.fc tmp/nginx.mod
```

Step 5: Install the resulting nginx.pp SELinux module:

none

```
/usr/sbin/semodule -i nginx.pp
```

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 API
- POST File Analysis API

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 API 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.

1.) Create a “ssl.conf.mdcore” (MetaDefender Core only accept this name) file

- On Windows, under ***<Installation Directory>\nginx***

none

```
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, under **/etc/ometascan/nginx.d/**

none

```
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

1.) Explicitly allow specific TLS versions, optionally with preferred ciphers. For example:

none

```
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_protocols tlsv1.1 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/

none

```
ssl on;

ssl_certificate "C:/Program Files/OPSWAT/Metadefender Core/nginx/cert.pem";

ssl_certificate_key "C:/Program Files/OPSWAT/Metadefender
Core/nginx/your_encrypted.key";
ssl_password_file "/etc/keys/private.pass";

ssl_protocols tlsv1.1 tlsv1.2;
ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-
GCM-SHA384;
```

For more SSL-options please consult Nginx documentation.

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/**

none

```
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.

none

```
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.

none

```
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**

none

```
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: <b3>` For example: `Active connections:``1``server accepts handled requests``144``144``358``Reading:``0``Writing:``1``Waiting:``0`	Request processed successfully. • Active connections <n> The current number of active 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.

Logging

To configure the log outputs and levels: MetaDefender Configuration

For Linux systems the installer configures the **logrotate** service to handle the Metadefender Core log files.

The default configuration will rotate daily and store the last 30 days.

If the log file path is modified, the logrotate config file should be updated as well.

> **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

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:

- **DUMP** : The most verbose severity level, these entries are for debuggers only.
- **DEBUG** : Debuggers severity level, mostly used by support issues.

- **INFO** : Information from the software, such as scan results.
- **WARNING** : An unexpected event occurred 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.

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
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]

Prefix field	Sample value	Description
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

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 • 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 message ID	Error description
8	Unknown error occurred in the daemon/service.
16	Error happened in the communication
21	Can't connect to the server
26	Can't send a message to the target, it's probably disconnected
28	The connection is timed out
73	Error happened in the communication
101	Error happened in the communication
113	Invalid file deletion
130	Unknown error occurred in the daemon/service.
149	Unable to gather OS information
162	Sending scan requests failed
166	File given can not be opened
169	REST frontend timeout.
176	Agent status is not available
179	Agent status is unavailable
181	REST response format is invalid
217	Engine name unreachable
218	Engine definition unreachable
219	Engine leak
220	Engine leak
221	Engine leak
222	Engine leak
231	Engine error

Error message ID	Error description
246	Can't transfer file, data channel is not open
248	File transfer response error
250	Error transferring file
252	Error in package staging process
253	Error in package staging process
254	Error in package staging process
255	Error in package staging process
256	Error in package staging process
275	Email sending error
277	Email sending error
282	Invalid main update descriptor
283	Invalid package descriptor
285	Invalid file downloaded
297	Invalid state option given in GetEngineInfoBlock
303	Downloaded package is not complete
304	Downloaded package is invalid
305	Downloaded package is invalid
306	Downloaded package is invalid
307	Downloaded package is invalid
308	Downloaded package is invalid
314	Agent with identifier is unavailable
319	Can not open downloaded package
320	Downloaded package is invalid

Error message ID	Error description
321	No respond from agent
325	No respond from agent
326	Error loading database
330	Error in uploaded descriptor
338	No respond from agent
341	Error in getting update package
342	Error in getting update package
343	Error in package staging process
355	Invalid reference to user data
379	Database error: user query
382	Database error: user query
383	Database error: users query
385	Database error: role query
386	Database error: roles query
425	Invalid state option given in GetDatabaseInfoBlock
428	Error in checking package updatable
431	Error validating engine
439	Agent status is not available
453	Agent with identifier is unavailable
459	No respond from agent
465	Database error: getting scan request
473	Agent does not have specific engine
474	Downloaded packagedescriptor is invalid

Error message ID	Error description
485	Error while downloading files from update server
486	Invalid certificate found while downloading files from update server
495	File missing during validation
496	File size mismatch during validation
497	SSL errors for https://update.dl.opswat.com
506	File type checking timed out
520	Could not trigger update
523	Invalid file reference
541	Invalid type reference for entry
542	Invalid type reference for entry
543	Invalid type reference for entry
545	Invalid type reference for entry
547	Invalid type reference for entry
551	Internal error occurred during the program run
552	System error occurred during the program run
553	Unknown error occurred during the program run
554	Can't create resource file
556	HTTP redirection count exceeded limit
563	Database error: get linked extracts
598	Invalid file reference
599	Engine type unreachable
602	Download timed out.
611	Can't remove the downloaded zip archive

Error message ID	Error description
613	File extraction failed while updating
620	Invalid content received for update
621	Can't download file for update
624	File open file for update
628	Can't read file to validate it's content
639	No respond from agent
641	Error in getting package state information
642	Error in checking package updatable
652	License is invalid or not containing licensed_engines
657	No activation_key is received
658	Invalid activation option value
659	Activation error
662	Connected agent count reached the license limit
663	License is invalid or not containing max_agent_count
670	Package download failed
678	Could not upload file
679	Update could not be applied
686	Trying to load engine which is not installed to the agent
687	Trying to load engine which is corrupt
708	Engine load failed
711	Updated engine can't find eicar file as threat
716	Installing engine resulted in error
718	Installing database resulted in error

Error message ID	Error description
721	Error in install database
722	Installing database resulted in error
723	File is missing for update in agent
727	Error loading engine
732	Error installing engine, unable to copy descriptor file
734	After updating the database the package descriptor is invalid
736	Error in prepare database
737	Error installing engine, unable to copy descriptor file
749	Engine process crashed
802	Could not create path to save updates
808	Could not save package, skipping
809	Could not save package, skipping
812	Could not save package, skipping
813	Could not save package, skipping
814	Error creating database package
816	Error creating database package
819	Error creating database package
820	Error creating database package
827	Database error: retrieving audit log
828	Database error: inserting audit log
829	Database error: retrieving audit log count
833	Exception during operation
835	Could not trigger update

Error message ID	Error description
837	Could not trigger update
838	Update package download failed
842	Error adding license information to engine descriptor
843	Could not copy legacy package, skipping
844	Could not save legacy package, skipping
846	Packagedescriptor could not be read
847	Could not read descriptor file
848	Could not write descriptor file
852	Error opening CERT file
854	No legacy database wrapper package present
855	Could not write licenses to database descriptor
858	Metadescriptor could contains tampered information, abort processing
861	Can't process new licence information
862	Error in package staging process
863	Can't initialize source instance
864	Can't create resource file
865	Can't send data to agent
866	Can't transfer file, can't open resource file
868	Can't get temporary file name
869	Can not find data channel
870	Can not load engine shared library
871	Cannot create temporary directory for engine files
892	Could not export legacy package, skipping

Error message ID	Error description
931	Unexpected network reply arrived
945	Error in getting update package by state
947	Error receiving data, request log from customer
948	Not enough disk space to export package(s)
949	Not enough disk space to export package(s)
950	Not enough disk space to export package(s)
951	Error sending data to agent, request log from customer
952	Not enough disk space to apply update package on agent
953	Not enough free disk space for agent to handle updates
955	Error sending data to agent, request log from customer
969	Creating of symbolic link has failed
970	Error in prepare database
978	Could not establish remote communication
979	Could not establish local communication
981	Deactivation error
995	Activation key not found in license
996	Stored license is invalid
997	Invalid activation option value
998	Automatic rectivation error
999	Error saving license information
1002	License activation error
1003	Error during license activation
1009	Network error occurred while trying to lookup hash

Error message ID	Error description
1010	Invalid response retrieved from server
1017	Due to errors in previous requests, disabling hash lookup for a given interval
1019	Too many timeout error occurred in a short period of time
1025	Error occurred in engine process
1027	Issue occurred while transfering file to agent
1030	Invalid update folder found
1031	Error Sending hashes message, request log from customer
1037	Engine capabilities unreachable
1042	Engine's descriptor is invalid
1043	Could not open update archive
1046	Custom engine task starting timed out
1047	Custom engine task finishing timed out
1050	Issue with agent, see troubleshooting guide
1051	Engine issue on agent
1052	Error in installing custom engine, engine with id not found
1053	Issue happened with engine
1055	Agent with identifier is unavailable
1057	No respond from agent for custom engine installation
1058	Error validating engine, no engine descriptor found
1059	Error validating engine, invalid engine descriptor found
1064	Not handling file as local, not must be an issue
1068	Could not apply udpates from picku folder
1081	Could not create folder for pickup

Error message ID	Error description
1095	Testing engine failed
1096	File transfer response error
1100	Error in package staging process
1126	Could not create quarantine database
1134	Error in quarantine database
1135	Error in quarantine database
1136	No such file to remove
1141	Error in quarantine database
1142	Error in quarantine database
1144	Error in quarantine database
1147	Error reading quarantine file
1148	Download aborted
1151	Error adding file to quarantine
1152	Error adding data chunk to quarantine file
1153	Error receiving file data, removing file from sanitized storage
1154	Given path is not absolute
1155	Could not create storage folder
1156	File storage path is not writable
1157	No owner given
1158	Id is not specified
1159	Id is too short
1160	Could not create directory
1161	Could not create directory

Error message ID	Error description
1163	Error opening file
1164	Could not write file
1165	File id is not specified
1166	No such file found
1167	Could not remove file
1168	File id or owner is not specified
1169	No such file to finalize
1170	Error sending file data to core
1171	Can't request file, data channel is not open
1173	Could not request file from agent
1174	Can't request file from agent
1176	Processing file from agent encountered an error
1177	Could not read salt value
1178	Could not read salt value
1179	Error creating salt file
1180	Error generating salt value for quarantine
1182	Error occurred quarantining file
1183	File not found in quarantine
1184	Same owner was already reading another file
1185	Error reading file in quarantine
1186	Error writing data to quarantined file
1188	Could not finalize quarantined file
1190	No file with given id is present for removal

Error message ID	Error description
1191	Could not remove file from quarantine
1192	Could not remove file from quarantine
1193	Could not gather quarantine information
1196	Database error: user query
1198	Error parsing disabled times information
1201	See reason for more information about the reason
1204	Can't open metadescriptor file for writing
1220	The certificate chain maybe invalid
1221	Info about the certificates in the chain
1226	Vulnerability lookup timed out
1227	Error in checking package updatable
1229	Error receiving file data, removing file from quarantine
1230	Engine capabilities unreachable
1231	Engine leak
1232	Engine capabilities unreachable
1233	Engine leak
1235	Could not create sanitized database
1252	Error in sanitize database
1253	Error in sanitize database
1254	Error getting sanitized files
1258	Error querying sanitized files
1266	Error occurred adding file to sanitized storage
1267	File not found in sanitized storage

Error message ID	Error description
1268	Same owner was already reading another file
1269	Error reading file in sanitized storage
1270	Error writing data to sanitized file
1272	Could not finalize sanitized file
1277	Could not gather sanitized files information
1280	Error reading quarantine file
1281	Download aborted
1284	Wrong enabled template in sanitize block
1285	Wrong convert template in sanitize block
1287	Error in sanitization, retries emptied
1290	Error adding file to sanitized storage
1291	Error adding data chunk to sanitized file
1295	Error saving sanitized file
1305	Database's descriptor is invalid
1603	Error in package staging process
1604	Filestore unreachable
1605	Could not upload file
1606	Could not upload file
1607	File not exist to load schema from
1608	Schema is not a proper JSON
1609	Schema format is invalid
1670	Error while checking if package is md.com only
1677	File not exist to load default value from

Error message ID	Error description
1678	Default value file is not a proper JSON
1679	Default value file format is invalid
1693	Database error occurred
1695	Database error occurred
1696	Database error occurred
1697	Database error occurred
1699	Unable to apply patch to revert configuration
1700	Reverted configuration is invalid, hash not matching
1701	Database error occurred
1702	Unable to load configuration from default values
1703	Unable to save configuration from default values
1704	Default value file is not a proper JSON
1705	Default value file format is invalid
1706	Unable to load configuration
1711	Error when sending bytes through socket
1712	Default value file is not a proper JSON
1740	Error reserving data id
1741	Process local file encountered an error
1742	No data id was set in HandleLocalFileBlock
1777	Unable to save new configuration
1781	Database error: requesting export on empty processing history
1782	Removing engine folder failed
1783	Removing engine folder failed

Error message ID	Error description
1784	Removing engine folder failed
1785	Removing folder failed
1787	Removing invalid update folder failed
1790	Removing folder failed
1791	Removing folder failed
1792	Removing package folder failed
1793	Removing database folder failed
1794	Removing folder failed
1795	Removing folder failed
1798	Removing downloading packages failed
1801	Error when selecting from downloaded
1803	Error when deleting from downloaded
1804	No result when selecting from downloaded
1805	Removing engine folder failed
1825	Database error: role query
1828	Invalid username or password
1830	Error in writing configuration file
1833	User has been suspended
1835	Database error: users query
1836	Error in updating engine configuration
1850	Database error: could not create default local directory
1855	Database error: userdirectory query failed
1859	User validation error

Error message ID	Error description
1861	User validation error: missing password or salt
1862	Database error: user query
1869	Error in workflow manager
1870	Error in workflow manager
1871	Error in workflow manager
1877	Failed to remove stuck file after multiple attempts
1883	Database error: users of user directory could not be deleted
1911	No root CA certificates found for LDAP SSL/TLS connections on windows platform
1994	Content length header does not match with downloaded content
2031	Timed out adding file to sanitized archive
2035	Invalid sanitization configuration found for archive
2049	Engine compression capabilities unreachable
2050	Engine leak
2051	Next file extraction resulted in fail
2090	Database error: user query
2091	Database error: user query
2095	Sending scan requests failed
2102	Sending external action requests failed
2107	Block missing option 'origin'
2123	No respond from agent
2124	Not enough information for whitelisting
2194	Input folder got corrupted, removing it
2333	File transfer response error

Error message ID	Error description
2337	Local file path and body both set, ambiguous request
2352	Error adding file to quarantine
2354	Error in quarantine database
2357	Error in quarantine database
2360	Could not update quarantined file
2361	Could not gather quarantine information
2382	Selected rule not matches batch's rule
2384	Error storing scan batch
2388	Database error: error querying batch
2390	Database error: get batched requests
2394	Error in writing license file
2395	Error in writing license file
2424	Could not create job object
2425	Could not set information on job object
2426	Could not assign process to job
2484	Cert or key file is not readable
2485	Configured certificate cannot be used
2486	Configured certificate cannot be used
2487	Configured certificate cannot be used
2532	Found unregistered file inside sanitized storage
2547	Stopped extracting archive file due to reaching a limit
2548	Archive engine gave invalid response
2555	Unable to save updated configuration

Error message ID	Error description
2637	Invalid node id while trying to load engine
2640	Invalid node id while trying to query node status
2646	Database error: error querying what to delete
2651	Cannot find file in FinalizeScanResultBlock
2652	Given dataid can not be cancelled
2653	Given dataid can not be cancelled
2654	Database error: users query
2737	Database error: userdirectory query failed
2748	Database error: finalizing statistics
2888	Something is wrong when opening file
2892	Possible Insufficient memory to operate, system throw bad_alloc
2895	LDAP error: DN of group has changed in remote directory service. Group can not be considered valid any more
2897	Database error: Can not cache user. Already exists as non-cached
2898	Database error: Can not cache user. Already cached in DB with different DN attribute
2905	File can not be opened for writing
2906	Unable to write to file
2910	Certificate name is invalid, not using HTTPS
2973	Invalid INIT state setting for megapackage
2974	Invalid same state setting for megapackage
2983	No megapackage to delete with id
2988	Error removing package
2996	Package sent for validation is invalid
2997	Error validating checksum of package

Error message ID	Error description
2998	Invalid path to generate megapackage
2999	Not enough free space to generate megapackage
3000	Error creating package
3001	Invalid package found during megapackage generation, aborting
3002	Error adding file to megapackage
3003	Error adding file to megapackage
3004	Error adding report to megapackage
3005	Error generating sha256 checksum for megapackage
3006	Could not open file for reading
3007	Error creating database package
3008	Error creating database package
3009	Error creating database package
3013	Megapackage could not be locked for download
3014	Error opening megapackage for reading
3015	Megapackage file is missing from disk
3016	Could not initiate download
3040	An engine failed to initialize properly
3042	Node could not create new folder for engine
3043	Node could not create new folder for engine
3044	Error in updating engine schema
3045	Error in package storing process
3048	Invalid engine configuration
3049	Schema based configuration received for non schema based engine

Error message ID	Error description
3050	Schema based engine received legacy configuration
3051	Legacy configuration received is invalid
3052	Invalid configuration received for engine
3053	Schema based engine config is invalid
3059	Failed to send extract task to Agent
3060	Failed to extract
3061	Extraction timeout reached
3109	Database installation failed
3110	Engine installation failed
3111	Engine installation failed
3112	Engine installation failed
3113	Custom Engine installation failed
3114	Engine installation failed
3115	Communication possible disconnected
3116	Communication possible disconnected
3118	No valid node status found
3120	No result for DLP lookup
3121	Engine capabilities unreachable
3122	Engine leak
3123	Engine leak
3126	Invalid DLP engine preferences found
3127	Invalid DLP engine preferences defaults found
3128	Invalid DLP engine preferences schema found

Error message ID	Error description
3133	No valid node status found
3136	Database error: finalizing statistics
3140	Can't process shared resource file
3141	Can't transfer file, data channel is not open
3142	Can't transfer file, no result from node
3143	Resource file sharing response error
3185	Error reading quarantine file
3186	Error sending data to agent, request log from customer
3188	Error transferring file
3195	Sandbox scheduling error
3197	Error in quarantine database
3198	Invalid result from sandbox engine
3199	Invalid result from sandbox engine
3201	Invalid result from sandbox engine
3205	Sending scan requests failed
3206	Sending scan requests failed
3207	Sending scan requests failed
3208	Invalid sandbox result
3209	Invalid sandbox result
3210	Invalid result from sandbox engine
3211	Sending scan requests failed
3214	Invalid sandbox result
3215	Invalid sandbox result

Error message ID	Error description
3216	Invalid result from sandbox engine
3241	Extraction open task timed out
3242	Extraction next task timed out
3257	An error occurred while generating package
3259	Package generation is already in progress
3260	Package generation is already in progress
3261	Error while generating package
3264	Source type is unknown
3267	Can't extract zip source
3268	Http source is not a zip
3269	Can't open package descriptor for writing
3273	No status for node found
3274	No Yara engine found on node
3275	No result received
3276	Database error: sources query.
3277	Database error: issues query
3288	Expiration field is not a proper date
3289	License is expired
3291	Expiration field is present but it is not a timestamp
3303	EngineManager: RPC message received, but has no sender
3305	EngineManager: RPC executor not found, sending back to caller
3307	EngineManager: RPC caller not found, result orphaned
3311	Engine failed to call remote procedure call

Error message ID	Error description
3313	Remote procedure call timed out
3315	Engine called RPC, but an error happened
3317	Engine: There is no active engine to send RPC message to
3326	Cloud scheduling error
3331	Error in quarantine database
3338	Database error: cloud
3340	Database error: cloud
3346	Database error: cloud
3347	Database error: cloud
3349	Database error: cloud
3350	Database error: cloud
3352	Database error: cloud
3354	Database error: cloud
3357	Sending scan requests failed
3358	Sending scan requests failed
3359	Sending scan requests failed
3360	Invalid result from cloud engine
3363	Sending scan requests failed
3365	No cloud engine can be found in Core's engine set
3367	No cloud engine is running on the node
3368	Invalid result from cloud engine
3373	Ignition file processing failed
3378	Failed to finish welcome wizard

Error message ID	Error description
3379	Welcome wizard user not found
3380	Failed to remove welcome wizard user
3414	Probably incorrect db state
3415	Probably incorrect db state
3454	An error occured after data was transfered to node
3461	Engine compression capabilities unreachable
3467	Database error: product query failed
3473	No respond from agent
3474	Installing database resulted in error
3475	Engine: There is no passive engine
3476	Engine: There is no active engine to deactivate
3477	Engine: Engine is already deactivated
3479	Error in updating custom engine, engine with id not found
3480	Error in updating custom engine, engine with id not found
3481	Error in updating custom engine, engine with id is active
3482	Installing database resulted in error
3483	Removing custom engine db folder failed
3484	Error installing database
3485	Error installing database, preupdate custom engine failed
3486	Error installing database, unable to copy descriptor file
3487	Error installing database, unable to copy database files
3488	Error installing database, postupdate custom engine failed
3490	Error in updating custom engine, engine with id not found

Error message ID	Error description
3494	Node couldn't create resources folder in given timeout
3500	Error validating user. Current server time is wrong.
3501	User validation error: Missing token data
3502	User validation error: Token data has expired
3503	Failed to remove reset password token from database
3506	Failed to validate user belonging to a disabled directory
3507	Non-local users are not allowed to login by token
3508	User validation error
3510	Error updating password
3515	Email sending error
3517	Email sending error
3519	Invalid login token
3523	Can't login mail server with the configuration
3524	Can't connect to mail server with the configuration
3525	Exception during closing connection to mail server
3532	Engine process is killed when timed out
3534	Can't find watermarked file
3535	File was still open when started reading from another
3536	Can't open watermarked file
3537	Error adding data to dlp file
3539	Error finalizing file
3541	Error occurred adding file to dlp storage
3544	Error reading watermarked file content

Error message ID	Error description
3545	Downloading watermarked file aborted
3546	Could not create dlp database
3553	Error in dlp database
3554	Error in dlp database
3555	No such file to remove
3559	Warning, could not remove file
3561	Error in dlp database
3567	Error querying watermarked files
3571	Error adding file to dlp storage
3572	Error adding data chunk to dlp file
3573	Error receiving file data, removing file from dlp storage
3575	Engine capabilities unreachable
3576	Engine leak
3580	Error adding new watermarked file to dlp storage
3644	Trying to send a compress task not exist in running compression tasks
3645	Trying to send an extract task not exist in running extraction tasks
3661	Error occurred while accessing scan details to add quarantine action
3664	Trying to load un-existed engineprocess*.exe
3666	Error when creating symlink of engineprocess*.exe
3677	Found unregistered file inside dlp storage
3678	Null metadata
3679	Invalid format metadata
3680	Error at parsing metadata

Error message ID	Error description
3681	Invalid format metadata
3687	Extraction size limit reached
3688	Cannot remove session id
3692	Extraction timeout reached
3693	Engine extraction capabilities unreachable
3694	Engine leak
3736	Failed to get version of preference schema
3738	Failed to get converted preference schema
3746	Failed to receive result after sending task to handle scan preferences
3747	Failed to convert scan preferences
3753	Error in creating result containter.
3754	Archive engine return a not empty folder
3756	Error in response while hash lookup such as invalid apikey
3758	Failed to hash
3762	Timeout when loading engine
3763	Timeout when testing engine
3774	Client disconnected, the data has not been transferred successfully
3775	Client disconnected, the data has not been transferred successfully
3780	Found unregistered file inside quaratine storage
3807	Error in setting update url
3809	Re-generate download ID
3811	Clean up outdated download IDs
3827	User roles validation error

Error message ID	Error description
3828	User roles validation error
3832	The file is not an archive or corrupted
3833	The file is not an archive or corrupted
3835	Error in getting package
3845	Failed to rotate nginx log, cannot rename log file
3847	Failed to rotate nginx log, cannot rename log file
3850	Failed to rotate nginx log, failed to reopen log
3854	Rotate nginx log successfully but cannot compress
3863	Could not remove file
3870	Could not remove file from sanitized storage
3871	Could not remove file from sanitized storage
3872	Could not request file from agent
3873	Processing file from agent encountered an error
3876	After updating the database the engine folder could not be found
3877	Node is under high load
3888	Missing suffix of original file
3893	No response from agent
3917	Open file resulted in fail
3918	Error when compressing.
3919	Compression has got error
3925	Undefined webhook reply
3926	Webhook response failed
3933	Stop extraction resulted in fail

Error message ID	Error description
3934	Directory not supported
3955	Failed to invoke extension
3958	Failed to login with Single Sign-On enabaled
3959	Do single sign-on with an unsupported directory
3971	Error in statistic info db
3972	Error querying statistic info db rowid
3973	Error querying statistic info db rowid
3996	File upload rejected due to insufficient disk space.
3999	Database error: configuration query
4000	Database error: configuration query
4001	Unable to apply patch to revert configuration
4002	Reverted configuration is invalid, hash not matching
4003	Database error: configuration query
4005	Database error: insert configuration
4006	Database error: configuration query
4007	Database error: configuration query
4008	Database error: configuration query
4009	Migration failed. Cannot create log directory
4022	Database error: user query
4023	Database error: Failed to delete user directory
4024	Database error: Failed to delete user directory
4025	Database error: Failed to delete user directory
4026	Database error: Failed to delete user directory

Error message ID	Error description
4027	Database error: Failed to delete user directory
4031	Error delete physical quarantine file
4033	Database error: error deleting all records
4034	Database error: Failed to delete records
4041	Error adding file to quarantine
4044	Failed to insert new request
4045	Failed to insert scan result
4046	Failed to clean incompleted requests
4047	Database error: update verdict count per hour
4048	Database error: update DLP count per hour
4049	Could not finalize scan requests
4050	Failed to insert new batch
4051	Database error: getting scan request id from hours
4052	Database error: getting verdicts statistics by hours
4053	Database error: getting DLP statistics by hours
4054	Database error: getting verdicts statistics by hours
4055	Database error: getting statistics info
4056	Database error: getting statistics info
4057	Database error: error deleting all records
4058	Database error: Failed to delete records
4062	Database error: error querying what to delete
4066	Database error: cloud
4069	Error in statistics DB

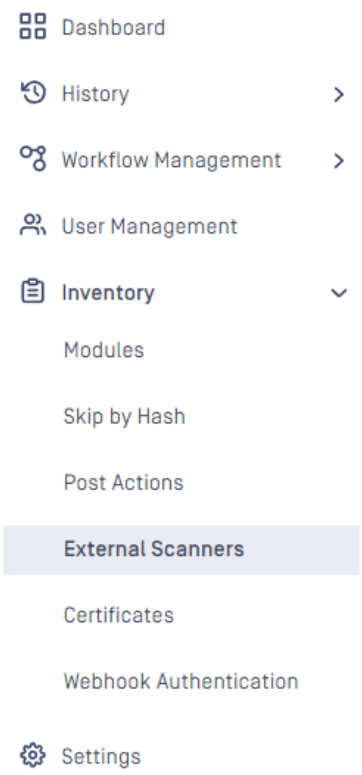
Error message ID	Error description
4070	Error in statistics DB: get rule id by name
4071	Error in statistics DB: rule not found
4072	Error in statistics DB: get metascan statistics
4073	Error in statistics DB: get CDR statistics
4074	Error in statistics DB: get DLP statistics
4075	Error in statistics DB: get vulnerabilities statistics
4076	Error in statistics DB: wrong type for top file types
4077	Error in statistics DB: get file types statistics
4081	Could not remove file from sanitized storage
4082	Could not remove file from sanitized storage
4084	Postgres issue
4089	Error in dlp database
4091	Coud not found the key to decrypt proxy password
4093	Error in sanitize database
4096	Database error: Failed to delete user directory
4097	Error in package staging process
4098	Error in get engine by state
4099	Error in get engine configuration
4100	Error in get engine schema
4101	Database error: user query
4102	Database error: Failed to delete user directory
4104	Database error: Failed to delete user directory
4106	Database error: Failed to delete user directory

External Scanners And Post Actions

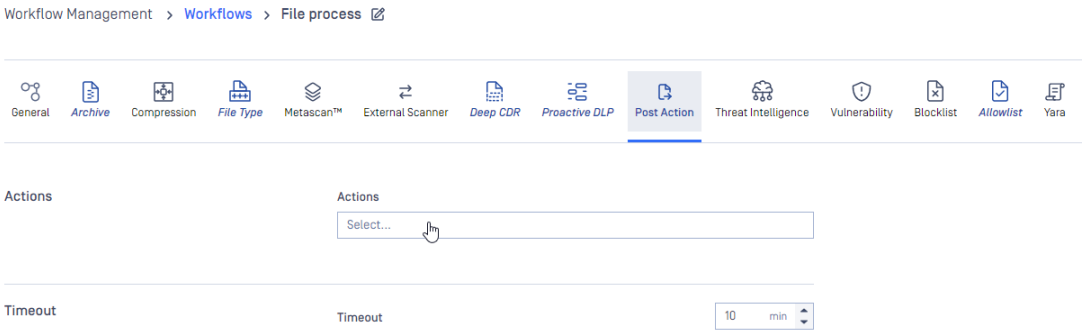
Under **Inventory** menu it is possible to configure custom **External Scanners** and custom **Post Actions**.

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



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

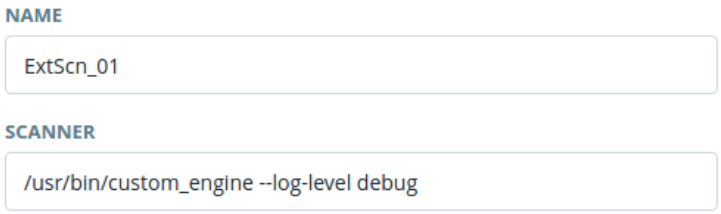
External Scanners are handled as scan engines from product side but are not updatable through the product.

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 API > 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

json

```
{
"def_time": 1491288912392,
"scan_result_i": 0,
"threat_found": ""
}
```

Example scan result where External Scanner found the file to be clean

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 run after the scan of the file for any post functionality such as copying the file etc...

Specification for post action process

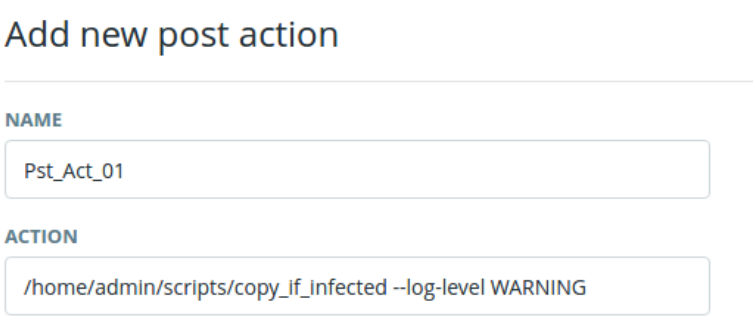
- **INPUT**
  - On standard input it gets the currently available scan result JSON without the `extracted_files` field
  - As last argument on the command line it gets the absolute path for the file
- **OUTPUT**

- 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.

Example of a Post Action



The scan result JSON if the Post Action returns 0

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

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"
  },
...
```

Hardening 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

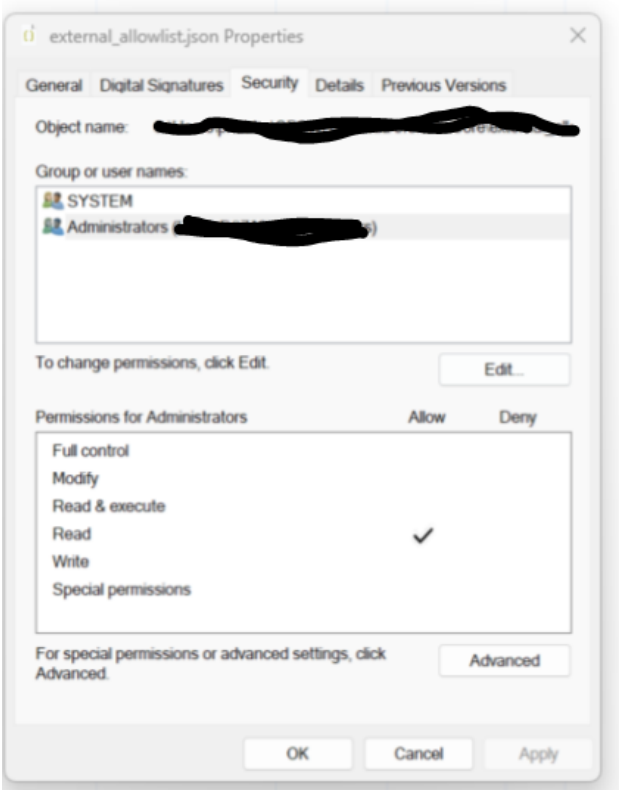
Linux

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

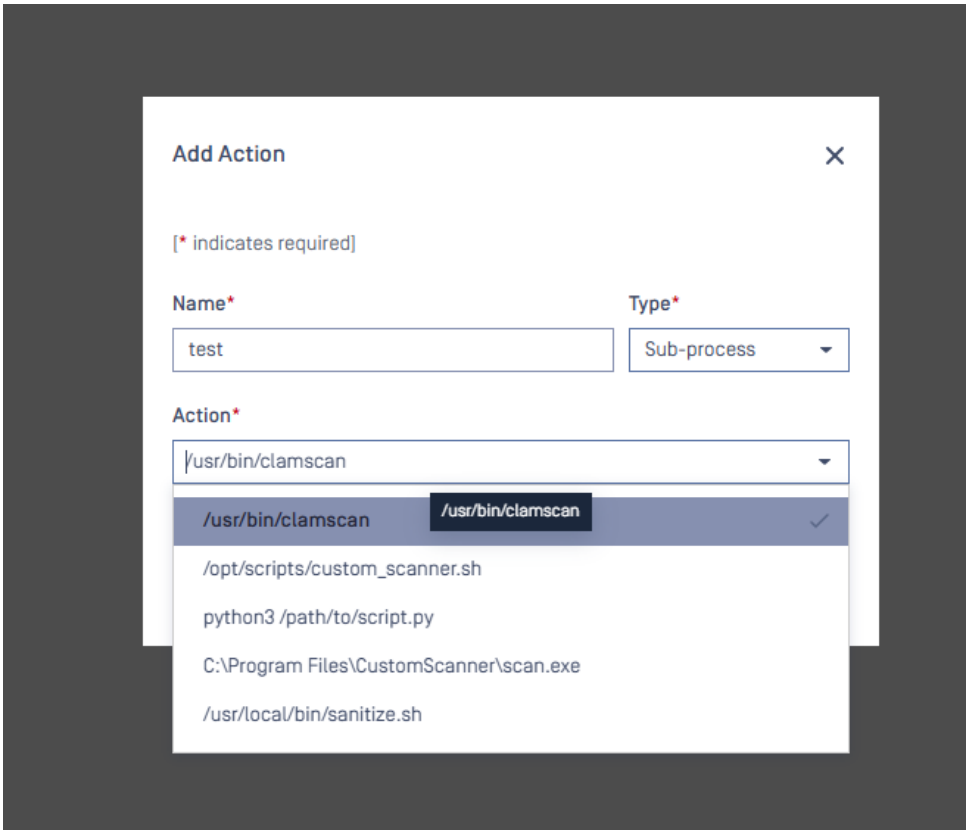
The feature is automatically enabled when the `external_allowlist.json` file exists:

- 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."


Add Action
[* indicates required]
Name*
test
Type*
Sub-process
Action*
/usr/bin/clamscan
/usr/bin/clamscan
/usr/bin/clamscan
/opt/scripts/custom_scanner.sh
python3 /path/to/script.py
C:\Program Files\CustomScanner\scan.exe
/usr/local/bin/sanitize.sh

Database Management

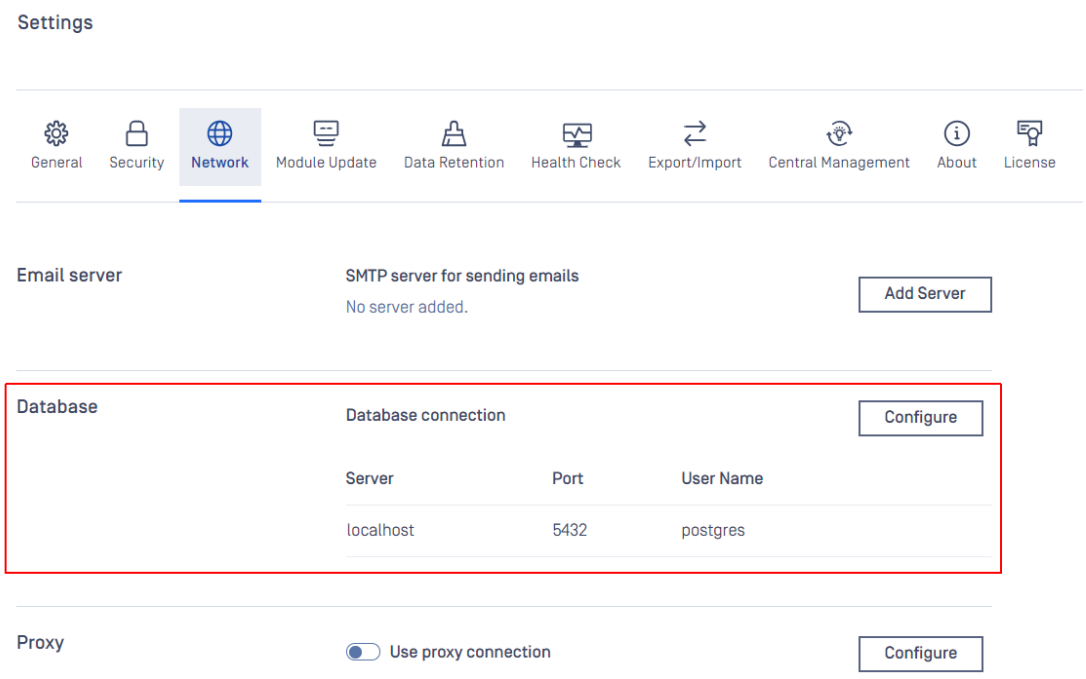
Starting MetaDefender Core 4.20.1, we have supported users to update PostgreSQL server information.

It could be done via either MetaDefender Core management console or separate CLI tool.

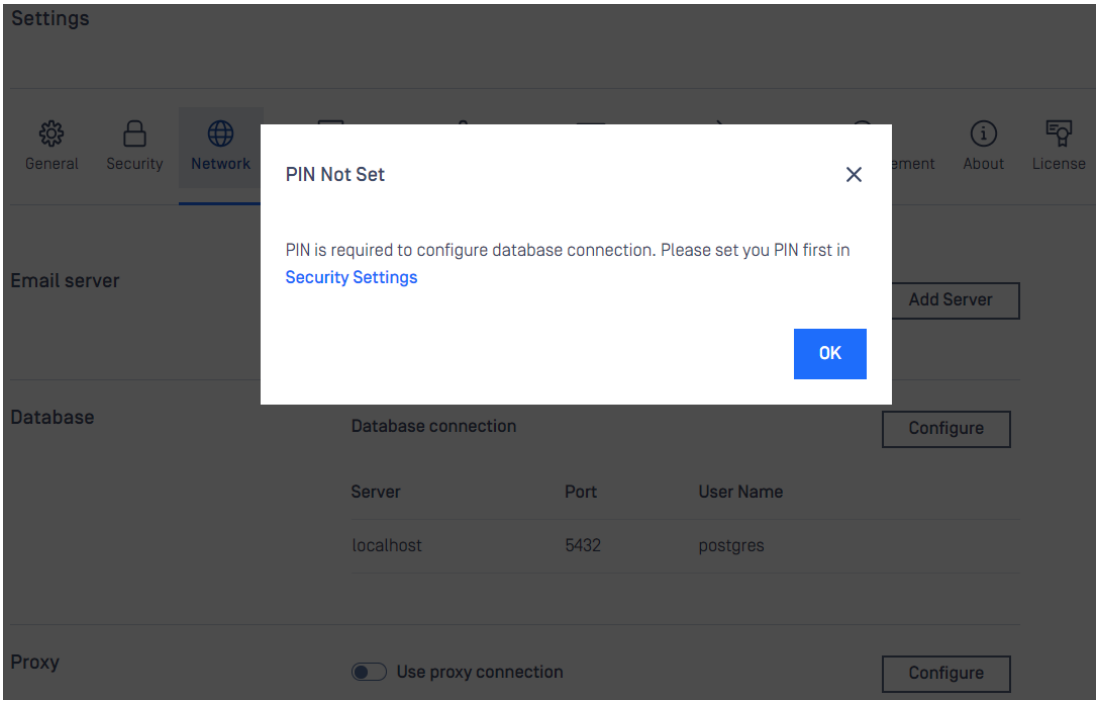
- MetaDefender Core management console
- CLI tool (ometascan-change-db-credentials)

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

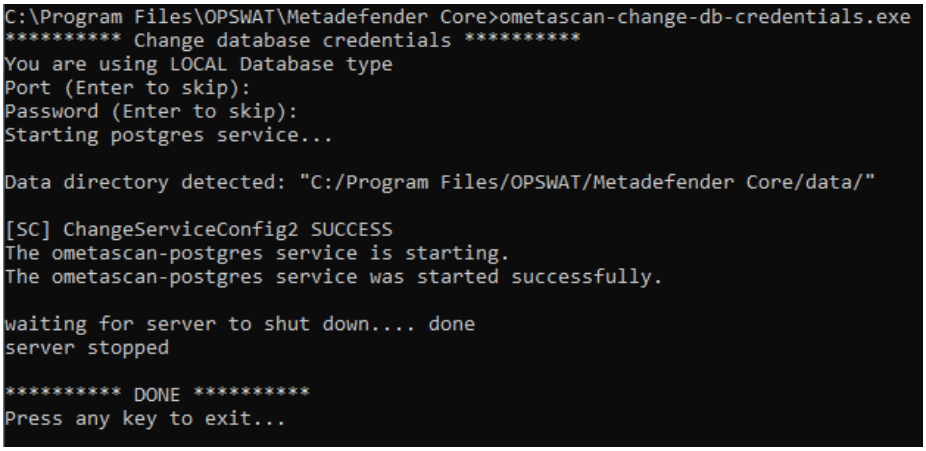
Info

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:

none

```
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
```

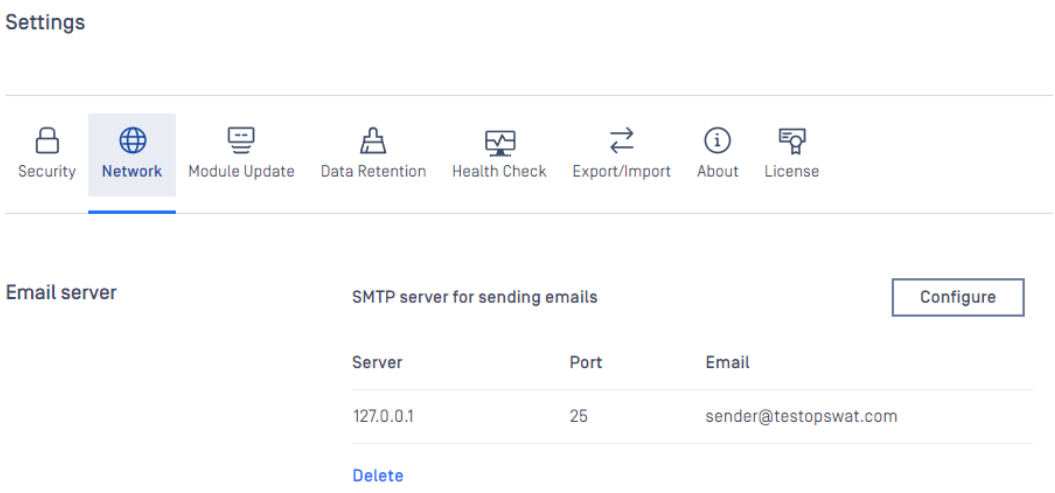
By default, this CLI tool will help change the PostgreSQL database admin credentials on both MetaDefender Core and PostgreSQL server sides. If you do not want to change the password on PostgreSQL server side, make sure you will use “-s” parameter.

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.

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).



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 transmission protocol)
AUTH METHOD	Specify the authorization protocol connecting to SMTP server	• AUTH PLAIN • AUTH LOGIN (preferred) • NONE (no authentication)

User authentication

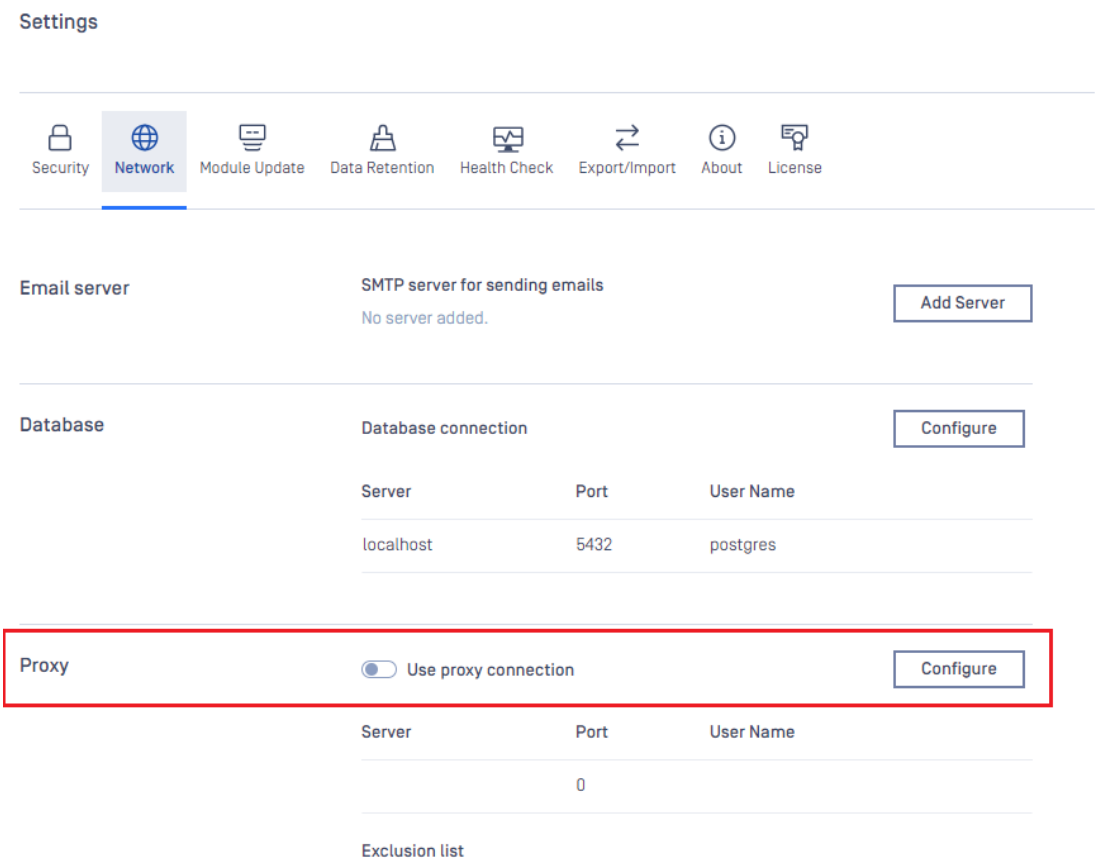
Properties	Description
EMAIL	Email address
USERNAME	Username for SMTP authentication (No required if AUTH METHOD = NONE)
PASSWORD	Password for SMTP authentication (No required if AUTH METHOD = NONE)

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.

Proxy Configuration

Since MetaDefender Core 4.19.0, 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.



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.*

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 API
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 API 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 API) 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 API

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 API
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 API

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 API

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.

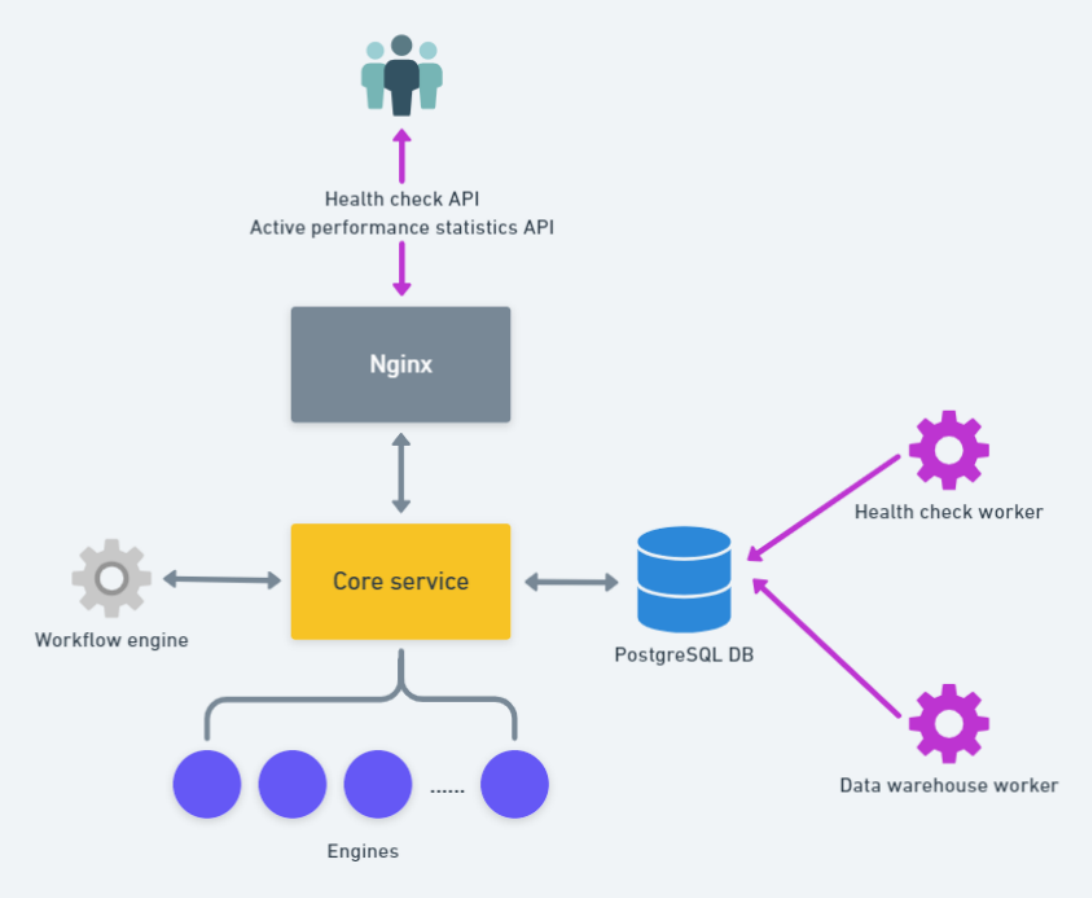
Nominating MetaDefender Core backup instance in offline mode
Instead, following actions needs to be made as alternative procedure:

- 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.

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.

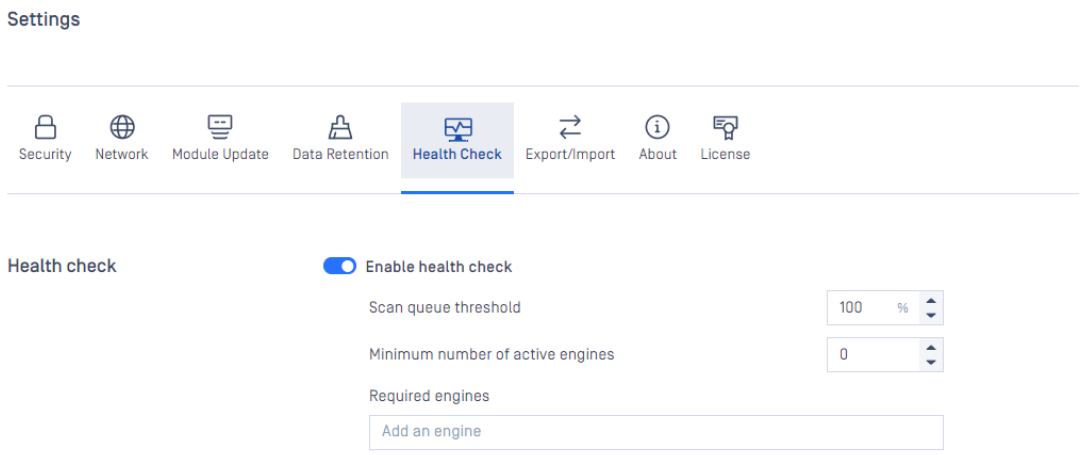
Learn more on how to use the heath check API at: GET - Get health check status API

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.
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		Selecting which engines mandatory for health check pass logic.

Using administrative REST API

Learn more at: GET - Get health check API

My OPSWAT Central Management

On this session, we will introduce how to enroll MetaDefender Core (MD Core) into My OPSWAT Central Management (MOCM), configure MD Core on MOCM, and the settings related to the connection between MD Core and MOCM.

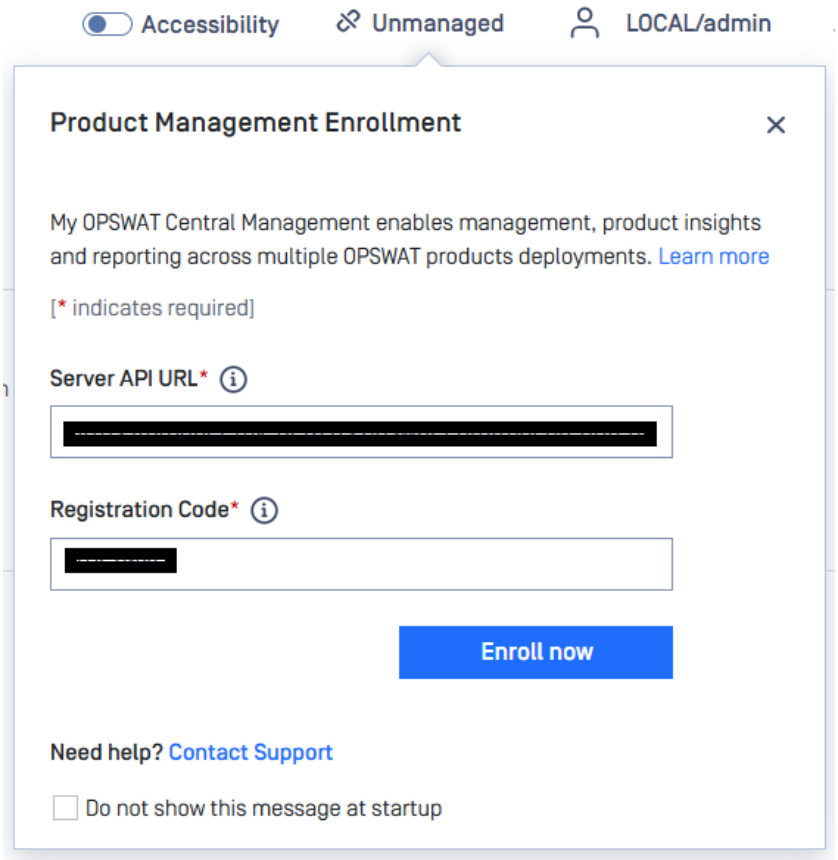
Support DB mode
Please note that currently, only MD Core instances using **Standalone DB** mode are supported for enrollment in MOCM

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.`

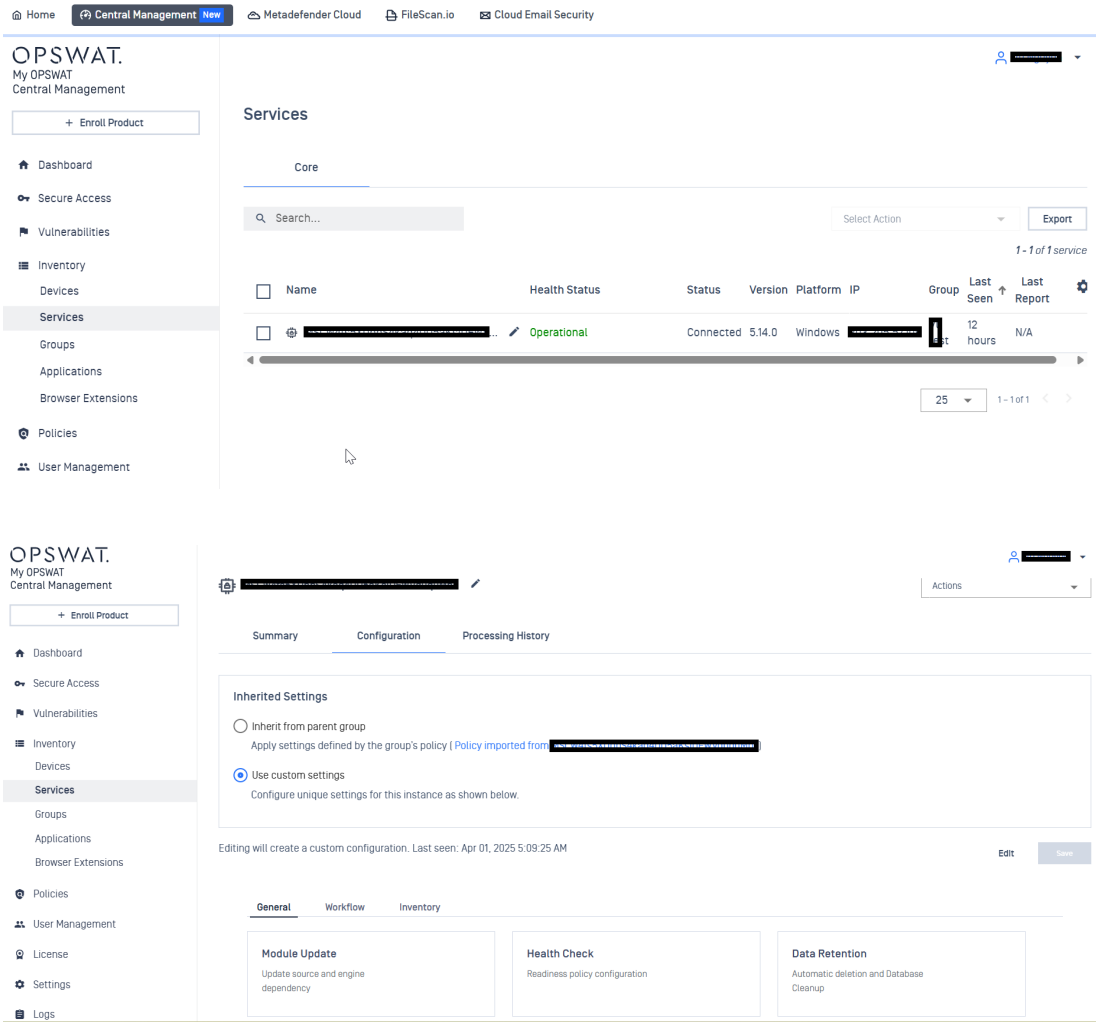


Configuring MetaDefender Core within My OPSWAT Central Management

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

Not supported yet

General

 Module Update

 Health Check

 Data Retention

Workflow

 Workflow Rules

 Workflow Templates

 Security Zones

Inventory

 Modules

- Configuration
- Lock/Unlock engine package
- Disable/Enable engine
- Delete engine

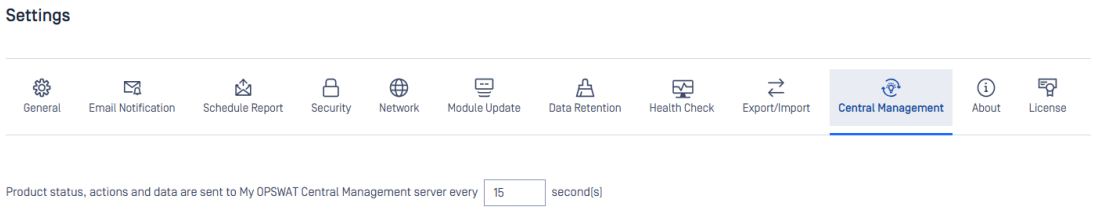
 External Scanners

 Webhook Authentication

 Post Actions

Settings related to My OPSWAT Central Management

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.



Email Notification

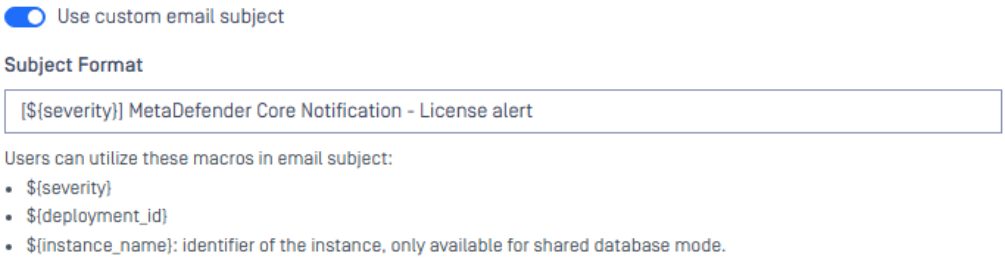
Administrator can configure Email Notification settings to receive notifications for detected threats and incidents, providing real-time updates on potential security risks.

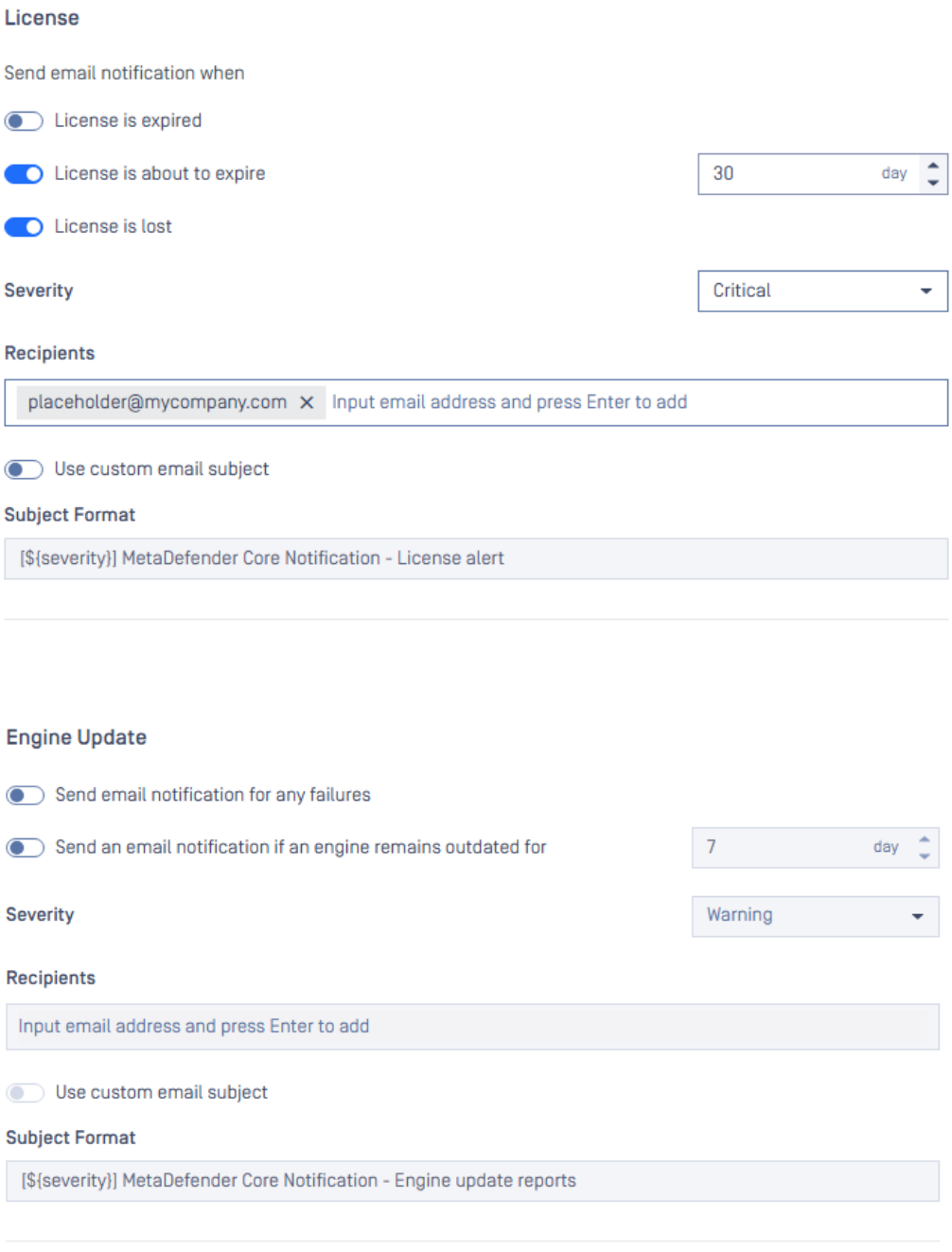
- **Soon-to-Expire Licensing:** Helps avoid service interruptions by facilitating timely license renewals.
- **Failed-to-Update Engine:** Notifies on “Download Failed”, “Permanently Failed”, Temporary Failed”, and “Content Invalid".
- **Outdated engines**: Notifies when there is any engine that not updated (update new database version) in a period of time.
- **Pre-defined Threats Detected :** Defines which scan results, threats, or incidents should trigger a notification. These can include infections, suspicious activities, or the presence of sensitive data.
- **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.
- **Database monitor:** Alerts users about current status of Database, notify for users for running vacuum process if needed

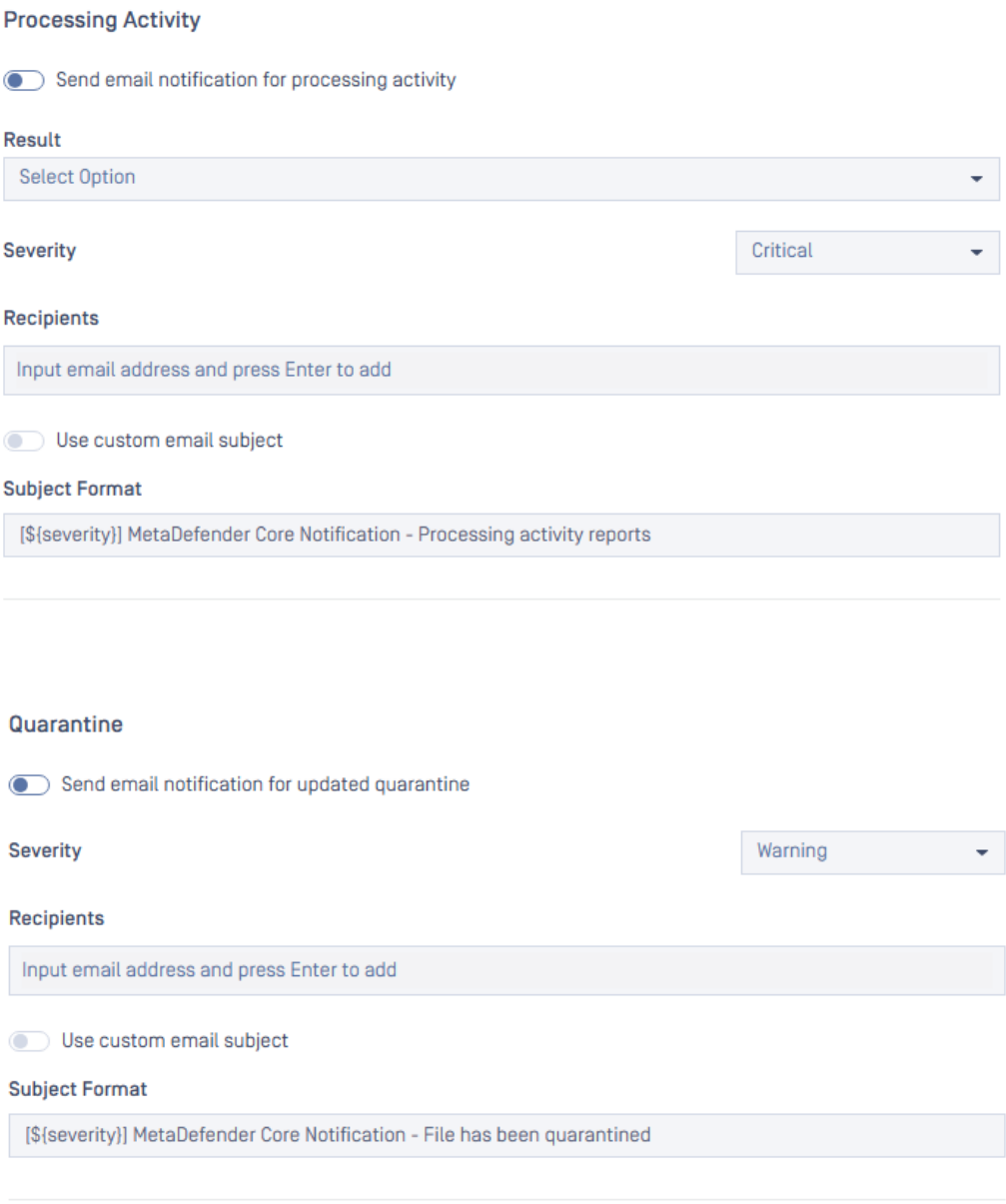
Custom Email Subject Format

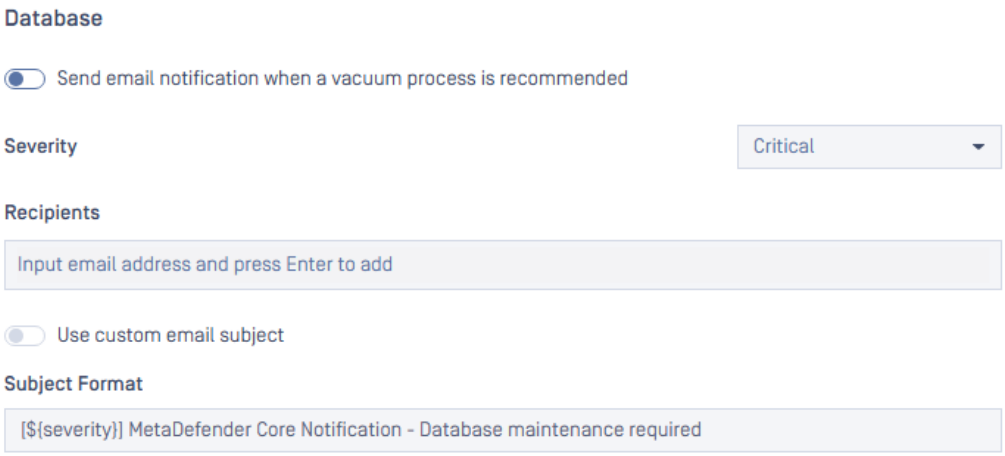
For each section, user can define custom subject format. MetaDefender Core provide 3 pre-define macro

- ${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**




License
Send email notification when
License is expired
License is about to expire
30
day
License is lost
Severity
Critical
Recipients
placeholder@mycompany.com
Input email address and press Enter to add
Use custom email subject
Subject Format
[${severity}] MetaDefender Core Notification - License alert
Engine Update
Send email notification for any failures
Send an email notification if an engine remains outdated for
7
day
Severity
Warning
Recipients
Input email address and press Enter to add
Use custom email subject
Subject Format
[${severity}] MetaDefender Core Notification - Engine update reports


Processing Activity
Send email notification for processing activity
Result
Select Option
Severity
Critical
Recipients
Input email address and press Enter to add
Use custom email subject
Subject Format
[${severity}] MetaDefender Core Notification - Processing activity reports
Quarantine
Send email notification for updated quarantine
Severity
Warning
Recipients
Input email address and press Enter to add
Use custom email subject
Subject Format
[${severity}] MetaDefender Core Notification - File has been quarantined


Database
Send email notification when a vacuum process is recommended
Severity
Critical
Recipients
Input email address and press Enter to add
Use custom email subject
Subject Format
[${severity}] MetaDefender Core Notification - Database maintenance required

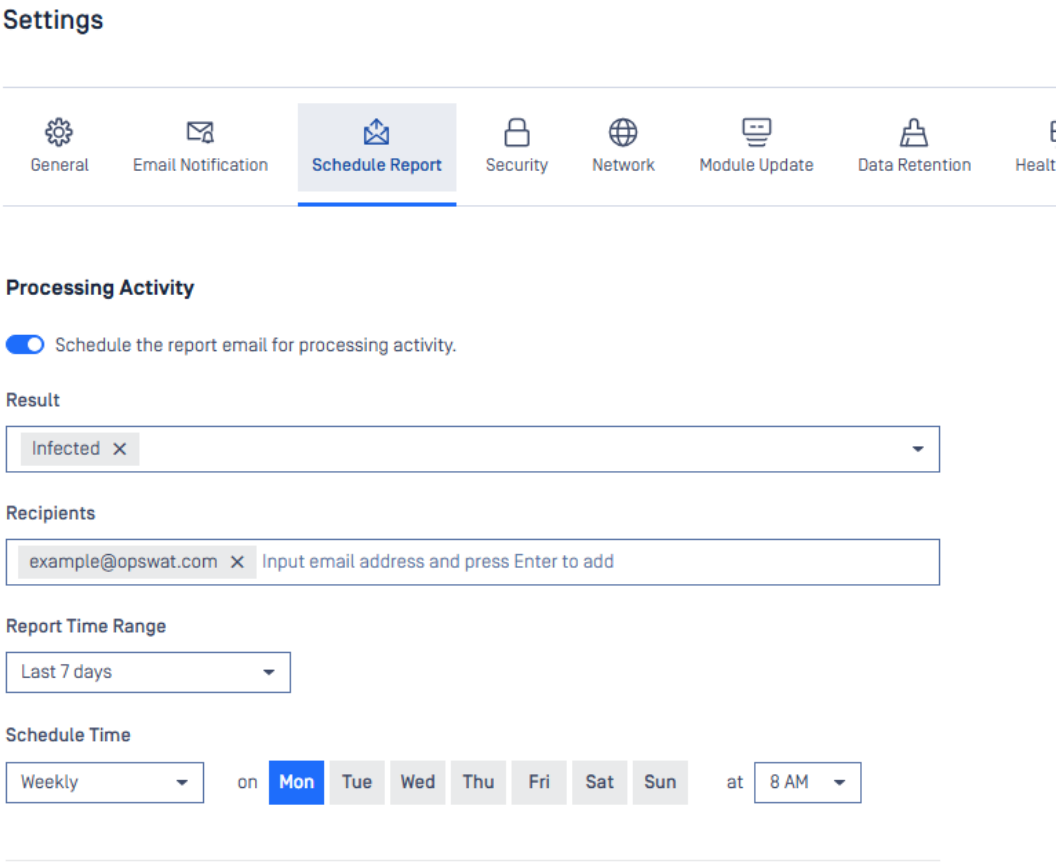
Schedule Report

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.

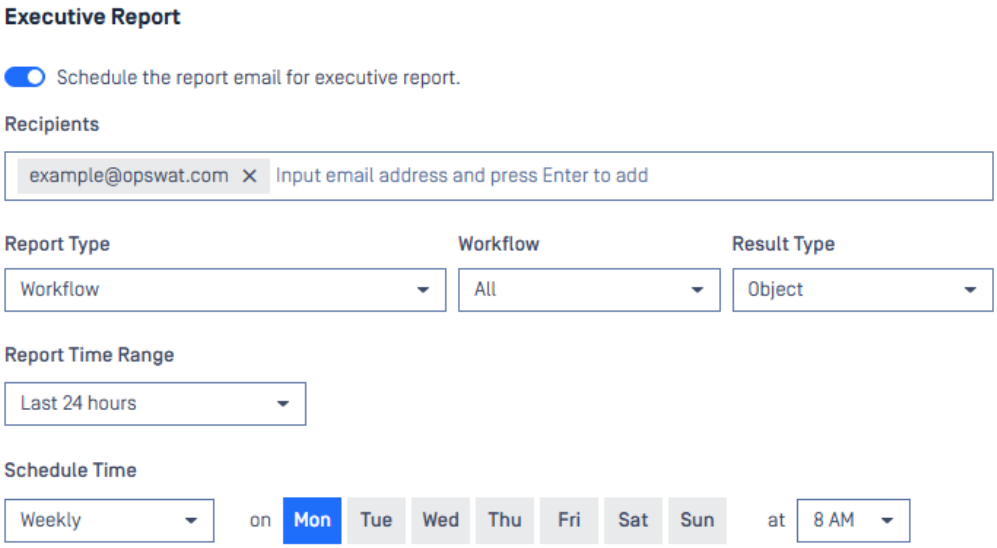
- **Result**: Allows users to specify a predefined outcome to filter data when generating reports for processing activities. This ensures that only relevant information is included in the report.
- **Report Time Range**: Defines the specific time period for which the report will collect data. This helps 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
    - Onetime - send the report only one time



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
  - Onetime - send the report only one time


Executive Report
Schedule the report email for executive report.
Recipients
example@opswat.com
Input email address and press Enter to add
Report Type
Workflow
Workflow
All
Result Type
Object
Report Time Range
Last 24 hours
Schedule Time
Weekly
on
Mon
Tue
Wed
Thu
Fri
Sat
Sun
at
8 AM

Encryption for Quarantined Files

Overview

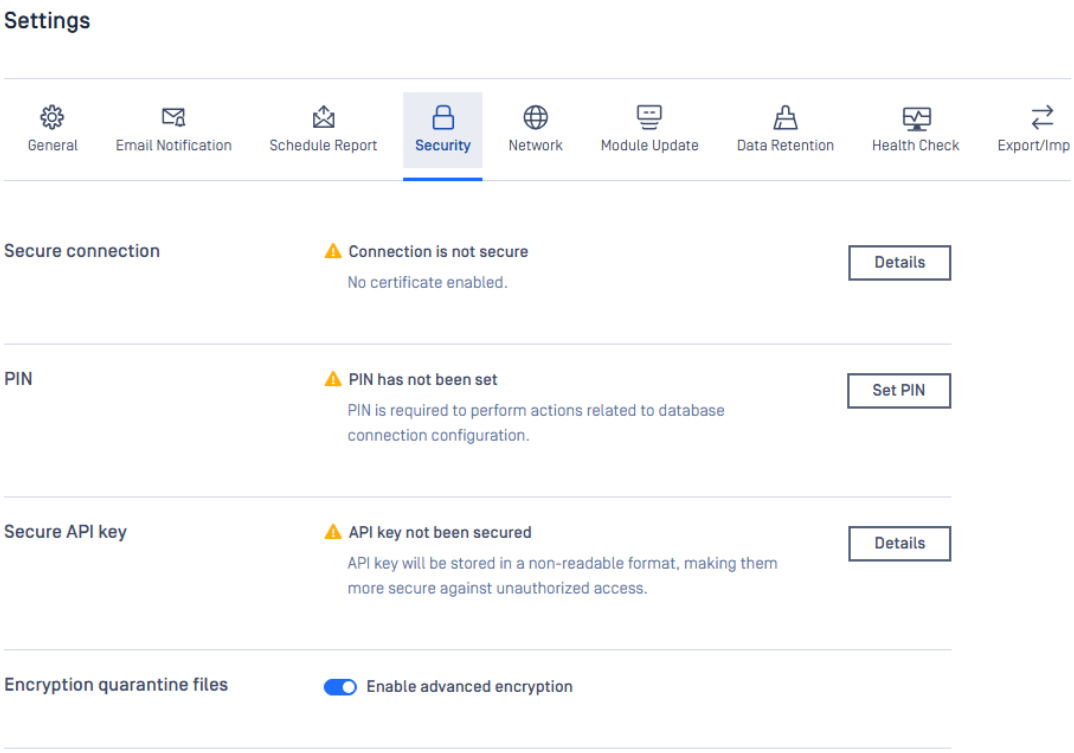
Since 5.14.0 - MetaDefender Core offers an enhanced encryption option for quarantined files, providing a significant security improvement over the standard encryption method.

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.

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

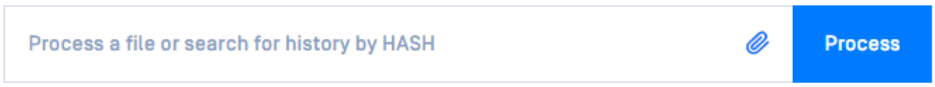
Beginning with MetaDefender Core 4.x, OPSWAT recommends using the JSON-based REST API. The available methods are documented (Full REST API documentation API).

File analysis process

1. Upload a file to scan (`POST /file`), then receive data_id from response: (Scan File API). 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 API). 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 API)

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

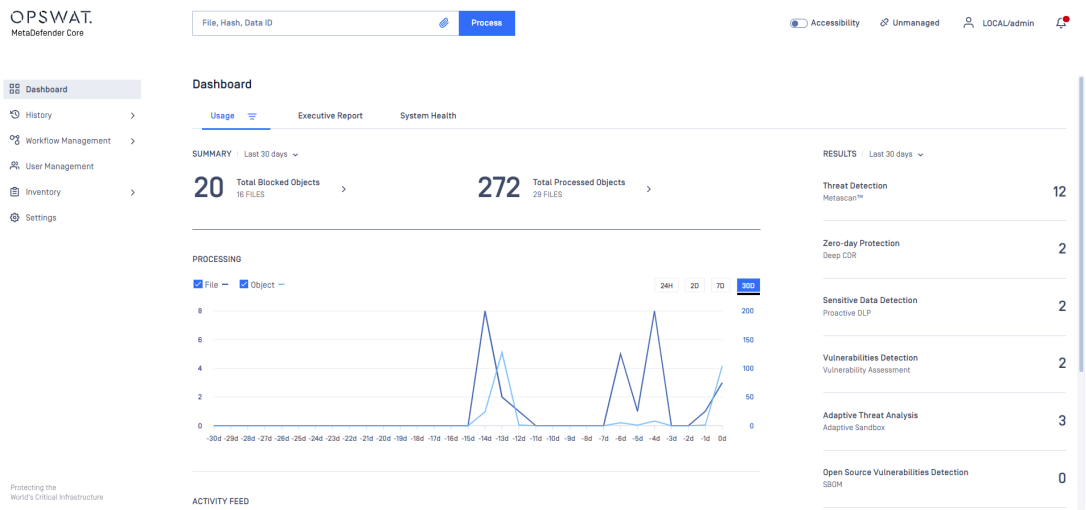
There are two option fields in the middle of the page. Next to them there is the **PROCESS FILE** button. With the leftmost option you can select between the available workflows for the public file processing. These workflows are determined by the MetaDefender Core administrators, so it is possible that 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.

Note that the default auto refresh rate of displayed information is 30 seconds.



Overview page

The Overview page shows information on

- Number of threats detected
- Number of files blocked
- Number of files sanitized
- Number of detected files having vulnerabilities
- Total number of files processed
- Number of threats analyzed by Adaptive Sandbox
- Number of countries of origin detected
- Number of scanned objects in the last 30 days
- Statistics on number of processed files in time (line chart)
- Statistics on processing results (two doughnut charts)
- Statistics on number and types of sensitive data detected (bar chart)
- Statistics on vulnerability severities (bar chart)

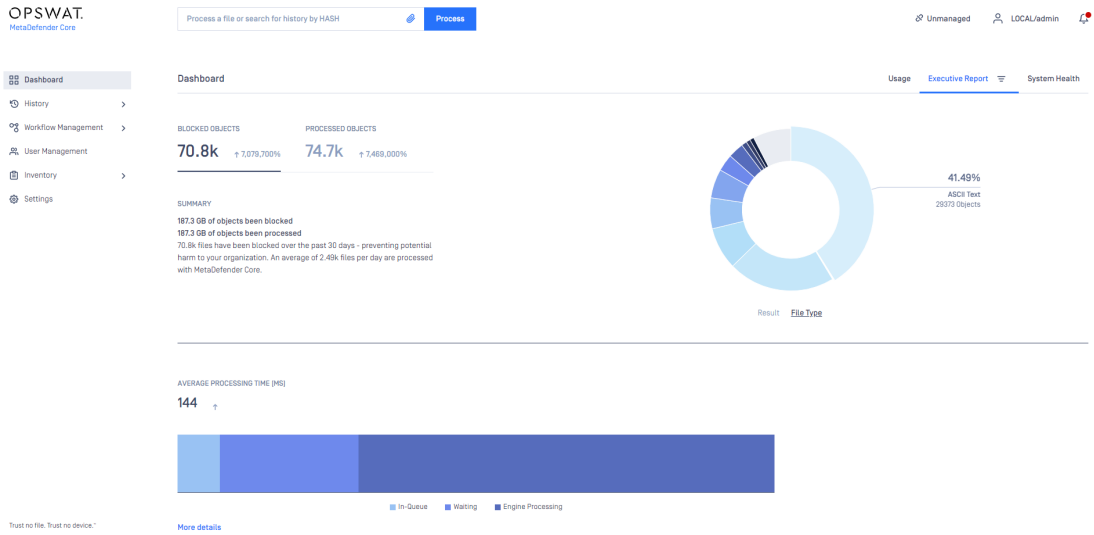
Both the default refresh rate (default is 1 minute) and the span of time displayed (24 hourss) can be changed.

Executive Report

The executives report page is to provide all needed statistics processed data.

The span of time displayed (24 hourss) can be changed.

Overview



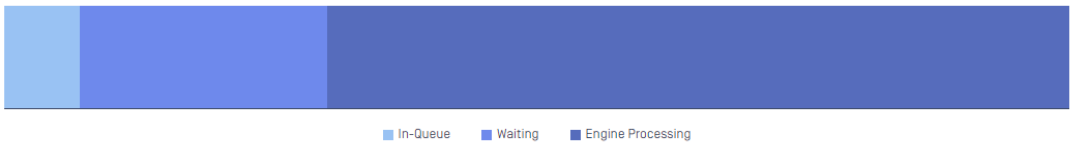
Statistics by block reason



Statistics for processing time

AVERAGE PROCESSING TIME [MS]

144



Processing Stage	Average	Max	Min
In-Queue	11	9,712	2
Waiting	36	1,073	1
Metascan™	16	716	1
AhnLab	3	431	0
Avira	2	40	1
ClamAV	16	716	0
ESET	1	358	0
Deep CDR	84	13,407	0
Proactive DLP	35	1,244	9
Filetype	20	1,187	0
Vulnerability Assessment	5	122	0

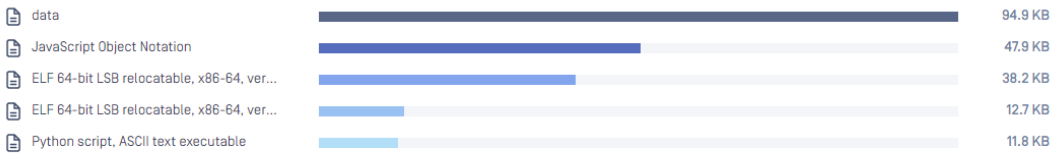
Show less

Statistics for file size

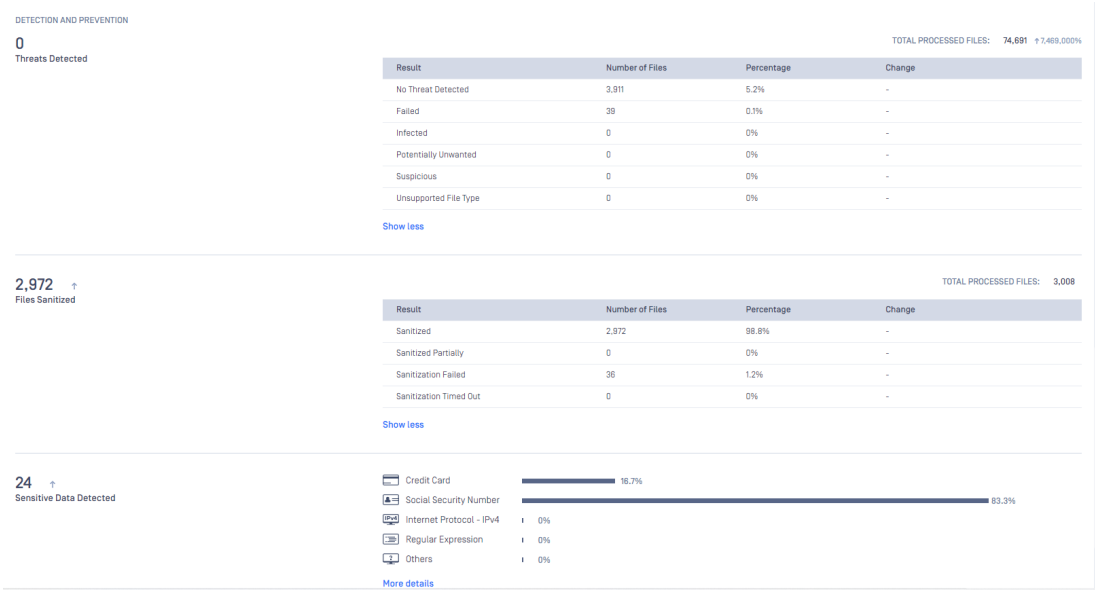
AVERAGE FILE SIZE

2.5 MB

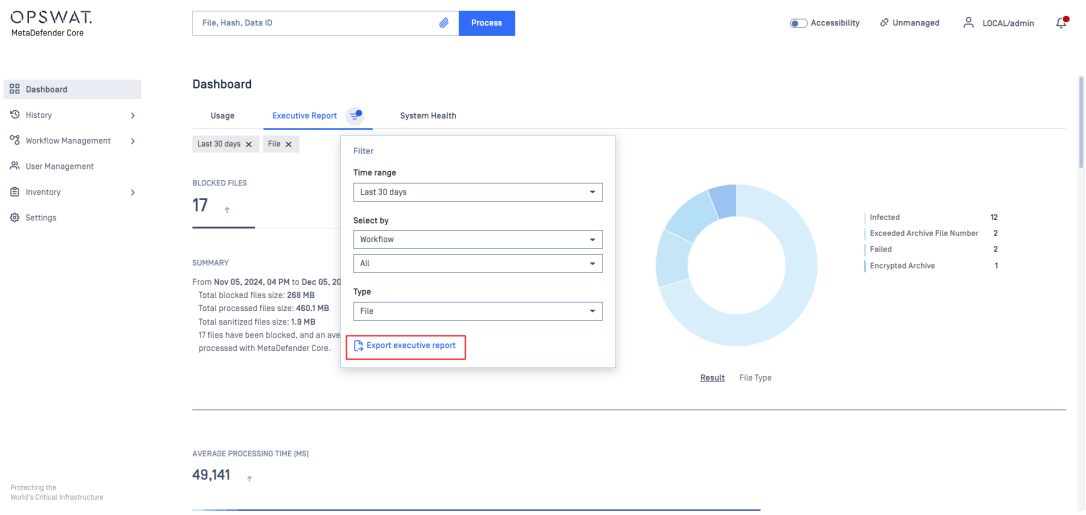
Only take into account the top 10 processed file types



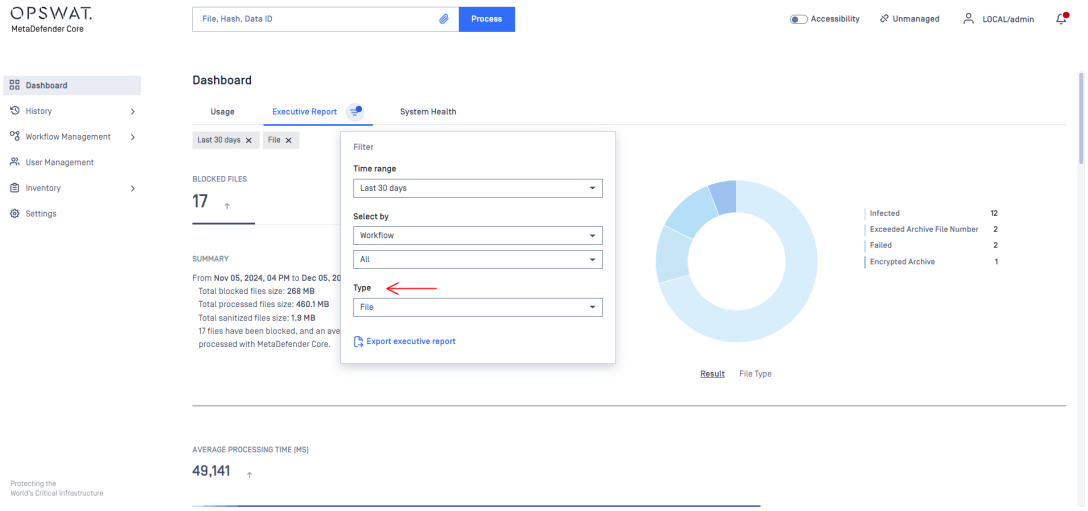
More details



You can also export the executive report to PDF file for sharing:

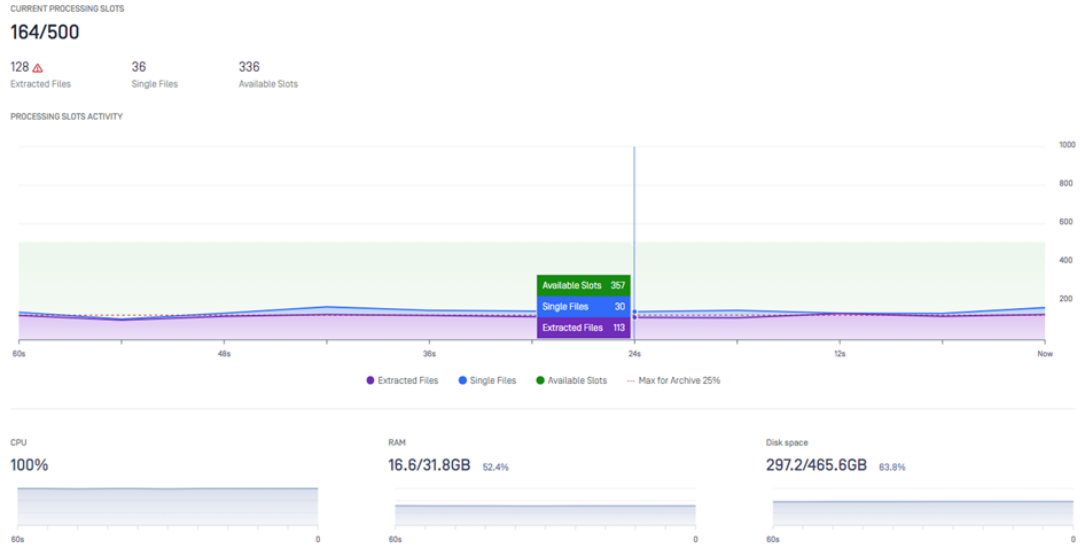


Product offers options where users can search by Workflow/Source/User and filter by File/Object:



System Health

The system health 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.

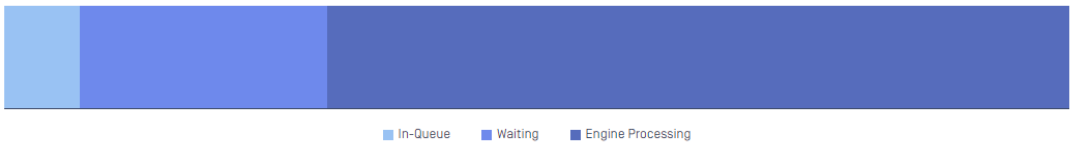
Average processing time

Overview

The Executive Report in MetaDefender Core provides statistical data on file scanning performance and processing metrics. This document explains the key terms, metrics, and values displayed in the report to help you better understand your MetaDefender Core deployment's performance.

AVERAGE PROCESSING TIME (MS)

144



Processing Stage	Average	Max	Min
In-Queue	11	9,712	2
Waiting	36	1,073	1
Metascan™	16	716	1
AhnLab	3	431	0
Avira	2	40	1
ClamAV	16	716	0
ESET	1	358	0
Deep CDR	84	13,407	0
Proactive DLP	35	1,244	9
Filetype	20	1,187	0
Vulnerability Assessment	5	122	0

Show less

Processing Stages

Definition	Explanations
Hashing	The process of calculating cryptographic hash values (such as MD5, SHA-1, SHA-256) of the scanned files.
In-Queue	The time a file spends waiting in the processing queue before active scanning begins.
Archive Extraction	The process of unpacking and analyzing compressed or archived files (such as ZIP, RAR, 7z, etc.).
Waiting (including reasons for waiting)	The waiting time when processing by Metascan™™™ engines
Nested Files	The waiting time for completely processing all nested files of an archive file

Duration 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 Time Spent in Stage ÷ Number of Files Processed

Example:

File A with Hashing time = 5

File B with Hashing time = 10

File A with Hashing time = 12

- Average Hashing = (5 + 10 + 12) / 3 = 9
- Max Hashing = 12
- Min Hashing = 5

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

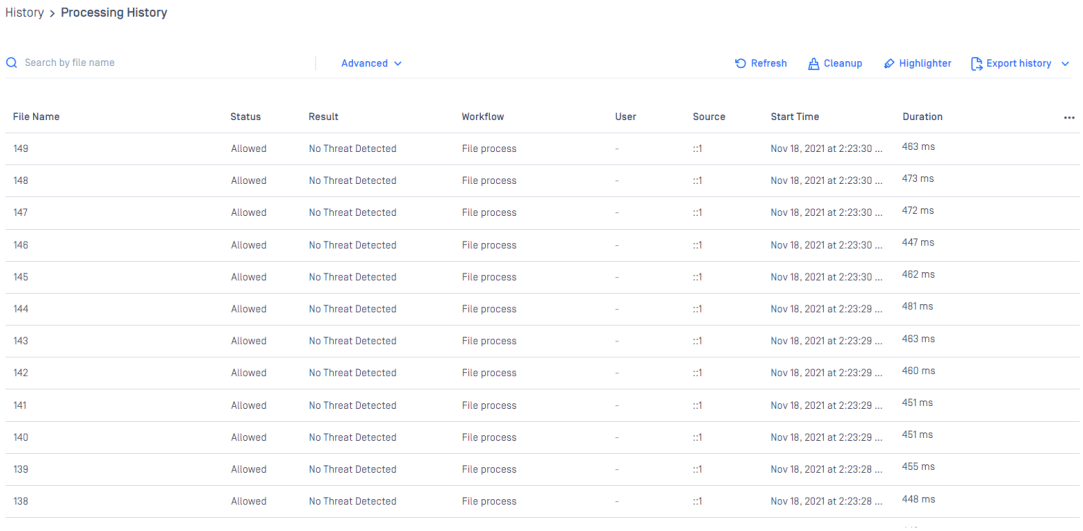
Important Notes

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.

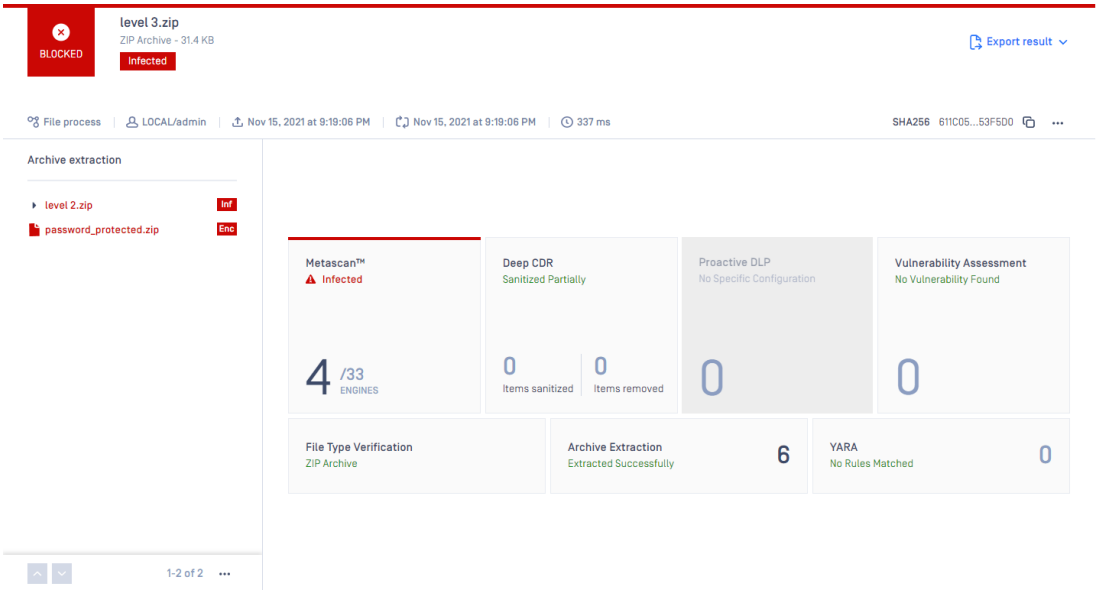
By understanding how block reasons are prioritized and counted, you can better interpret security statistics and focus remediation efforts on the most critical issues first.

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:

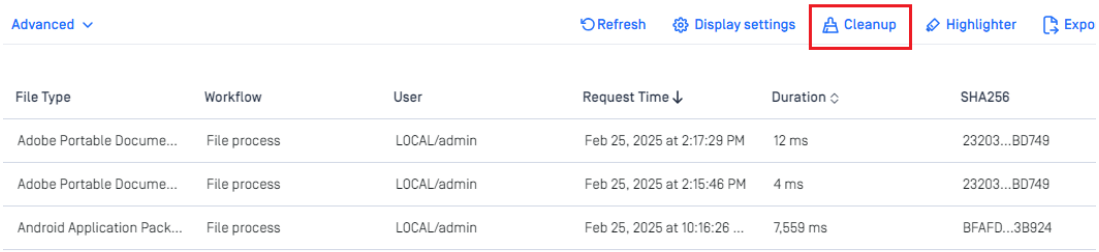
- MD5, SHA1, SHA256, SHA512 hashes.
- File name (and you can limit search result for a specific scan result, and for specific username who submitted files)
- Source (IP address)
- User name

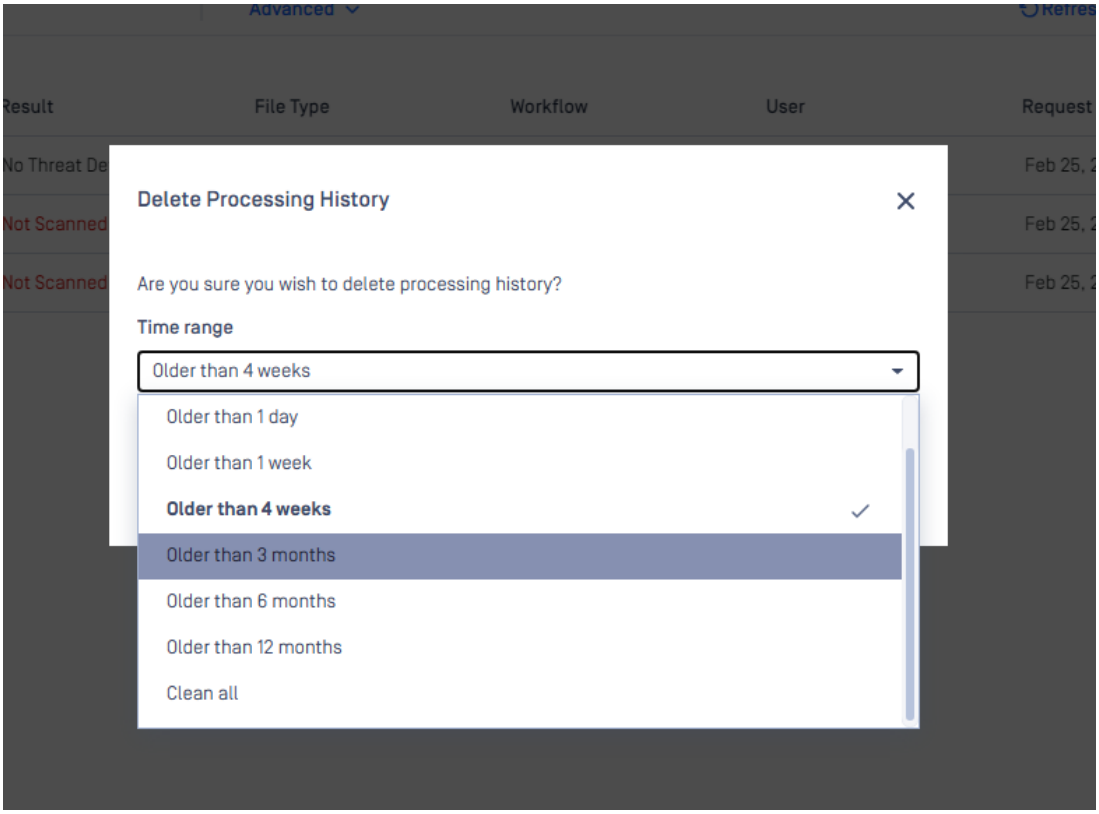
There is an option to export scan history in CSV or STIX format. For the export, the Result filter can also be applied.

- 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.

Inventory Management

Metadefender Core displays detailed information on scan nodes and the status of engine updates including anti-malware engines, archive engines, etc.

Certificates

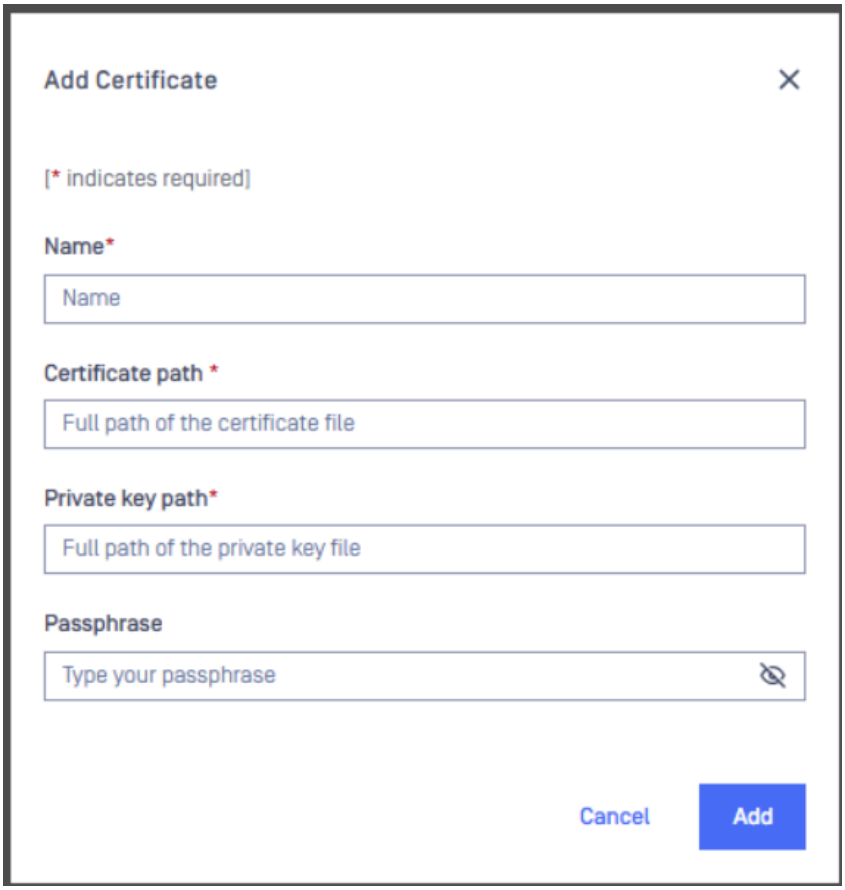
On this page, path to certificates and private keys for signing scan batches or HTTPS configuration can be given.

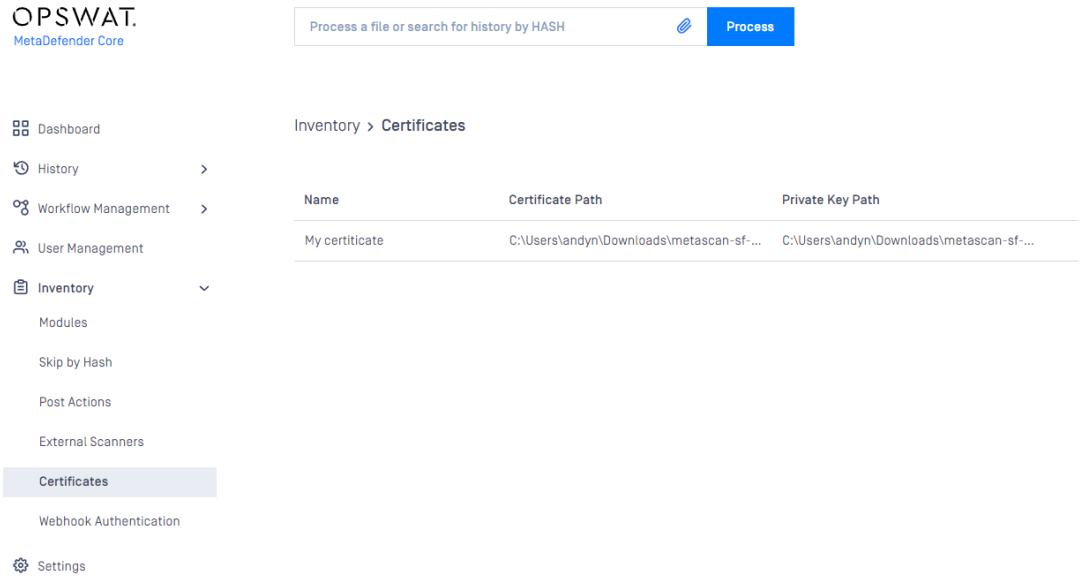
> **Note**
>
> - Certificate should be in a Base64-encoded X.509 certificate file (.crt, .cer) format.
> - Private key should be a privacy-enhanced electronic mail file (.pem) format and it should not be locked by password.

Adding certificate-key pair to the inventory

1. Go to **Inventory→Certificates** page
2. Click on **Add new certificate** button
3. Fill the **Add new certificate** form by giving a name, a path to certificate file and a path to the corresponding key file
4. Click **Add** button

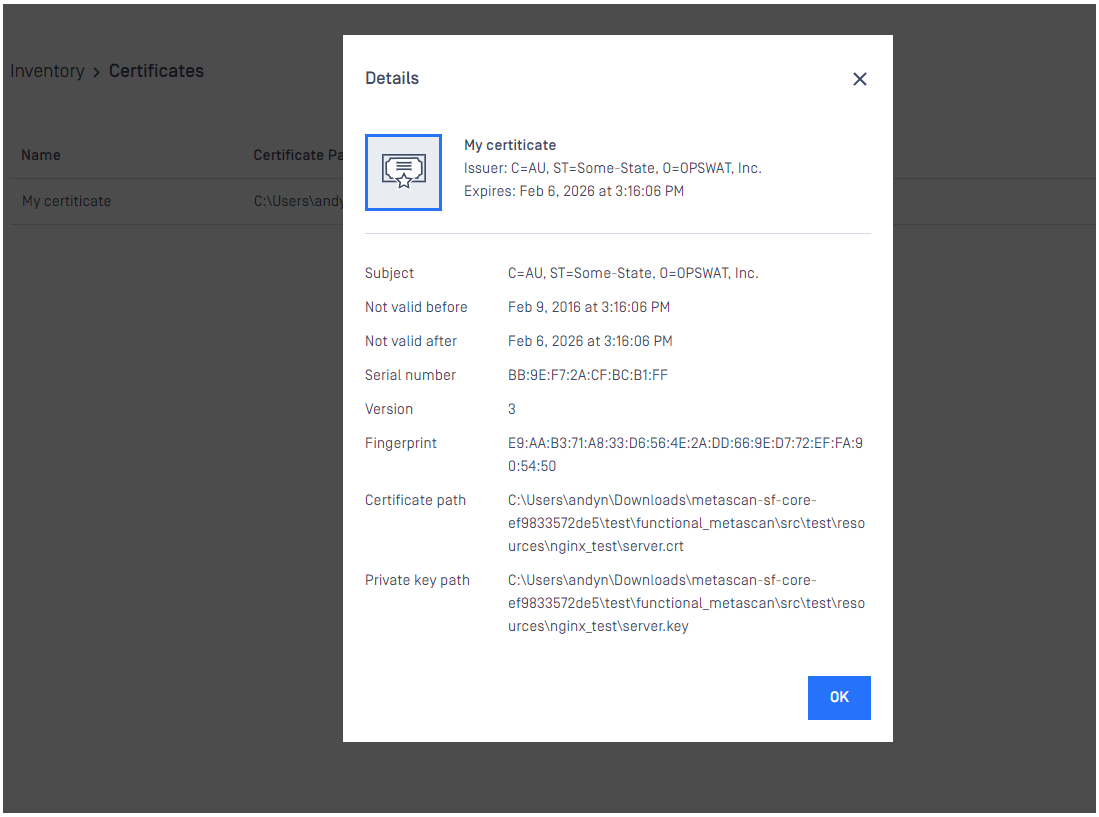
- **Name:** Certificate name
- **Certificate Path:** Absolute path to the certificate file
- **Private Key Path:** Absolute path to the private key file
- **Passphrase:** Passphrase for the private key (not a file path)





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 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

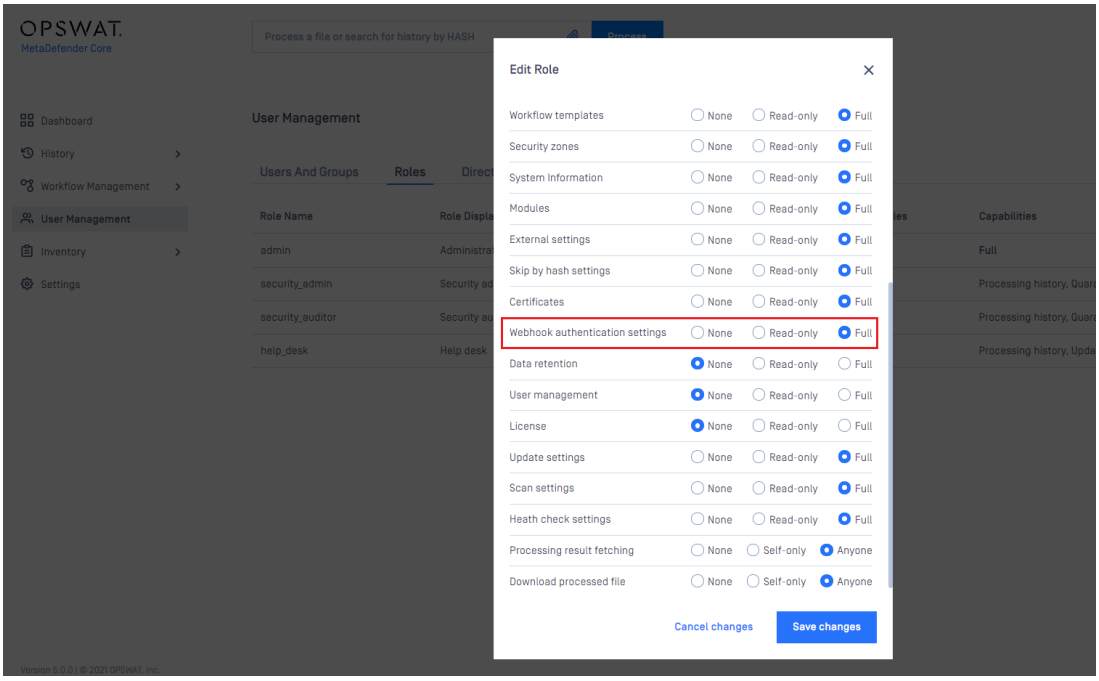
While leveraging webhook on MetaDefender Core, the callback connection stays at un-secured by default. To harden the security, MetaDefender Core supports a mutual authentication mode between the server (MetaDefender Core) and client (where the callback is sent to), and prevents the man-in-the-middle 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:

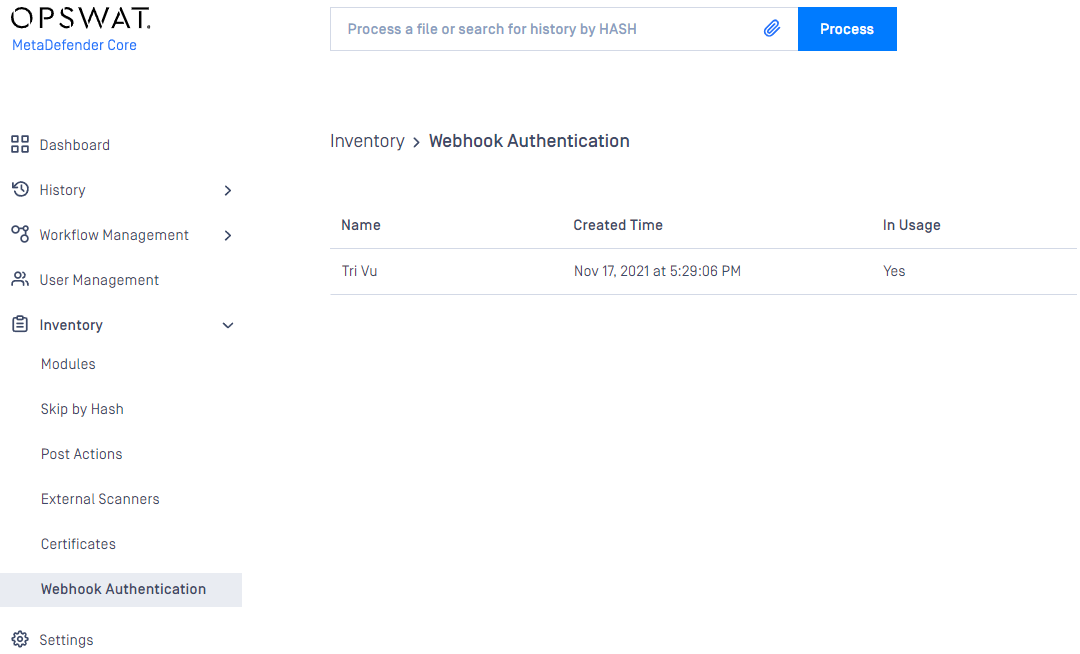
- MetaDefender Core to calculate the signature of scan result using the selected private key.
- MetaDefender Core to include the calculated signature into the callback response header before sending back to client.
- Client side (where the callback is sent to) to use the stored public key to:
  - Decrypt 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.



Create a new certificate-key pair for webhook authentication

1. Go to **Inventory→Webhook Authentication** page
2. Click on **Add certificate** button

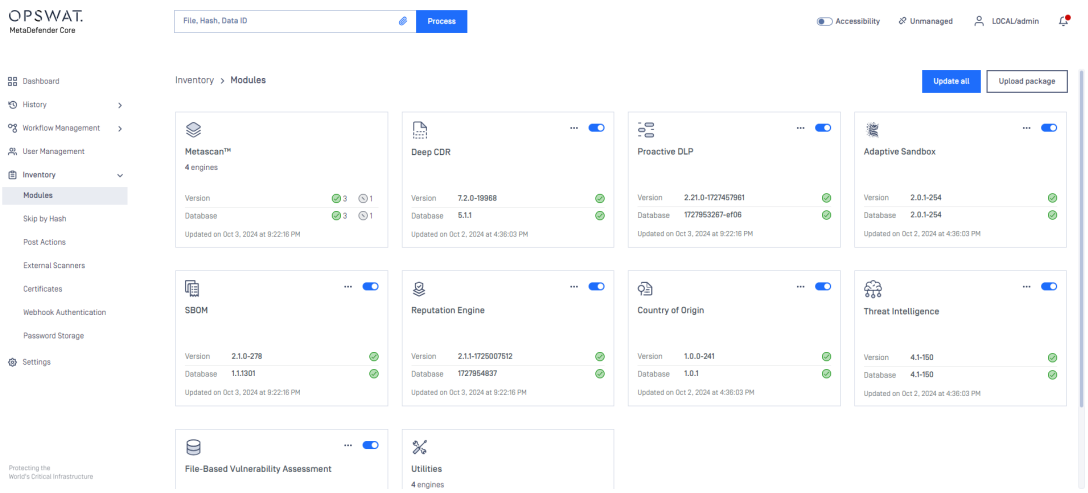


Modules

Engine type details

Under the **Modules** menu all the installed engines are listed with their details such as

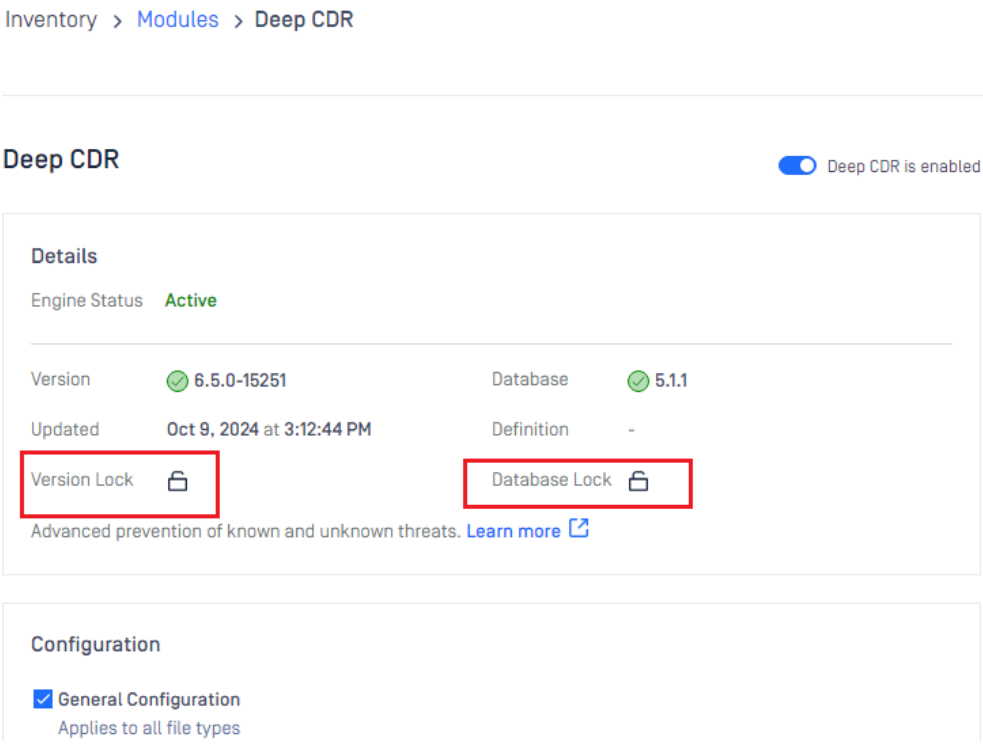
- Type of engine. Possible types are
  - Anti-malware engine
  - Archive engine
  - Proactive DLP engine
  - Deep CDR engine
  - Adaptive Sandbox engine
  - Software Bill of Materials engine
  - Filetype detection engine
  - Reputation engine
  - Country of Origin engine
  - Utility engine
  - Vulnerability 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 currently 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)

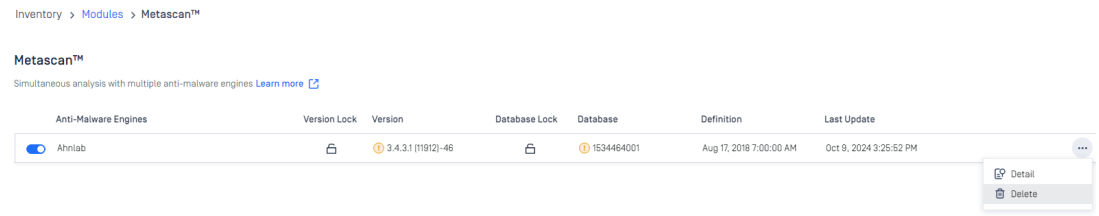
Engine and its database can be locked (previous called pin and unpin) to prevent it from being applied new updates when you allow auto update on Core.

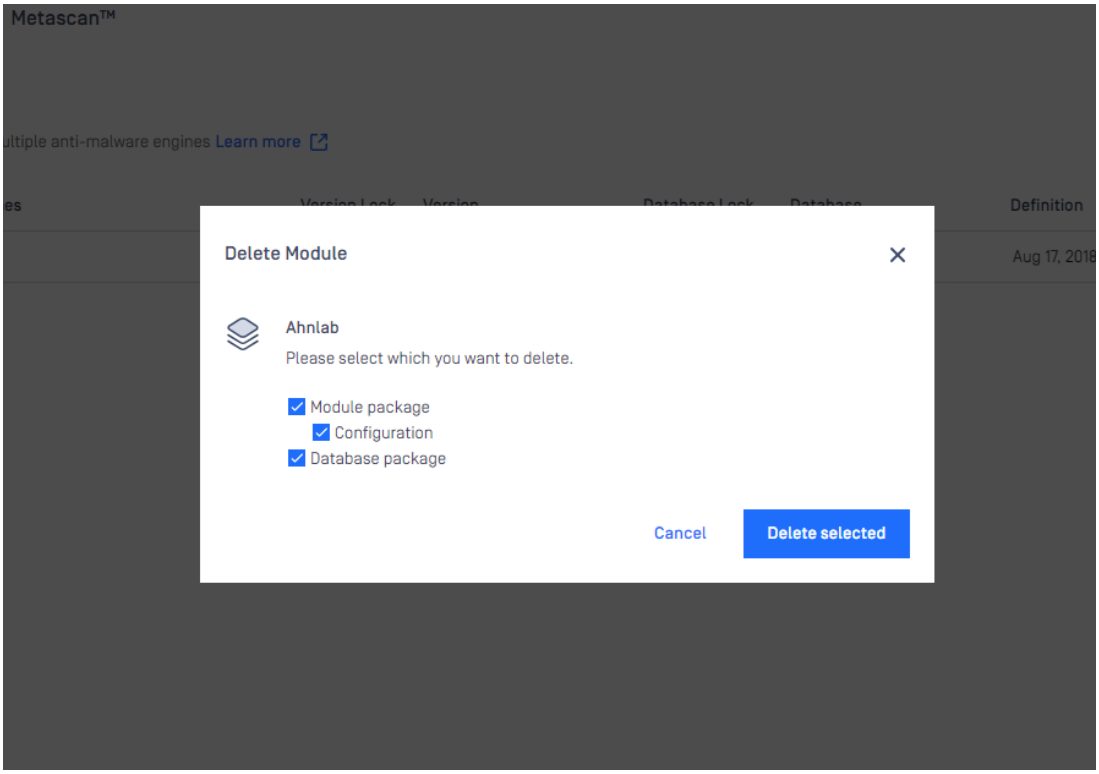


> **Note**
>
> - You can lock engine and database individually on same engine
> - When locked, that means no auto update can be applied on that part (engine / database), even when user triggers "Update All" button

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.

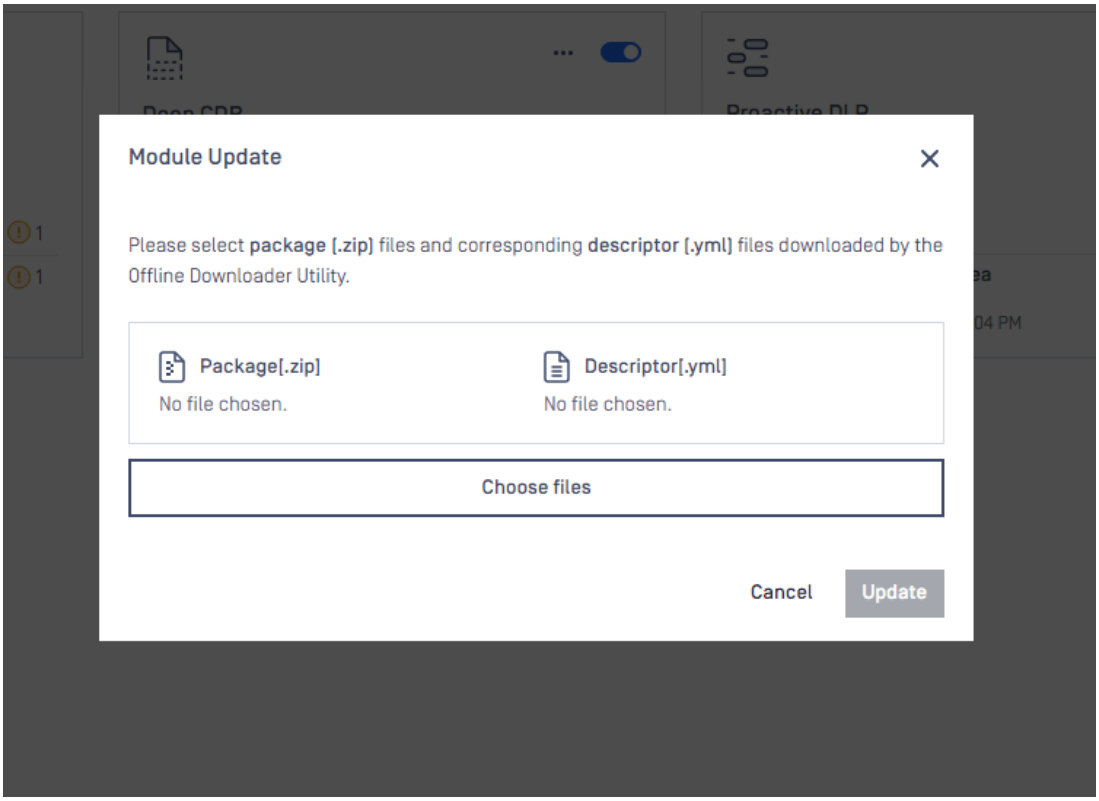




Manual updates

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.



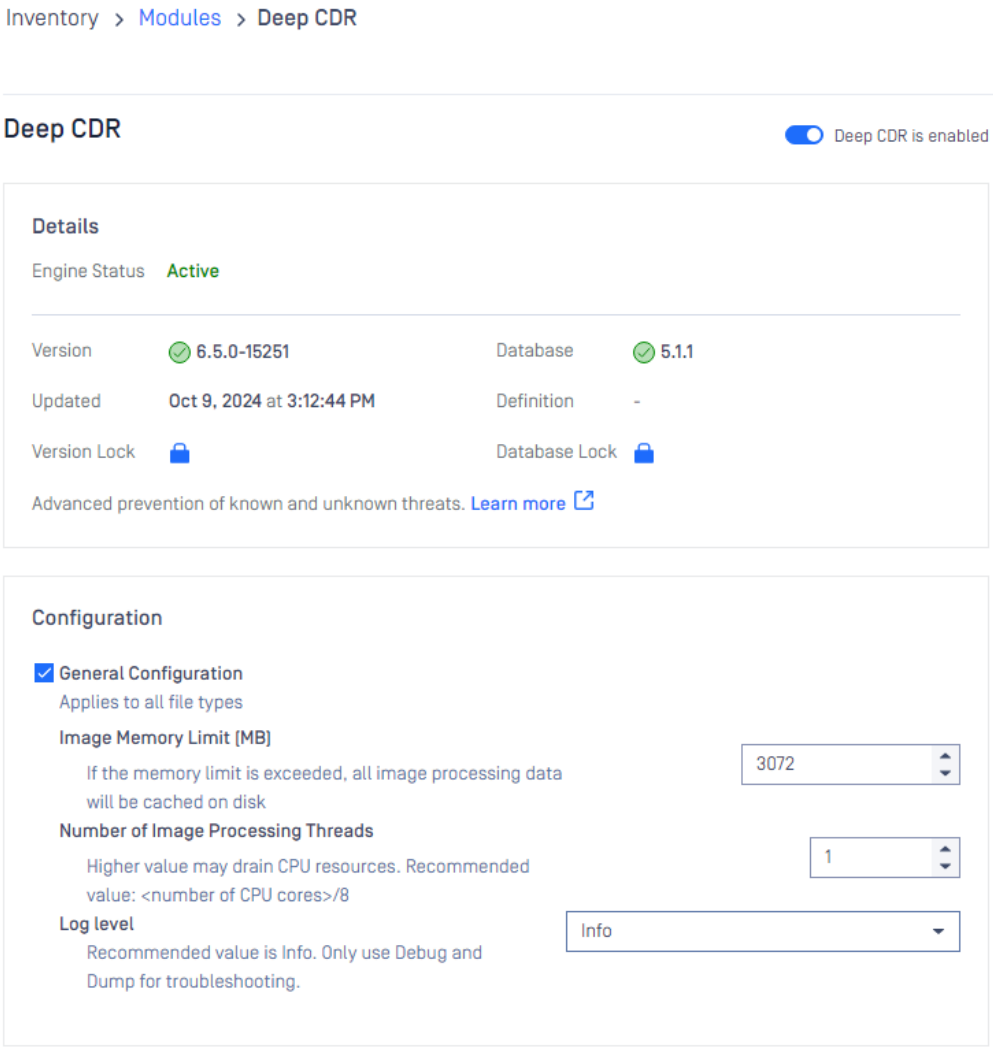
Upload packages

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.

Configuring engine advanced settings

Some engines advanced settings can be configured.



Choose your desired configurations and click **Save changes**, then Close.

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

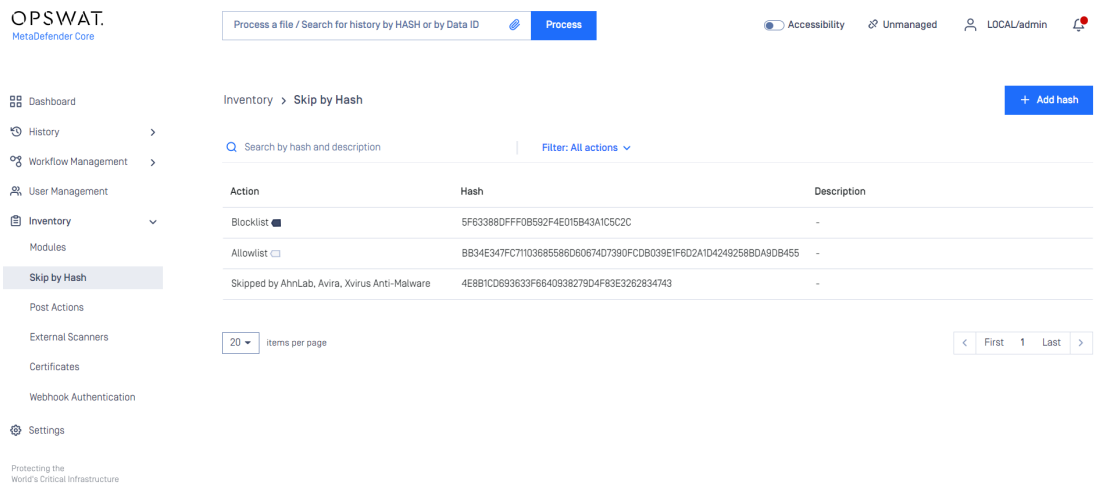
Files whose hashes listed here will be globally allowlisted, so they won't be processed in any workflow and will be allowed.

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.

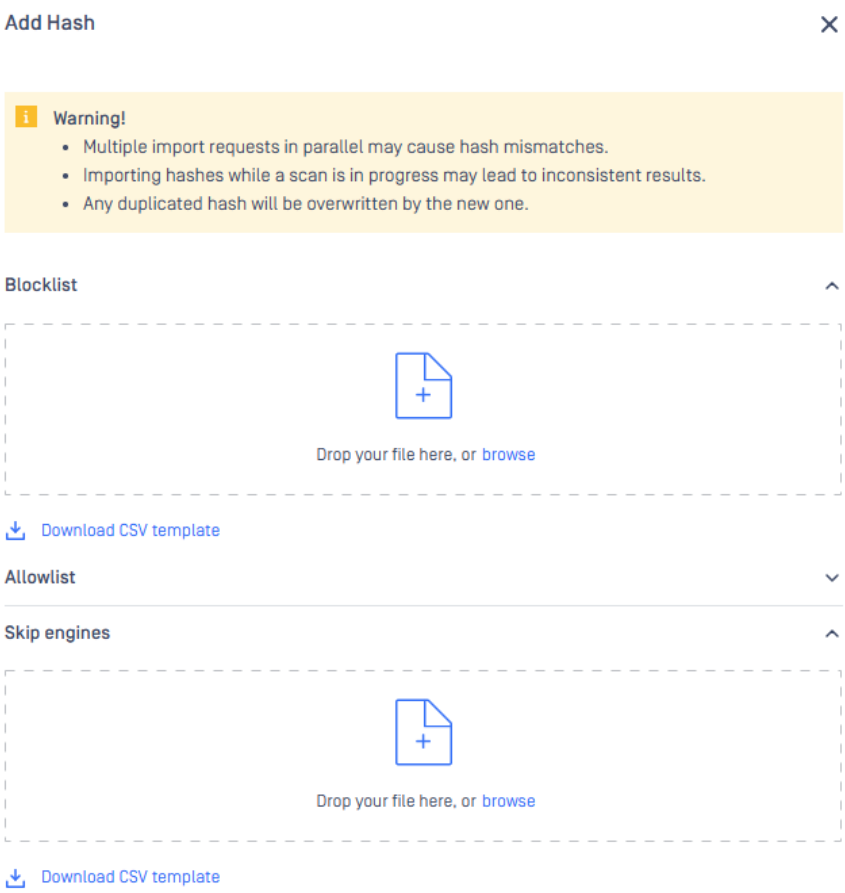


Adding a hash to skip list

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.

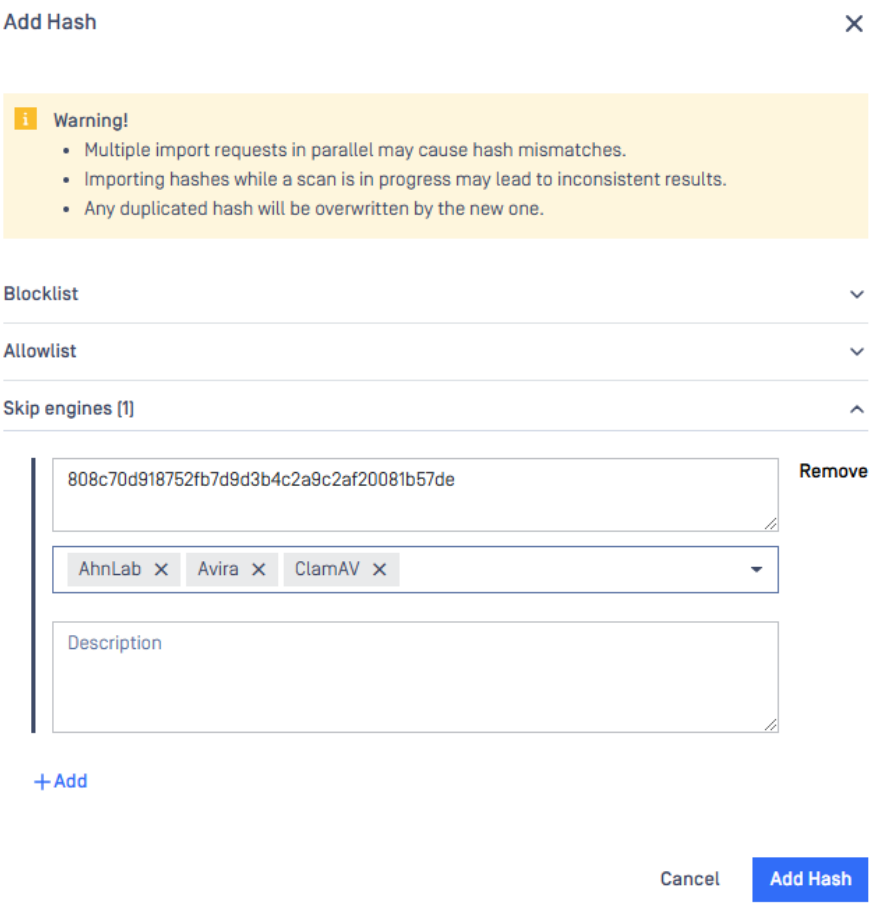


- Manually Add:
  - Allow you to input hashes and information manually.
- Import CSV:
  - Enable you to prepare a list of hashes and upload all at once to the application.
  - A CSV template is downloadable for easy content input.



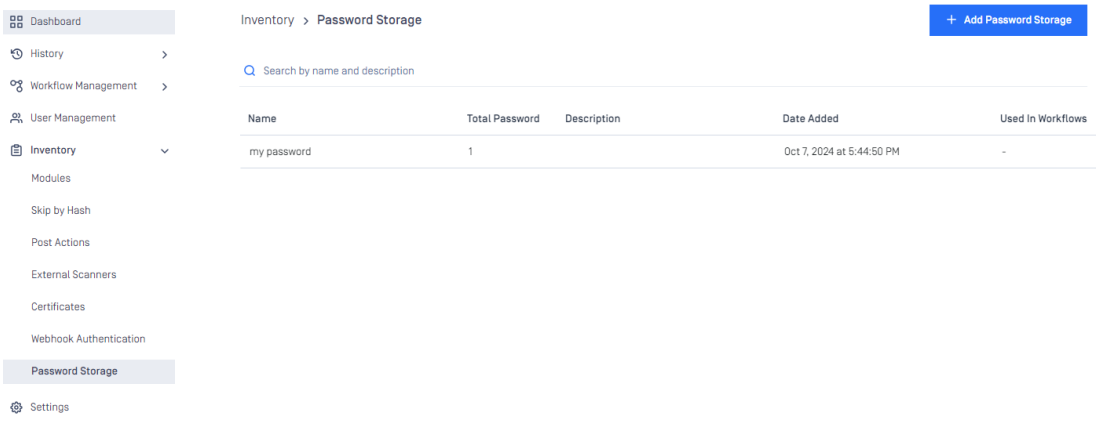
Select your desired way, then "Add hash" page appears.

After giving the necessary information, click on the **Add** button.



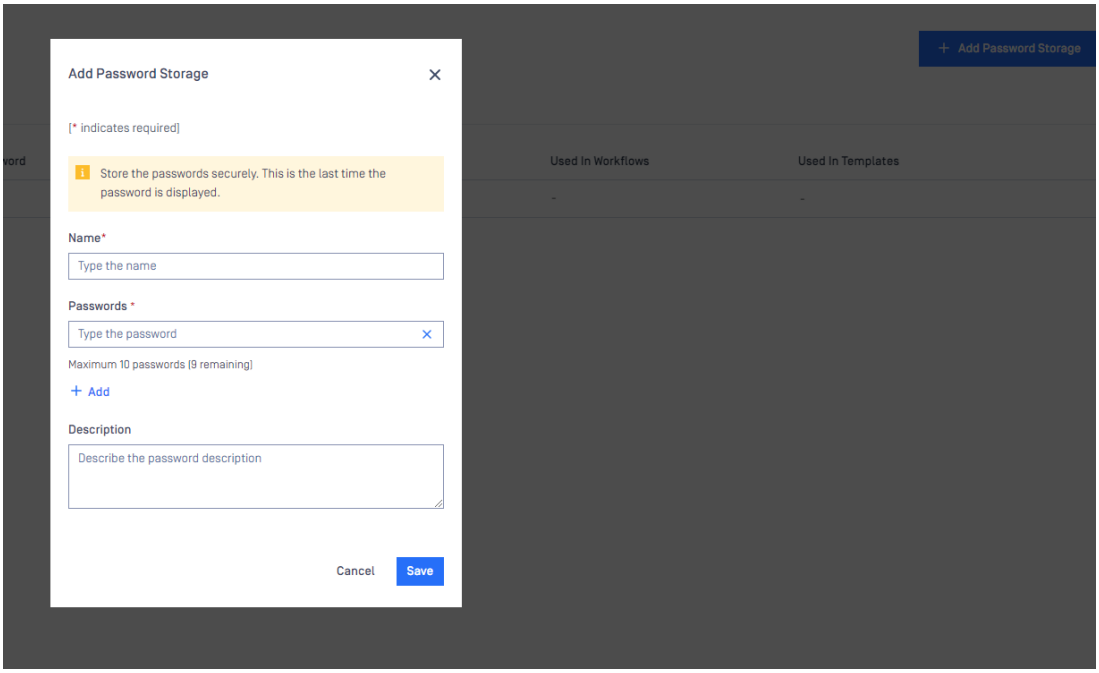
Password Storage

Users can define and manage lists of passwords used for file decryption, streamlining the process when handling encrypted or password-protected files.

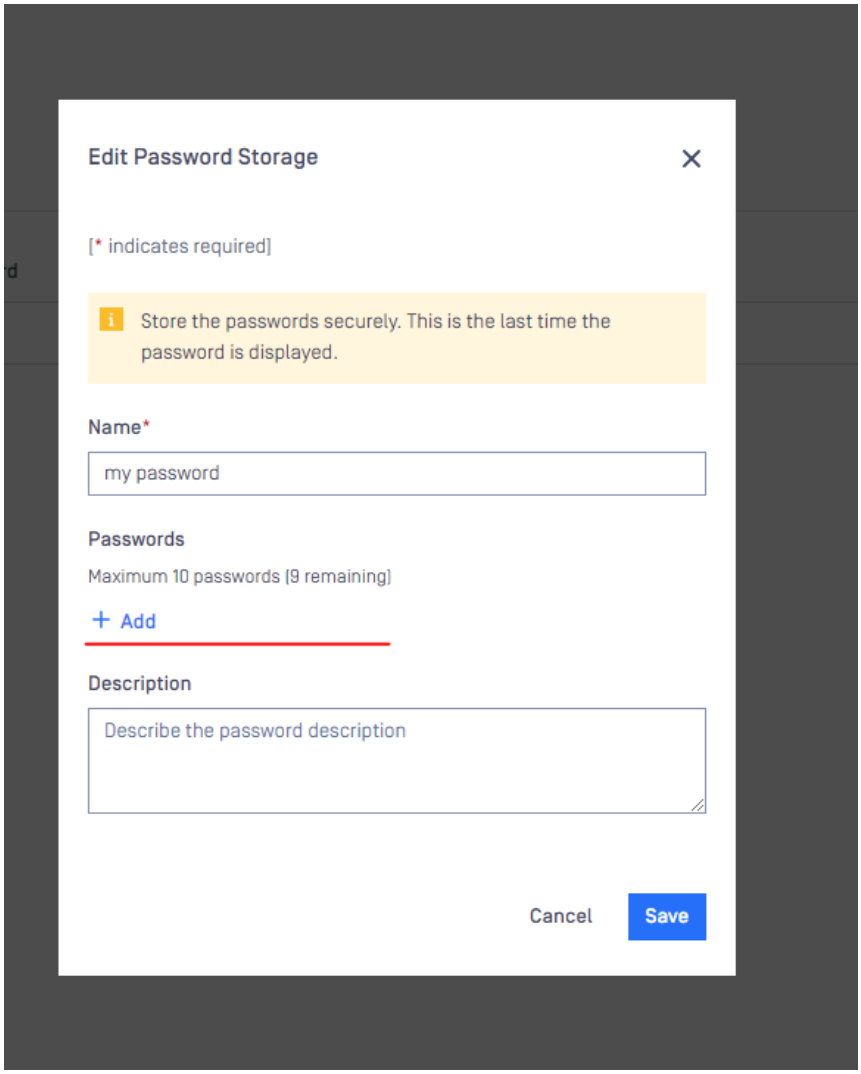


We can add up to 50 password storage and each password storage can contains up to 10 passwords

Click the button Add Password Storage for adding new password storage



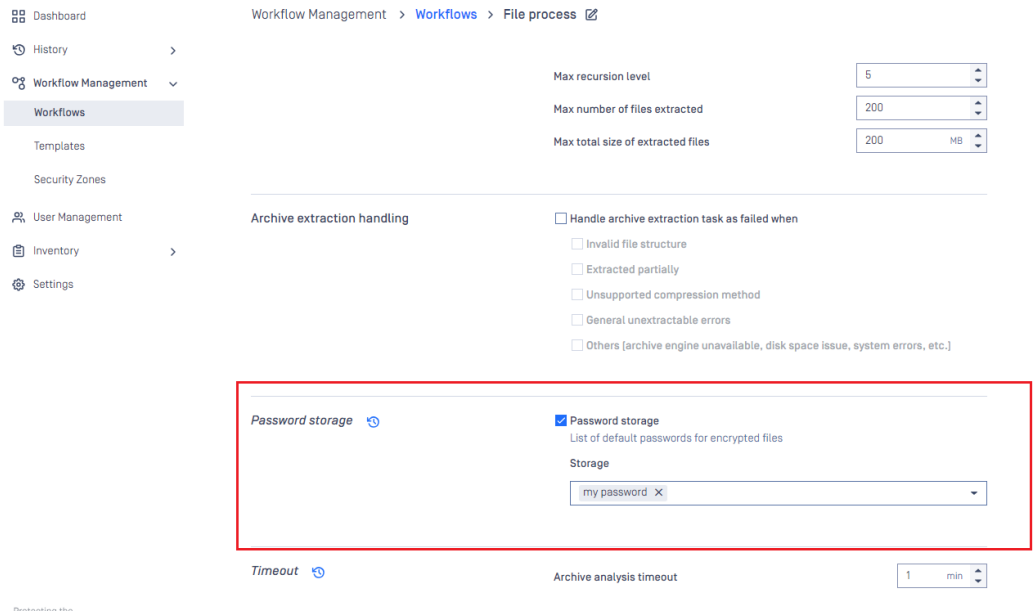
We also can add more password in the list for later, click to any password storage to modify



We can only add more password to the list, we cannot modify the old passwords or remove it from the list. To remove the password lists, we need remove the whole password storage.

After created the password storages, we need to go to workflow rule setting to select the password storages that we want to using for a specific workflow rule.

Workflow Management > Workflows > Select a workflow rule > Archive tab > Scroll down and check Password storage


Dashboard
History
Workflow Management
Workflows
Templates
Security Zones
User Management
Inventory
Settings
Workflow Management > Workflows > File process
Max recursion level
5
Max number of files extracted
200
Max total size of extracted files
200
MB
Archive extraction handling
Handle archive extraction task as failed when
Invalid file structure
Extracted partially
Unsupported compression method
General unextractable errors
Others (archive engine unavailable, disk space issue, system errors, etc.)
Password storage
Password storage
List of default passwords for encrypted files
Storage
my password
Timeout
Archive analysis timeout
1
min

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

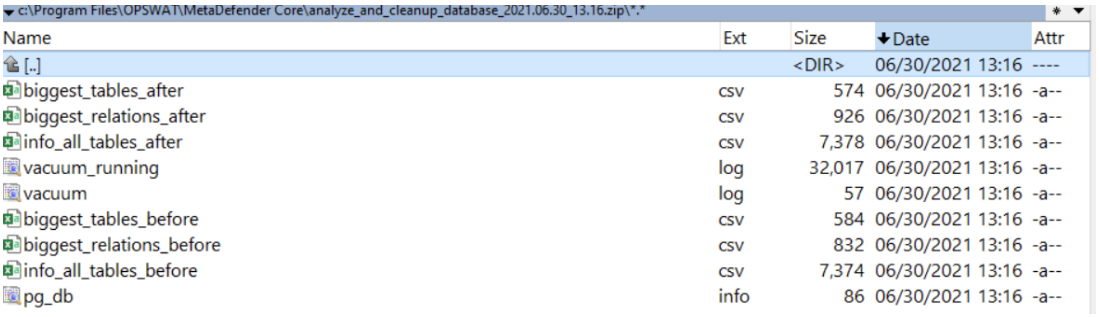
Windows

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`

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.

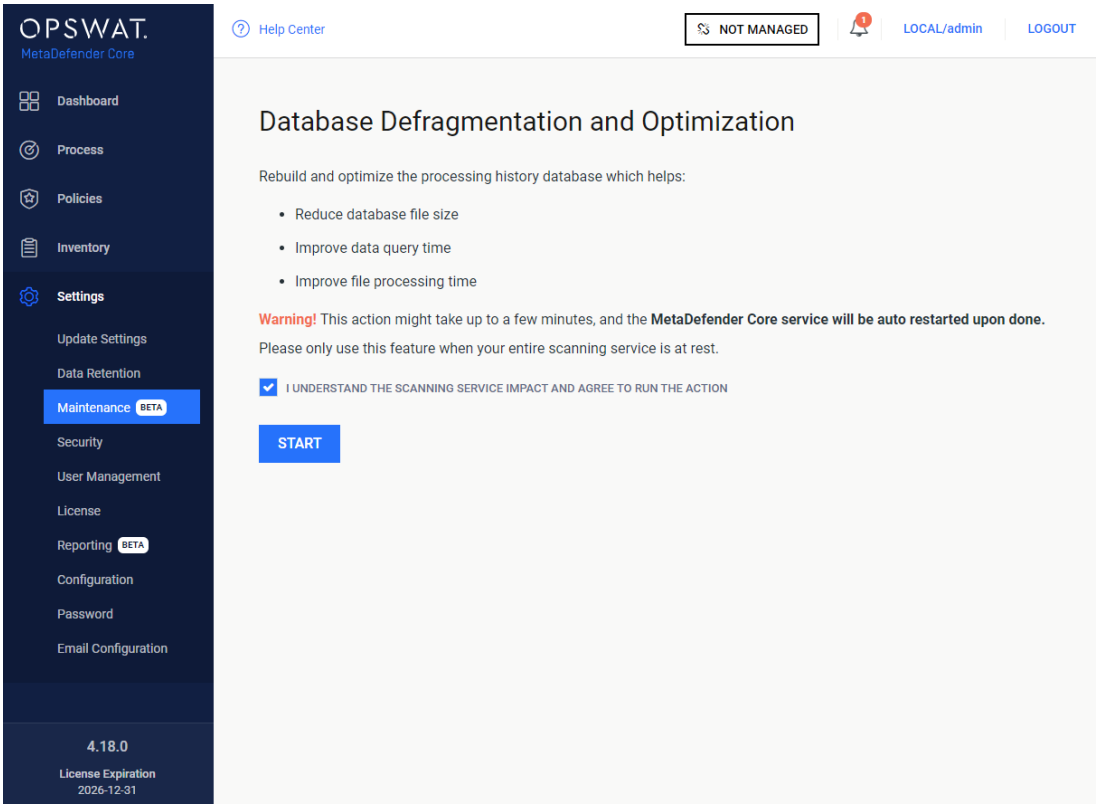
Starting with version 4.19.0, the product has changed to using PostgreSQL database. The applicable vacuum instructions could be found at MetaDefender Core 4.19.0 or above

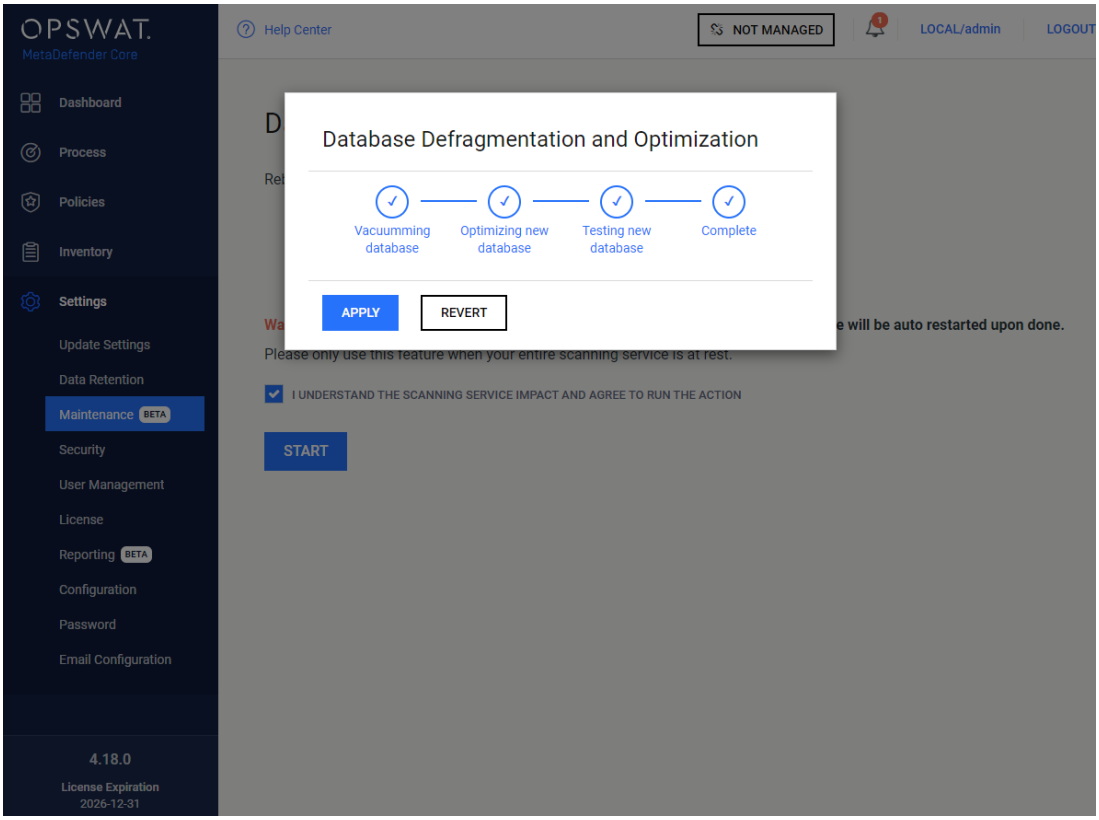
When your scan database grows big, it might cause performance degradation (e.g. timeout on client requests).

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.

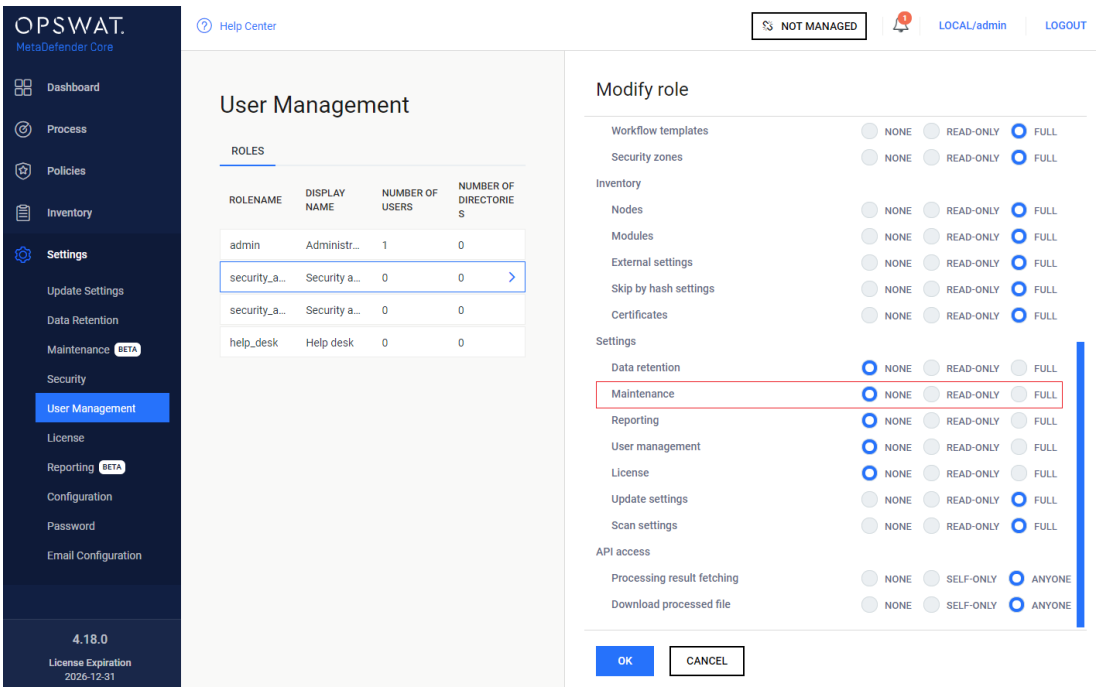
Caution: Please only use this feature when your entire scanning service is at rest.





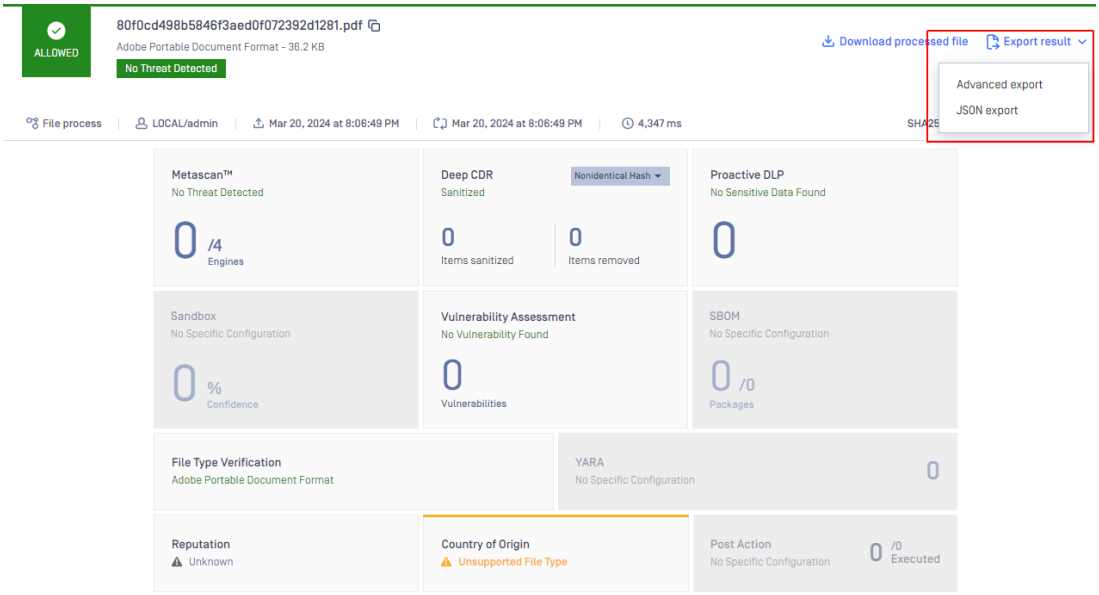
Hitting “APPLY” to apply all changes and restart MetaDefender Core service (or “REVERT” to roll back changes).

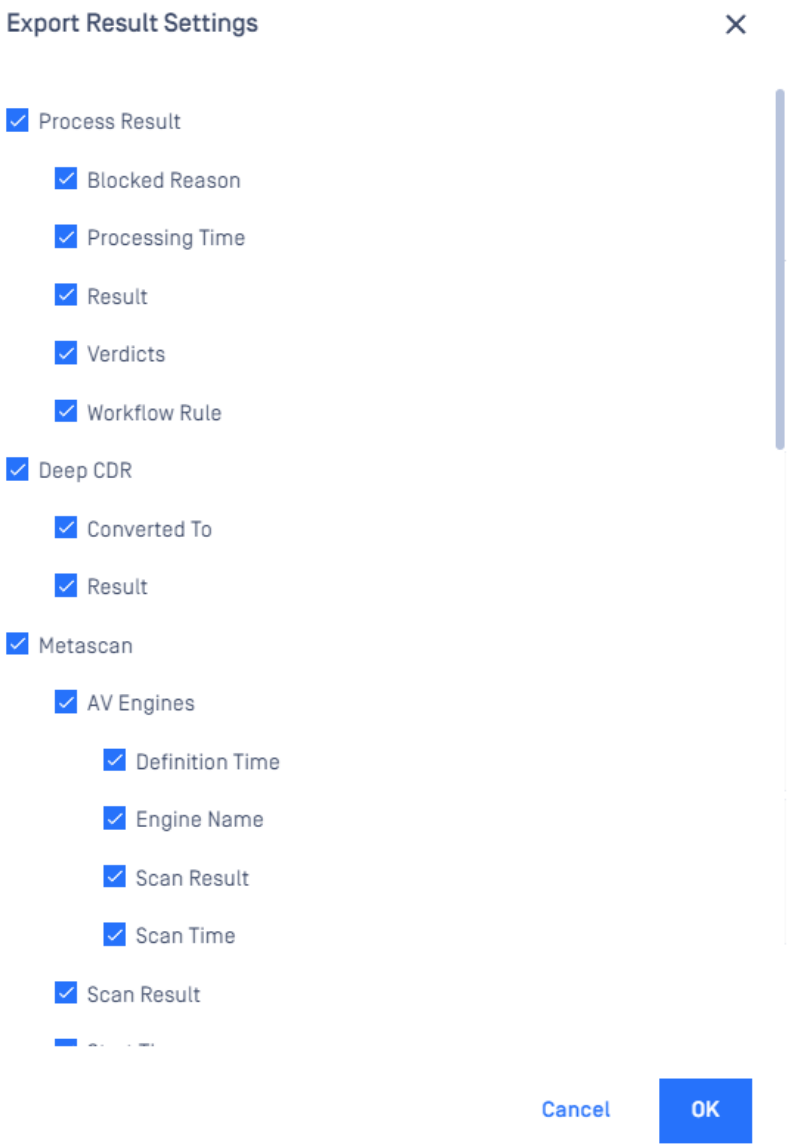
By default, only the “admin” user role can see this Maintenance page and make the changes, but you can configure to grant access to other user roles if desired.



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

Data ID	3a03101e00a54b9f9fdbabd3a95d45b7

File Details

Display Name	80f0cd498b5846f3aed0f072392d1281.pdf
Size	36196 bytes
Type	Adobe Portable Document Format
Type ID	PDF
MIME Type	application/pdf
MD5	804c0d8d9de4fd6c8f1077f207205ca1
SHA1	6f27f1e2c6f6d8467ee4daf63b2ec1b8a2614a93
SHA256	2648f04071a4f4c62c204b5715438885631b225be06c9551cbe553ec7fa9c540
Receive Data Timestamp	2024-03-20 20:06:49 UTC+07:00
Upload Timestamp	2024-03-20 20:06:49 UTC+07:00
Upload Time	6

Process Result

Result	Allowed
Processing Time	4347 ms
Verdicts	No Threat Detected
Workflow Rule	File process
Hash Time	1

Reputation

Result	Reputation Unknown

Country of Origin

Result	COO Unsupported File Type

Deep CDR

Result	Sanitized
Converted To	pdf

Metascan

Total AV Engines	4
Start Time	2024-03-20 20:06:49 UTC+07:00
Scan Result	No Threat Detected

AV Engines

Engine Name	Scan Result	Definition Time	Scan Time
ESET	No Threat Detected	2024-03-20 15:38:19	3 ms
Avira	No Threat Detected	2024-03-20 16:59:00	22 ms
AhnLab	No Threat Detected	2024-03-21 07:00:00	27 ms
ClamAV	No Threat Detected	2024-03-19 22:25:53	31 ms

Proactive DLP

Result	No Sensitive Data Found

Redaction

Result	Not Redacted

Metadata Removal

Result	Not Removed

Watermark

Result	Not Added

Crop Embedded Images

Result	Not Cropped

Sandbox

Result	Sandbox Not Configured

SBOM

Result	SBOM Not Configured

Vulnerability

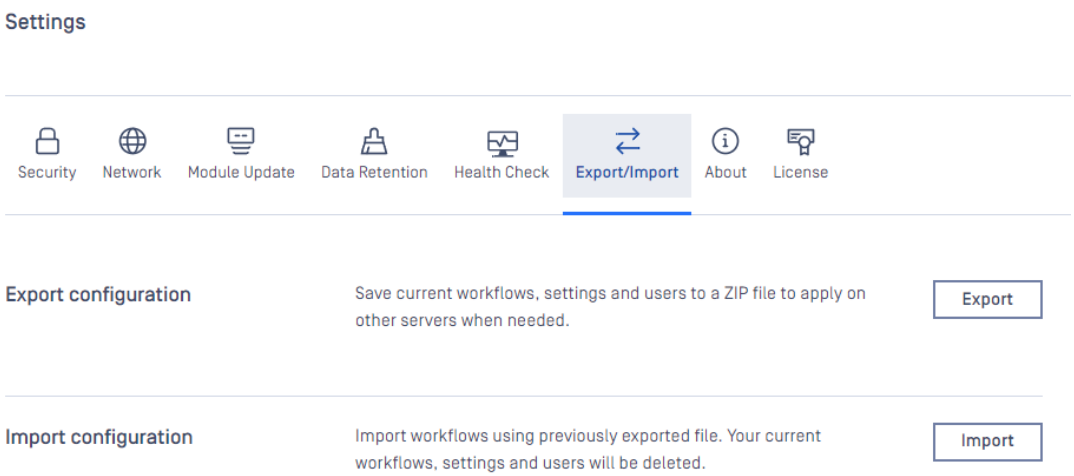
Result	No vulnerability found

Yara

Result	Yara Not Configured

Import/Export configuration

MetaDefender Core current configuration can be exported or a new one can be imported under the **Settings > Export/Import** page.

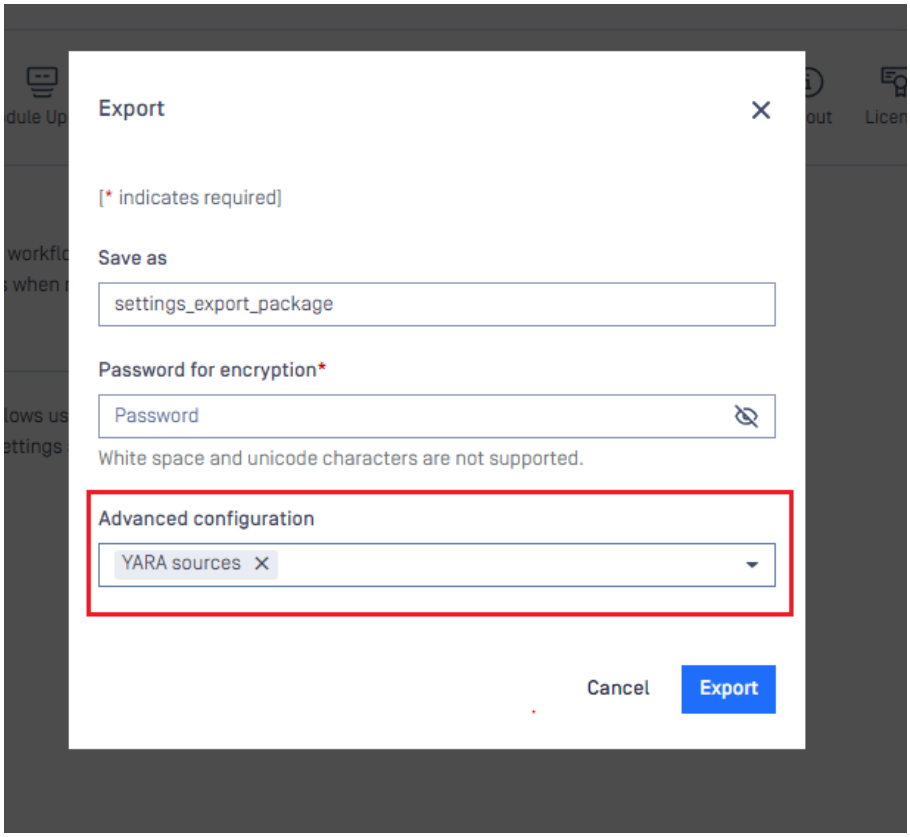


Export

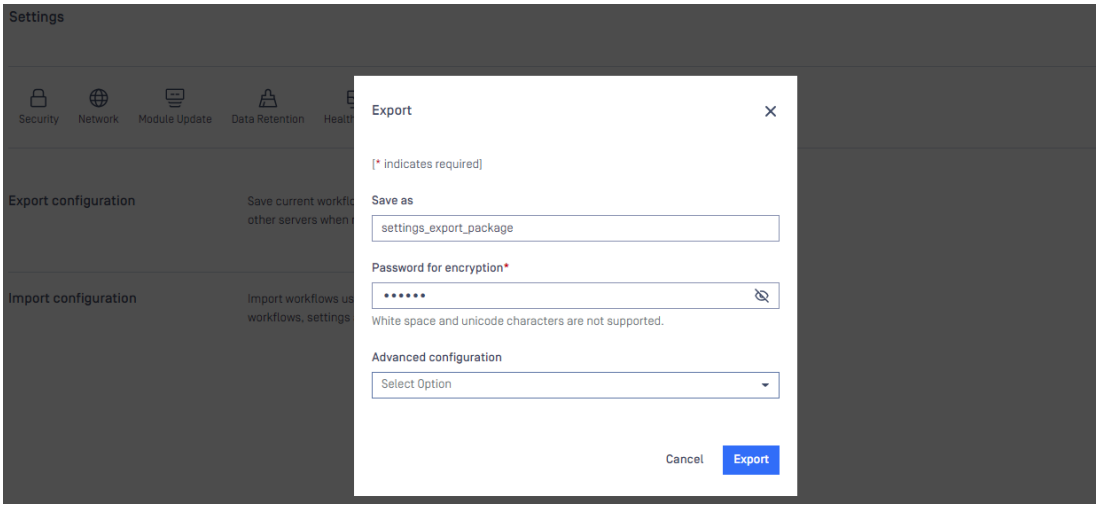
Click the export button to set the password to encrypt the archive package for exported configurations. This package will contain the following configurations:

- Workflow rules and workflow templates and security zones
- Configuration in the **Inventory** menu
  - Skip by Hash
  - External Scanners
  - 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
- Users management settings: users, role, directories

- 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) API

Import

Choose the exported package and also input proper password to import.

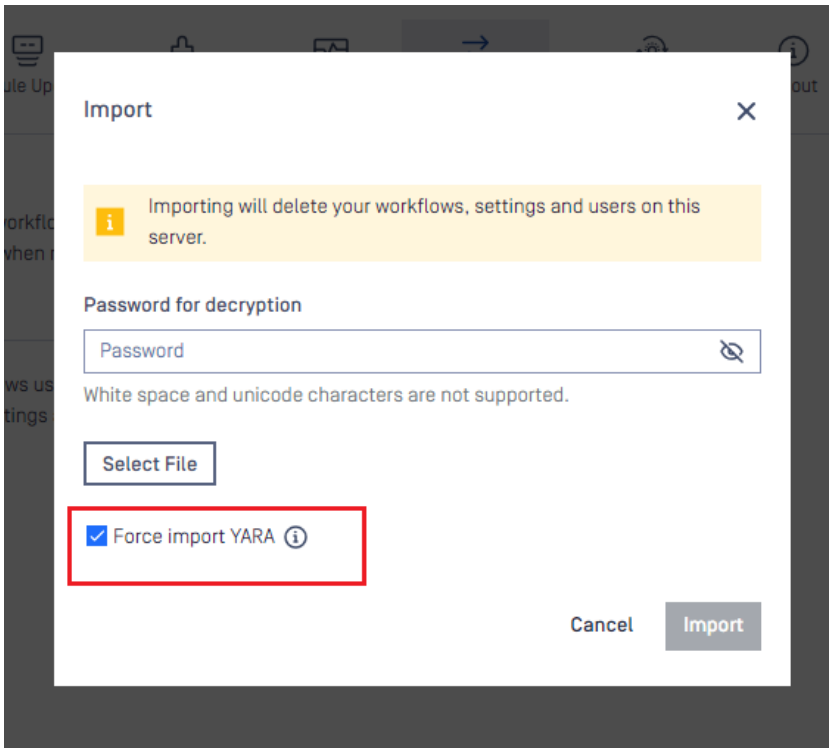
JSON config file import supported
Besides package import support, you can also import JSON config file like in MetaDefender Core v4.

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) API

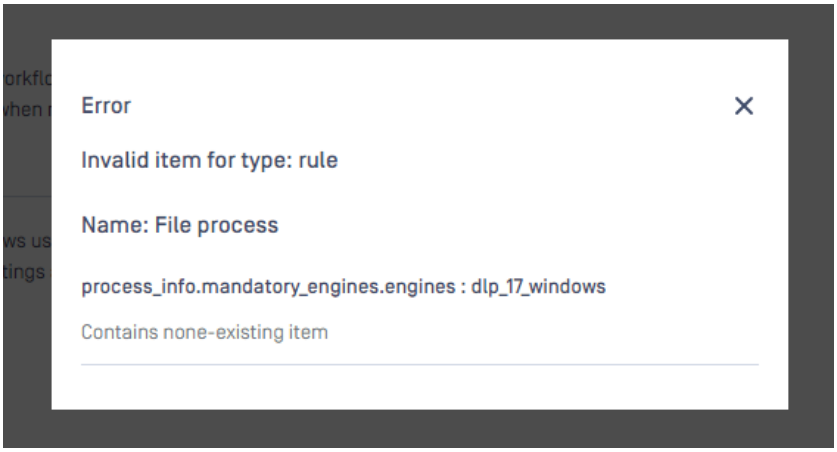
YARA sources will be not imported by default. User need to select `Force import YARA` to import it



Troubleshooting

Occasionally, errors may occur during the configuration import process. This section aims to provide a clear understanding of the issue.

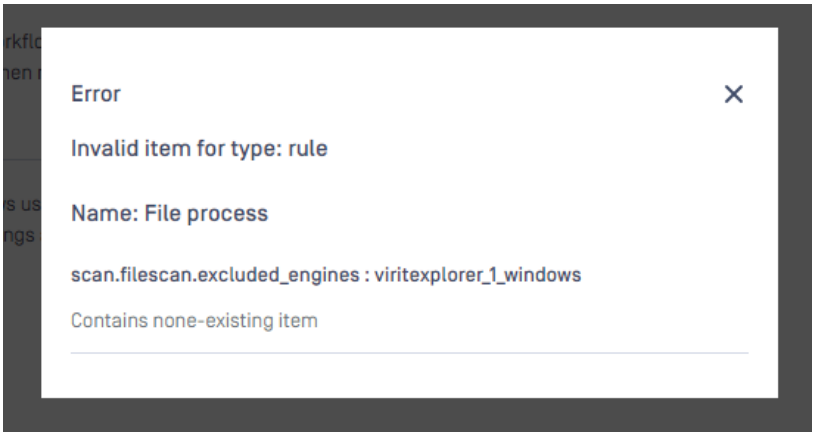
Mandatory engines are missing

Name	Description
Error details	
Root-cause	The imported zip contains the workflow rule named `File process` . In this workflow rule, under tab **General** > **Mandatory engines** had choose Proactive DLP as mandatory engine but in the current MD Core, there is no such engine, so the import failed to process. This issue also happen with all other engines: AV engines, Filetype, Archive extraction, Deep CDR, etc. Proactive DLP here is just an example.
Remediation	There are 2 options: • Option 1: User can install the missing engine in the current MD Core, after that -> re-import the zip file. • Option 2: The user can manually modify the zip file - please refer to the section "Manually modify the zip file" at the end of this page. • Open the file **wf_config_export.json** search for all the string: **process_info.mandatory_engines.engines** and remove the engine, for example:

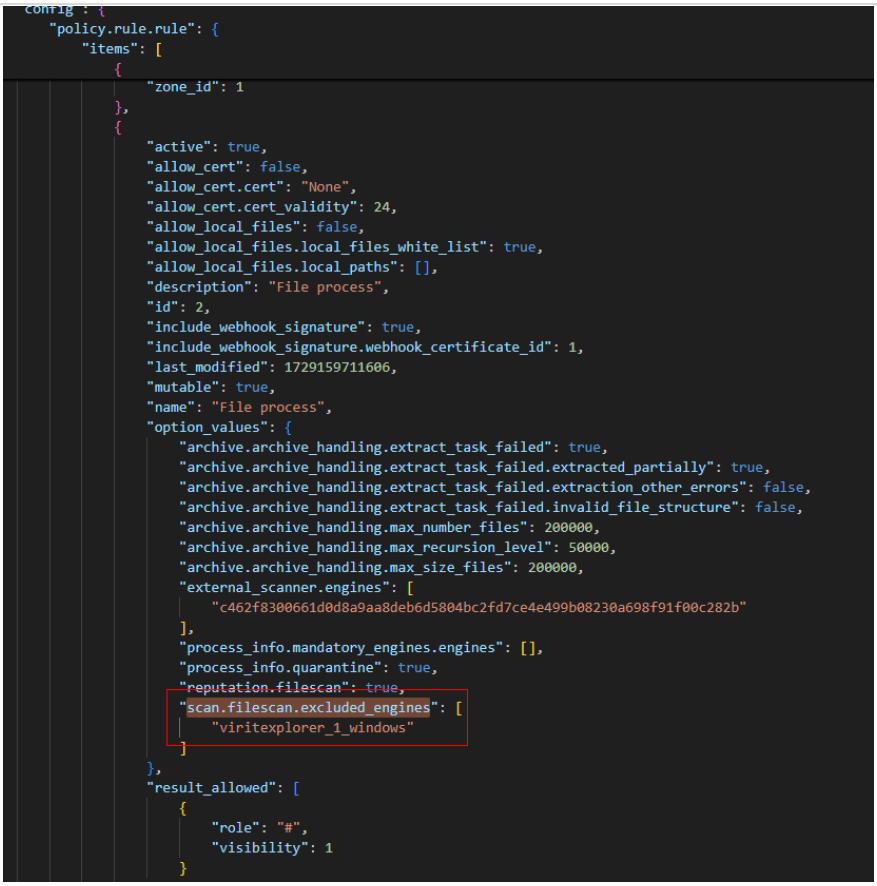

```
"config": {
    "policy.rule.rule": {
        "items": [
            {
                "allow_local_files.local_files_white_list": true,
                "allow_local_files.local_paths": [],
                "description": "File process",
                "id": 2,
                "include_webhook_signature": false,
                "include_webhook_signature.webhook_certificate_id": 0,
                "last_modified": 1729132867245,
                "mutable": true,
                "name": "File process",
                "option_values": {
                    "ds.filescan": true,
                    "filetype_analysis.timeout": 1,
                    "process_info.global_timeout": true,
                    "process_info.global_timeout.value": 120,
                    "process_info.mandatory_engines": true,
                    "process_info.mandatory_engines.engines": [
                        "dlp_17_windows"
                    ],
                    "scan.filescan.download_timeout": 1,
                    "scan.filescan.global_scan_timeout": 1,
                    "scan.filescan.per_engine_scan_timeout": 1
                },
                "result_allowed": [
```

Name	Description

Excluded engines are missing

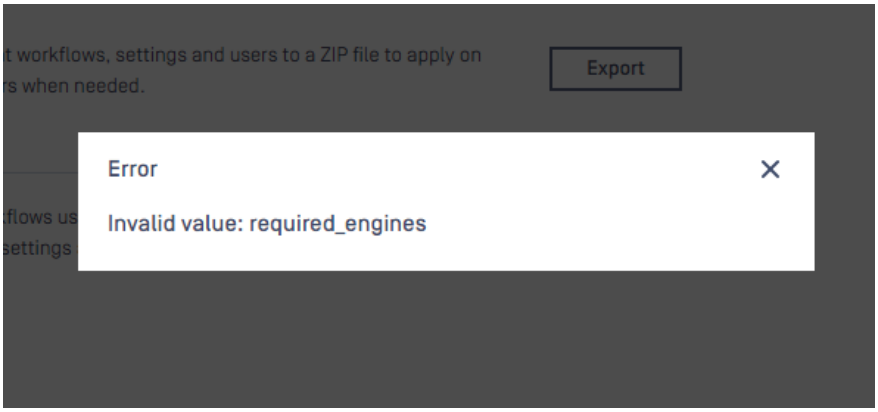
Name	Description
Error details	
Root-cause	The imported zip contains the workflow rule named `File process` . In this workflow rule, under tab **Metascan** > *Excluded AV engines* had choose Vir.IT eXplorer as excluded engine but in the current MD Core, there is no such engine, so the import failed to process. This issue also happen with all other AV engines.
Remediation	There are 2 options: • Option 1: User can install the missing engine in the current MD Core, after that -> re-import the zip file. • Option 2: The user can manually modify the zip file - please refer to the section "Manually modify the zip file" at the end of this page. • Open the file **wf_config_export.json** search for all the string: **scan.filescan.excluded_engines** and remove the engine, for example

Name	Description

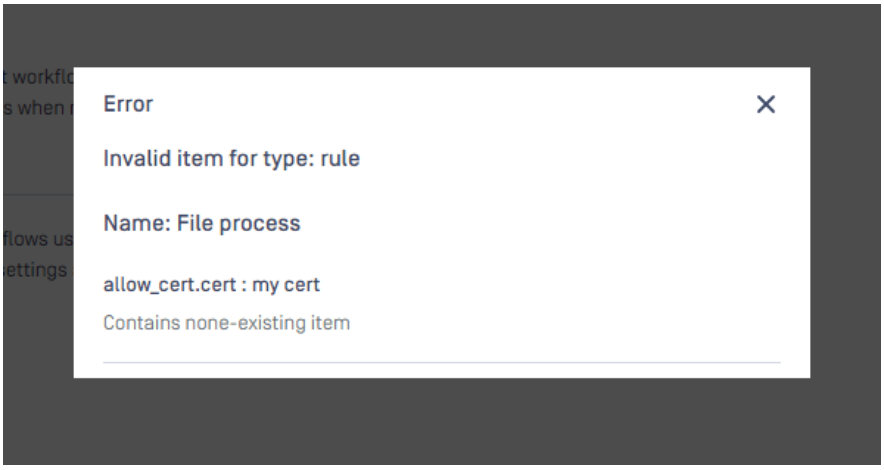
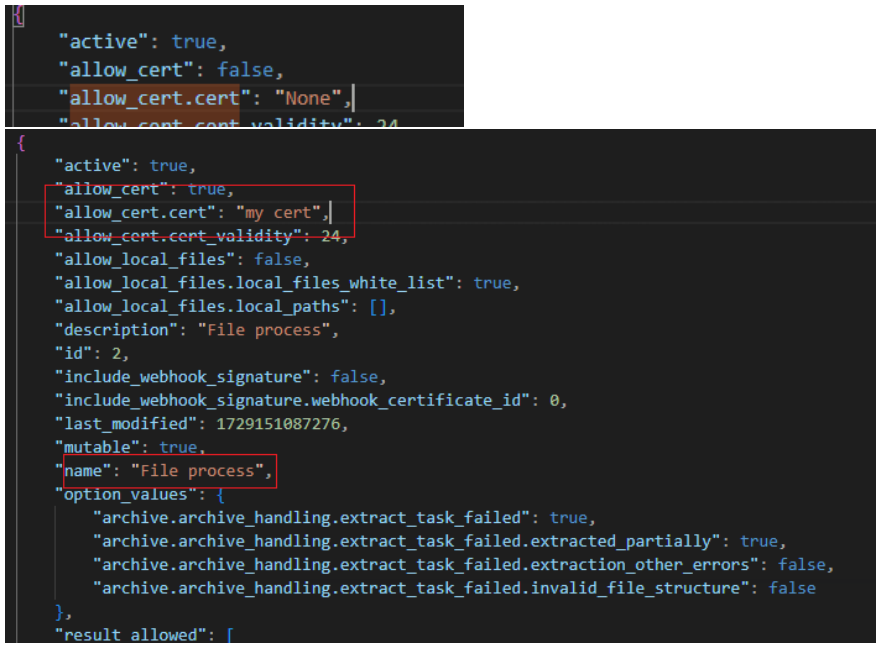


```
config : {
    "policy.rule.rule": {
        "items": [
            {
                "zone_id": 1
            },
            {
                "active": true,
                "allow_cert": false,
                "allow_cert.cert": "None",
                "allow_cert.cert_validity": 24,
                "allow_local_files": false,
                "allow_local_files.local_files_white_list": true,
                "allow_local_files.local_paths": [],
                "description": "File process",
                "id": 2,
                "include_webhook_signature": true,
                "include_webhook_signature.webhook_certificate_id": 1,
                "last_modified": 1729159711606,
                "mutable": true,
                "name": "File process",
                "option_values": {
                    "archive.archive_handling.extract_task_failed": true,
                    "archive.archive_handling.extract_task_failed.extracted_partially": true,
                    "archive.archive_handling.extract_task_failed.extraction_other_errors": false,
                    "archive.archive_handling.extract_task_failed.invalid_file_structure": false,
                    "archive.archive_handling.max_number_files": 200000,
                    "archive.archive_handling.max_recursion_level": 50000,
                    "archive.archive_handling.max_size_files": 200000,
                    "external_scanner.engines": [
                        "c462f8300661d0d8a9aa8deb6d5804bc2fd7ce4e499b08230a698f91f00c282b"
                    ],
                    "process_info.mandatory_engines.engines": [],
                    "process_info.quarantine": true,
                    "reputation.filescan": true,
                    "scan.filescan.excluded_engines": [
                        "viritexplorer_1_windows"
                    ]
                },
                "result_allowed": [
                    {
                        "role": "#",
                        "visibility": 1
                    }
```

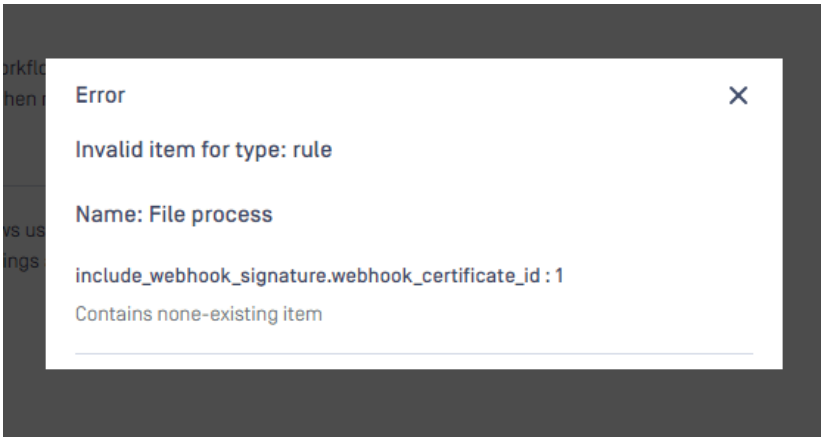
The required engines in Health Check are missing

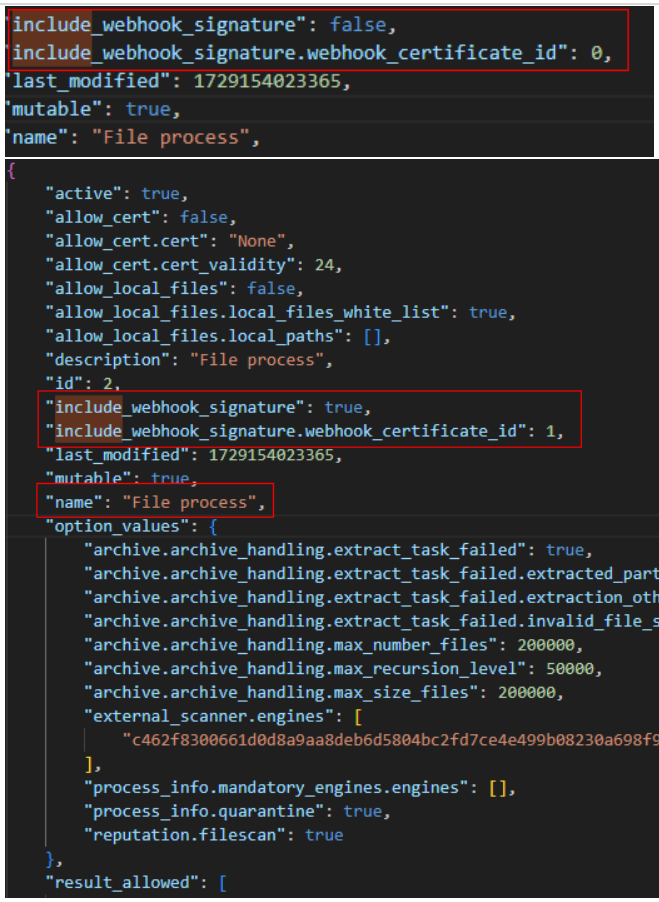
Name	Description
Error details	
Root-cause	The imported zip has Health Check setting, in this section, it contains a setting for "Required engines" For example, in the zip file, it has configured Ahnlab as required engine for Health Check, but in the current MD Core, the license does not contains this engine, or simply this MD Core does not add license yet.
Remediation	There are 2 options: • Option 1: User can add the license that has contains the required engines. • Option 2: The user can manually modify the zip file - please refer to the section "Manually modify the zip file" at the end of this page. • Open the file **settings_export.json** search for all the string: **required_engines** and remove the engine, for example: • 

The batch cert that using in the workflow is missing

Name	Description
Error details	
Root-cause	The imported zip contains the workflow rule named `File process` . In this workflow rule, under tab **General** > **Generate batch signature with certificate** had choose a search named "my cert". But in the current MD Core, there no such cert name "my cert"
Remediation	There are 2 options: • Option 1: User can add the missing cert in Inventory > Certificates > Add the cert like the exported configuration. • Option 2: The user can manually modify the zip file - please refer to the section "Manually modify the zip file" at the end of this page. • Open the file **wf_config_export.json** and search for all the string: allow_cert and change it to false, **allow_cert.cert** and change it to None • 

Name	Description
The webhook signature using in the workflow is missing	

Name	Description
Error details	
Root-cause	The imported zip contains the workflow rule named `File process` . In this workflow rule, under tab **General** > **Webhook** had choose a signature ID: 1. But in the current MD Core, there no such signature in Inventory> Webhook signature
Remediation	There are 2 options: • Option 1: User can add the missing Webhook signature in Inventory. • Option 2: The user can manually modify the zip file - please refer to the section "Manually modify the zip file" at the end of this page. • Open the file **wf_config_export.json** search for all the string: **include_webhook_signature** and change it to **false**, "**include_webhook_signature.webhook_certificate_id ": 1** and reset this id to **0**, for example:

Name	Description
•	


```
include_webhook_signature": false,
include_webhook_signature.webhook_certificate_id": 0,
last_modified": 1729154023365,
mutable": true,
name": "File process",
```

```
{
    "active": true,
    "allow_cert": false,
    "allow_cert.cert": "None",
    "allow_cert.cert_validity": 24,
    "allow_local_files": false,
    "allow_local_files.local_files_white_list": true,
    "allow_local_files.local_paths": [],
    "description": "File process",
    "id": 2,
    "include_webhook_signature": true,
    "include_webhook_signature.webhook_certificate_id": 1,
    "last_modified": 1729154023365,
    "mutable": true,
    "name": "File process",
    "option_values": {
        "archive.archive_handling.extract_task_failed": true,
        "archive.archive_handling.extract_task_failed.extracted_partially": true,
        "archive.archive_handling.extract_task_failed.extraction_other_errors": false,
        "archive.archive_handling.extract_task_failed.invalid_file_structure": false,
        "archive.archive_handling.max_number_files": 200000,
        "archive.archive_handling.max_recursion_level": 50000,
        "archive.archive_handling.max_size_files": 200000,
        "external_scanner.engines": [
            "c462f8300661d0d8a9aa8deb6d5804bc2fd7ce4e499b08230a698f91f00c282b"
        ],
        "process_info.mandatory_engines.engines": [],
        "process_info.quarantine": true,
        "reputation.filescan": true
    },
    "result_allowed": [
```


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:
    a. wf_config_export.json → workflow rules/template related settings.
    b. settings_export.json → general configuration.
    c. 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**

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 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.

Code	Scan result	Relation	Description
16	Exceeded Archive Timeout	Archive Extraction engine	The archive extraction process on a certain 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 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	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.
27	Likely Malicious Verdict by Sandbox	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.

Code	Scan result	Relation	Description
28	Malicious Verdict by Sandbox	Adaptive Sandbox	Adaptive Sandbox has definitively determined that the file is harmful.
29	Blocked Verdict by Sandbox	Adaptive Sandbox	This indicates that the Adaptive Sandbox has determined that this file is dangerous and needs 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	SBOM	This indicates that software components contain known vulnerabilities that could potentially be exploited and therefore may require attention or 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	SBOM	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	SBOM	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.

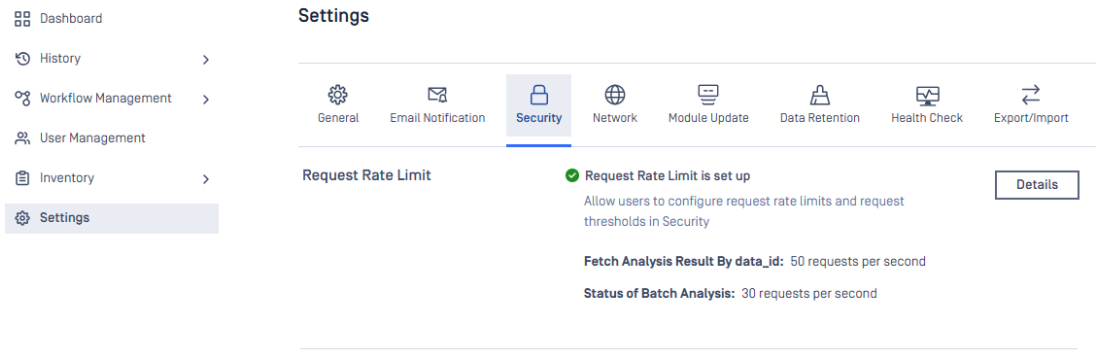
Code	Scan result	Relation	Description
38	Known Bad	Reputation engine	The Reputation engine found the file as a known (recognizable) harmful one. For more details, the Reputation Engine cross-references file hashes with a database of known malicious files and utilizes advanced analysis to quickly address false positives. 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.
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	SBOM	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.

Code	Scan result	Relation	Description
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 blocked. This verdict has the lowest priority compared to other block verdicts.
49	InSights Indicator Matches	InSights Threat Intelligence	This indicates that InSights Threat Intelligence has detected malicious domains and IP addresses in the file.
50	No Indicator Matches in InSights	InSights Threat Intelligence	This indicates that InSights Threat Intelligence has detected **no** malicious domains and IP addresses in the file.
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.
255	In Progress	MetaDefender Core	The scan is still in progress, not yet finished.

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

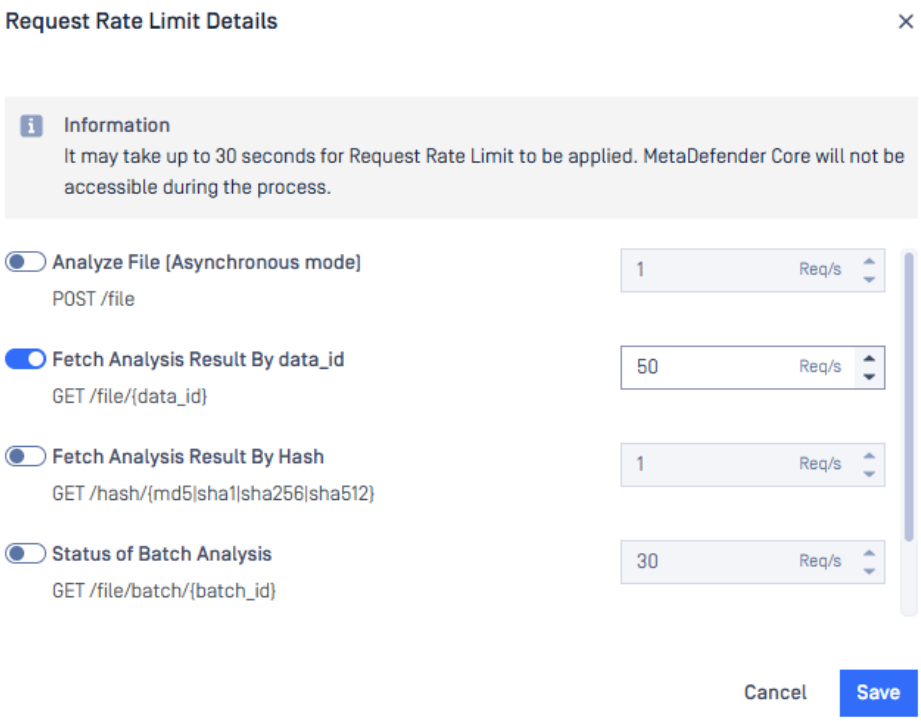
Version	5.13.0
Max scan queue	4000
Engines	File type, Archive, AhnLab, Avira, Bitdefender, ESET, K7, Quick Heal, and Vir.IT Explorer

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
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



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.

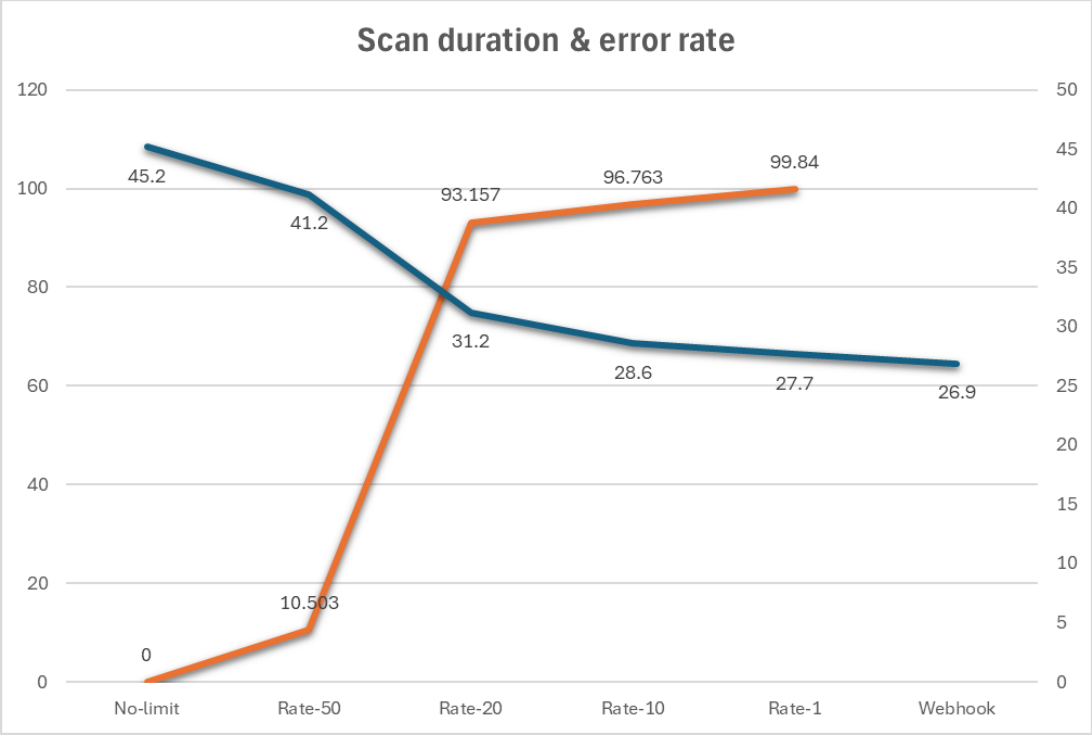
Reference here for more details about header `callbackurl`.

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.

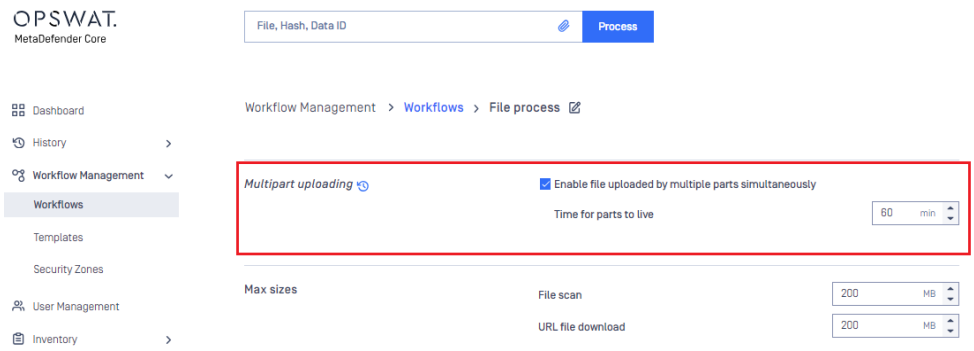
This feature composes a set of following APIs

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.

json

```
{
  "err": "Multipart uploading session cancelled"
}
```

How to configure

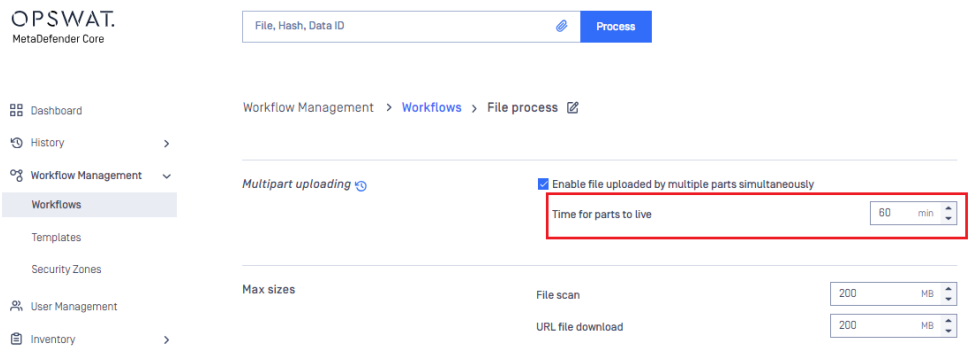
By default, a multipart upload session, once initiated successfully, will be available for parts uploaded within 60 minutes. Outside that duration, HTTP code 409 CONFLICT is returned for each call to POST `/file/multipart/{data_id}`.

json

```
{
  "err": "Multipart uploading session timed out"
}
```

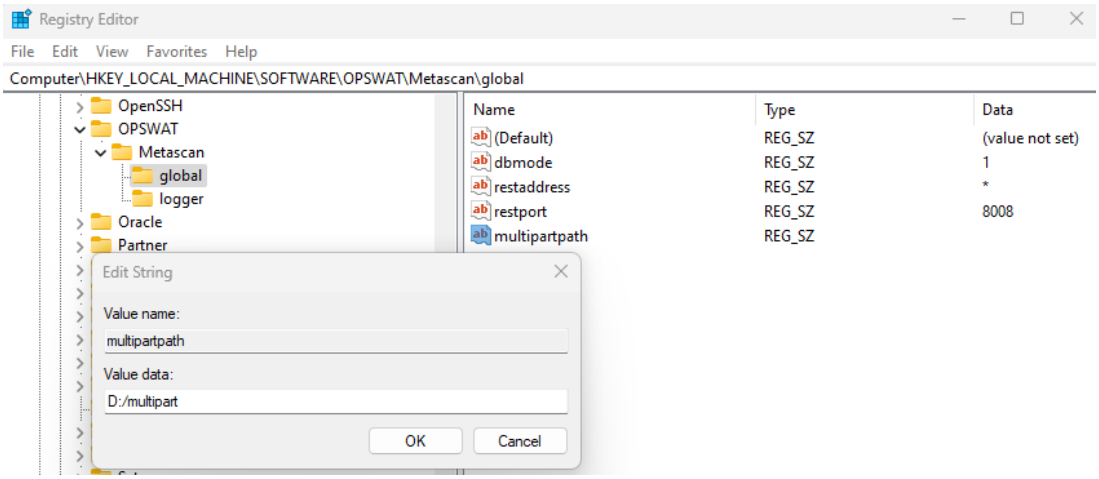
You can extend or collapse the time range as needed by following these steps:

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

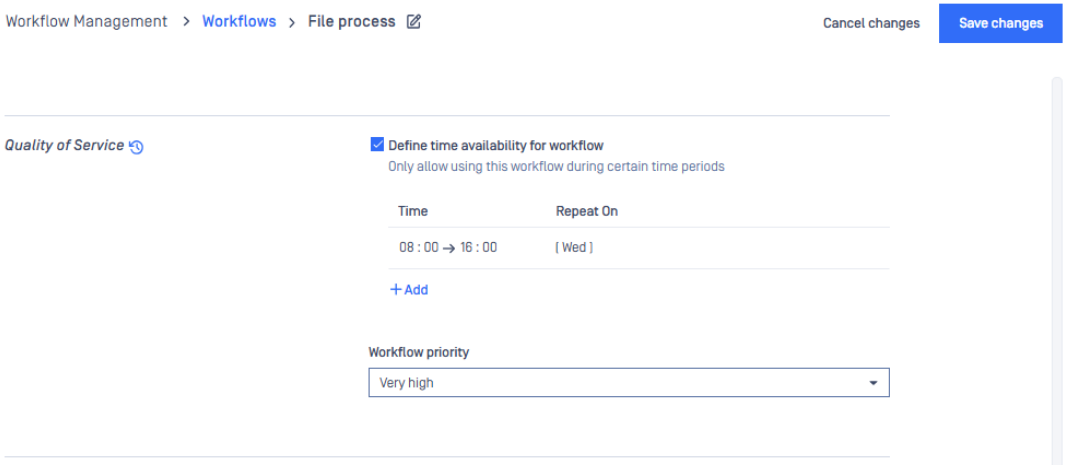
When all parts of a file are uploaded fully to MetaDefender Core, you can cancel the file processing with POST POST `/file/{data_id}/cancel`. Please do not be confused by the two APIs. DELETE `/file/multipart/{data_id}` is used when parts of the file are being uploaded, while POST

`/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 must remain within the 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. 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.

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**
- **How to deploy**
- **Recommended database service**
- **How to monitor**
- **Troubleshooting and maintenance**

Recommended Architectures

This page explains what are the architecture recommended for running MetaDefender Core in AWS EC2 instance or AWS EKS (Kubernetes)

Infrastructure Options for MetaDefender Core in AWS EC2

The possible infrastructure vary depending on what are the number of EC2 instances where MetaDefender Core will be installed

Options:

1. Single EC2
2. Autoscaling EC2 (On RoadMap)

Infrastructure Options for MetaDefender Core in AWS EKS

The possible infrastructure vary depending on what are the AWS services for hosting 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**

Above options are explained at EKS 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 MetaDefender Core service. See EKS Cluster Deployment

EKS Cluster

The possible infrastructure vary depending on what are the AWS services for hosting 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 provision the cluster for you and install the K8S components needed for MetaDefender Core service.

MetaDefender PostgreSQL database details

Recommendations

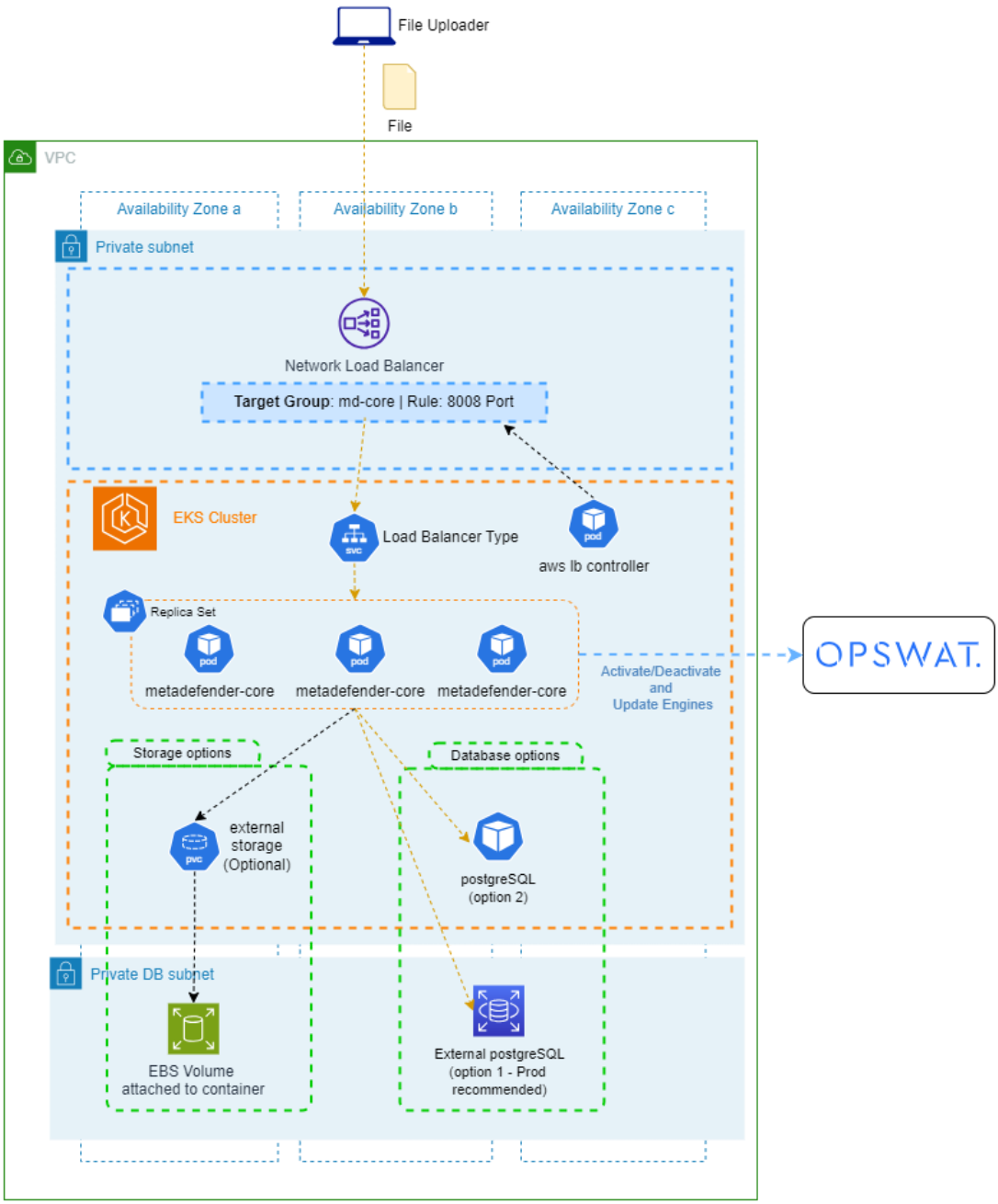
For running MetaDefender Core in a Kubernetes cluster we recommend to use, for production environments, an external database service for PostgreSQL. In case of **AWS** it is **AWS RDS** which the script will install and configure to be accessible from the cluster.

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.

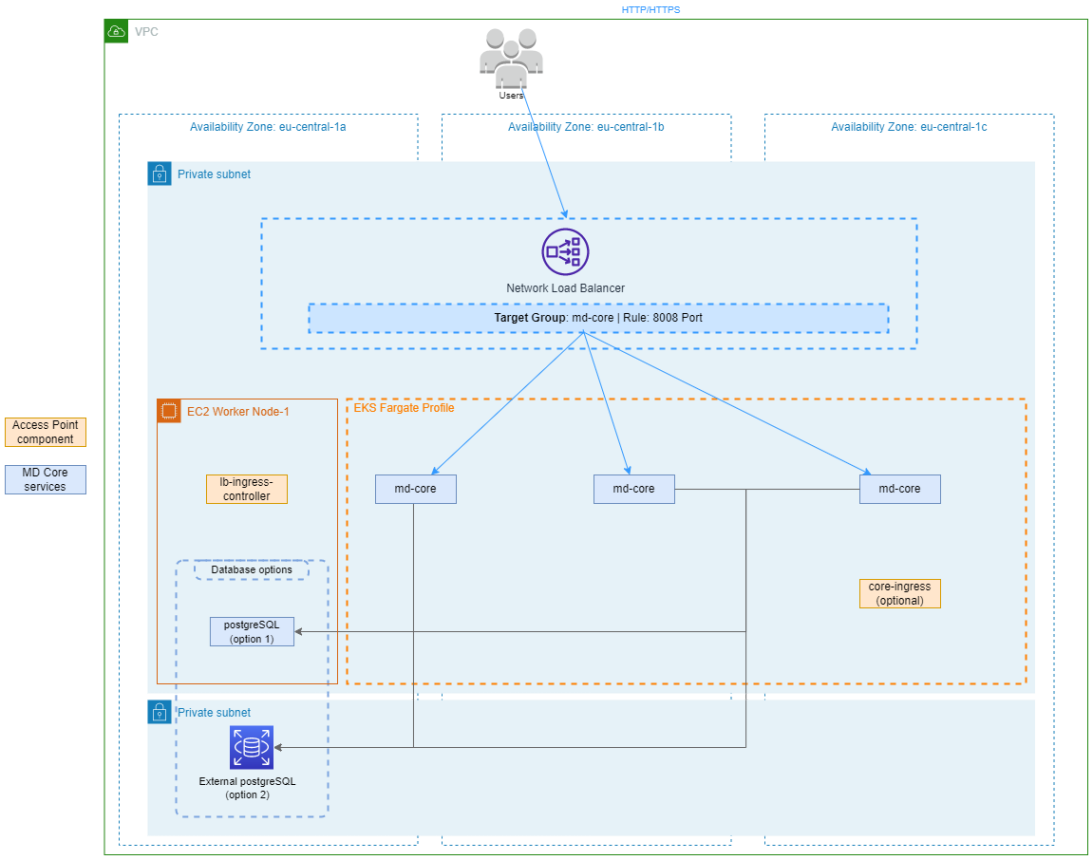
EC2 Instances + AWS RDS / PostgreSQL Pod

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
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

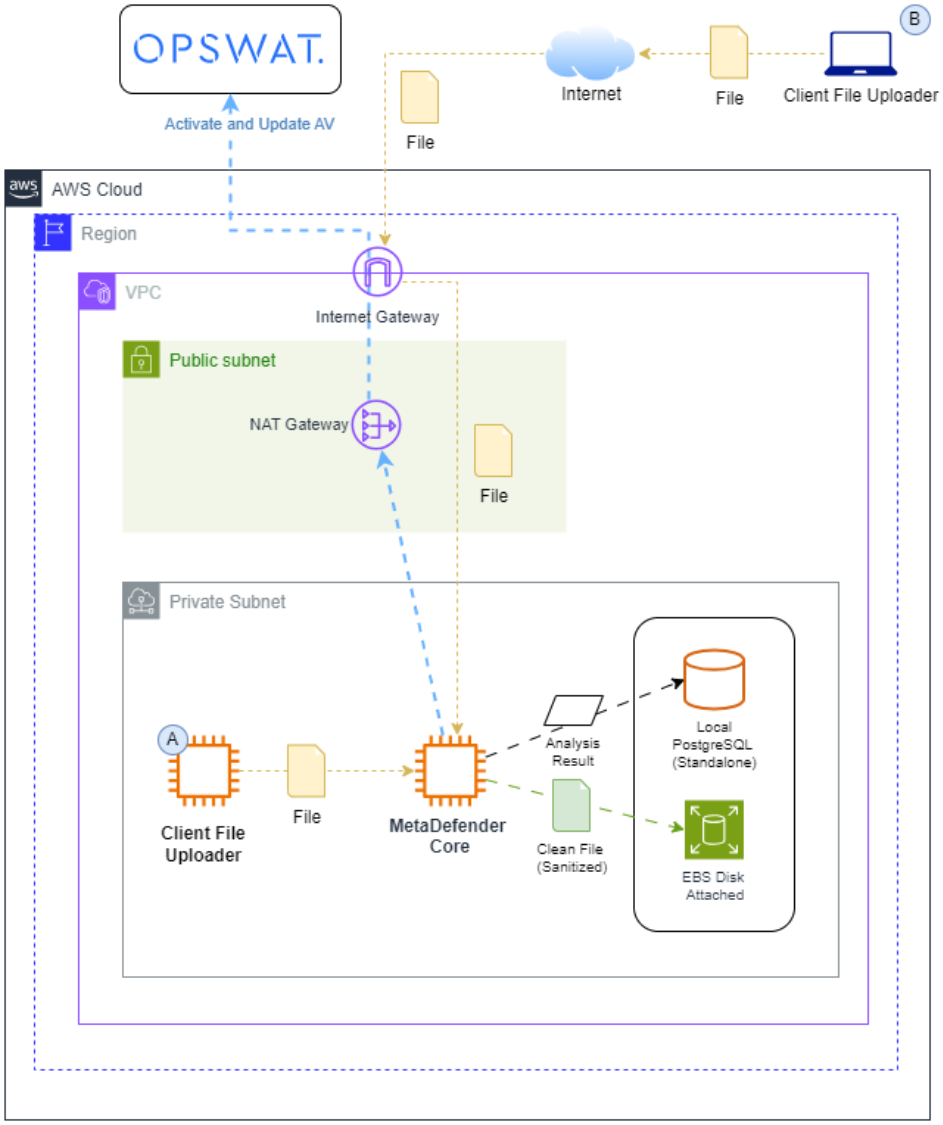
Orange flows

- **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}/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

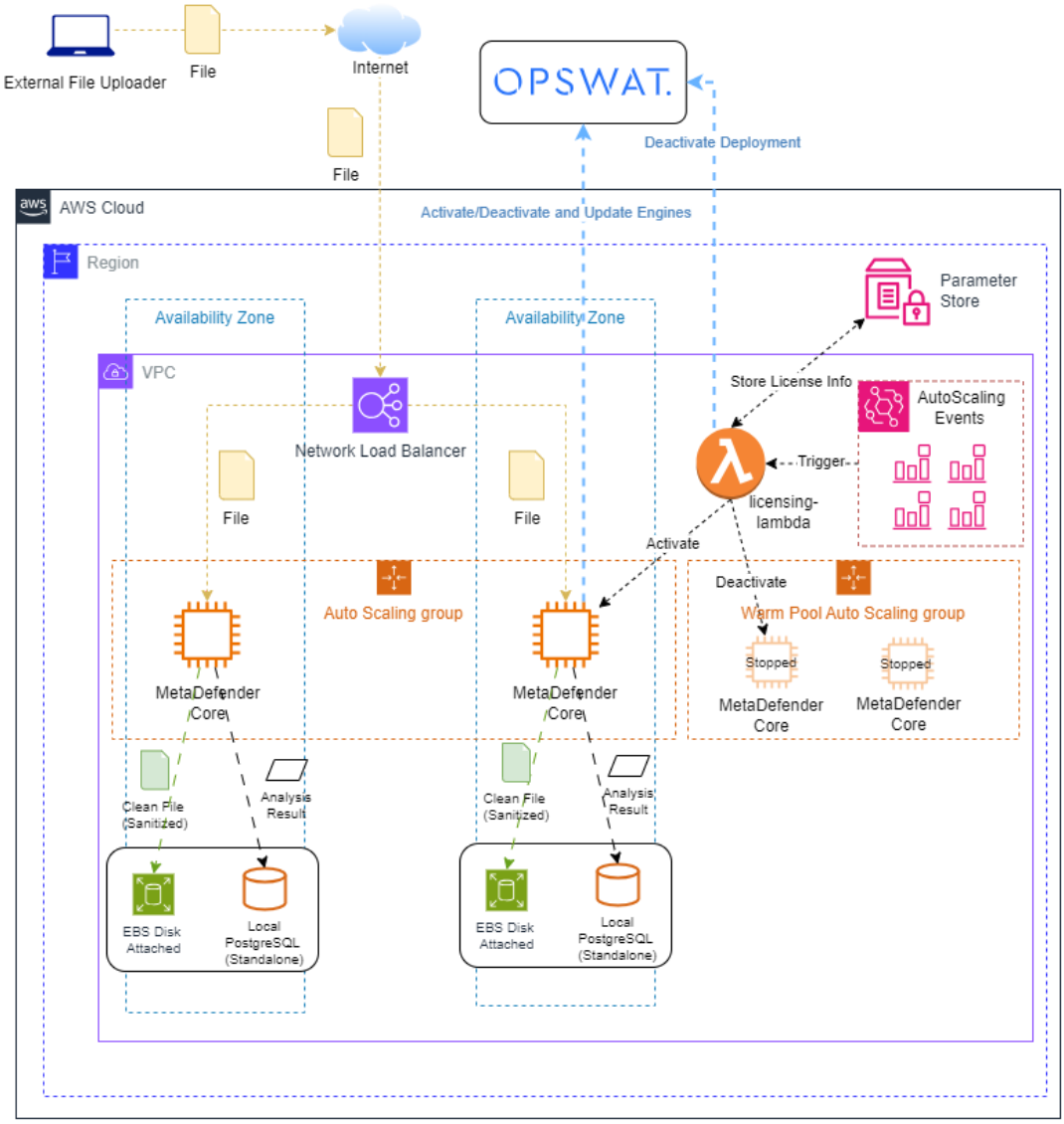
Autoscaling EC2

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

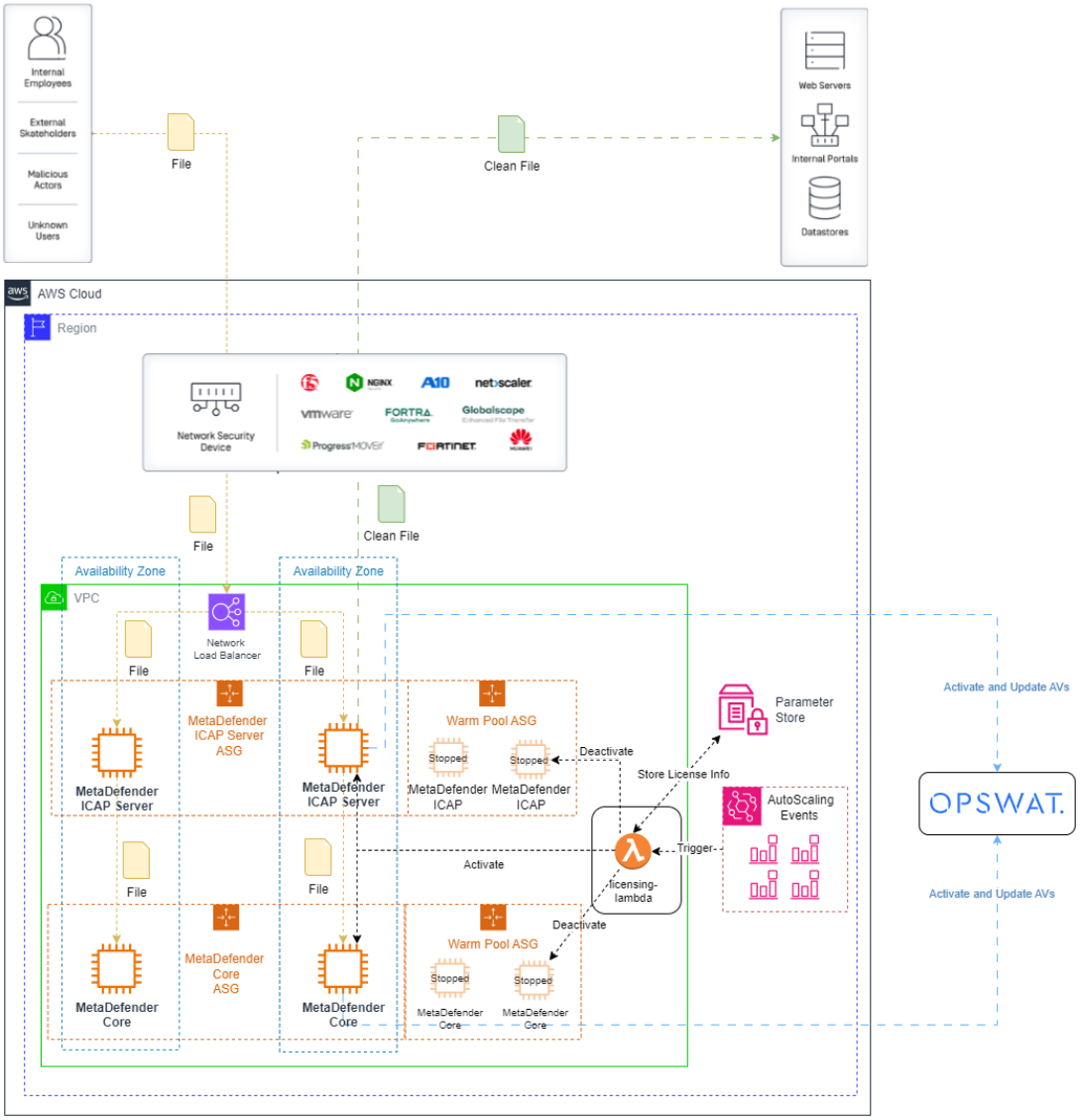
There are 2 flows for MetaDefender Core to work

- **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 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

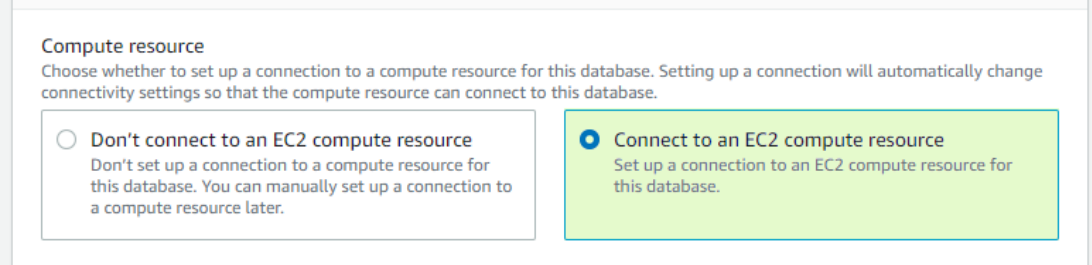
Database Connection

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

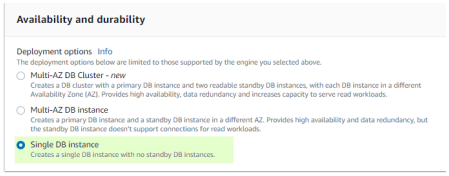
To connect MetaDefender Core with RDS hosted in a private subnet, the configuration will depend on the type of deployment.

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.

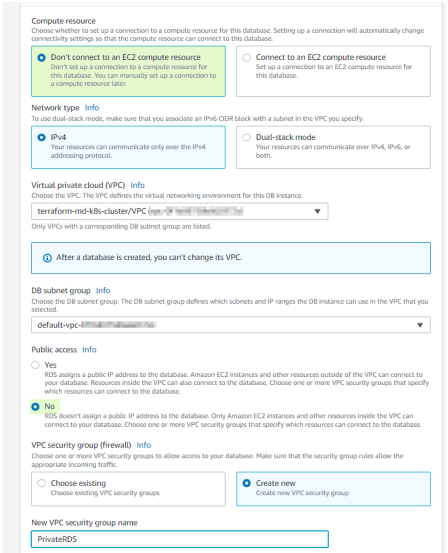


For EKS deployment, using the terraform project provided by OPSWAT it is deployed in a private subnet only allowing access from VPC CIDR where the EKS cluster is deployed. In case of creating the RDS instance from the console follow these steps

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

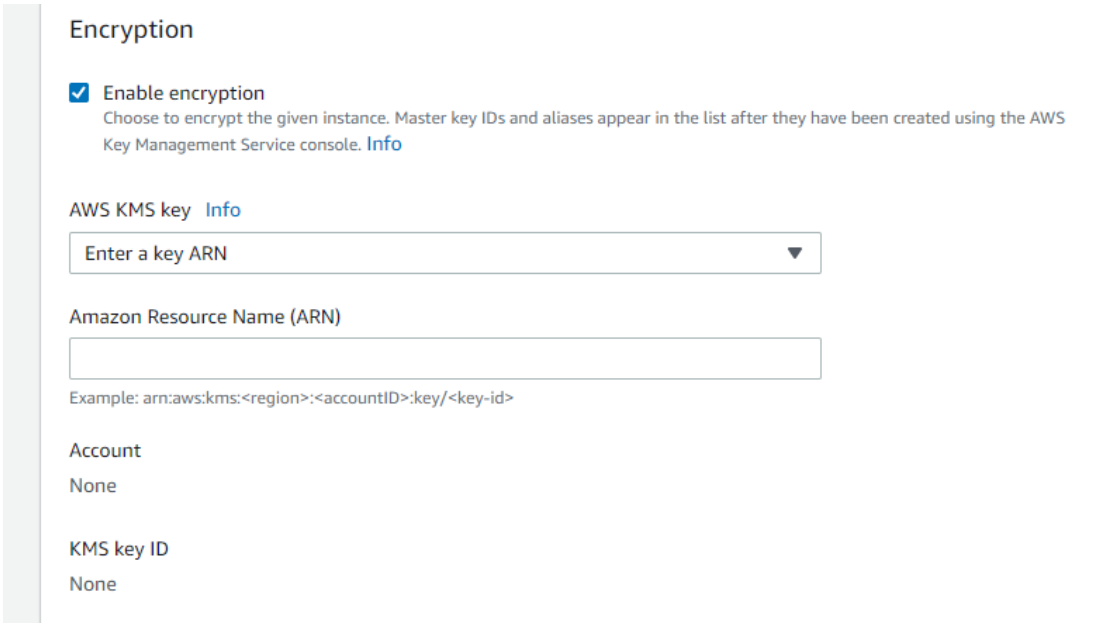
Network traffic processed by MetaDefender Core is SSL encrypted during transmission. The product does not collect or store any customer data outside of analyzing the contents of files submitted for scanning.

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


Encryption
Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. Info
AWS KMS key Info
Enter a key ARN
Amazon Resource Name (ARN)
Example: arn:aws:kms:<region>:<accountID>:key/<key-id>
Account
None
KMS key ID
None

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

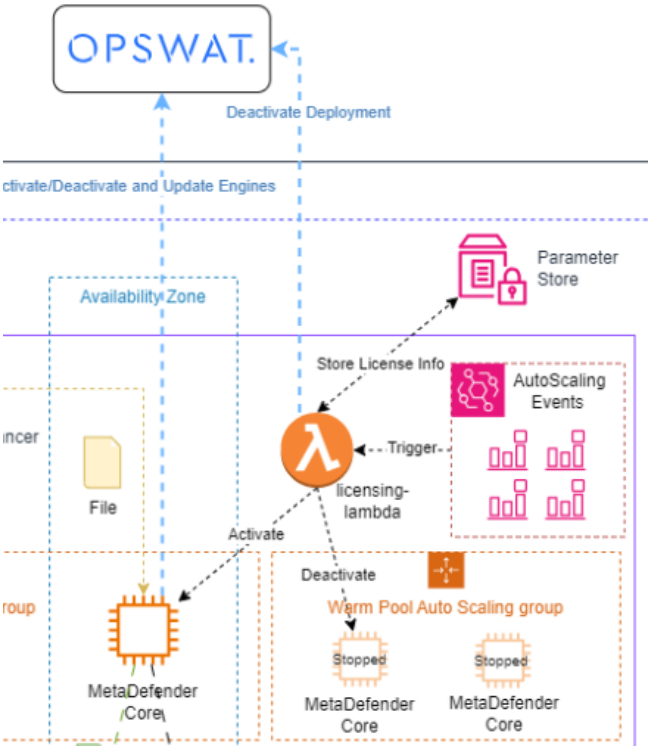
> **Warning**
> Do not enable the Lambda function automation and set up the environment variables from the user data of the instances, as this will generate a conflict and neither option will work.

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
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

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"

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
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 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

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.

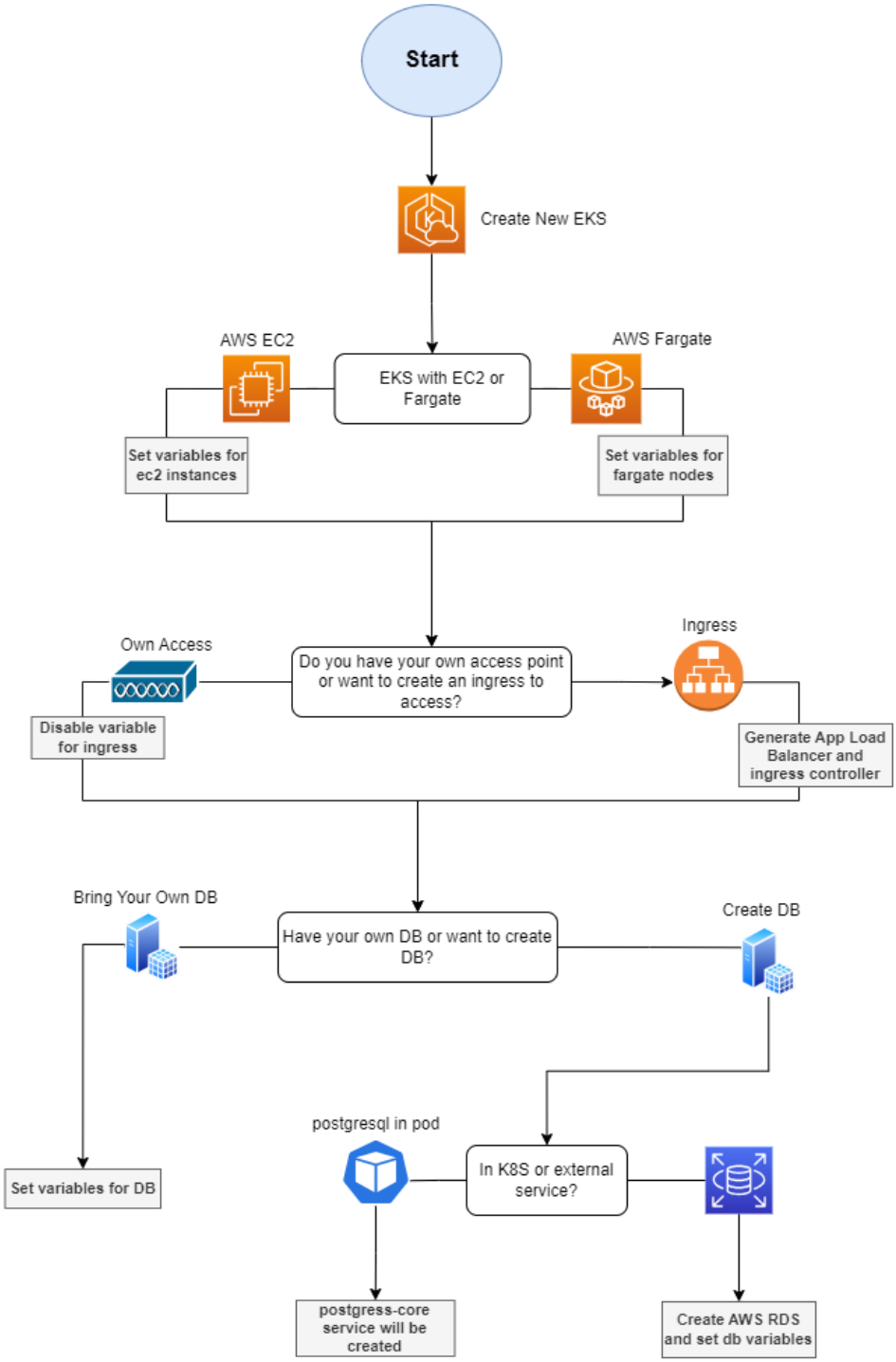
> **Info**
> The script will deploy a single Worker Node for the cluster with enough space for 1 replica of MD Core. AWS EC2 Instance size is **c5.2xlarge (8 vCPU & 16 GiB Memory).** Each pod would request 7 vCPU and 8 GiB Memory. To change the request to adapt each pod to the specific case go to values.yml 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.
   a. **An Ingress** will be create per each product flag added as parameter to the script
   b. **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**
   a. **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.
   b. Create new **DB in K8S** or **external DB** that for AWS we will provision a ***RDS instance***


Start
Create New EKS
AWS EC2
EKS with EC2 or Fargate
AWS Fargate
Set variables for ec2 instances
Set variables for fargate nodes
Own Access
Do you have your own access point or want to create an ingress to access?
Ingress
Disable variable for ingress
Generate App Load Balancer and ingress controller
Bring Your Own DB
Have your own DB or want to create DB?
Create DB
Set variables for DB
postgresql in pod
In K8S or external service?
postgress-core service will be created
Create AWS RDS and set db variables

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

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"

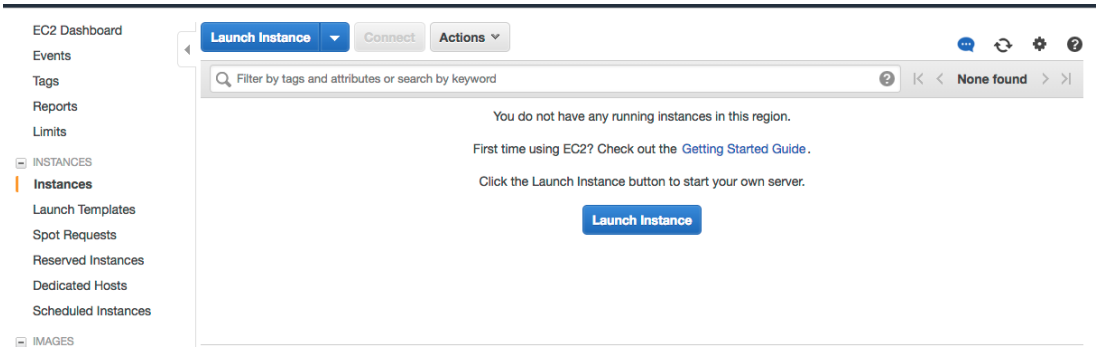
bash

```
terraform init
terraform apply
```

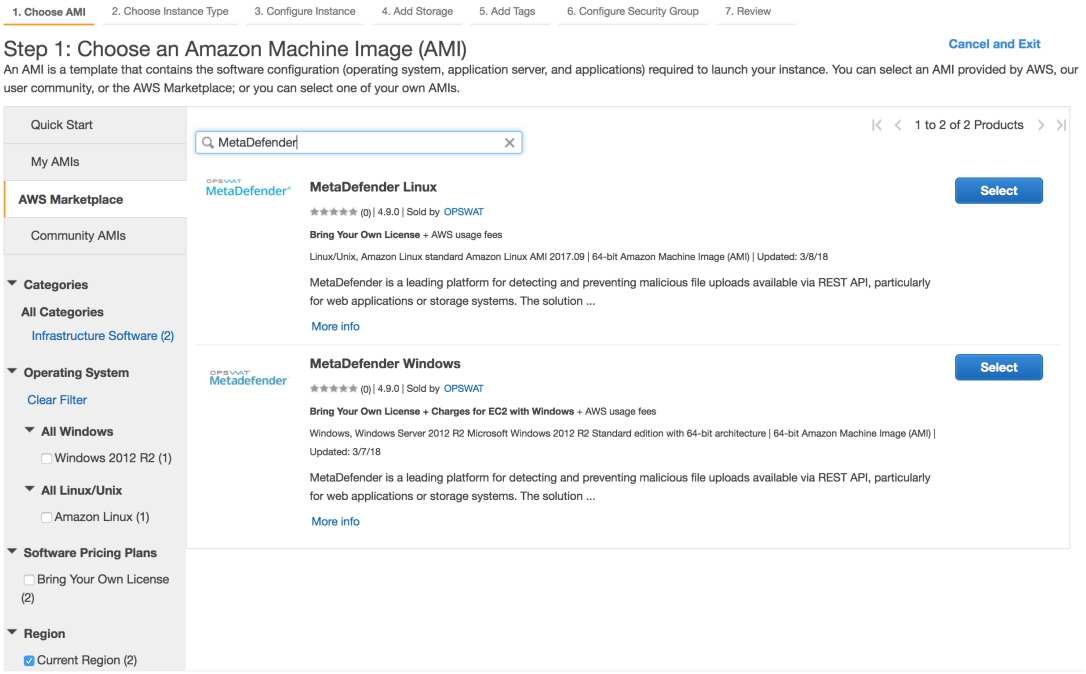
Deploy using AWS Console

Select MetaDefender Core AMI

- Go to the EC2 Management Console in AWS and select Launch Instance

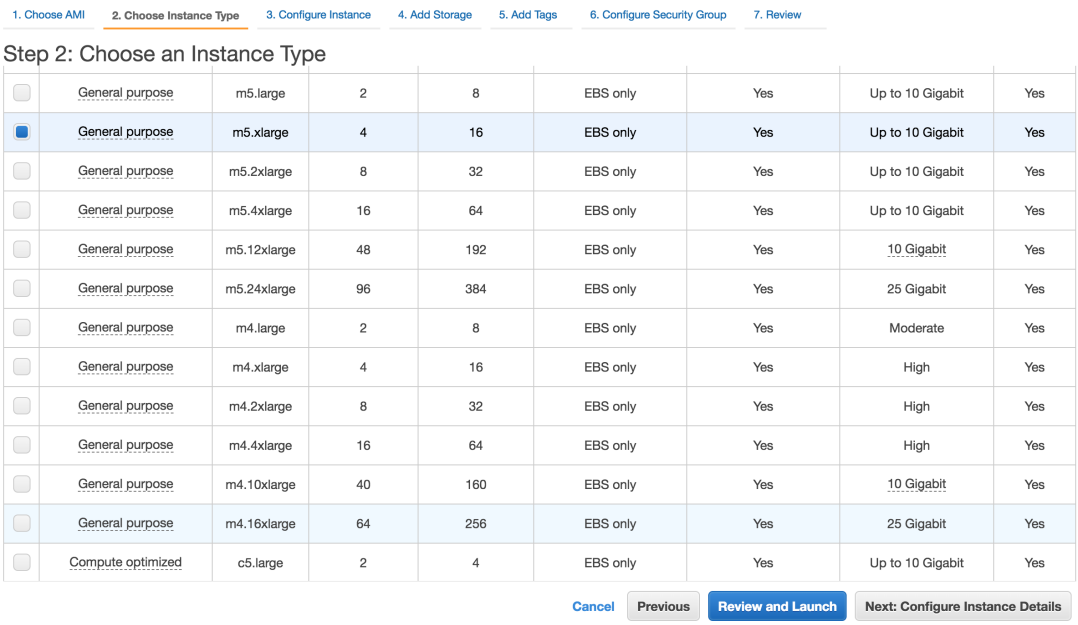


- Select the desired MetaDefender, based on the OS support:



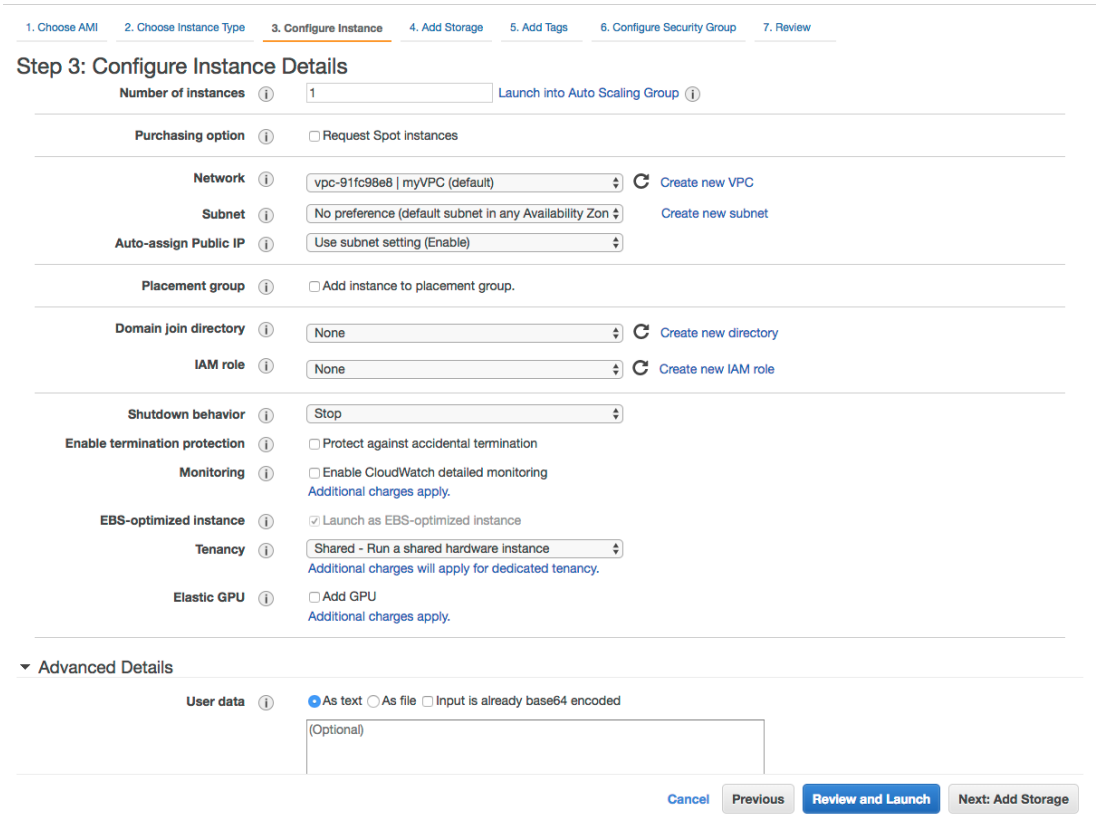
- Choose the Instance type

Please review MetaDefender's system requirements (OS and hardware requirements) before choosing the desired AMI and instance type.



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.

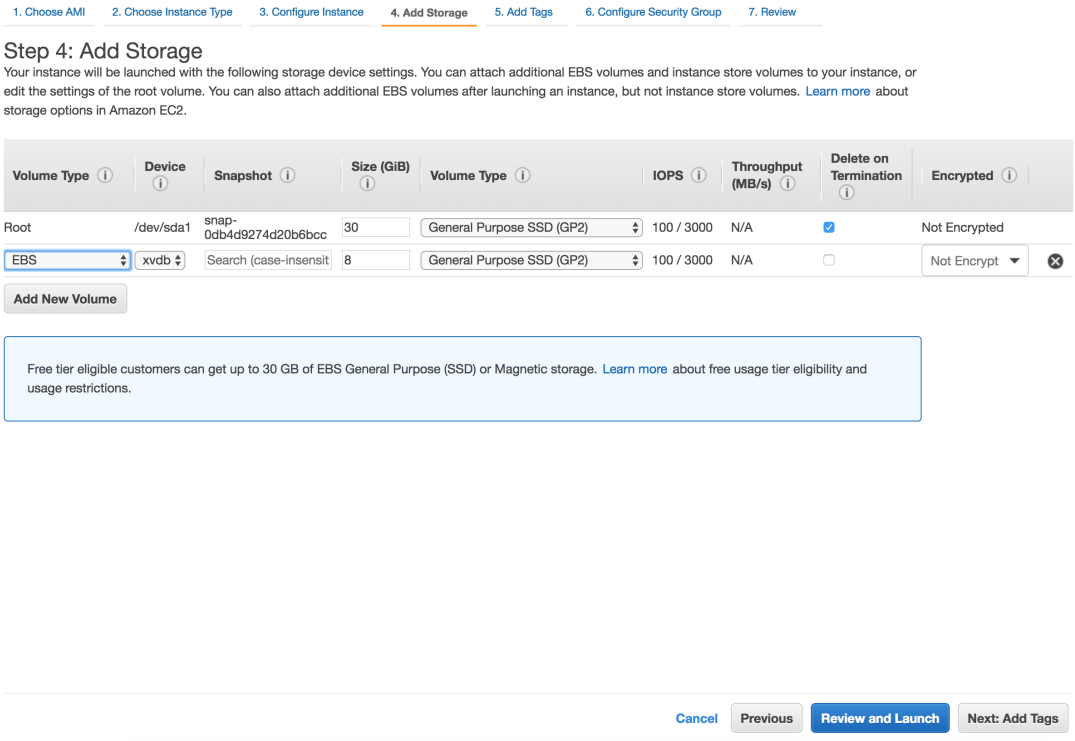
Storage configuration

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.

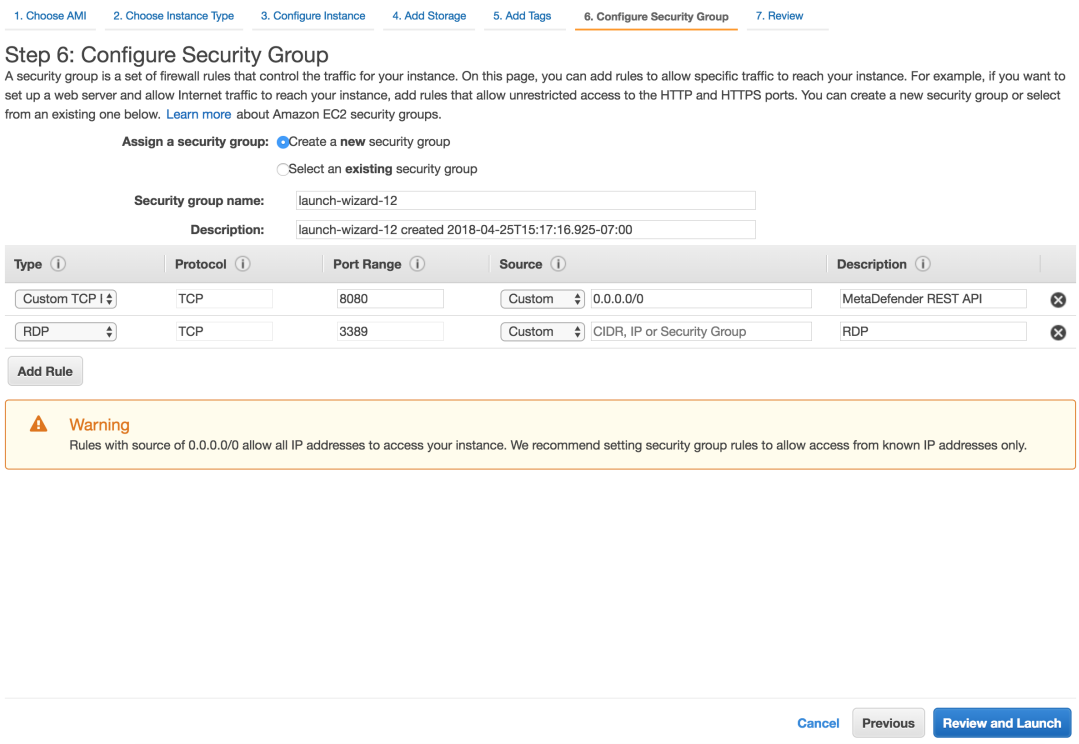


Security Groups configuration

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

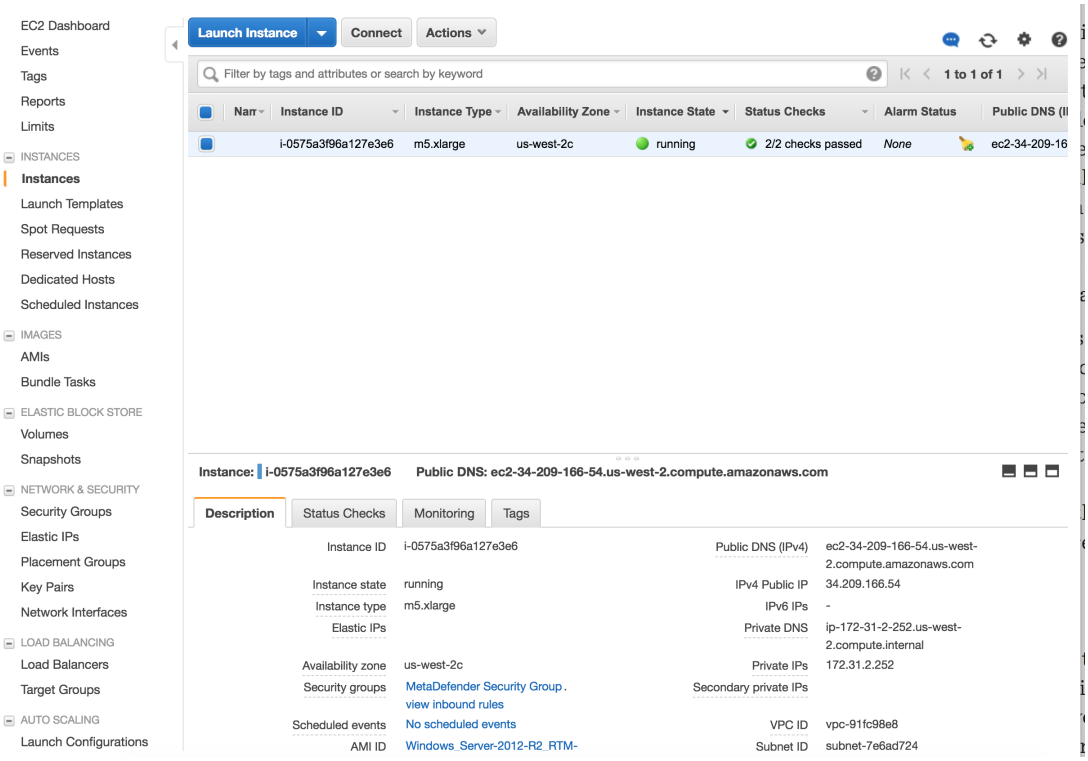


Warning

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

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"

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 v5.2.0 is available the "/readyz" endpoint that helps to fetch the current health check status of MetaDefender Core server

Health Check for EC2 instance deployment

For generic hardware or AZ fault, we recommend to deploy MetaDefender always 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 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

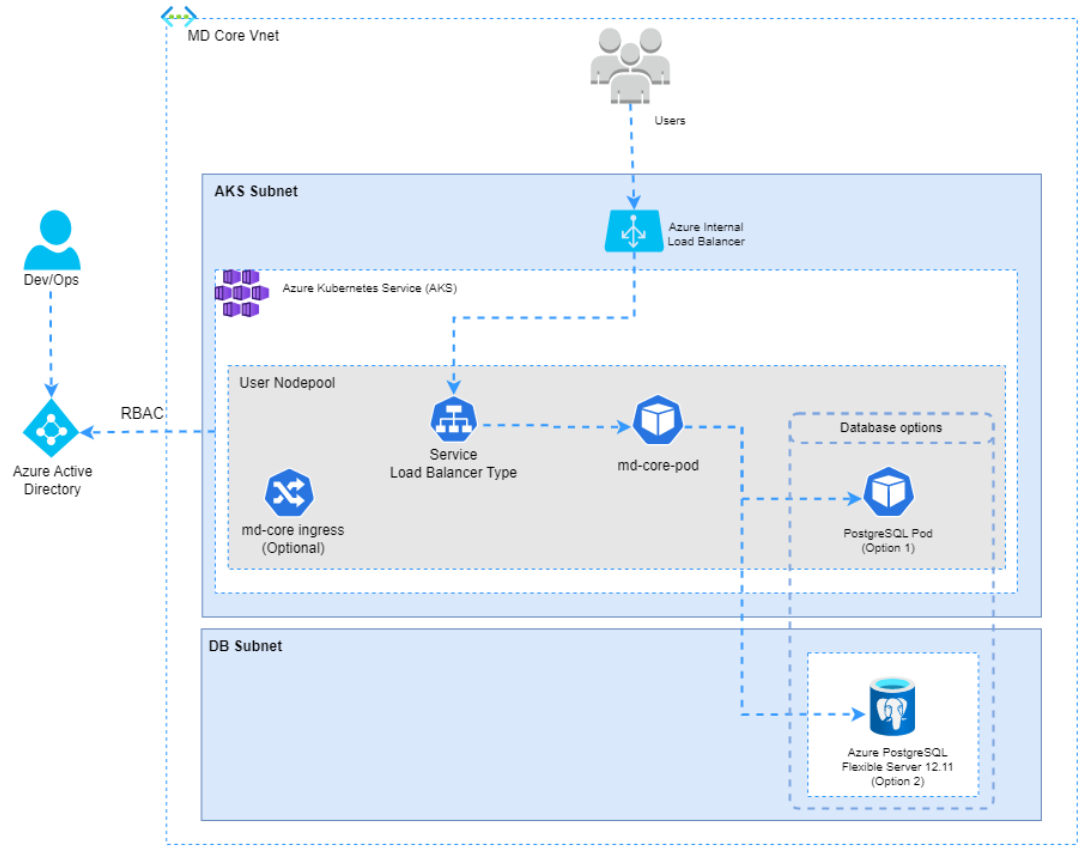
MetaDefender PostgreSQL database details

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	Description
Azure Virtual Network	<resource-group-name>-vnet	
Azure Subnet	<resource-group-name>-aks-subnet	
Azure Subnet	<resource-group-name>-db-subnet	
Azure Private DNS Zone	<postgres_db_account_name>.postgres.database.azure.com	
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

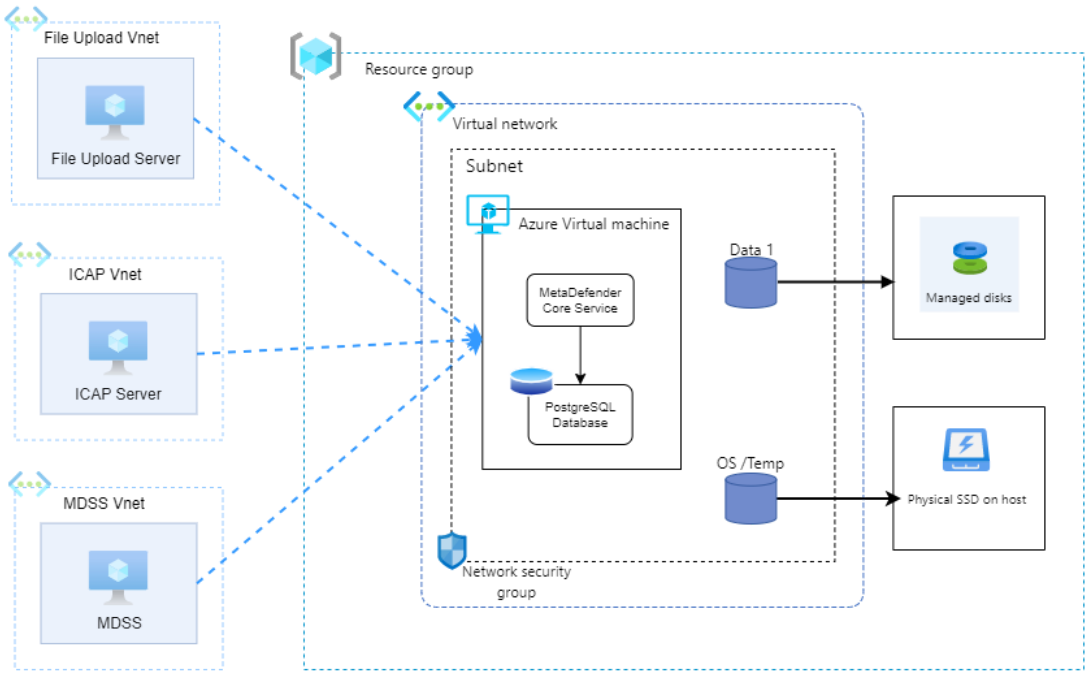
Service Type	Resource Name	Description	Option
Azure AKS	<cluster_name>	Azure Kubernetes Service Cluster	All
Azure ProstgreSQL Flexible Server	postgresql-<postgres_db_account_name>	PostgreSQL Server for MetaDefender Core	Option with Azure PostgreSQL
Azure ProstgreSQL Flexible Server Configuration	azure.extensions	Configuration for installing PG_TRGM,DBLINK,BTREE_GIN_	Option with Azure PostgreSQL

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
SubNet	metadefender-core-subnet	

Compute

Service Type	Resource Name	Description
Azure VM	metadefender-core-<os>	Azure VM Instance to run Windows MetaDefender Core AMI

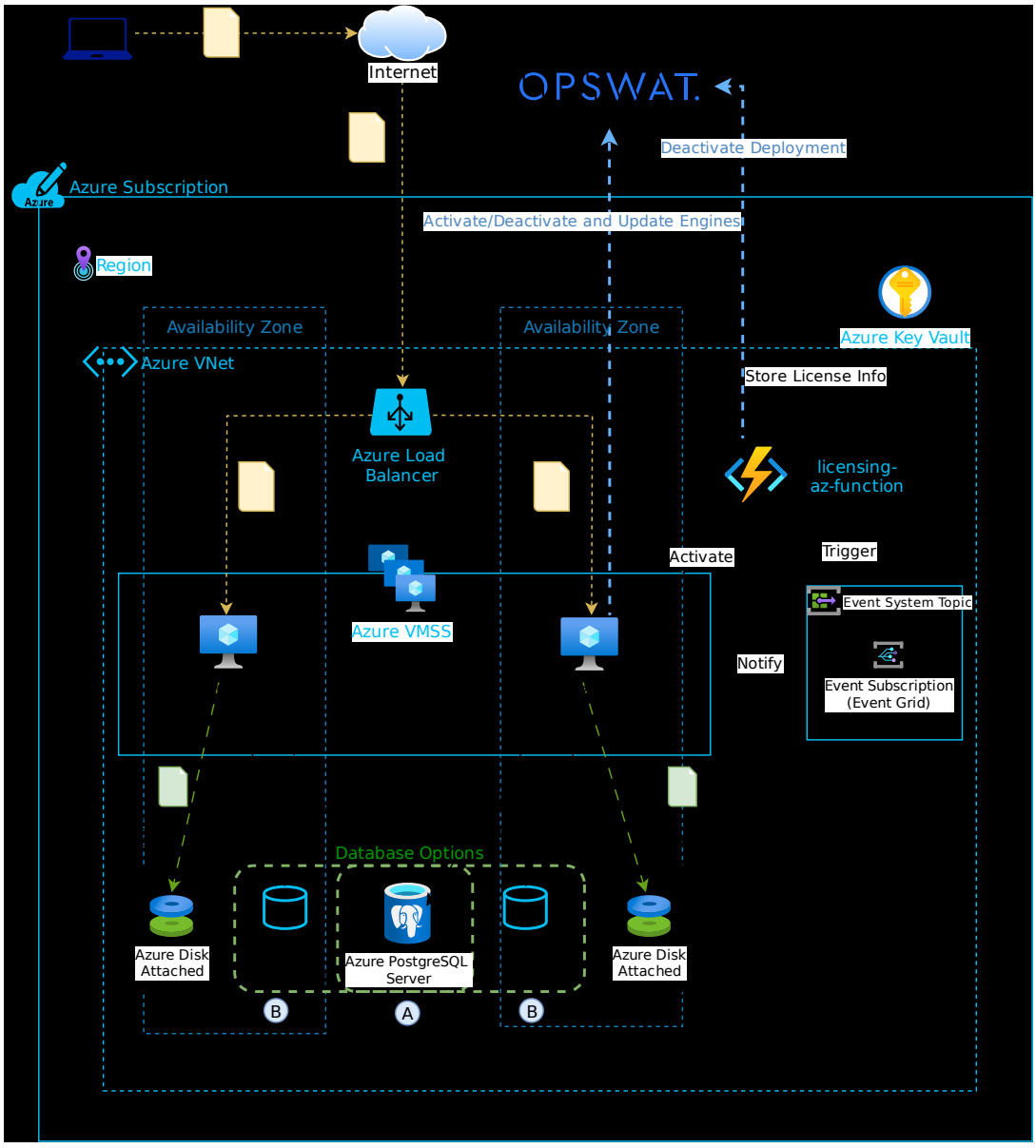
Storage

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

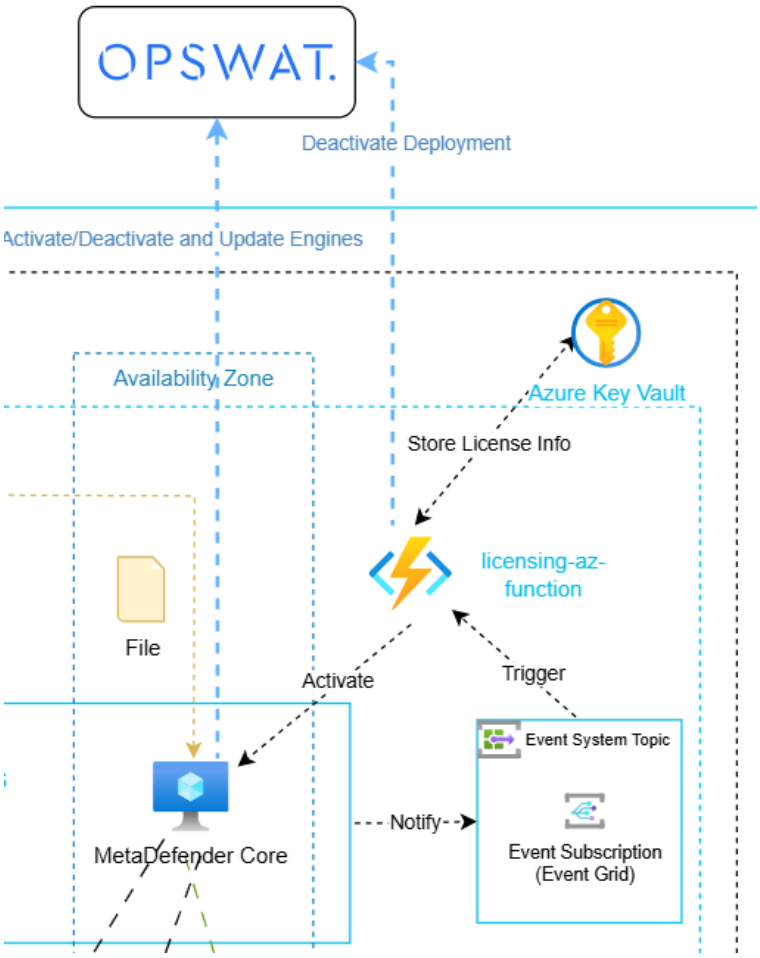
Licensing Automation on Azure

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
    a. License Key (licenseKey)
    b. API Key (apiKey)
    c. Core UI User (coreUser)
    d. 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
    a. SUBSCRIPTION_ID
    b. 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
    a. Subject Begins With: "/subscriptions/{subscription_id}/resourceGroups/{RG_NAME}/providers/Microsoft.Compute/virtualMachines"
6. Under Advanced Filters
    a. Key: data.operationName
    b. Values:
        i. "Microsoft.Compute/virtualMachines/start/action"
        ii. "Microsoft.Compute/virtualMachines/deallocate/action"
        iii. "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 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
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

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

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`

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"

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

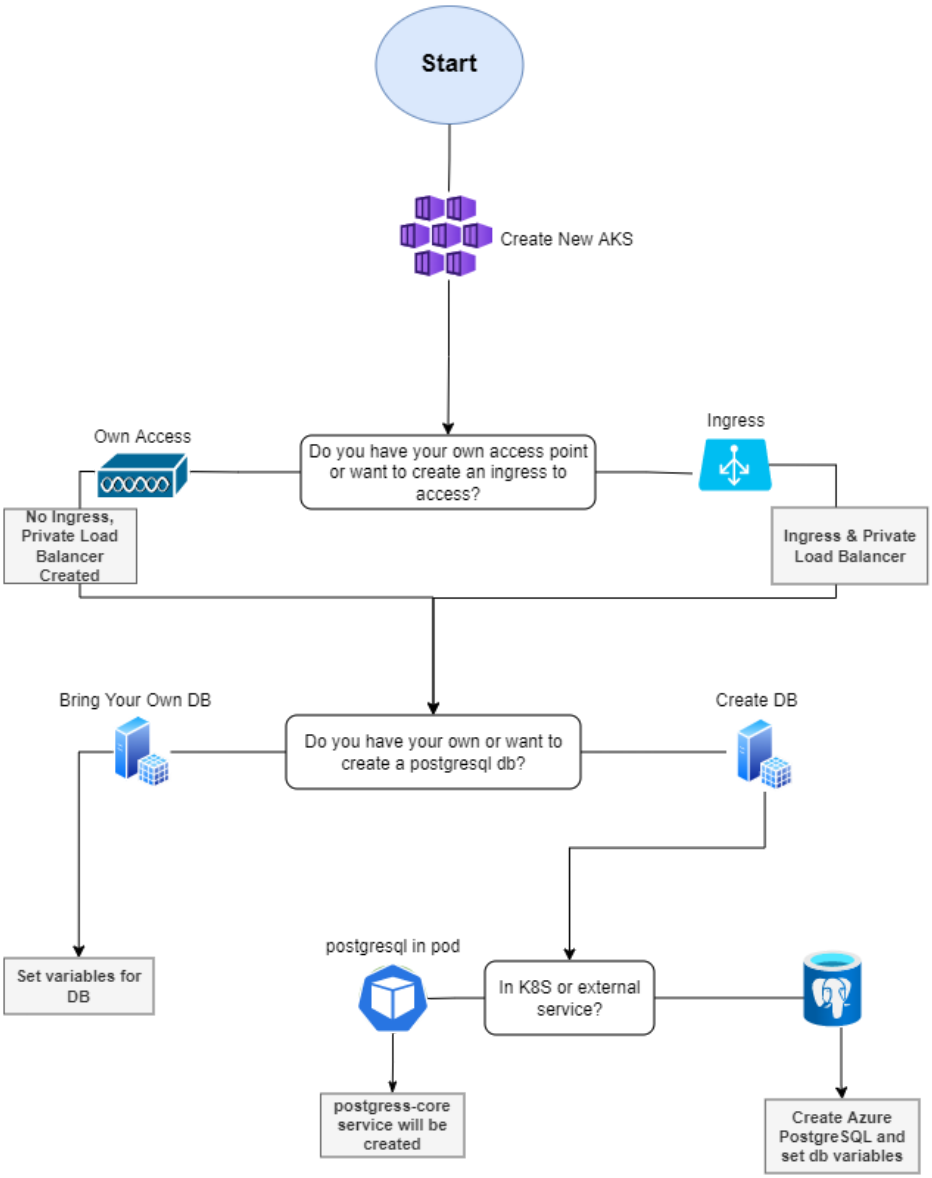
Parameter	Flags	Options	Default	Description	Required/Optional
Action		• provision • install		Action to indicate the script if we want to provision (Create resources + install Core) or install (Install Core)	**Required**
Location	-l or --location	• AWS • Azure • GCP		Where is going to be the K8S cluster	**Required**
MetaDefender Flag Installation	Combination of • --mdcore • --mdss • --icap		-	Install MetaDefender Core in the cluster provisioned	**Required**
Image Version	--image	latest 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 MetaDefender products will be installed in the K8S Cluster	Optional Max Characters: 10

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.
    a. **An Ingress** and an internal load balancer will be created per each product flag added as parameter to the script
    b. **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**
    a. **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.
    b. 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

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"

bash

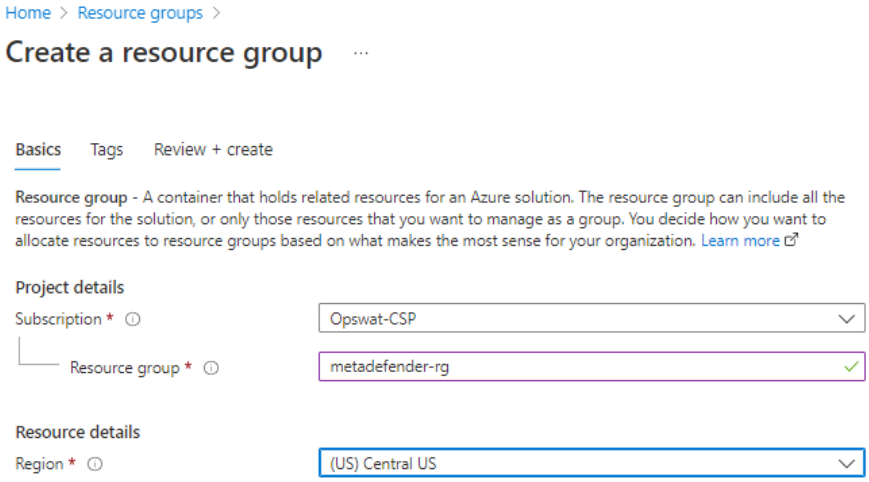
```
terraform init
terraform apply
```

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

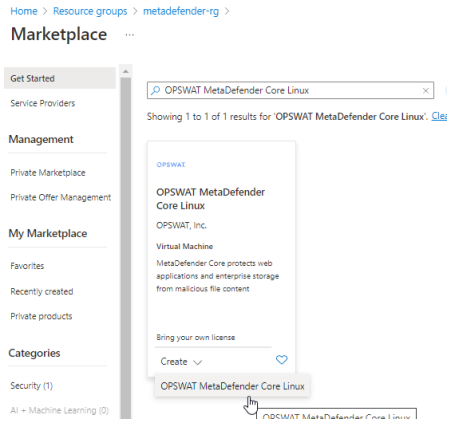
1. Create Resource Group

- Go to you Azure account and create a new resource group (we will refer to it as "metadefender-rg")

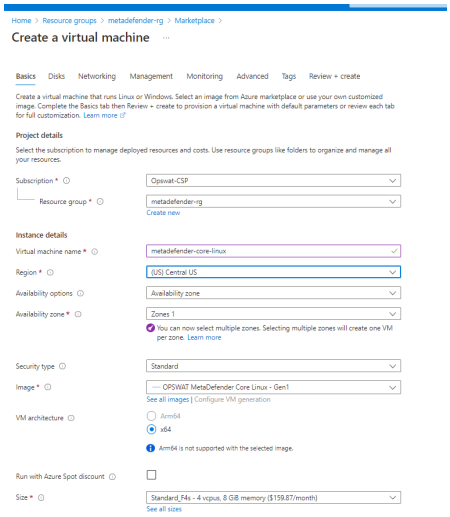


2. Create Azure VM

- 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

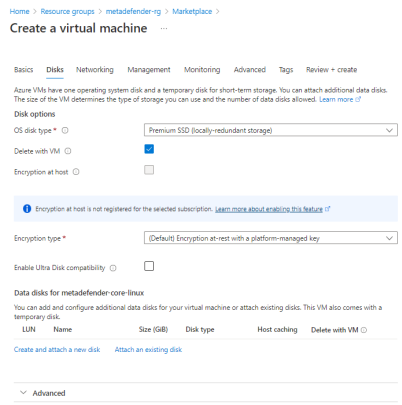
Storage step can be skipped.

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.

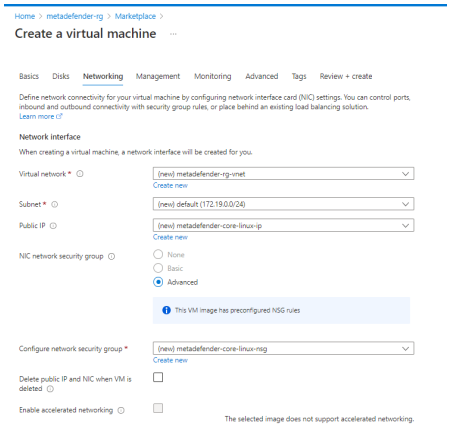
In case quarantining the files in the MetaDefender instance or analyzing high volumes of files is a requirement, please consider adding an additional Azure Managed Disk

- Premium SSD is recommended for having the best performance

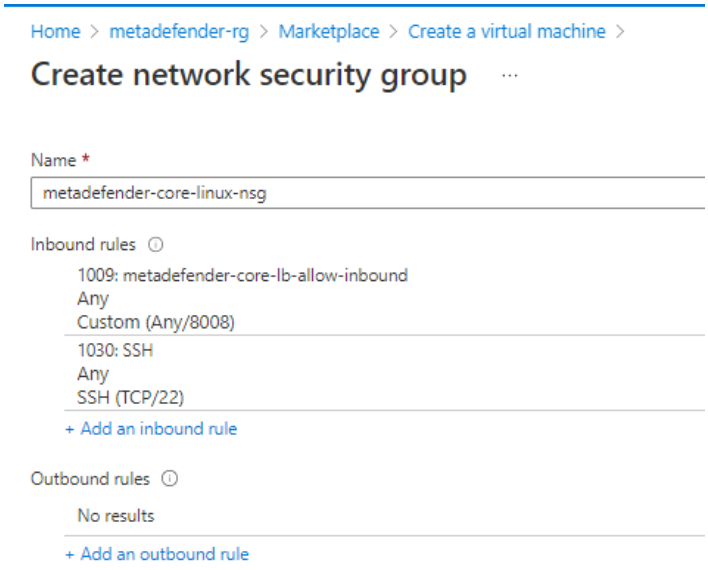


3. Network Configuration

- 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
    a. Generate an APIKEY from the UI and save it for later.
    b. 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

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"

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"

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

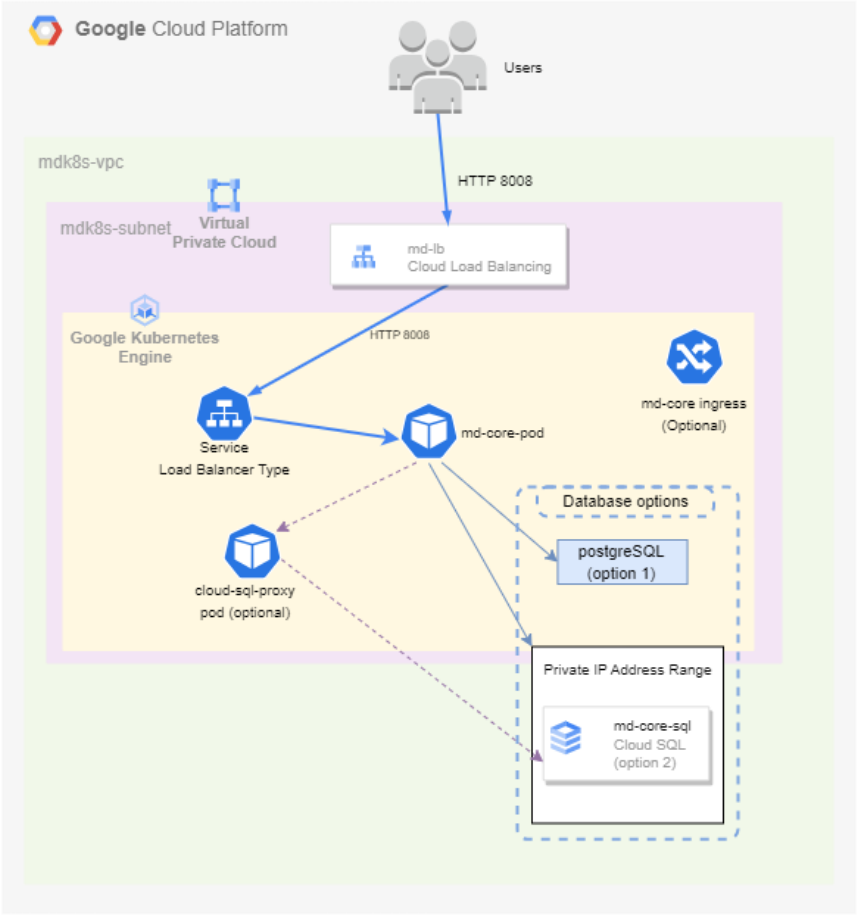
For running MD Core in a Kubernetes cluster we recommend to use, for production environments, an external database service for PostgreSQL. In case of **GCP** it is **GCP Cloud SQL** which the script will deploy and configure to be accessible from the cluster.

We offer different ways to access to the Cloud SQL Server from the GKE Cluster, these are the following ordered by recommended solutions

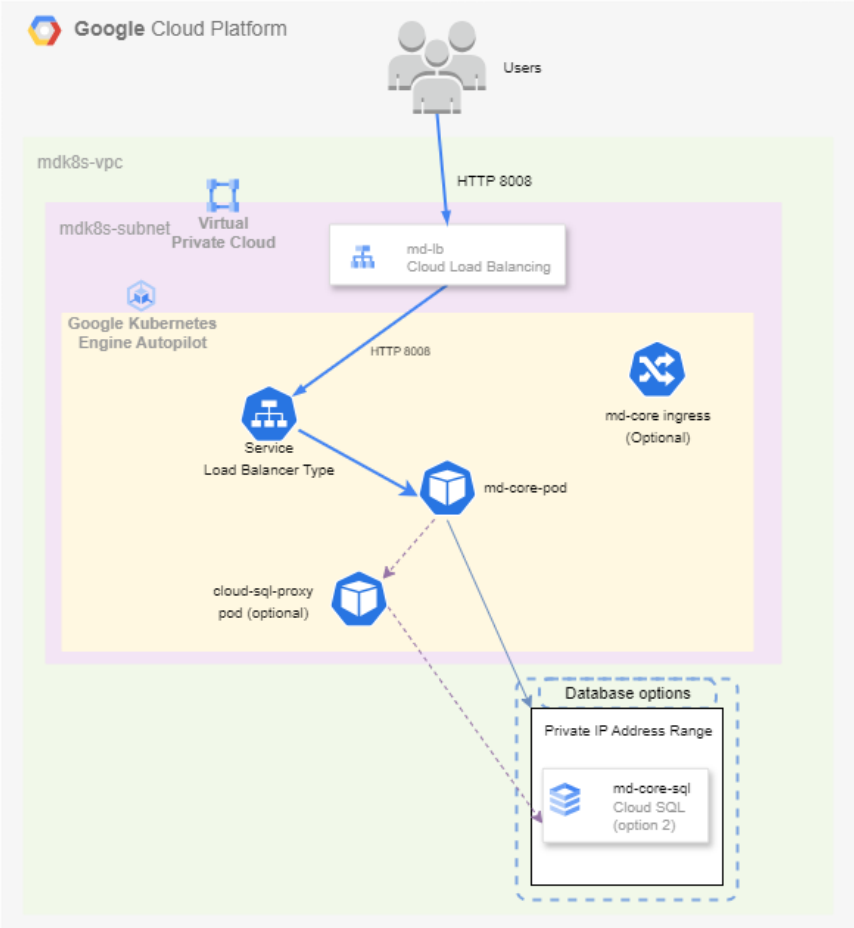
1. Private Connection
2. GCP Provided Connector in a Pod
3. GCP Provided Connector in a Sidecar container

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 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 SQL

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
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 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 MetaDefender products will be installed in the K8S Cluster	Optional Max Characters: 10

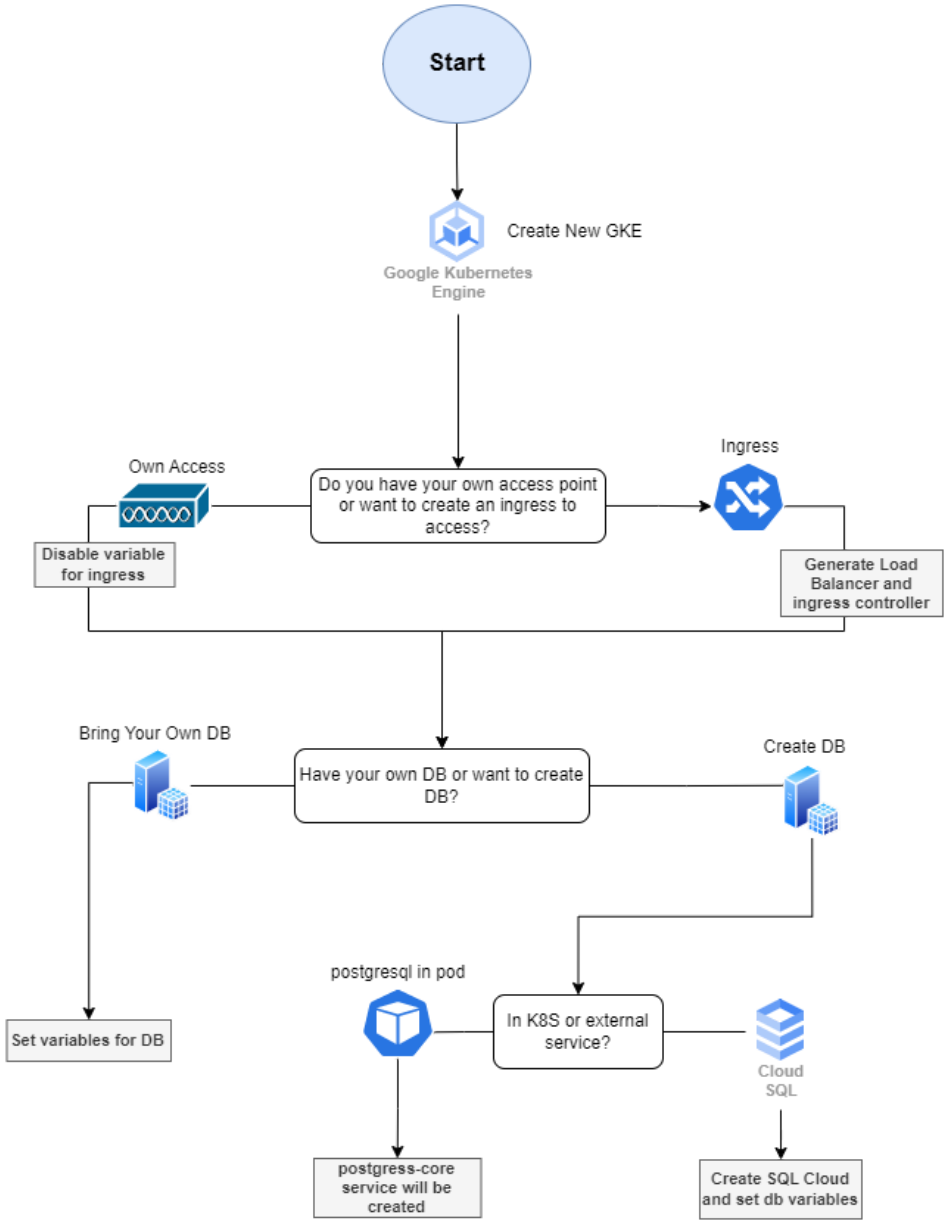
Connection to Cloud SQL Database from GKE

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.
    a. **An Ingress** and an internal load balancer will be created per each product flag added as parameter to the script
    b. **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**
    a. **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.
    b. Create new **DB in K8S** or **external DB** that for GCP we will provision a **Cloud SQL Server**
        i. **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:

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:

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:

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:

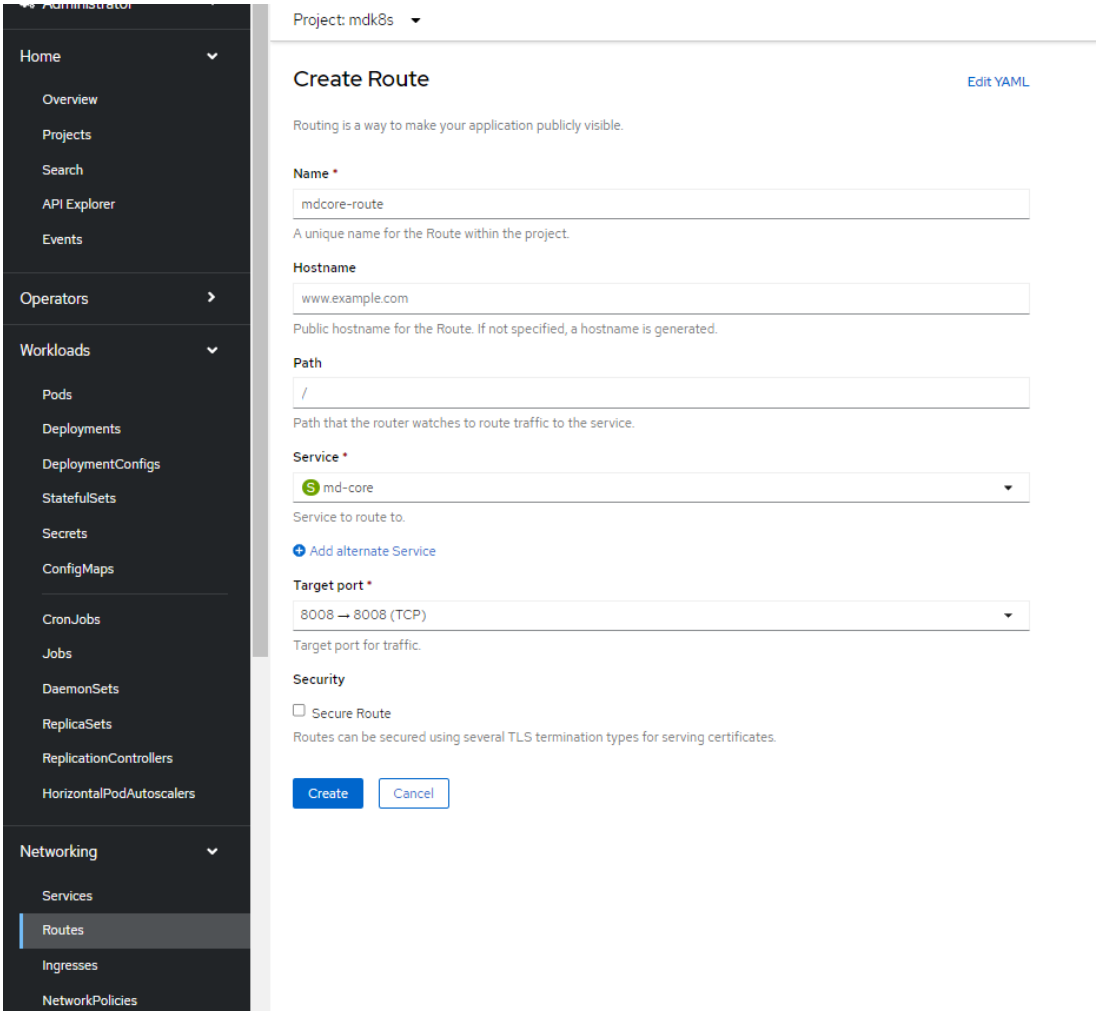
bash

```
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`

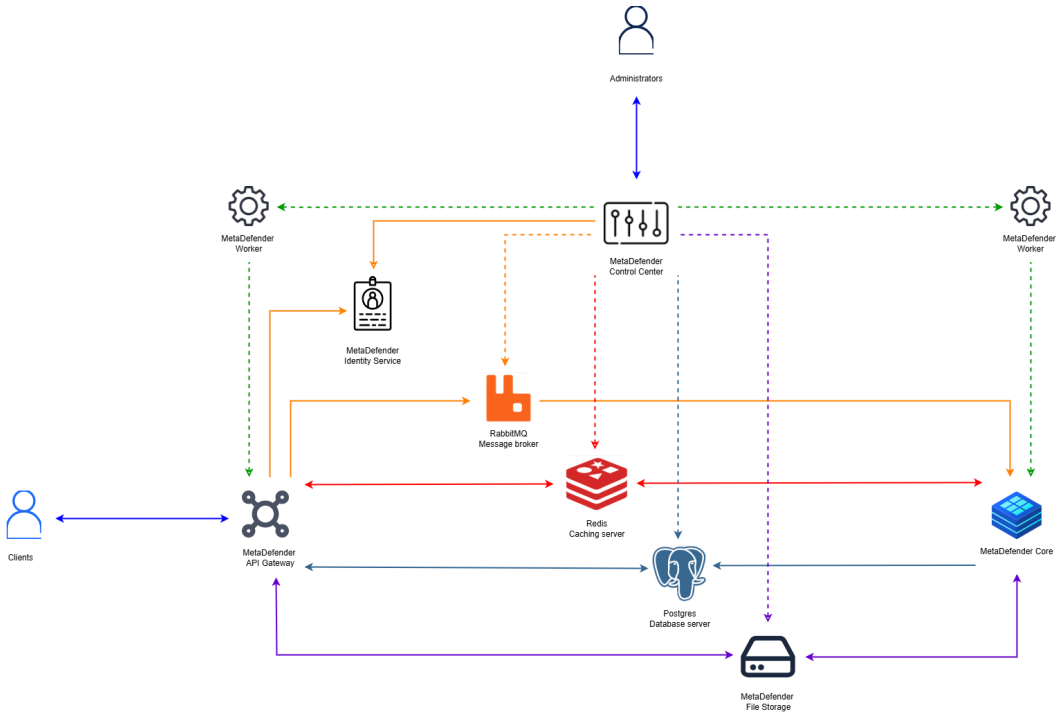


An ingress is also created by default and can be disabled by setting the `core_ingress.enabled` value to `false` .

MetaDefender Distributed Cluster Overview

Introduction

> **Roadmap**
> More features will be supported along the road. Learn more details at Roadmap.



The MetaDefender Distributed Cluster (MDDC) is an approach to serve very large deployments while offering improved auto-scaling, high availability and fault tolerant capabilities for MetaDefender Core.

Distributed Cluster consists of several components:

Component	Functionalities
MDDC Control Center	Assist administrators with user management, system health monitoring, and deploying or upgrading MetaDefender Core or API Gateway without any downtime.
MDDC Identity Service	Assist Control Center and API Gateway in client authentication, managing user activity sessions and authorization.
MDDC File Storage	Securely store and share files asynchronously across components in the cluster. The component manages the duration and duplication of files.
MDDC Worker	Deploy and monitor activities of MetaDefender Core and API Gateway.
MDDC API Gateway	Accept file scans, fetch scan statuses, and process download requests from clients.
MetaDefender Core	Scan the accepted files.
RabbitMQ - Message broker	Receives tasks from the API Gateway and forwards them to MetaDefender Core instances for processing.
Redis - Caching server	Store in-progress results in memory for rapid retrieval.
PostgreSQL - Database sever	Permanently store scan results, configuration and executive reports.

The Distributed Cluster offers users two distinct interfaces. The first is a RESTful interface provided by MDDC API Gateway for applications to upload files for scanning, retrieve scan status, download processed files, or abort file scanning. The other is a Web UI provided by MDDC Control Center for the system administrator to manage licenses and users, modify workflow configurations, monitor the overall system, and remotely deploy or upgrade MDDC API Gateway or MetaDefender Core.

When a file is submitted to the MDDC API Gateway for scanning, its body content is securely transmitted to MDDC File Storage for subsequent use. API Gateway submits a scan task in RabbitMQ queue, and responds to the application with `data_id`. The task is delivered to healthy MetaDefender Core instances and one of them will accept the task. The file corresponding to the task is transmitted from MDDC File Storage to the instance's local storage, and the processing of the file takes place. Scan results produced by the processing are continuously recorded in Redis for fast retrieval and are finally stored in the PostgreSQL database for long-term storage. If created, the sanitized or watermarked file is securely transmitted to MDDC File Storage for future download by API Gateway.

In certain rate situations, if one of the MetaDefender Core instances unexpectedly ceases operation, its 'broken' files are delivered to other MetaDefender Core instances for continued processing without the need for applications to resubmit the files. By leveraging MDDC File Storage and RabbitMQ, MetaDefender Core instances within MetaDefender Distributed Cluster can collaborate in distributing the workload of archive extraction, greatly decreasing the overall time required to process archive files while utilizing the resources much more efficiently.

Using the Web Console from by MDDC Control Center, the system administrator is able to adjust workflow settings centrally and, after which the updates are automatically synced across all MetaDefender Core instances. The administrator can scale out the number of MDDC API Gateway or MetaDefender Core instances if additional

power is required. He or she can also upgrade the instances seamlessly while the file processing is occurring. All statistical data and health information for components, along with executive reports, can be accessed easily through the Web UI of MDDC Control Center.

Recommended System Configuration

MetaDefender Distributed Cluster components

The architecture requires all of the following components to be set up:

1. PostgreSQL Database Server.
2. Redis Caching Server.
3. RabbitMQ Messaging Broker.
4. MetaDefender Distributed Cluster File Storage.
5. MetaDefender Distributed Cluster API Gateway.
6. MetaDefender Distributed Cluster Identity Service.
7. MetaDefender Distributed Cluster Control Center.
8. MetaDefender Distributed Cluster Worker.
9. MetaDefender Core, version 5.14.2 or above.

Each component should be set up separately on each networked server.

Besides the components mentioned above, the following installers must be submitted to Control Center in order to be deployed on Worker remotely.

1. MetaDefender Core, version 5.14.2 or above.
2. MetaDefender Distributed Cluster API Gateway, version 2.0.0 or above.

System requirements

	Platform	Dependencies	Recommended System Specs
PostgreSQL Database Server Version: 14.18 or above	64-bit Windows or Linux only.		Check details on relevant vendor 's site.
Redis Caching Server Version 7.0.5 or above	64-bit Linux only.		Check details on relevant vendor 's site.
RabbitMQ Messaging Broker Version 3.13.0 or above	64-bit Windows or Linux only.	64 bit Erlang/OTP 25.0 or above (erlang).	Check details on relevant vendor 's site.
MetaDefender Distributed Cluster File Storage	**64-bit Windows**: Windows 11 IoT 21-24H2; Windows Server 2019 latest build; Windows Server 2022. **64-bit Linux**: Debian 12.x; Ubuntu 22.04; RHEL 9.x; Rocky Linux 9.	Windows: Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above. Linux: `uuid` package.	At least 8 CPU cores and 8GB RAM available.
MetaDefender Distributed Cluster Control Center	**64-bit Windows**: Windows 11 IoT 21-24H2; Windows Server 2019 latest build; Windows Server 2022. **64-bit Linux**: Debian 12.x; Ubuntu 22.04; RHEL 9.x; Rocky Linux 9.	Windows: Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above. Linux: `uuid` package.	At least 4 CPU cores and 4GB RAM available.
MetaDefender Distributed Cluster Identity Service	**64-bit Windows**: Windows 11 IoT 21-24H2; Windows Server 2019 latest build; Windows Server 2022. **64-bit Linux**: Debian 12.x; Ubuntu 22.04; RHEL 9.x; Rocky Linux 9.	Windows: Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above. Linux: `uuid` package.	At least 4 CPU cores and 4GB RAM available.

	Platform	Dependencies	Recommended System Specs
MetaDefender Distributed Cluster Worker for **MetaDefender Distributed Cluster API Gateway**	**64-bit Windows**: Windows 11 IoT 21-24H2; Windows Server 2019 latest build; Windows Server 2022. **64-bit Linux**: Debian 12.x; Ubuntu 22.04; RHEL 9.x; Rocky Linux 9.	Windows: Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above. Linux: `uuid` package.	At least 4 CPU cores and 8GB RAM available.
MetaDefender Distributed Cluster Worker for **MetaDefender Core**	**64-bit Windows**: Windows 11 IoT 21-24H2; Windows Server 2019 latest build; Windows Server 2022. **64-bit Linux**: Debian 12.x; Ubuntu 22.04; RHEL 9.x; Rocky Linux 9.	Windows: Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above. Linux: `uuid` package.	Recommended System Configuration

Roadmap

API Endpoints

To offer seamless integration where customers do not need to change their existing integrations/applications, MetaDefender Distributed Cluster replicates most of MetaDefender Core's API endpoints such as

POST /file

GET /file

POST /file/batch

and many more at MetaDefender Distributed Cluster API documentation API.

Features

Feature	Support plan
I. File Submission methods	**Ongoing**
Streaming	Supported
Downloading (scan-from-link)	Q3 2025
Multi-part uploading	Q4 2025
Password-protected file	Supported
Local scan	Not supported
Synchronous scan	Not supported
Cancel a scan	Supported
II. File in Batch	**Ongoing**
Initiate, Close, Cancel a batch	Supported
Retrieve batch status	Supported
Download a signed batch result	Supported
Close a batch using Webhook	Q3 2025
III. Processing Result Retrieval	**Ongoing**
Fetch processing result by data ID	Supported
Fetch processing result by hash	Supported
Download sanitized file or DLP-processed file	Supported
View original file in case of being reused by hash	Supported
Transmit result using Webhook	Q3 2025
Transmit sanitized file using Webhook	Q3 2025
Export result as PDF / JSON	Q3 2025
Download SBOM full report	Q3 2025

Feature	Support plan
Download Adaptive Sandbox full report	Q3 2025
Download quarantined file	Q3 2025

IV. Licensing	Ongoing
Online activation	Supported
Offline activation	TBD
Check license information	Supported
Deactivation	Supported

V. Exclusive features	Ongoing
Load Balancing	Supported
Distribute extraction load among MetaDefender Core instances	Supported
Auto upgrade MetaDefender Core & Hub instances with zero downtime	Supported
Scaling with Ease for MetaDefender Core & Hub instances	Supported
At-rest data encryption	Supported
In-transit data encryption	Supported
High Availability supports File Storage, API Gateway and MetaDefender Core	Supported
High Availability for Redis and RabbitMQ	June 2025
Distribute compression load among MetaDefender Core instances	Q3 2025

VI. User Management	Ongoing
Authentication with local user account	Supported
Authentication with AD/LDAP account	Q4 2025
Authentication with SSO	Q1 2026

Feature	Support plan
Role-based access control	Supported
VII. Web Management Console	**Ongoing**
System Health	Supported
Executive Report	Supported
Processing History	Supported
Audit Logs	Partially Supported
Quarantine	Q3 2025
Workflow Management	Supported
Inventory/Services management	Supported
Inventory/Workers management	Supported
Module Update settings	Q3 2025
Data Retention settings	Partially Supported
Health Check settings	Supported
HTTPS enablement	Supported
Advanced TLS settings	Q3 2025
Proxy settings	Q3 2025
Configuration backup & restore	Q3 2025
Email Notification	Q4 2025
Schedule Reporting	Q4 2025
Rate Limiting for API Gateway	Q4 2025

Deployments

	Support plan
I. VM Deployment	Supported
Installers for Windows and Linux are available on My OPSWAT	Supported
All-in-one package & deploying script	TBD
II. Docker-based Deployment	Supported
Debian 12	Supported
III. Kubernetes Deployment	Jun 2025

Integration with MetaDefender Ecosystem

Product	Support plan
I. Integrate with MetaDefender ICAP	Supported
II. Integrate with MetaDefender Email Security	Supported
III. Integrate with MetaDefender Kiosk	Q3 2025
IV. Integrate with MetaDefender Storage Security	Q3 2025
V. Integrate with MetaDefender Software Supply Chain	Q4 2025
VI. Integrate with MetaDefender Managed File Transfer	Q4 2025

Installation

Supported deployment fashions

For MetaDefender Distributed Cluster deployment, we support both two deployment approaches:

1. Setup manually via component installers at your choice environments (on-prem, containerization, cloud): VM Setup.
2. Or via using docker images on the containerization environment; Container-Based Setup.

Artifacts location

MetaDefender Distributed Cluster components' **docker images** could be also found on OPSWAT central repository at Docker Hub: https://hub.docker.com/u/opswat

- For Redis, RabbitMQ and PostgreSQL docker images: please refer to installation guide Container-Based Setup for details

Installers could be found and downloaded on the OPSWAT portal at https://my.opswat.com/portal/home/metadefender-core.

- For RabbitMQ and PostgreSQL installers: please refer to installation guide Container-Based Setup for details

Alternatively you can find related component installers by accessing MetaDefender Core product tab https://my.opswat.com/portal/home/metadefender-core, Modules & Utilities section lists all the components and its docker toolkits.

For manual installation, let's download the following installers to your machine and head to VM Setup.

- MetaDefender Distributed Cluster Identity Service
- MetaDefender Distributed Cluster File Storage
- MetaDefender Distributed Cluster Control Center
- MetaDefender Distributed Cluster Worker

Overview / MetaDefender Core



MetaDefender Core enables you to integrate advanced malware prevention and detection capabilities into your existing IT solutions and infrastructure for better handling common attack vectors: securing web portals from malicious file upload attacks, augmenting cybersecurity products, and developing your own malware analysis systems.

Important Notice: End of Support for Java 8 and Java 11: The Adaptive Sandbox engine requires Java 17 from March 31, 2025. Please upgrade your Core system if you are currently using Java 8 or 11.

Type Software

Solutions Application Security, Security Operations

Resources

- Product page on OPSWAT Corporate Site
- Documentation
- Differences in MetaDefender Core v4 and v5
- MetaDefender Core Sample Code
- Anti-malware Engine Status

Download | View Licenses

Modules & Utilities

Software Utility
MetaDefender Distributed Cluster Identity Service
MetaDefender Distributed Cluster is a scalable, disruption-resistant deployment architecture for MetaDefender Core,...

Software Utility
MetaDefender Distributed Cluster Control Center
MetaDefender Distributed Cluster is a scalable, disruption-resistant deployment architecture for MetaDefender Core,...

Software Utility
MetaDefender Distributed Cluster Worker for API Gateway Docker Toolkit
MetaDefender Distributed Cluster is a scalable, disruption-resistant deployment architecture for MetaDefender Core,...

Software Utility
MetaDefender Distributed Cluster File Storage Docker Toolkit
MetaDefender Distributed Cluster is a scalable, disruption-resistant deployment architecture for MetaDefender

MetaDefender Distributed Cluster Control Center

Platform	Version | Release date
Debian 12 / Ubuntu 22.04 ✓	2.0.0 ✓
Redhat Enterprise Linux / Rocky Linux 9	
Windows 11 IoT / Windows Server 2022	

Release date:	May 29, 2025
MD5:	f65242e34409a74a87fca660944c15cf
SHA1:	50501ffe48beb4ae815028196e1dcdc34c6a8779
SHA256:	592d8077fcdbdf71032dfac4fdf0ce4a5eb2dcb78ab8a1c8729bbffb7e7cec1b

WGET Link | CURL Link | Download - 28 MB

Contact OPSWAT Sales to purchase licenses.

VM Setup

Prerequisite
Before executing the setup, please ensure the service's Minimum System Requirements are met and that any necessary dependencies are installed.

Setup order requirement

MetaDefender Distributed Cluster consists of the following components along with their corresponding default ports.

Step	Component	How to install (short name)	Default port
1	Redis Caching Server	Redis	6379
2	RabbitMQ Message Broker	RabbitMQ	5672
3	PostgreSQL Database Server	Postgres	5432
4	MetaDefender Distributed Cluster File Storage	MDDC File Storage	8890
5	MetaDefender Distributed Cluster Identity Service	MDDC Identity Service	8891
6	MetaDefender Distributed Cluster Control Center	MDDC Control Center	8892
7	MetaDefender Distributed Cluster Worker	MDDC Worker	8893

The system administrator should adhere to the following service installation sequence to prevent conflicts:

1. Install Redis, RabbitMQ, Postgres, MDDC File Storage, MDDC Identity Service.
2. Install MDDC Control Center.
3. Install MDDC Worker on the targeted machines (for deploying MDDC API Gateway or MetaDefender Core).

Warning
An exception rule for the firewall needs to be created to permit both incoming (inbound) and outgoing (outbound) connections to every component.

> **Info**
> While many components can be set up on a single machine, they should be installed individually on different machines according to their features. Kindly consult Best practices for further information.

Install Redis Caching Server

> **Info**
> Redis version 7.0 or higher is required.
>
> Only Redis on Linux is officially recommended.

1. Follow steps to install Redis Caching server.
2. Access Redis configuration file `/etc/redis/redis.conf` for editing.
3. Comment out the `bind` setting and set `protected-mode` option to **no**.

redis.conf none

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

4. Restart the service.

bash

```
# Red Hat/Rocky
 $ sudo systemctl enable redis
 $ sudo systemctl restart redis

# Debian/Ubuntu
$ sudo systemctl enable redis-server
$ sudo systemctl restart redis-server
```

Install RabbitMQ Message Broker

Info
RabbitMQ version 3.13.0 or higher is required.

Warning
RabbitMQ functions effectively only with specific supported versions of Erlang. Please refer to the link for the Erlang-RabbitMQ compatibility matrix.

Windows

1. Download Erlang and follow the instructions to install Erlang.
2. Download RabbitMQ for Windows.
3. Run the executable file as administrator, follow instructions to complete the RabbitMQ installation.
4. In Command Prompt, change working directory to `<RabbitMQ installation folder>/rabbitmq_server-<version>/sbin` and run the following command:

None bash

```
> rabbitmqctl.bat add_user <username> <password>
> rabbitmqctl.bat set_permissions -p / <username> "." "." "."
> rabbitmqctl.bat set_user_tags <username> administrator
```

Linux

1. Download Erlang and follow the instructions to install Erlang and its dependencies.
2. Download RabbitMQ for Red Hat/Rocky or Debian/Ubuntu .
3. In Terminal, run the following command:

bash

```
# Red Hat/Rocky
$ sudo rpm -Uvh --nodeps rabbitmq-server-<rabbitmq version>.el8.noarch.rpm
$ sudo systemctl enable rabbitmq-server
$ sudo systemctl start rabbitmq-server

# Debian/Ubuntu
$ sudo dpkg -i rabbitmq-server_<rabbitmq version>_all.deb
```

4. In Terminal, run the following command:

None bash

```
$ sudo rabbitmqctl add_user <username> <password>
$ sudo rabbitmqctl set_permissions -p / <username> "." "." "."
$ sudo rabbitmqctl set_user_tags <username> administrator
```

Install PostgreSQL Database Server

> **Info**
> PostgreSQL version 14.18 or higher is required.
>
> `pg_trgm` extension is required for PostgreSQL running on Linux.

1. Download PostgreSQL Database Server.
2. Follow steps to setup Postgres Database Server to allow connections from external applications.
3. Restart Postgres Database Server.

Install MetaDefender Distributed Cluster File Storage

1. Build Ignition file for MDDC File Storage service.
2. Start Command Prompt as Administrator on Windows or Terminal on Linux and run the following command:

bash

```
# Windows
> msiexec.exe /i <mddc_file_storage_package> /qn

# Debian or Ubuntu
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_file_storage_package> || sudo apt install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_file_storage_package> -y
```

3. Check the service status.

Install MetaDefender Distributed Cluster Identity Service

1. Build Ignition file for MDDC Identity Service.
2. Start Command Prompt as Administrator on Windows or Terminal on Linux and run the following command:

bash

```
# Windows
> msiexec.exe /i <mddc_identity_service_package> /qn

# Ubuntu or Debian
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_identity_service_package> || sudo apt install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_identity_service_package> -y
```

3. Check the service status.

Install MetaDefender Distributed Cluster Control Center

1. Build Ignition file for MDDC Control Center service.
2. Start Command Prompt as Administrator on Windows or Terminal on Linux and run the following command:

bash

```
# Windows
> msiexec.exe /i <mddc_control_center_package> /qn

# Ubuntu or Debian
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_control_center_package> || sudo apt install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_control_center_package> -y
```

3. Check the service status.

Setup Data Lake and Data Warehouse

1. Go to `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Control Center` directory in Windows Command Prompt or `/usr/sbin` directory in Linux Terminal.
2. Run the following command:

bash

```
# Windows
> mddc-dbready.exe --host=<postgres-host> --port=<postgres-port> --user=
<postgres-user> --password=<postgres-password> --target=lake,warehouse

# Linux (Ubuntu, Debian, Red Hat or Rocky)
$ mddc-dbready --host=<postgres-host> --port=<postgres-port> --user=<postgres-
user> --password=<postgres-password> --target=lake,warehouse
```

Info
Make sure the `postgres-user` possesses **superuser** rights to successfully create the database.

Install MetaDefender Distributed Cluster Worker

You need to prepare at least 2 VMs for Worker: one for MetaDefender Core and one for API Gateway.

1. Build Ignition file for MDDC Worker service.
2. Start Command Prompt as Administrator on Windows or Terminal on Linux and run the following command:

bash

```
# Windows
> msiexec.exe /i <mddc_worker_package> /qn

# Ubuntu or Debian
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_worker_package> || sudo apt install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_worker_package> -y
```

3. Check the service status.
4. Repeat the above steps for other MDDC Workers.

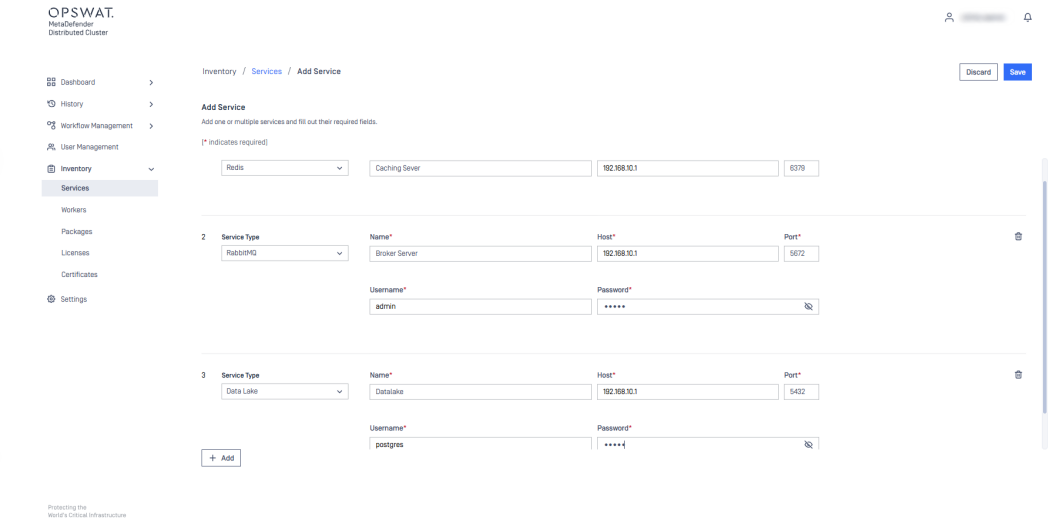
Configurations

When all the Workers are installed successfully, it marks a completed installation. Now, heading to essential configuration steps.

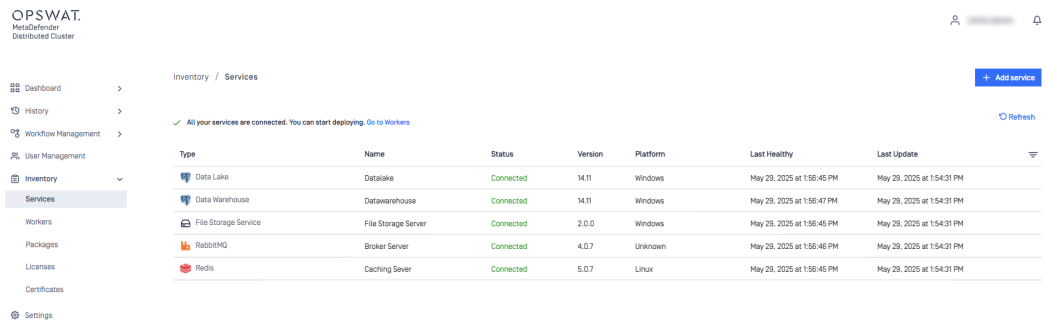
Connect essential services

> **Info**
> Essential services for the Distributed Cluster includes Redis, Postgres, RabbitMQ, and File Storage.
>
> The system is operational only when MDDC Control Center can effectively connect to all essential services.

1. Sign in to MDDC Control Center web console with the initial administrator user account that you created in Install MetaDefender Distributed Cluster Identity Service.
2. Go to `Inventory` > `Services` and select **`Add Service`**.
3. Choose the service type and fill out the fields required by the services.
4. Click **Save**.



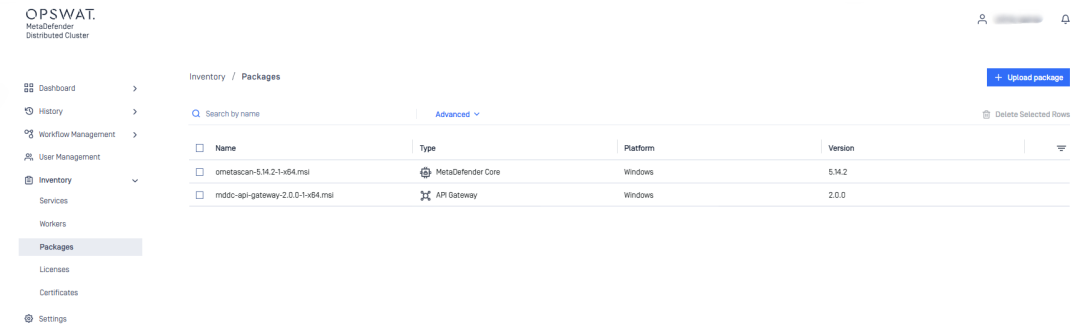
5. Check status of the service connections.



Submit MetaDefender Core and MDDC API Gateway packages

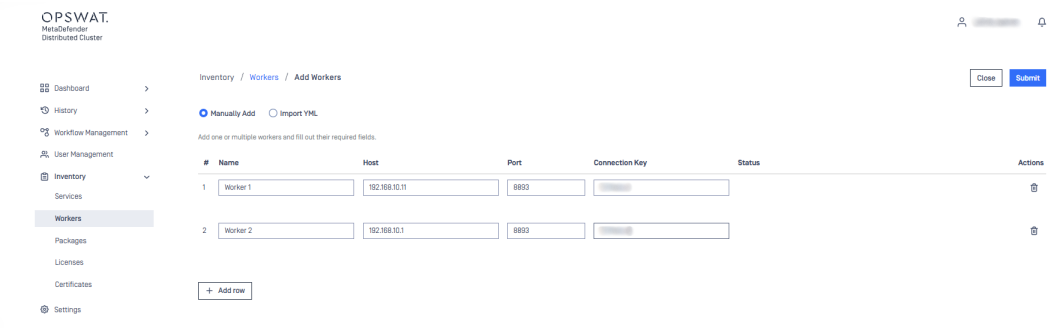
> **Info**
> The packages are installation files of MetaDefender Core and MDDC API Gateway, which will be subsequently deployed on MDDC Worker remotely by MDDC Control Center.
>
> Various versions of installation files may be submitted to MDDC Control Center. The correct version to install will be chosen during Deployment phase.

1. Sign in to MDDC Control Center web console with the initial administrator user account that you created in Install MetaDefender Distributed Cluster Identity Service.
2. Go to `Inventory` > `Packages` and select `Upload package`.
3. Select MetaDefender Core/MDDC API Gateway installation files.
4. Click **`Upload`**.

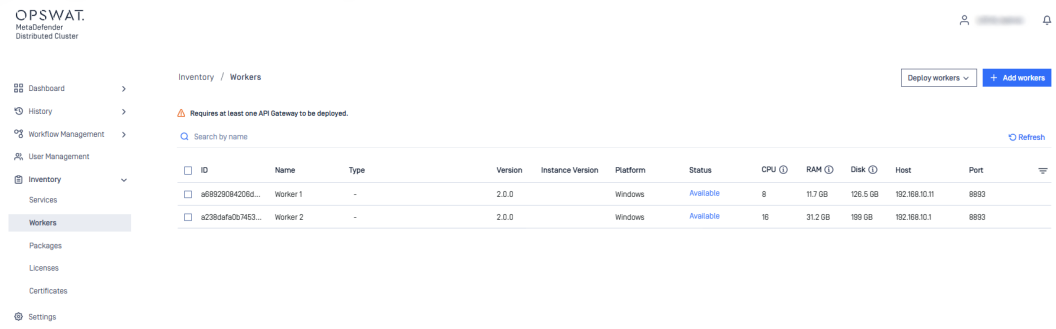


Connect to MDDC Workers

1. Sign in to MDDC Control Center web console with the initial administrator user account that you created in Install MetaDefender Distributed Cluster Identity Service.
2. Go to `Inventory` > `Workers` and select `Add workers`.
3. Complete the required fields to add new workers and select `Submit`.

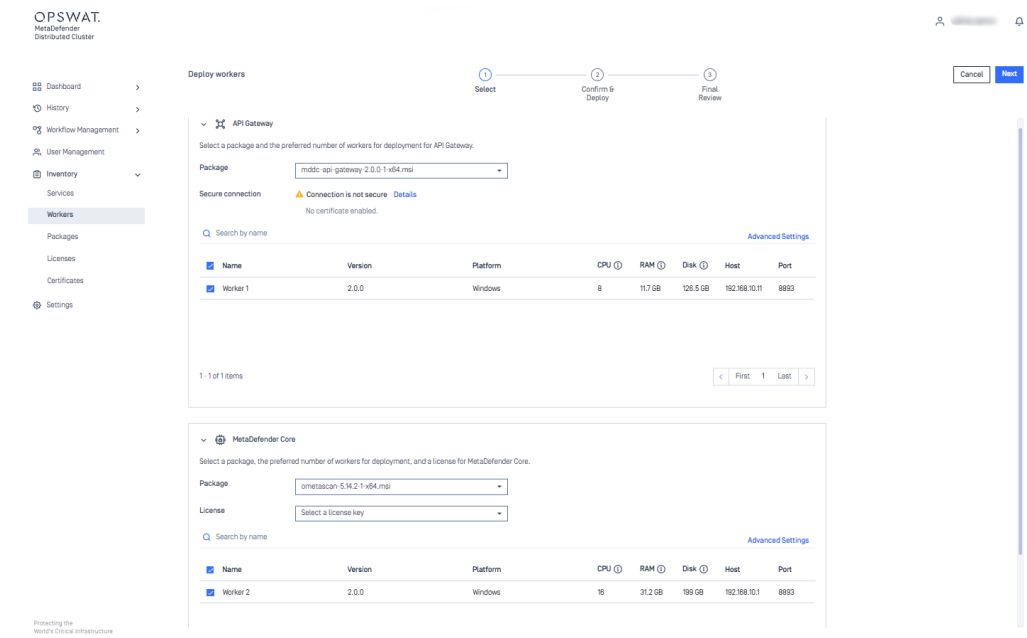


3. Check the status of the Worker connections.



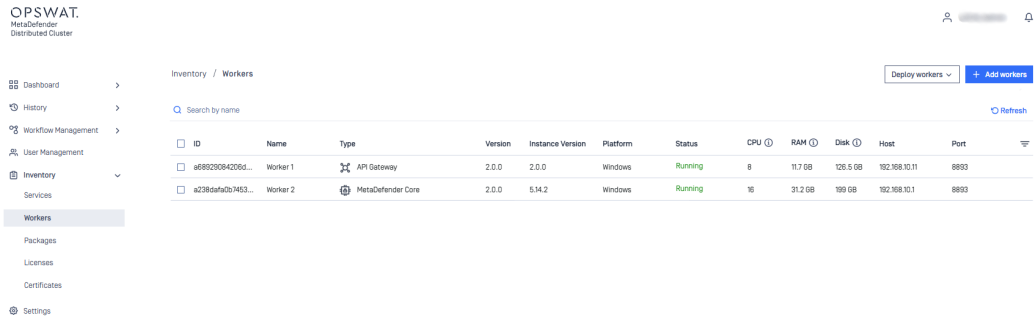
Deploy MetaDefender Core and API Gateway instances

1. Sign in to MDDC Control Center web console with the initial administrator user account that you created in Install MetaDefender Distributed Cluster Identity Service.
2. Go to `Inventory` > `Workers` and select `Deploy workers`.
3. Choose the workers for deployment and decide which package will be deployed on the workers.



> **Info**
> The license to activate MetaDefender Core instances can be selected in this phrase.
>
> Advanced settings (enable HTTPS, select log level, define engine parallel count, etc.) are configurable.

3. Choose `Next`.
4. Confirm the deployment details, then click `Deploy` and `Finish`.
5. Hold off until the deployment is completed successfully.



6. Once the system is up, MDDC API Gateway can efficiently accept scan requests.

Container-Based Setup

Prerequisite
Before running the setup, please check Recommended System Configuration to install all required dependencies of MetaDefender Distributed Cluster (MDDC).

Setup order requirement

Please follow installation steps **in order** to complete the system setup properly:

Order no.	Service	Notes
1	Redis, RabbitMQ, PostgreSQL, Identity Service and File Storage	• Could be setup in parallel in any order among them. • Make sure they are all fully functional and accessible before proceeding to the next setup order #2.
2	Control Center	• Ensure it's able to connect to those services in #1 • Make sure it is fully functional and accessible.
3	Worker for API Gateway and Worker for MetaDefender Core	• Could be setup in parallel in any order among them. • Ensure they're able to connect to Control Center • Make sure they are all fully functional and accessible.

Image name and version

All the images can be found at OPSWAT Docker Hub with the following information:

Note: **version** is the currently release version.

MetaDefender Distributed Cluster Identity Service

Docker image bash

```
opswat/metadefender-distributed-cluster:identity-service-<version>-debian-12
```

MetaDefender Distributed Cluster File Storage

Docker image bash

```
opswat/metadefender-distributed-cluster:file-storage-<version>-debian-12
```

MetaDefender Distributed Cluster Control Center

Docker image bash

```
opswat/metadefender-distributed-cluster:control-center-<version>-debian-12
```

MetaDefender Distributed Cluster Worker for API Gateway

Docker image bash

```
opswat/metadefender-distributed-cluster:worker-api-gateway-<version>-debian-12
```

MetaDefender Distributed Cluster Worker for MetaDefender Core

Docker image bash

```
opswat/metadefender-distributed-cluster:worker-core-<version>-debian-12
```

Environment variables

1. MDDC Identity Service

Environment Variable	Necessity	Description
`MDDC_IDENTITY_SERVICE_DB_HOST`	Required	Provide the database host for Identity Service
`MDDC_IDENTITY_SERVICE_DB_PORT`	Optional	Provide the database port for Identity Service Default: 5432
`MDDC_IDENTITY_SERVICE_DB_USER`	Required	Provide the database user for Identity Service
`MDDC_IDENTITY_SERVICE_DB_PASSWORD`	Required	Provide the database password for Identity Service
`MDDC_USER`	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker

Environment Variable	Necessity	Description
`MDDC_PASSWORD`	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_EMAIL`	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker

Environment Variable	Necessity	Description
`MDDC_APIKEY`	Optional	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_IDENTITY_SERVICE_CONNECTION_KEY`	Required	Define the connection key in order to register to Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
`MDDC_IDENTITY_SERVICE_PORT`	Optional	Define the expose port for Identity Service Default: 8891
`LOG_LEVEL`	Optional	Define the log level. Default value: info Accepted values: info/debug/error/warning

Start Identity Service container with docker run:

bash

```
docker run -d --name mddc-identity-service \
        -e MDDC_IDENTITY_SERVICE_DB_HOST=postgres-is \
  -e MDDC_IDENTITY_SERVICE_DB_USER=admin \
  -e MDDC_IDENTITY_SERVICE_DB_PASSWORD=admin \
  -e MDDC_IDENTITY_SERVICE_CONNECTION_KEY=1234abcd \
  -e MDDC_USER=admin \
  -e MDDC_PASSWORD=admin \
  -e MDDC_EMAIL=admin@local \
  -p 8891:8891 opswat/metadefender-distributed-cluster:identity-service-2.0.0-
debian-12
```

2. MDDC File Storage

Environment Variable	Necessity	Description
`MDDC_FILE_STORAGE_CONNECTION_KEY`	Required	Define the connection key in order to register to Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
`MDDC_FILE_STORAGE_PORT`	Optional	Define the expose port for File Storage Default: 8890
`LOG_LEVEL`	Optional	Define the log level. Default value: info Accepted values: info/debug/error/warning

Note: Persistent storage is located at `/opt/opswat/mddc-file-storage`. If end-users require data to be retained across container lifecycles, they must mount a volume to this path with `777` permissions to ensure full read/write access for all processes.

Start File Storage container with docker run

bash

```
docker run -d --name mddc-file-storage \
        -e MDDC_FILE_STORAGE_CONNECTION_KEY=1234abcd \
  -e MDDC_FILE_STORAGE_PORT=8890 \
  -p 8890:8890 opswat/metadefender-distributed-cluster:file-storage-2.0.0-debian-
12
```

3. MDDC Control Center

Environment Variable	Necessity	Description
MDDC_LAKE_DB_HOST	Optional	Provide the database host for Data Lake. In case that the end-user does not have the Data Lake, it's required to provide this variable to automate the database preparation
MDDC_LAKE_DB_PORT	Optional	Provide the database port for Data Lake Default: 5432
MDDC_LAKE_DB_USER	Optional	Provide the database user for Data Lake. In case that the end-user does not have the Data Lake, it's required to provide this variable to automate the database preparation
MDDC_LAKE_DB_PASSWORD	Optional	Provide the database password for Data Lake. In case that the end-user does not have the Data Lake, it's required to provide this variable to automate the database preparation
MDDC_WAREHOUSE_DB_HOST	Optional	Provide the database host for Data Warehouse. In case that the end-user does not have the Data Warehouse, it's required to provide this variable to automate the database preparation
MDDC_WAREHOUSE_DB_PORT	Optional	Provide the database port for Data Warehouse Default: 5432
MDDC_WAREHOUSE_DB_USER	Optional	Provide the database user for Data Warehouse. In case that the end-user does not have the Data Warehouse, it's required to provide this variable to automate the database preparation
MDDC_WAREHOUSE_DB_PASSWORD	Optional	Provide the database password for Data Warehouse. In case that the end-user does not have the Data Warehouse, it's required to provide this variable to automate the database preparation
MDDC_CACHE_HOST	Optional	Provide the caching host (Redis).
MDDC_CACHE_PORT	Optional	Provide the caching port (Redis)
MDDC_BROKER_HOST	Optional	Provide the broker host (RabbitMQ)
MDDC_BROKER_PORT	Optional	Provide the broker port (RabbitMQ)
MDDC_BROKER_USER	Optional	Provide the broker username (RabbitMQ)
MDDC_BROKER_PASSWORD	Optional	Provide the broker password (RabbitMQ)

Environment Variable	Necessity	Description
`MDDC_CONTROL_CENTER_DB_HOST`	Required	Provide the database host for Control Center
`MDDC_CONTROL_CENTER_DB_PORT`	Optional	Provide the database port for Control Center Default: 5432
`MDDC_CONTROL_CENTER_DB_USER`	Required	Provide the database username for Control Center
`MDDC_CONTROL_CENTER_DB_PASSWORD`	Required	Provide the database password for Control Center
`MDDC_USER`	Required	Provide the administrator account that is defined in Identity Service. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker

Environment Variable	Necessity	Description
`MDDC_PASSWORD`	Required	Provide the administrator account that is defined in Identity Service. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_APIKEY`	Optional	Provide the administrator account that is defined in Identity Service. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_IDENTITY_SERVICE_HOST`	Required	Provide the Identity Service host in order to add it to Control Center

Environment Variable	Necessity	Description
`MDDC_IDENTITY_SERVICE_PORT`	Optional	Provide the Identity Service port in order to add it to Control Center Default: 8891
`MDDC_IDENTITY_SERVICE_CONNECTION_KEY`	Required	Provide the Identity Service connection key in order to add it to Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
`MDDC_CONTROL_CENTER_ENCRYPTION_KEY`	Required	Define the encryption key for communication between Control Center and the services. Must be 32 characters long and contain only lowercase letters (a–z) and digits (0–9)
`MDDC_FILE_STORAGE_HOST`	Required	Provide the File Storage host in order to add it to Control Center
`MDDC_FILE_STORAGE_PORT`	Optional	Provide the File Storage port in order to add it to Control Center Default: 8890
`MDDC_FILE_STORAGE_CONNECTION_KEY`	Required	Provide the File Storage connection key in order to add it to Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
`MDDC_CERT_PATH`	Optional	Provide the directory path that contains the certificate and private key in order to enable https Note: when provide this variable, it's supposed to mount this path to `/certs/` as volume For example: `--volume /your-path:/certs`
`LOG_LEVEL`	Optional	Define the log level. Default value: info Accepted values: info/debug/error/warning

Start Control Center container with Docker run

bash

```
docker run -d --name mddc-control-center \
        -e MDDC_CONTROL_CENTER_DB_HOST=postges-cc \
  -e MDDC_CONTROL_CENTER_DB_USER=admin \
  -e MDDC_CONTROL_CENTER_DB_PASSWORD=admin \
  -e MDDC_IDENTITY_SERVICE_HOST=mddc-identity-service \
  -e MDDC_USER=admin \
  -e MDDC_PASSWORD=admin \
  -e MDDC_IDENTITY_SERVICE_CONNECTION_KEY=1234abcd \
  -e MDDC_CONTROL_CENTER_ENCRYPTION_KEY=12345678123456781234567812345678 \
  -e MDDC_FILE_STORAGE_HOST=mddc-file-storage \
  -e MDDC_FILE_STORAGE_CONNECTION_KEY=1234abcd \
  -e MDDC_CERT_PATH=/certs \
  -v /new-certificates:/certs
  -p 8892:8892 opswat/metadefender-distributed-cluster:control-center-2.0.0-
debian-12
```

4. MDDC Worker for API Gateway

Environment Variable	Necessity	Description
`MDDC_WORKER_CONNECTION_KEY`	Required	Define the connection key in order to register to Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
`MDDC_WORKER_PORT`	Optional	Define the expose worker's port. Default: 8893
`MDDC_WORKER_HOST`	Optional	Define the worker's host address. If it's not specified, it will get the container's internal IP address
`MDDC_CONTROL_CENTER_HOST`	Required	Provide the Control Center's host address
`MDDC_CONTROL_CENTER_PORT`	Optional	Provide the Control Center's port Default: 8892
`MDDC_USER`	Required	Provide the administrator account that is defined in Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker

Environment Variable	Necessity	Description
`MDDC_PASSWORD`	Required	Provide the administrator account that is defined in Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_APIKEY`	Optional	Provide the administrator account that is defined in Identity Service. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_API_GATEWAY_PORT`	Optional	Define the expose port to scan files via API Gateway Default: 8899
`LOG_LEVEL`	Optional	Define the log level. Default value: info Accepted values: info/debug/error/warning

Start Worker for API Gateway container with Docker run

bash

```
docker run -d --name mddc-worker-api-gateway \
        -e MDDC_WORKER_CONNECTION_KEY=1234abcd \
  -e MDDC_WORKER_HOST=mddc-core-api-gateway \
  -e MDDC_CONTROL_CENTER_HOST=mddc-control-center \
  -e MDDC_USER=admin \
  -e MDDC_PASSWORD=admin \
  -e MDDC_API_GATEWAY_PORT=8899 \
  -p 8893:8893 -p 8899:8899 opswat/metadefender-distributed-cluster:worker-api-
gateway-2.0.0-debian-12
```

5. MDDC Worker for Core

Environment Variable	Necessity	Description
`MDDC_WORKER_CONNECTION_KEY`	Required	Define the connection key in order to register to Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
`MDDC_WORKER_PORT`	Optional	Define the expose worker's port. Default: 8893
`MDDC_WORKER_HOST`	Optional	Define the worker's host address. If it's not specified, it will get the container's internal IP address
`MDDC_CONTROL_CENTER_HOST`	Required	Provide the Control Center's host address
`MDDC_CONTROL_CENTER_PORT`	Optional	Provide the Control Center's port. Default: 8892
`MDDC_USER`	Required	Provide the administrator account that is defined in Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker

Environment Variable	Necessity	Description
`MDDC_PASSWORD`	Required	Provide the administrator account that is defined in Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`MDDC_APIKEY`	Optional	Provide the administrator account that is defined in Identity Service. This account is to automatically do the following tasks: • Add Redis to Control Center if specified • Add RabbitMQ to Control Center if specified • Add Data Lake to Control Center if specified • Add Data Warehouse to Control Center if specified • Add File Storage to Control Center if specified • Add Worker to Control Center, upload API Gateway installer to Control Center, and deploy API Gateway to Worker • Add Worker to Control Center, upload MetaDefender Core installer to Control Center, and deploy MetaDefender Core to Worker
`LOG_LEVEL`	Optional	Define the log level. Default value: info Accepted values: info/debug/error/warning
`MDDC_LICENSE_KEY`	Optional	Provide the license key to activate MetaDefender Core
`MDDC_LICENSE_DESCRIPTION`	Optional	Define description of the license key

Start Worker for Core container with Docker run:

bash

```
docker run -d --name mddc-worker-core \
        -e MDDC_WORKER_CONNECTION_KEY=1234abcd \
  -e MDDC_WORKER_HOST=mddc-core-core \
  -e MDDC_CONTROL_CENTER_HOST=mddc-control-center \
  -e MDDC_USER=admin \
  -e MDDC_PASSWORD=admin \
  -p 8893:8893 opswat/metadefender-distributed-cluster:worker-core-2.0.0-debian-
12
```

Start MetaDefender Distributed Cluster with Docker Compose

1. Create a local file named `docker-compose.yaml` and copy the following content to this file:

yaml yaml

```
services:
  redis:
    image: redis:7.0.5
    container_name: redis
    ports:
      - "6379:6379"
    networks:
      - mddc
  rabbitmq:
    image: rabbitmq:3.13.0
    container_name: rabbitmq
    restart: always
    healthcheck:
      test: ["CMD", "rabbitmq-diagnostics", "-q","ping"]
      interval: 10s
      timeout: 10s
      retries: 30
      start_period: 20s
    environment:
      - RABBITMQ_DEFAULT_USER=admin
      - RABBITMQ_DEFAULT_PASS=admin
    ports:
      - "5672:5672"
      - "15672:15672"
    networks:
      - mddc
  postgres:
    image: postgres:14.17
    container_name: postgres
    ports:
      - 5432:5432
    networks:
      - mddc
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=admin
    healthcheck:
      test: [ "CMD", "pg_isready", "-U", "admin", "-d", "postgres" ]
      interval: 10s
      timeout: 10s
      retries: 30
  identity-service:
    env_file:
    - .env.example
    image: opswat/metadefender-distributed-cluster:identity-service-2.0.0-debian-
12
    container_name: identity-service
    ports:
      - 8891:8891
    networks:
      - mddc
    deploy:
      restart_policy:
```

```
        condition: on-failure
    depends_on:
      postgres:
        condition: service_healthy
        restart: true
  file-storage:
    env_file:
    - .env.example
    image: opswat/metadefender-distributed-cluster:file-storage-2.0.0-debian-12
    container_name: file-storage
    ports:
      - 8890:8890
    networks:
      - mddc
    deploy:
      restart_policy:
        condition: on-failure
    depends_on:
      postgres:
        condition: service_healthy
        restart: true
  control-center:
    env_file:
    - .env.example
    image: opswat/metadefender-distributed-cluster:control-center-2.0.0-debian-12
    container_name: control-center
    ports:
      - 8892:8892
    networks:
      - mddc
    deploy:
      restart_policy:
        condition: on-failure
    healthcheck:
      test: ["CMD", "true"]
      interval: 60s
      start_period: 90s
      start_interval: 60s
    depends_on:
      - identity-service
      - redis
      - rabbitmq
      - file-storage
  worker-api-gateway:
    env_file:
    - .env.example
    image: opswat/metadefender-distributed-cluster:worker-api-gateway-2.0.0-
debian-12
    container_name: worker-api-gateway
    ports:
      - "8893"
      - 7777:7777
    networks:
```

```
      - mddc
    deploy:
      restart_policy:
        condition: on-failure
    healthcheck:
      test: ["CMD", "true"]
      interval: 5s
      timeout: 2s
      start_period: 30s
    depends_on:
      control-center:
        condition: service_healthy
  worker-core:
    env_file:
    - .env.example
    image: opswat/metadefender-distributed-cluster:worker-core-2.0.0-debian-12
    container_name: worker-core
    ports:
      - "8008"
    networks:
      - mddc
    deploy:
      restart_policy:
        condition: on-failure
    depends_on:
      control-center:
        condition: service_healthy
      worker-api-gateway:
        condition: service_healthy

networks:
  mddc:
    driver: bridge
    ipam:
      config:
        - subnet: 10.0.0.0/24
          gateway: 10.0.0.1
```

2. Prepare an environment variable file named `.env.example` and provide with your own values
3. Run the application with the command:

yaml

```
docker compose up -d
```

Known limitation

- When the host experiences resource limitations or degraded performance, some containers may fail to start properly. In such cases, restarting the container is recommended to restore normal operation.

Best practices

Although it is possible to install Redis Caching Server, RabbitMQ Message Broker and Postgres Database Server and MetaDefender Distributed Cluster File Storage on the same machine, they should be installed separately on various machines to optimize their performance.

Redis Caching Server

The caching server consumes a large amount of memory while operating; hence, a machine with ample and high-speed memory is best suited to this component.

RabbitMQ Message Broker

The broker is one of keys that powers MetaDefender Distributed Cluster architecture, it ensures tasks are delivered to MetaDefender Core instances in an equitable manner, delivering a "broken" task to a healthy MetaDefender Core instance and spreading tasks to new instances if more MetaDefender Core instances are added to system. For that reason, the broker should be hosted on a separate machine.

Postgres Database Server

MetaDefender Distributed Cluster database is split into three main clusters.

Data Lake stores scan results and other details related to requests such data_id, hashes, etc.

Since Data Lake is shared among MetaDefender Core and API Gateway instances, it should be hosted on a large-volume and high-speed disk. The network is also essential to Data Lake; a high-speed network is necessary.

Data Warehouse, which prepares materials for building executive reports, uses a single connection to Data Lake and collects data periodically.

Since executive reports may be stored for a long period of time for Control Center to access, Data warehouse should be hosted on a large-volume machine.

Control Center-related database, storing all user details, configurations and other settings. Control Center database can be hosted on the same machine as Control Center.

MetaDefender Distributed Cluster File Storage

File Storage is shared among MetaDefender Core and API Gateway instances. The server consumes a large amount of disk to store the submitted files from instances. Since all file-related traffic goes through File Storage, a high speed network is essential. Rocky 9.0 is recommended to host File Storage.

MetaDefender Distributed Cluster API Gateway

Due to differences in Operating System and Nginx support on Windows and Linux, API Gateway should be hosted on a Linux machine running Rocky 9.0 for high throughput of file scan submissions.

MetaDefender Core

One of strong aspects of MetaDefender Distributed Cluster is that it can support a hybrid architecture in which API Gateway and File Storage instances may be hosted on Linux while MetaDefender Core instances can be on Windows. Therefore, based on customer requirements, MetaDefender Core instances can be hosted on Windows or Linux machines.

Warning

It is recommended to setup all MetaDefender Core instances on Windows or on Linux.

Mixed OS run is unsupported.

Configurations

MetaDefender Distributed Cluster services (Control Center, File Storage, Identity Service, Worker) require an **ignition file** to define essential settings. These files typically include **minimum required key-value pairs** for basic functionality, along with **optional key-value pairs** that users can tailor to their needs.

After installing the service, a **configuration file** is created. The file includes adjustable settings, allowing users to tailor parameters such as host, port, log level, and more to suit their requirements.

Check the details of each service to find the ignition file with required settings, the configuration file created after installation, and the configurable settings.

> **Info**
> Restarting the service is necessary for configuration changes to take effect.

MetaDefender Distributed Cluster File Storage

Ignition file

> **Info**
> The ignition file is required only for a fresh installation.

To install MetaDefender Distributed Cluster File Storage server, ignition file in YML format is required at the following location:

- Windows: `C:\opswat\mddc_file_storage.yml`
- Linux: `/etc/opswat/mddc_file_storage.yml`

The ignition file includes keys:

Key path	Value type	Accepted values	Required	Description
secure.connection_key	string	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	required	An arbitrary string that enables clients to connect to the server. Use this value as input when adding a MDDC File Storage in the UI of the MDDC Control Center.
secure.private_key	string		required	Content of private key in X509 format.
secure.certificate	string		required	Content of certificate in X509 format.
storage.path	string		required	Path to an **existing** directory where the MDDC File Storage server stores its files. The server requires full permissions to access the path in Linux.
rest.host	string		optional	IP address (V4/V6) or host where the server resides on. Default value is `'*'` **Notes:** value `'*'` allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to `0.0.0.0`
rest.port	number		optional	The port where the server resides on. Default value is `8890`
log.streams[@].log_type	string	• file • syslog	optional	Type of log device.

Key path	Value type	Accepted values	Required	Description
log.streams[@].log_level	string	• dump • debug • info • warning • error	optional	Level of log message.
log.streams[@].log_path	string	If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then • `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. • `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).	optional	Location where logs are written.

Configuration file

After successfully installing, MetaDefender Distributed Cluster File Storage generates a configuration file with changeable settings at the following location:

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster File Storage\mddc_file_storage.yml`
- Linux: `/etc/mddc-file-storage/mddc_file_storage.yml`

> **Info**
> The service must be restarted to take the new configurations into effect.

Sample

> **Info**
> OpenSSL or a similar tool (e.g., ssh-keygen) can create a pair of public and private keys in X.509 format.

> **Warning**
> The value of `storage.path` should be updated with the path of an **existing** folder according to your selected operating system.

yaml

```
storage:
  path: 'C:/opswat/mddc/file_storage'   # or '/var/lib/mddc-file-storage/' for
Linux
secure:
  connection_key: "1234abcd" # [0-9a-zA-Z]{4,64}
  private_key: |
        -----BEGIN PRIVATE KEY-----
        MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
        PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
        nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
        toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
        0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
        J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
        Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
        uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
        fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
        zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
        ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
        QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
        qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
        vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
        fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
        Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
        G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
        HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
        YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
        xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
        esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
        zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
        mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
        au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
        40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
        YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
        PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
        W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
        naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
        2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
        39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
        J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
        R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
        Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
        BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
        fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
        x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
        UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
        OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
        65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
        aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
        SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
        me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
        G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
        TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
        56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
```

```
      gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
      Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
      NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
      fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
      -----END PRIVATE KEY-----
certificate: |
      -----BEGIN CERTIFICATE-----
      MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
      VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
      MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
      HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
      WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
      d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
      cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
      aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
      aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
      Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
      LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
      KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
      jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
      Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
      TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
      Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
      nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
      GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
      sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
      9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
      jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
      YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
      xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
      wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
      H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
      xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
      ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
      1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
      AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
      mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
      8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
      e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
      -----END CERTIFICATE-----
```

MetaDefender Distributed Cluster Identity Service

Ignition file

> **Info**
> The ignition file is required only for a fresh installation.

To install MetaDefender Distributed Cluster Identity Service server, ignition file in YML format is required at the following location:

- Windows: `C:\opswat\mddc_identity_service.yml`
- Linux: `/etc/opswat/mddc_identity_service.yml`

The ignition file includes keys

Key path	Value type	Accepted values	Required	Description
secure.connection_key	string	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	required	An arbitrary string that enables clients to connect to the server. Use this value for the key `identity.connection_key` in configuration file of MetaDefender Distributed Cluster Control Center.
secure.private_key	string		required	Content of private key in X509 format.
secure.certificate	string		required	Content of certificate in X509 format.
database.host	string		required	IP address / domain name of the server where PostgreSQL server locates.
database.port	number		required	Port of PostgreSQL server is listening for connections from clients.
database.user	string		required	PostgreSQL server's user. SUPERUSER privilege is required to setup the server's database and extensions for the first time.
database.password	string		required	PostgreSQL server's user credentials.

Key path	Value type	Accepted values	Required	Description
rest.host	string		optional	IP address (V4/V6) or host where the server resides on. Default value is `'*'` **Notes:** value `'*'` allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to `0.0.0.0`
rest.port	number		optional	The port where the server resides on. Default value is `8891`
log.streams[@].log_type	string	• file • syslog	optional	Type of log device.
log.streams[@].log_level	string	• dump • debug • info • warning • error	optional	Level of log message.

Key path	Value type	Accepted values	Required	Description
log.streams[@].log_path	string	If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then • `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. • `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).	optional	Location where logs are written.
user.name	string		optional	User name for the initial administrator user account.
user.password	string		optional	Password for the initial administrator user account.
user.email	string	Basic email format, a string starts with non `whitespace`/non `@` characters, contains one `@` symbol, and ends with non `whitespace`/non `@` characters.	optional	E-mail address for the initial administrator user account.
user.apikey	string	string of exactly 36 characters composed of uppercase and lowercase letters (A-Z, a-z) and digits (0-9)	optional	API key for the initial administrator user account.

Configuration file

After successfully installing, MetaDefender Distributed Cluster Identity Service generates a configuration file with changeable settings at the following location

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Identity Service\mddc_identity_service.yml`
- Linux: `/etc/mddc-identity-service/mddc_identity_service.yml`

> **Info**
> The service must be restarted to take the new configurations into effect.

Sample

> **Warning**
> `database.host`, `database.port`, `database.user`, and `database.password` should be updated with the appropriate values of your Postgres host/IP, port, username, and password.

> **Info**
> OpenSSL or a similar tool (e.g., ssh-keygen) can create a pair of public and private keys in X.509 format.

yaml

```
database:
  host: "your_postgres_host"
  port: 5432
  user: "your_postgres_username"
  password: "your_postgres_admin_password"
secure:
  connection_key: "1234abcd"    # [0-9a-zA-Z]{4,64}
  certificate: |
    -----BEGIN CERTIFICATE-----
    MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
    VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
    MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
    HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
    WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJl
    d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
    cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
    aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
    aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
    Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
    LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
    KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
    jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
    Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
    TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
    Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
    nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
    GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
    sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
    9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
    jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
    YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
    xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
    wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
    H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
    xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
    ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
    1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
    AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
    mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
    8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
    e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
    -----END CERTIFICATE-----

  private_key: |
    -----BEGIN PRIVATE KEY-----
    MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
    PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
    nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
    toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
    0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
    J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
    Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
    uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
```

```
fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
-----END PRIVATE KEY-----
```

MetaDefender Distributed Cluster Worker

Ignition file

> **Info**
> The ignition file is required only for a fresh installation.

To install MetaDefender Distributed Cluster Worker server, ignition file in YML format is required at the following location:

- Windows: `C:\opswat\mddc_worker.yml`
- Linux: `/etc/opswat/mddc_worker.yml`

The ignition file includes keys

Key path	Value type	Accepted values	Required	Description
secure.connection_key	string	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	required	An arbitrary string that enables clients to connect to the server. Use this value as input when adding a MDDC Worker in the UI of the MDDC Control Center.
secure.private_key	string		required	Content of private key in X509 format.
secure.certificate	string		required	Content of certificate in X509 format.
rest.host	string		optional	IP address (V4/V6) or host where the server resides on. Default value is `'*'` **Notes:** value `'*'` allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to `0.0.0.0`
rest.port	number	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	optional	The port where the server resides on. Default value is `8893`
log.streams[@].log_type	string	• file • syslog	optional	Type of log device.

Key path	Value type	Accepted values	Required	Description
log.streams[@].log_level	string	• dump • debug • info • warning • error	optional	Level of log message.
log.streams[@].log_path	string	If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then • `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. • `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).	optional	Location where logs are written.

Configuration file

After successfully installing, MetaDefender Distributed Cluster Worker generates a configuration file with changeable settings at the following location::

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Worker\mddc_worker.yml`
- Linux: `/etc/mddc-worker/mddc_worker.yml`

> **Info**
> The service must be restarted to take the new configurations into effect.

Sample

> **Info**
> OpenSSL or a similar tool (e.g., ssh-keygen) can create a pair of public and private keys in X.509 format.

yaml

```
secure:
  connection_key: 1234abcd
  private_key: |
      -----BEGIN PRIVATE KEY-----
      MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0tJ
      PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAmHK
      nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5ZZ
      toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6DSU
      0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyMBT
      J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8pS
      Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2pHN
      uC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBKIzv9cgi9
      fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0nPN1IMSn
      zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzoGMTvP/Au
      ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTMaB
      QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAoICAFWe8MQZb37k2gdAV3Y6aq8f
      qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIoH+
      vXUpWOjnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0UOiSMMftpk9
      fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A
      Kqx9mld2Ovl7NjgiC1Fx9r+fZw/iOabFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGbMT
      G+ulFrsiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBtODU372a2CQK/u4Q/
      HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8
      YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50SeOcF2QL2xzKmmbZEpXgl
      xBHR0hjgon0IKJDGfor4bHO7Nt+1Ece8u2oTEKvpz5aIn44OeC5mApRGy83/0bvs
      esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRIlG56ErxC3l0s11wrIpTmXXckqw
      zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz
      mgHl24nTg00UH1OjZsABAoIBAQDOxftSDbSqGytcWqPYP3SZHAWDA0O4ACEM+eCw
      au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77
      40QdykllTzTVROqmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfs5
      YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBShnK8xrGaMEmN7W0qTMbFH
      PgQz5FcxRjCCqwHilwNBeLDTp/ZECEB7y34khVh531mBE2mNzSVIQcGZP1I/DvXj
      W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR
      naVJS6WxOKjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6
      2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBLOmbSUhN3PG2m
      39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5jOiPuYkFZ4gH79
      J7gXI0/YKhaJ7yXthYVkdrSF6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+RwDC
      R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZMp
      Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItNl1IZZMW2TPUiY0rl6yaCh
      BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxdJzjaVtxN7ZXwZvOVrY7I7fPWYnCE
      fXCr4+IVpZI/ZHZWpGX6CGSgT6EOjCZ5IUufIvEpqVSmtF8MqfXO9o9uIYLokrWQ
      x1dBl5UnuTLDqw8bChq7O5y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEfHI
      UlGdNVomwXwYEzgE/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3RM
      OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+SOYo26iBu3nw3L
      65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNgOwqu1gRuxcq7TOjt41UdqIKO8vN7/A
      aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jdl/t08EXs79S5IKPcgAx87sTTi7KDN5
      SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiIlNS9foWeLyVkBgCQ6S
      me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRoOJafCaVfGI+175q/eWcqTX4q+I
      G4tKls4sL4mgOJLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVDQy2ttFdsq9iK
      TncCggEBAMmt/8yvPflS+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9nY
      56s9w7OUO8perBXlJYmKZQhO4293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIxa2
      gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjcQoaKvymAoCmAs8V2oMr
      Ziw1YQ9uOUoWwOqm1wZqmVcOXvPIS2gWAs3fQlWjH9hkcQTMsUaXQDOD0aqkSY3E
      NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F5OTSEiGhhHesgRrs
```

```
    fBrpEY1IATtPq1taBZZogRqI3rOkkPk=
    -----END PRIVATE KEY-----
certificate: |
    -----BEGIN CERTIFICATE-----
    MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQYD
    VQQGEwJHQjEQMA4GA1UECAwHRXJ1d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEb
    MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3Qx
    HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNjA3
    WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRXJ1
    d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2V0
    cy10ZXN0MRIwEAYDVQQDDAlsb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbmVA
    aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYtuW
    aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8
    Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTek
    LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnH
    KT/m6DSU0xlZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6
    jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQ
    Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAz
    TK4l2pHNuC53QVc/EF++GBLAxmvCDq9ZpMIYi7OmzkkAKKC9Ue6Ef217LFQCFIBK
    Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqnV0
    nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtgzo
    GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9p
    sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFgQU
    9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAXar
    jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAgEANjIBMrow
    YNCbhAJdP7dhlhT2RUFRdeRUJD0IxrH/hkvb6myHHnK8nOYezFPjUlmRKUgNEDuA
    xbnXZzPdCRNV9V2mShbXvCyiDY7WCQE2Bn44z26O0uWVk+7DNNLH9BnkwUtOnM9P
    wtmD9phWexm4q2GnTsiL6Ul6cy0QlTJWKVLEUQQ6yda582e23J1AXqtqFcpfoE34
    H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVcjv
    xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2cKk
    ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv2g
    1WpTHlumlClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148QYA
    AOlbUOJQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6Gg
    mnIZLRerQCu1Oozr87rOQqQakPkyt8BUSNK3K42j2qcfhAONdRl8Hq8Qs5pupy+s
    8sdCGDlwR3JNCMv6u48OK87F4mcIxhkSefFJUFII25pCGN5WtE4p5l+9cnO1GrIX
    e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
    -----END CERTIFICATE-----
```

MetaDefender Distributed Cluster Control Center

Ignition file

> **Info**
> The ignition file is required only for a fresh installation.

To install MetaDefender Distributed Cluster Control Center server, ignition file in YML format is required at the following location:

- Windows: `C:\opswat\mddc_control_center.yml`
- Linux: `/etc/opswat/mddc_control_center.yml`

The ignition file includes keys

Key path	Value type	Accepted values	Required	Description
identity.host	string		required	IP address of the server where MDDC Identity Service server locates.
identity.port	string		required	Port of MDDC Identity Service server is listening for connections from clients.
identity.connection_key	string	• A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	required	The access key required to connect to the MDDC Identity Service server, ensuring it matches the value used by the server.
database.host	string		required	IP address / domain name of the server where PostgreSQL server locates.
database.port	number		required	Port of PostgreSQL server is listening for connections from clients.
database.user	string		required	PostgreSQL server's user. SUPERUSER privilege is required to setup the server's database and extensions for the first time.
database.password	string		required	PostgreSQL server's user credentials.
secure.encryption_key	string	A 32-character plain text composed of characters 'a'-'z' and digits '0'-'9'.	required	The encryption key is used to encrypt the sensitive data in the database.
rest.port	number		optional	The port where the server resides on. Default value is 8892
rest.log_path	string		optional	Location where logs are written.

Key path	Value type	Accepted values	Required	Description
rest.log_level	string	• dump • debug • info • warning • error	optional	Level of log message.
log.streams[@].log_type	string	• file • syslog	optional	Type of log device.
log.streams[@].log_level	string	• dump • debug • info • warning • error	optional	Level of log message.

Key path	Value type	Accepted values	Required	Description
log.streams[@].log_path	string	If `log.streams[@].log_type` is `"file"` then `log.streams[@].log_path` is the path to a file on file system where logs are written. If `log.streams[@].log_type` is `"syslog"` then • `log.streams[@].log_path` can be `[tcp/udp]://host:port` where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. • `log.streams[@].log_path` can be `"local"` to write log to local syslog server (Linux only).	optional	Location where logs are written.

Warning
Avoid using the loopback IP address (such as `localhost` or `127.0.0.1`) for key `identity.host` .

It may prevent the MetaDefender Distributed Cluster API Gateway from successfully establishing a connection to MetaDefender Distributed Cluster Identity Service.

Configuration file

After successfully installing, MetaDefender Distributed Cluster Control Center generates a configuration file with changeable settings at the following location:

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Control Center\mddc_control_center.yml`
- Linux: `/etc/mddc-control-center/mddc_control_center.yml`

> **Info**
> The service must be restarted to take the new configurations into effect.

Sample

> **Warning**
> `database.host`, `database.port`, `database.user`, and `database.password` should be updated with the appropriate values of your Postgres host/IP, port, username, and password.
>
> `identity.host` should be updated with the appropriate host or IP of your MetaDefender Distributed Cluster Identity Service.

yaml

```
database:
  host: "your_postgres_host_ip"
  port: 5432
  user: "your_postgres_username"
  password: "your_postgres_admin_password"
identity:
  host: "your_mddc_identity_service_host_ip"
  port: 8891
  connection_key: "1234abcd"
secure:
  encryption_key: "12345678123456781234567812345678" # [a-z0-9]{32}
```

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.

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)
- MetaDefender Distributed Cluster API Gateway version
- 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

Performance metrics

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

Performance metrics	Description
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 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.

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 object, and it also takes 10 seconds to process 10 objects, then performance is quantified as 1 second per object, rather than 10 seconds.

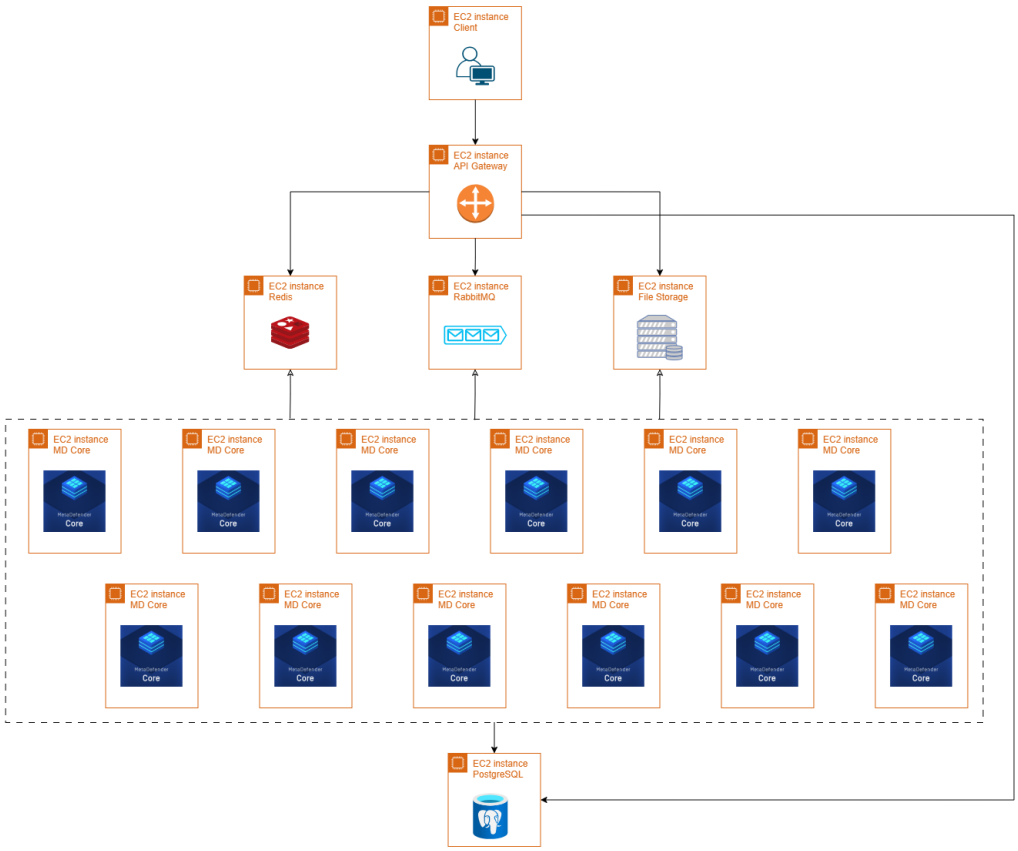
- total time / total number of objects processed: 10 seconds / 10 objects = 1 second / object.

Dataset

File category	File type	Number of files	Total size	Average file size
Document	DOC	3,820	534 MB	0.14 MB
Medium archive files	RPM CAB EXE	50	Compressed size: 2.8 GB Extracted size: 12.09 GB	Compressed size: 56.02 MB Extracted size: 0.036 MB
Big archive files	CAB	4	Compressed size: 2.9 GB Extracted size: 124 GB	Compressed size: 715 MB

Environment

Topology



Using AWS environment with the specification below:

MDDC system

	MD Core	File Storage	API Gateway	PostgreSQL	RabbitMQ	Redis
OS	Windows Server 2022	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9
AWS instance type	c5.2xlarge	c5n.4xlarge	c5n.2xlarge	c5.xlarge	c5.xlarge	c5.xlarge
vCPU	8	16	4	4	4	4
Memory	16GB	32GB	8GB	8GB	8GB	32GB
Disk Type	gp3	gp3	gp3	gp3	gp3	gp3
IOPS	3000	12000	3000	10000	3000	3000
Throughput	125MB/s	1000MB/s	256MB/s	550MB/s	125MB/s	125MB/s
Size	100GB	150GB	100GB	100GB	80GB	80GB
Network bandwidth (baseline & burst)	2.5 Gbps 10 Gbps	15 Gbps 25 Gbps	5 Gbps 25 Gbps	1.25 Gbps 10 Gbps	1.25 Gbps 10 Gbps	1.25 Gbps 10 Gbps
Benchmark (Geekbench)	EC2 c5.2xlarge	EC2 c5n.4xlarge	EC2 c5n.2xlarge	EC2 c5.xlarge	EC2 c5.xlarge	EC2 c5.xlarge

Client tool

	Detail
OS	Rocky Linux 9
AWS instance type	c5n.xlarge
vCPU	4
Memory	10GB
Disk	Type: gp3 IOPS: 3000 Throughput: 125MB/s Size: 80GB
Network bandwidth	Baseline: 5 Gbps Burst: 10 Gbps

Product information

- MetaDefender Core v5.14.2
- Engines:
  - Metascan 8: Ahnlab, Avira, ClamAV, ESET, Bitdefender, K7, Quick Heal, VirIT Explorer
  - Archive v7.4.0
  - File type analysis v7.4.0
- MDDC Control Center v2.0.0
- MDDC API Gateway v2.0.0
- MDDC File Storage v2.0.0
- PostgreSQL v14.17
- RabbitMQ v3.12.6
- Redis v7.2.1

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update
- Scan queue: 1000 (for Load Balancer deployment)

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 settings

- Max file size: 99999999
- Scan timeout: 10 minutes
- Per engine scan timeout: 1 minutes

Advanced settings

RabbitMQ

- RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS=-rabbit consumer_timeout unlimited default_consumer_prefetch {false,525}

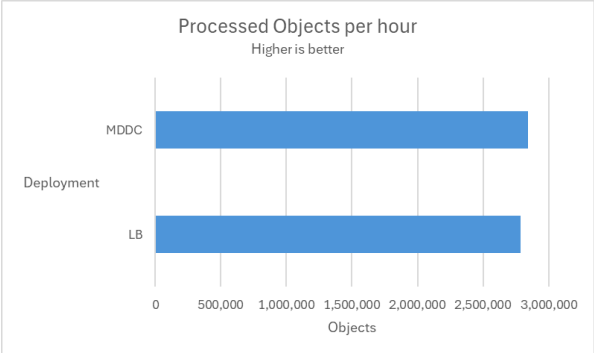
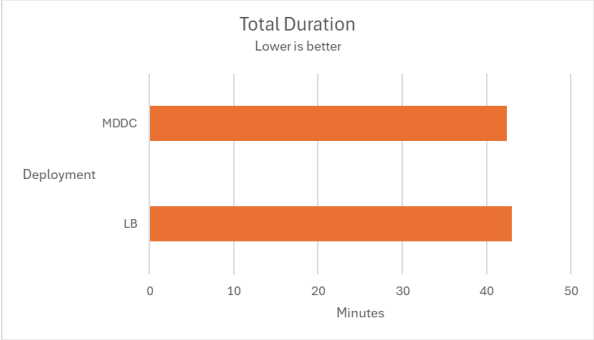
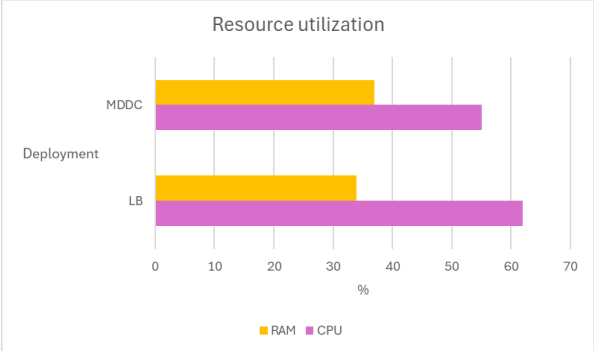
Redis

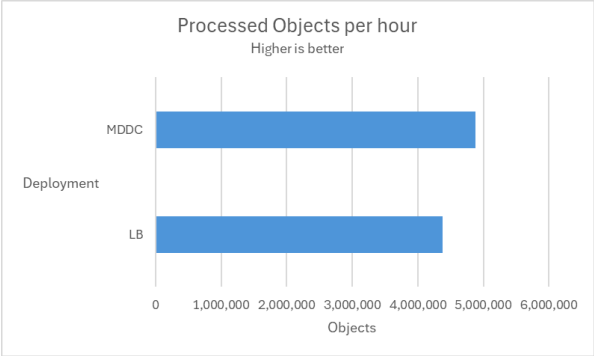
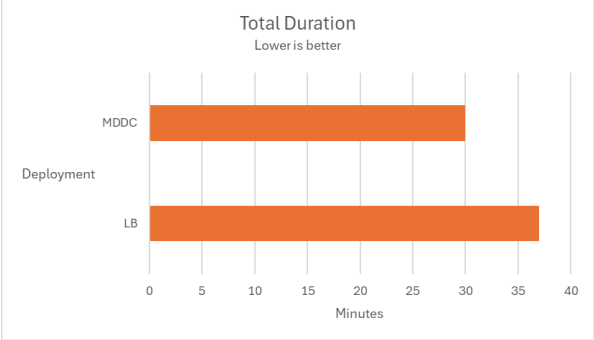
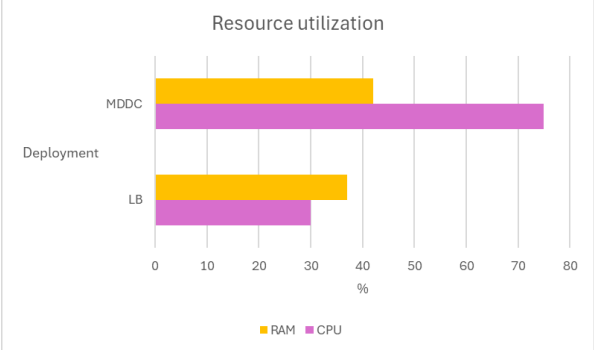
- redis-cli flushall
- redis-cli config set save ''
- redis-cli config set maxmemory 25gb
- redis-cli config set maxmemory-policy volatile-ttl

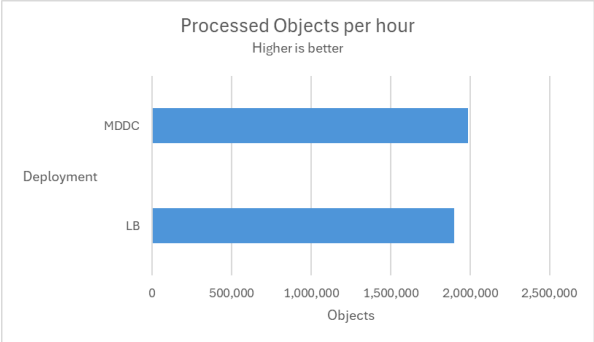
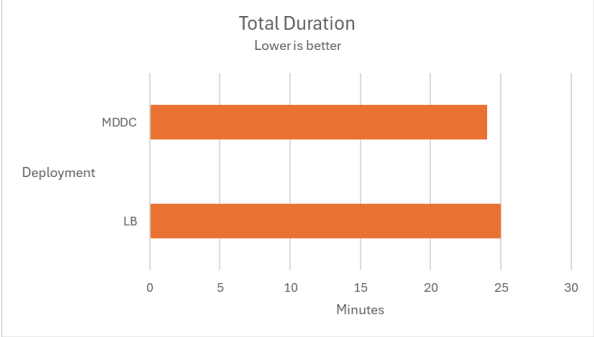
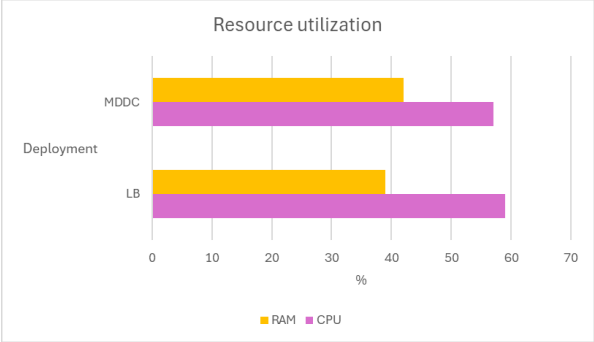
Performance results

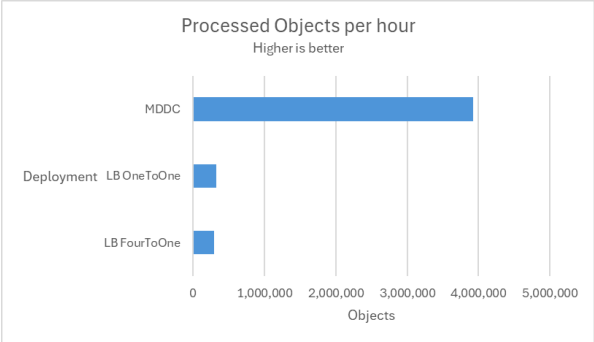
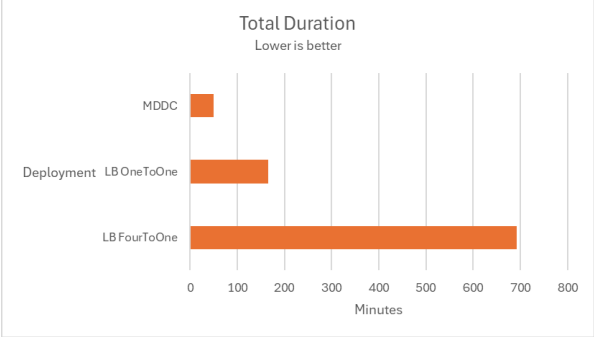
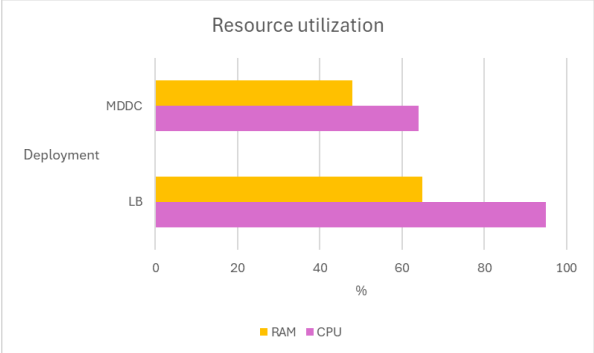
Load-balance deployment vs MDDC deployment

Multiple tests are conducted using 12 MetaDefender Core instances across two deployment types, MetaDefender Distributed Cluster (MDDC) and Load Balancer, to determine the superiority of the MDDC in 4 different datasets.

Scenario	Result
Aggressively submitted 2M non-archive files at a rate of 800 files per second.	  

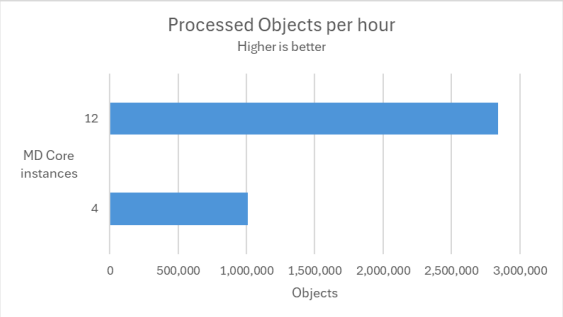
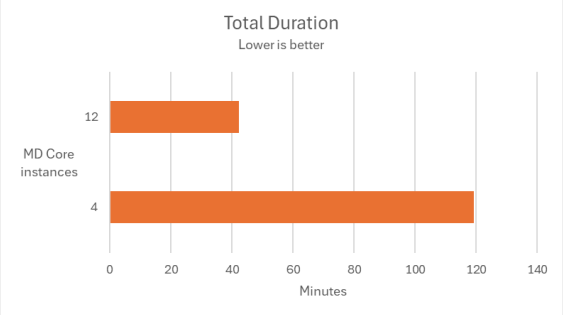
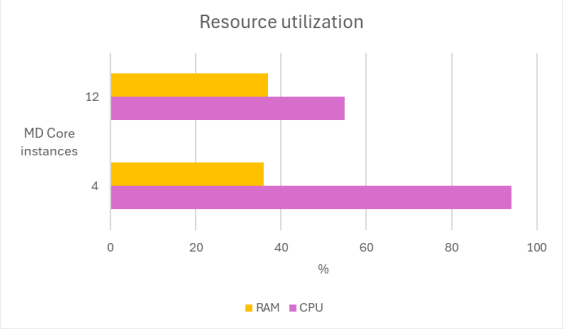
Scenario	Result
Submitted 400 medium archive files at a rate of 1 files per second.	  

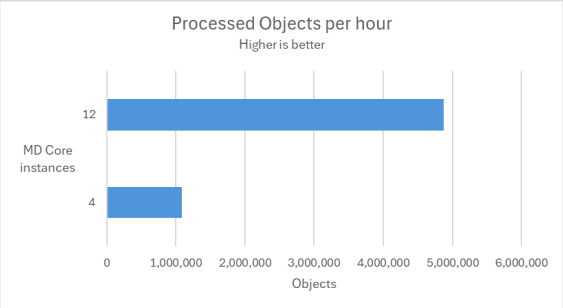
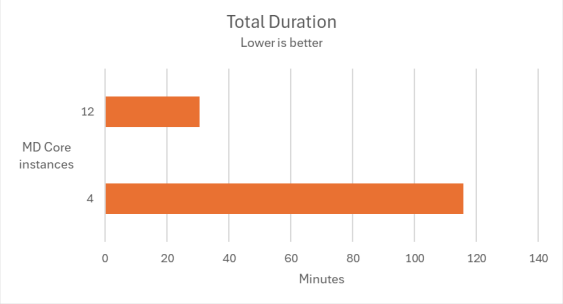
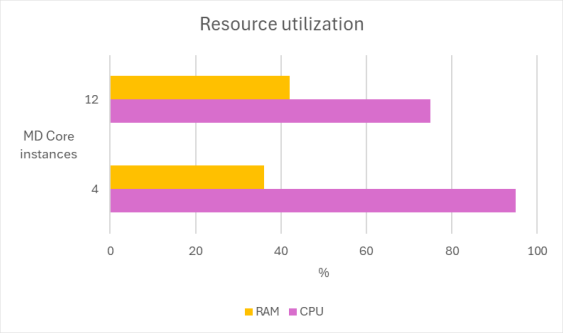
Scenario	Result
Submitted a mix of 189K non-archive and medium archive files at a rate of 180 files per second.	  

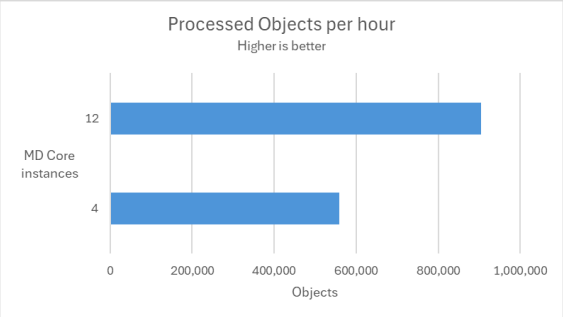
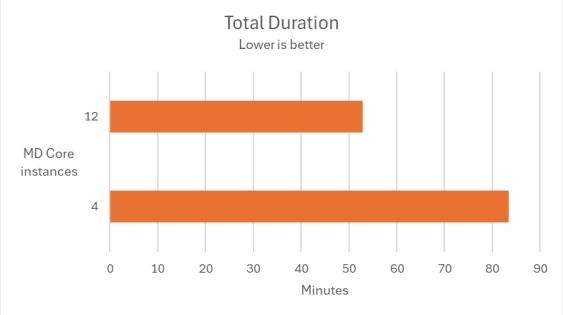
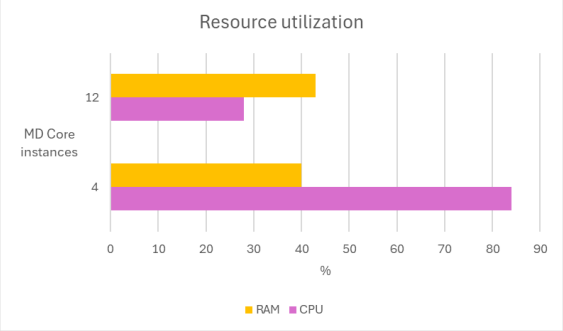
Scenario	Result
Submitted 4 large CAB files. The scenarios replicate 2 different routing cases of a common Load Balancer. ***LB OneToOne***: An ideal routing ensures that one CAB file is routed to a single MD Core. ***LB FourToOne***: The worst routing that delivered four CAB files to a single MD Core. # **Archive distribution** In workflow, setting "Load shared among MetaDefender Core instances for archive processing" is enabled. ☑ Load shared among MetaDefender Core instances for archive processing Applicable to Distributed Cluster deployment, nested files in archive could be processed in multiple MetaDefender Core instances, recommended when processing mostly big archive files.	  

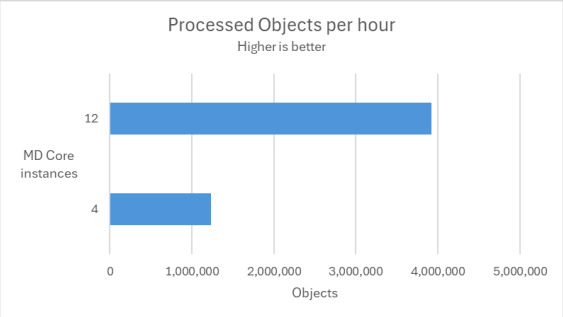
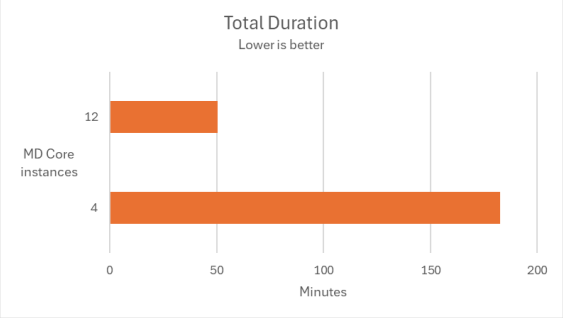
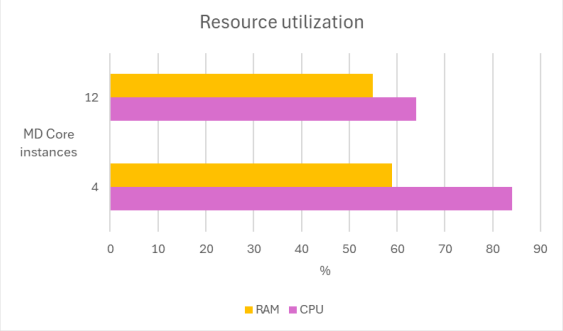
Scaling out

In the following test scenarios, we conducted experiments on four datasets using 4 and 12 of MD Core instances in MetaDefender Distributed Cluster (MDDC), demonstrating the benefits of increased instance counts.

Scenario	Result
Aggressively submitted 2M non-archive files at a rate of 800 files per second.	  

Scenario	Result
Submitted 400 medium archive files at a rate of 1 files per second.	  

Scenario	Result
Submitted a mix of 189K non-archive and medium archive files at a rate of 60 files per second.	  

Scenario	Result
Submitted 4 large CAB files. **Archive distribution** In workflow, setting "Load shared among MetaDefender Core instances for archive processing" is enabled. ☑ Load shared among MetaDefender Core instances for archive processing Applicable to Distributed Cluster deployment, nested files in archive could be processed in multiple MetaDefender Core instances, recommended when processing mostly big archive files.	  

Troubleshooting

How to investigate product alerts, dealing with installation, logging and maintenance related matters.

How To Collect Logs In MetaDefender Distributed Cluster Deployments

1. MetaDefender Core

Windows

- Please refer to How to Create Support Package With Bundle Tools?

> **Info**
> The installation directory of MetaDefender Core when deployed by MetaDefender Distributed Cluster, located in the installation directory of MDDC Worker.
>
> `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Worker\data\services\MetaDefender Core`

Linux

- The support package is created within the output file after running the command to collect the MDDC Worker support package.

2. MetaDefender Distributed Cluster API Gateway

Windows

- Run command prompt as administrator privilege.
- Change directory (`cd`) to MDDC API Gateway installation directory.
- Run following file script to generate MDDC API Gateway support package:

none

```
mddc-api-gateway-collect-support-data.bat
```

- Find and collect MDDC API Gateway generated support package (`mddc-api-gateway-<api-gateway_version>-support-<timestamp>.zip`) in the MDDC API Gateway installation folder

Info
The installation directory of MDDC API Gateway if deployed by MetaDefender Distributed Cluster, located in the installation directory of MDDC Worker:

`C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Worker\data\services\MetaDefender Distributed Cluster API Gateway`

Linux

- The support package is created within the output file after running the command to collect the MDDC Worker support package.

3. MetaDefender Distributed Cluster File Storage

Windows

- Run command prompt as administrator privilege.
- Change directory (`cd`) to MDDC File Storage installation folder (e.g. `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster File Storage`)
- Run following file script to generate MDDC File Storage support package:

none

```
mddc-file-storage-collect-support-data.bat
```

- Find and collect MDDC File Storage generated support package (`mddc-file-storage-<file-storage_version>-support-<timestamp>.zip`) in the Hub installation folder.

Linux

- Run terminal as root privilege (`sudo`).
- Run following file script to generate MDDC File Storage support package:

none

```
/usr/bin/mddc-file-storage-collect-support-data.sh
```

- Find and collect MDDC File Storage generated support package (`mddc-file-storage-<file-storage_version>-support-<timestamp>.zip`) in the current folder.

4. MetaDefender Distributed Cluster Identity Service

Windows

- Run command prompt as administrator privilege.

- Change directory (`cd`) to MDDC Identity Service installation folder (e.g. `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Identity Service`)
- Run following file script to generate MDDC Identity Service support package:

none

```
mddc-identity-service-collect-support-data.bat
```

- Find and collect MDDC Identity Service generated support package (`mddc-identity-service-<identity-service_version>-support-<timestamp>.zip`) in the Hub installation folder.

Linux

- Run terminal as root privilege (`sudo`).
- Run following file script to generate MDDC Identity Service support package:

none

```
/usr/bin/mddc-identity-service-collect-support-data.sh
```

- Find and collect MDDC Identity Service generated support package (`mddc-identity-service-<identity-service_version>-support-<timestamp>.zip`) in the current folder.

5. MetaDefender Distributed Cluster Control Center

Windows

- Run command prompt as administrator privilege.
- Change directory (`cd`) to MDDC Control Center installation folder (e.g. `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Control Center`)
- Run following file script to generate MDDC Control Center support package:

none

```
mddc-control-center-collect-support-data.bat
```

- Find and collect MDDC Control Center generated support package (`mddc-control-center-<control-center_version>-support-<timestamp>.zip`) in the MDDC Control Center installation folder.

Linux

- Run terminal as root privilege (`sudo`).
- Run following file script to generate MDDC Control Center support package:

none

```
/usr/bin/mddc-control-center-collect-support-data.sh
```

- Find and collect MDDC Control Center generated support package (`mddc-control-center-<control-center_version>-support-<timestamp>.zip`) in the current folder.

6. MetaDefender Distributed Cluster Worker

Windows

- Run command prompt as administrator privilege.
- Change directory (`cd`) to MDDC Worker installation folder (e.g. `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Worker`)
- Run following file script to generate MDDC Worker support package:

none

```
mddc-worker-collect-support-data.bat
```

- Find and collect MDDC Worker generated support package (`mddc-worker-<worker_version>-support-<timestamp>.zip`) in the MDDC Worker installation folder.

Linux

- Run terminal as root privilege (`sudo`).
- Run following file script to generate MDDC Worker support package:

none

```
/usr/bin/mddc-worker-collect-support-data.sh
```

- Find and collect MDDC Worker generated support package (`mddc-worker-<worker_version>-support-<timestamp>.zip`) in the current folder.

7. Redis (Caching Server)

Windows

- Find and collect Redis own logs in the Redis installation folder (e.g. `C:\Program Files\Redis\server_log.txt`)

Linux

- Run terminal as root privilege (`sudo`).

- Open Redis config file (`/etc/redis/redis.conf`) in edit mode e.g.:

none

```
vi /etc/redis/redis.conf
```

- Find and change `logfile` directive to your desired location

none

```
logfile "<path to redis log>.log"
```

- Save the file, and restart Redis daemon e.g.

none

```
sudo systemctl restart redis
```

- Find and collect Redis log at `<path to redis log>.log`

8. RabbitMQ (Messaging Broker Server)

Windows

- Find and collect RabbitMQ log at `%APPDATA%\RabbitMQ\log\rabbit@<computer name>.log`
- Find and collect RabbitMQ upgrade log at `%APPDATA%\RabbitMQ\log\rabbit@<computer name>_upgrade.log`

Linux

- Run terminal as root privilege (`sudo`).
- Run following command to retrieve RabbitMQ log location:

none

```
rabbitmq-diagnostics -q log_location
```

- Access RabbitMQ log folder and find log files:
  - `rabbit@<computer name>.log`
  - `rabbit@<computer name>_upgrade.log`

9. PostgreSQL (Database Server)

Windows

- Find and collect PostgreSQL log files in `C:\Program Files\PostgreSQL\12\data\log` with names `postgresql-<yyyy-mm-dd>_<HHMMSS>.log`

Linux

- Run terminal as root privilege (`sudo`).
- Open PostgreSQL config file (`/etc/postgresql/12/main/postgresql.conf`) in edit mode e.g.:

none

```
vi /etc/postgresql/12/main/postgresql.conf
```

- Find and change `logging_collector` directive to `on` :

none

```
logging_collector = on
```

- Save the file and restart PostgreSQL daemon, e.g.:

none

```
sudo systemctl restart postgresql
```

- Find and collect log files in `/var/lib/postgresql/12/main/log` with names `postgresql-<yyyy-mm-dd>_<HHMMSS>.log`

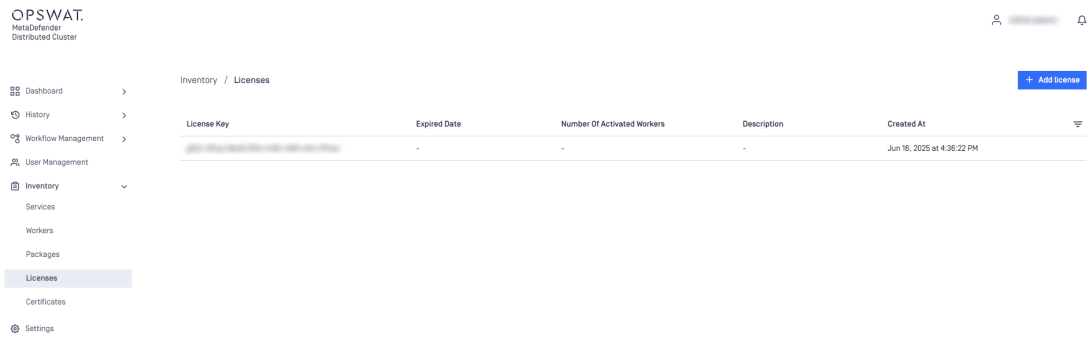
License Activation

MetaDefender Distributed Cluster supports seamless license activation for every deployed MetaDefender Core instance. The license key must be provided to the MetaDefender Distributed Cluster Control Center and will be activated on each individual MetaDefender Core instance either automatically during deployment or manually at a later time. If necessary, multiple license keys may also be supplied.

> **Info**
> In the present release, only **online activation** is supported. For additional information, please refer Activation Modes section.

Adding License to MetaDefender Distributed Cluster Control Center

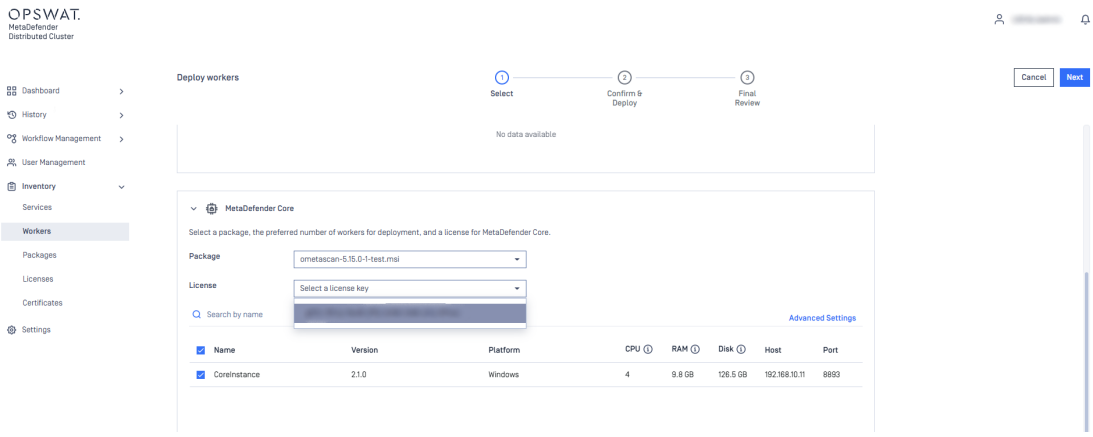
1. Sign in to MetaDefender Distributed Cluster Control Center Console.
2. Go to `Inventory` > `Licenses` and select `Activate License`.
3. Input your license key and click `Add`.



License Activation

Automatic activation during deployment

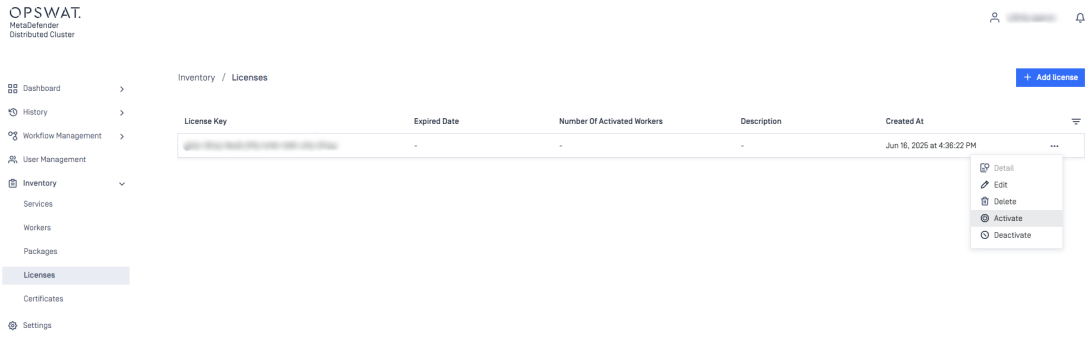
During the deployment process of MetaDefender Core instances, you can select a license key to automatically activate on the instances being deployed.



Activation after deployment

After deploying MetaDefender Core instances, follow these steps to activate your license:

1. Go to the main menu and select `Inventory` > `Licenses`.
2. From the list of available licenses, choose the key you wish to use for activation.
3. Click `Activate` to apply the license key to the appropriate instance(s).



> **Info**
> Once activation is selected, the license key will be applied to all unlicensed MD Core instances. Ensure that your license quota is sufficient to cover all unlicensed instances.
>
> You can view the number of activated instances and available slots by selecting **Details** on the license key. At the moment, **Details** can only be viewed after activation is successful.

Deactivating a License

Follow these steps to deactivate your license:

1. Go to the main menu and select `Inventory` > `Licenses`.
2. From the list of available licenses, choose the key you wish to use for deactivation.
3. Click `Deactivate` to remove the license from all MetaDefender Core instances currently activated with the license key.



Info

Once deactivated, the license slots will become available and can be reassigned when necessary.

When MetaDefender Core instances are undeployed through MetaDefender Distributed Cluster Control Center, their associated licenses are automatically deactivated.

System Upgrade

Overview

A primary focus of MetaDefender Distributed Cluster is to reduce the disruption of file processing during system upgrades. The levels of impact on file processing during component updates are outlined as follows:

Component	Impact level	Upgrade method
MetaDefender Core	0	Deferral by MDDC Control Center
MeteDefender Distributed Cluster Worker	1	Manual by installer file
MeteDefender Distributed Cluster API Gateway	1	Deferral by MDDC Control Center
MeteDefender Distributed Cluster Control Center	1	Manual by installer file
MeteDefender Distributed Cluster Identity Service	1	Manual by installer file
MeteDefender Distributed Cluster File Storage	2	Manual by installer file

The potential impacts of each level are detailed below.

Level	Impact
0	The upgrade does not impact file processing.
1	The upgrade does not impact the processing of existing files but may affect the submission of new files, fetching scan result, downloading processed files, monitoring or management.
2	The upgrade needs the entire system to go down.

MetaDefender Distributed Cluster hosts a vast majority of MetaDefender **Core** instances. During the MetaDefender **Core** upgrade, each instance í upgraded sequentially to prevent interference with the system's file processing. Intrinsically, the MetaDefender **Core** upgrade is controlled and managed strictly by MetaDefender Distributed Cluster Worker and MetaDefender Distributed Cluster Control Center. These services guarantee that the MetaDefender **Core** instance is safely isolated from new file submissions and continues processing until all files are finished on its end before the upgrade procedure takes place.

While not impacting file processing, the upgrade of the MetaDefender Distributed Cluster **Worker** may cause a hiccup in reporting the status and resources consumed by the MetaDefender Core or MetaDefender Distributed Cluster API Gateway instance to MetaDefender Distributed Cluster Control Center. Consequently, users may

slightly notice that one of their workers is occasionally missing from their dashboards.

During the MetaDefender Distributed Cluster **API Gateway** upgrade, clients may not be able to submit files, fetch scan statuses, or download processed files from the service. The deployment of multiple instances of the MetaDefender Distributed Cluster **API Gateway** should be considered to reduce service interruptions.

MeteDefender Distributed Cluster **Control Center** is designed for system administrators to manage and monitor operational services (MetaDefender Core, MetaDefender Distributed Cluster API Gateway, etc.), thus its upgrade solely affects the administrators and does not file processing.

Although file processing remains uninterrupted, the upgrade of MetaDefender Distributed Cluster **Identity Service** may affect the authentication of users accessing MetaDefender Distributed Cluster **Control Center**. It may also cause temporary failures in validating requests that contain API key header in MetaDefender Distributed Cluster **API Gateway**.

Most of services within the system establish connections to MetaDefender Distributed Cluster **File Storage**, thus its upgrade results in system downtime. Consequently, the upgrade of MetaDefender Distributed Cluster **File Storage** requires the system administrator to place the entire system in scheduled maintenance mode and this should be executed during a period when no files are sent for scanning.

Upgrade method

Manual by installer file

> **Info**
> Appliable for MetaDefender Distributed Cluster **Control Center**, **Worker**, **Identity Service** and **File Storage**.

1. Download installer package from My Opswat.
2. Access the machine that host component service pending for upgrade.
3. Start Command Prompt as Administrator on Windows or Terminal on Linux and run one of the following commands:

bash

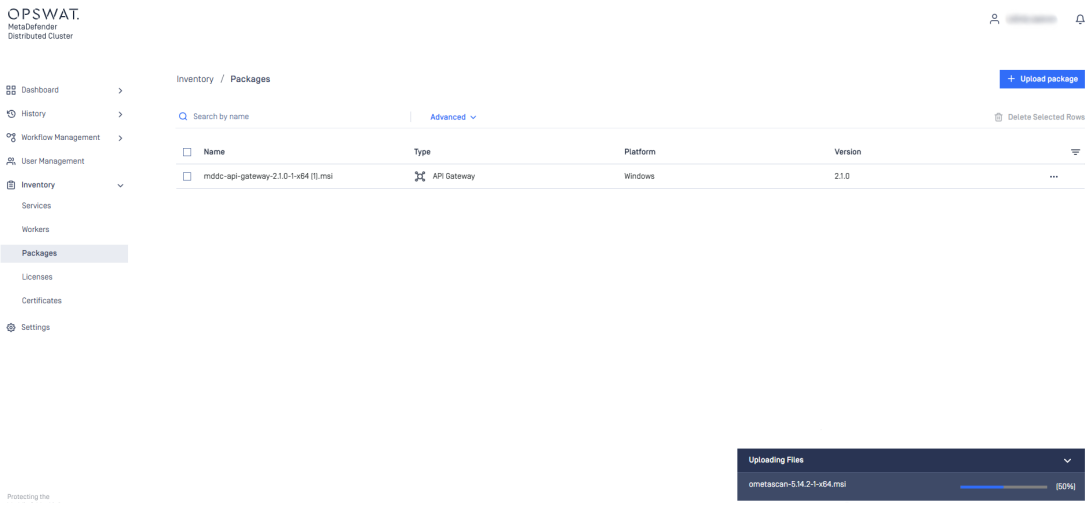
```
# Windows
> msiexec.exe /i <new_installer> /qn

# Debian or Ubuntu
$ sudo dpkg -i <new_installer> || sudo apt install -f

# Red Hat or Rocky
$ sudo yum install <new_installer> -y
```

4. Confirm the service starts successfully.

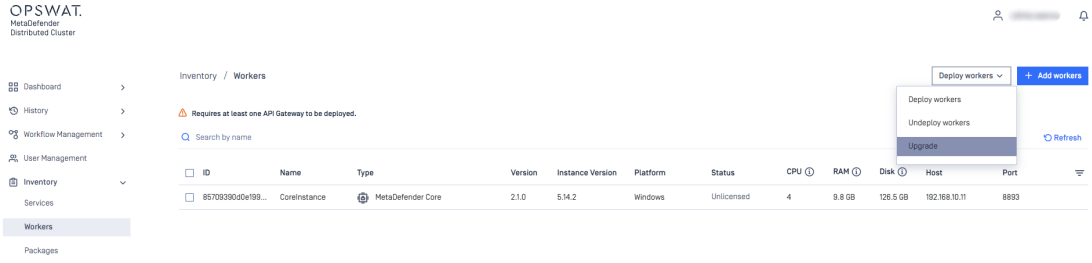
Deferral by MetaDefender Distributed Cluster Control Center

> **Info**
> Appliable for MetaDefender Distributed Cluster **API Gateway** and MetaDefender **Core**.

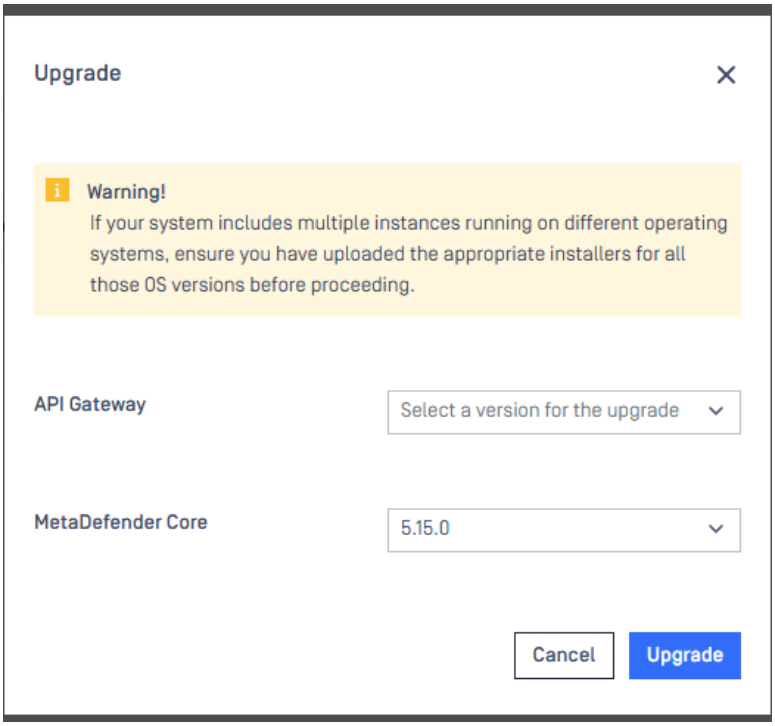
1. Download installer package from My Opswat.
2. Sign in to MetaDefender Distributed Cluster Control Center console with your administrator account.
3. Navigate to `Inventory` > `Packages`, click `Upload packages`.



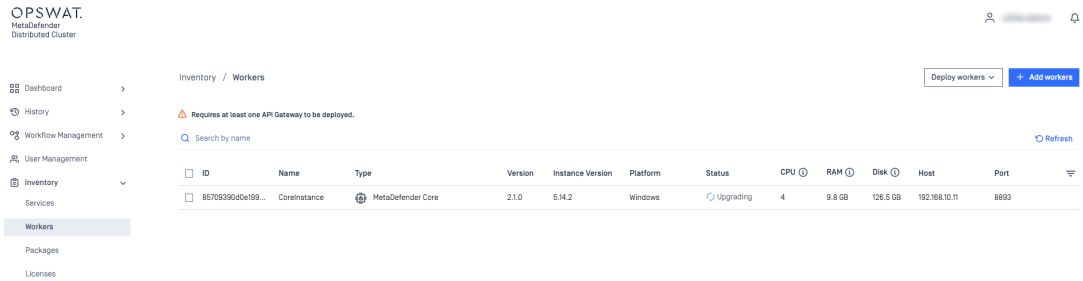
3. Go to `Inventory` > `Worker`, click `Deploy workers` and select `Upgrade`.



4. Select the correct installer version and click `Upgrade`.



5. The progress is displayed during upgrade.



6. Confirm components upgraded to correct versions.
7. Monitor system health and status.

Info
During the upgrade process, modifications to the settings outlined below are prohibited:

- Workflow
- Module Update
- Health check

Upgrade procedures

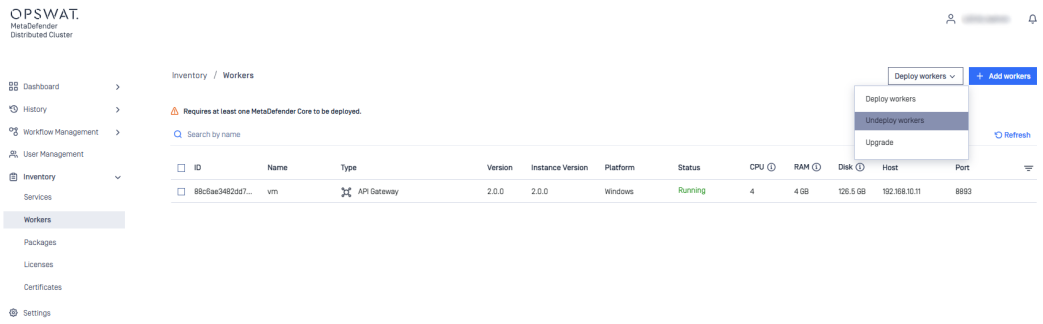
- Upgrade to MetaDefender Distributed Cluster v2.1.0

Upgrade to MetaDefender Distributed Cluster v2.1.0

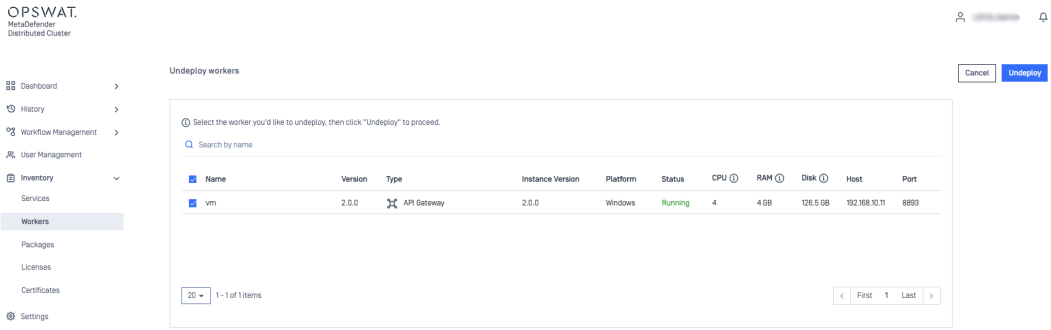
> **Warning**
> MetaDefender Distributed Cluster v2.1.0 includes a major change in MetaDefender Distributed Cluster **File Storage**, thus the system need to go down during upgrade.

Upgrade procedure

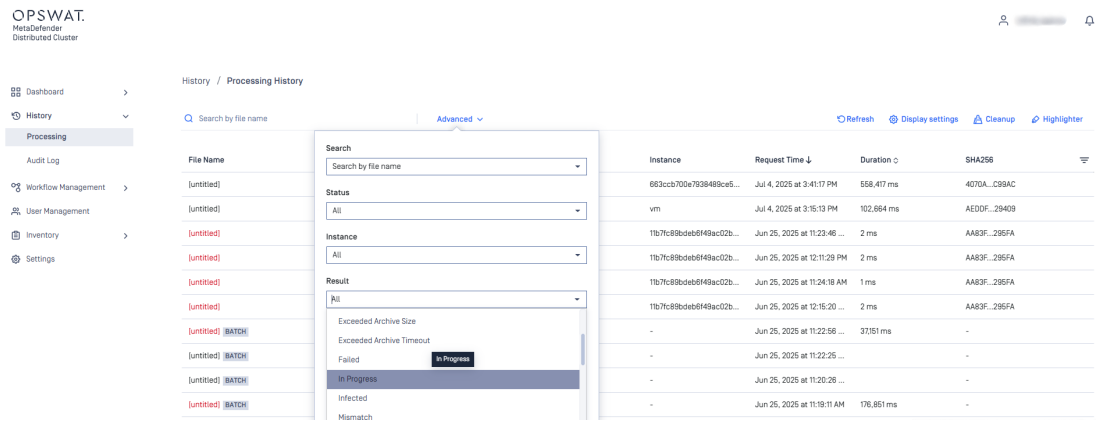
1. Sign to MetaDefender Distributed Cluster **Control Center** console with your administrator account, navigate to `Inventory` > `Worker` and select `Undeploy workers` .



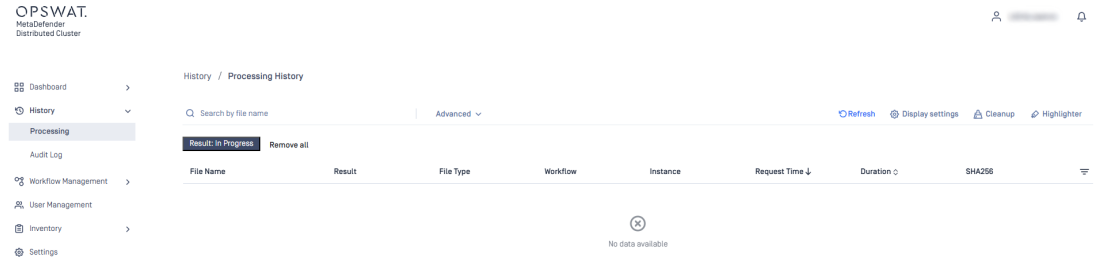
2. Select workers of MetaDefender Distributed Cluster **API Gateway**, then select `Undeploy` .



3. Navigate to `Dashboard` > `Processing History` and filter **In-Progress** requests.



4. Wait until all in-progress requests are complete.



5. Upgrade MetaDefender Distributed Cluster **File Storage** by installer.
6. Upgrade MetaDefender Distributed Cluster **Worker** by installer.
7. Upgrade MetaDefender Distributed Cluster **Identity Service** by installer.
8. Upgrade MetaDefender Distributed Cluster **Control Center** by installer.
9. Upgrade MetaDefender **Core** by MetaDefender Distributed Cluster Control Center.
10. Deploy the new version of MetaDefender Distributed Cluster API Gateway on MetaDefender Distributed Cluster Control Center.
11. In MetaDefender Distributed Cluster Control Center Dashboard, verify that all services are operational.

Info

The ignition file is **not** required during the service upgrade. Upon successful upgrades, connections between component services are automatically established.

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:

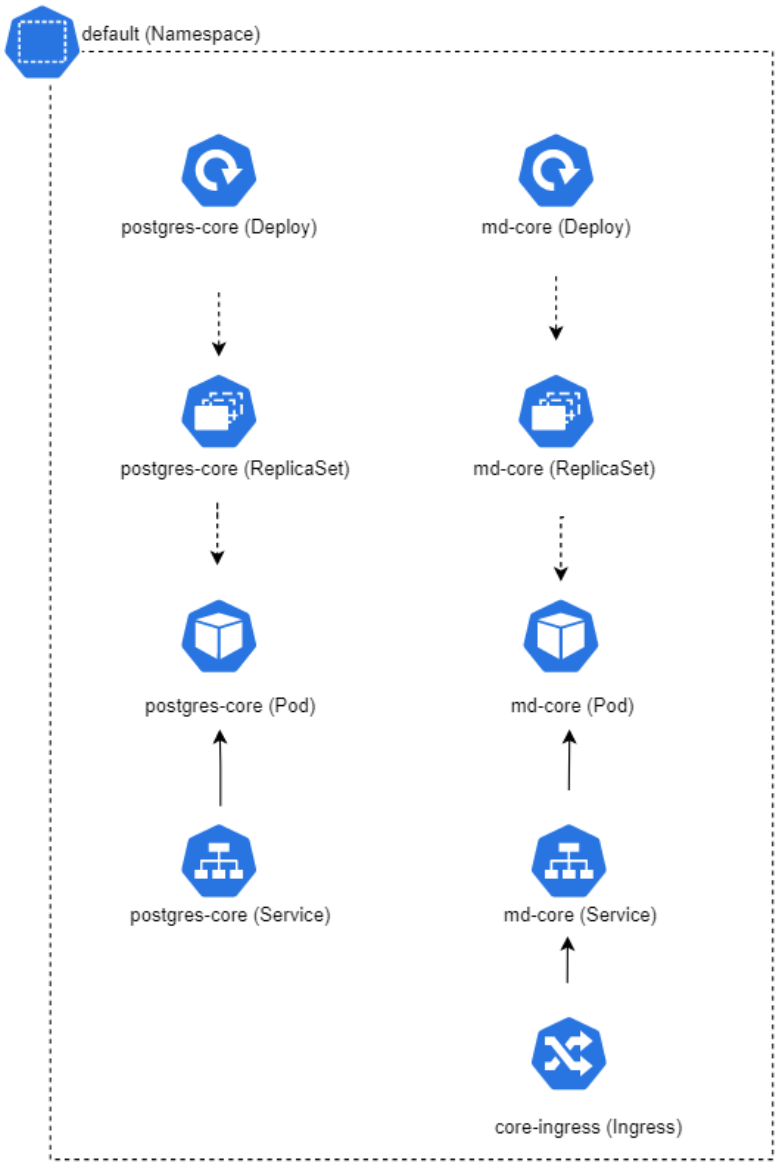
- MetaDefender Core Service + Ingress + PostgreSQL service
  - md-core service
  - postgres-core service
  - core-ingress component (Optional)
- MetaDefender Core Service + Ingress + External Database
  - md-core service
  - core-ingress service (Optional)
  - PostgreSQL external database

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 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

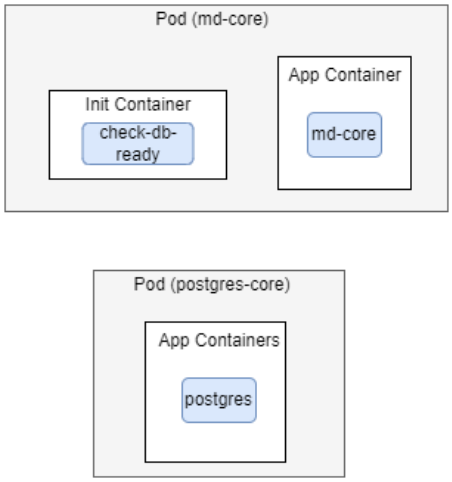


Info
postgres-core components will be needed to be generated just in case an external database is not connect to the md-core service

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 K8S

Using Kubernetes technology provides us with some features that distinct it from other type of deployments. These are the main features:

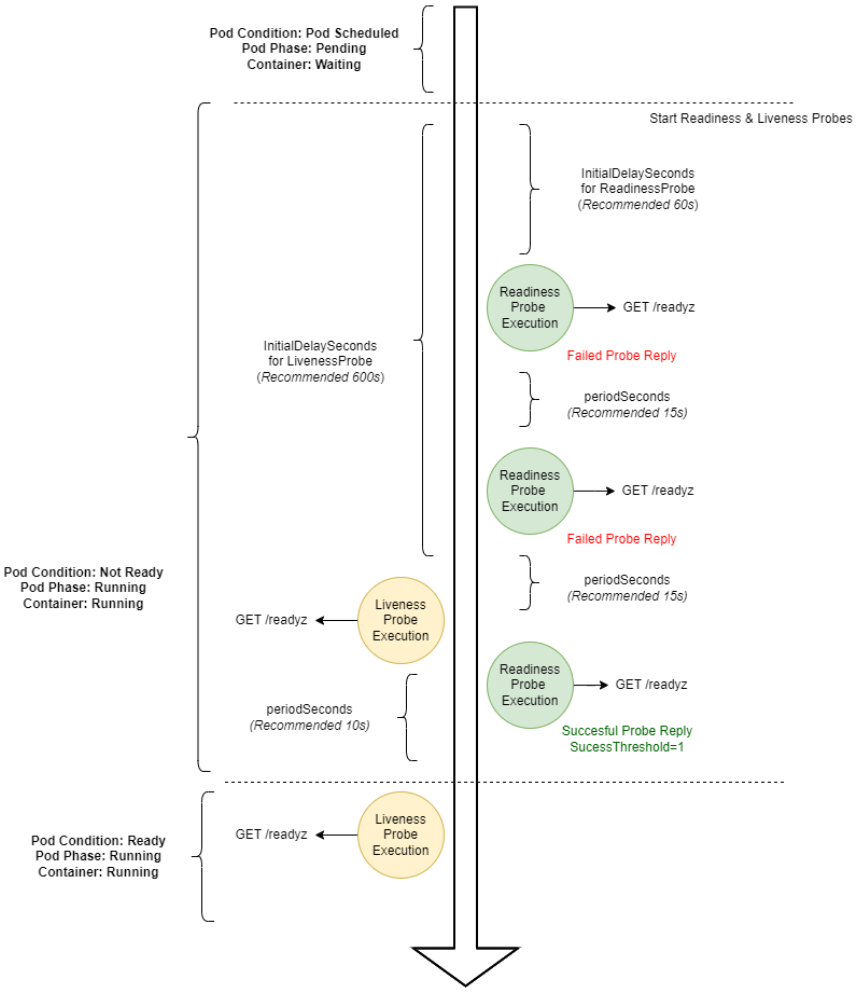
- **Service discovery and load balancing.** See Installing MetaDefender Core in K8S
- **Storage orchestration.** Allows you to automatically mount a storage system. See Storage Configuration on K8S
- **Automated rollouts and rollbacks.** See Upgrade/Deployment Strategy on K8S
- **Automatic bin packing.** 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 API (/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


Pod Condition: Pod Scheduled
Pod Phase: Pending
Container: Waiting
Start Readiness & Liveness Probes
InitialDelaySeconds
for ReadinessProbe
(Recommended 60s)
Readiness
Probe
Execution
GET /readyz
Failed Probe Reply
InitialDelaySeconds
for LivenessProbe
(Recommended 600s)
periodSeconds
(Recommended 15s)
Readiness
Probe
Execution
GET /readyz
Failed Probe Reply
Pod Condition: Not Ready
Pod Phase: Running
Container: Running
periodSeconds
(Recommended 15s)
GET /readyz
Liveness
Probe
Execution
Readiness
Probe
Execution
GET /readyz
periodSeconds
(Recommended 10s)
Succesful Probe Reply
SucessThreshold=1
Pod Condition: Ready
Pod Phase: Running
Container: Running
GET /readyz
Liveness
Probe
Execution

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:

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:

bash

```
git clone https://github.com/OPSWAT/metadefender-k8s.git metadefender
cd metadefender/helm_carts

#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:

yaml

```
core_components:
  md-core:
    service_type: LoadBalancer
```

MD Core can also be exposed using an ingress:

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:

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

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

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

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

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
    a. For Azure see example below
    b. For AWS you need to install AWS LB Controller and set up the ingress with the annotations indicated
    c. GCP install it automatically, see below configuration needed

Azure AWS GCP

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
```

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

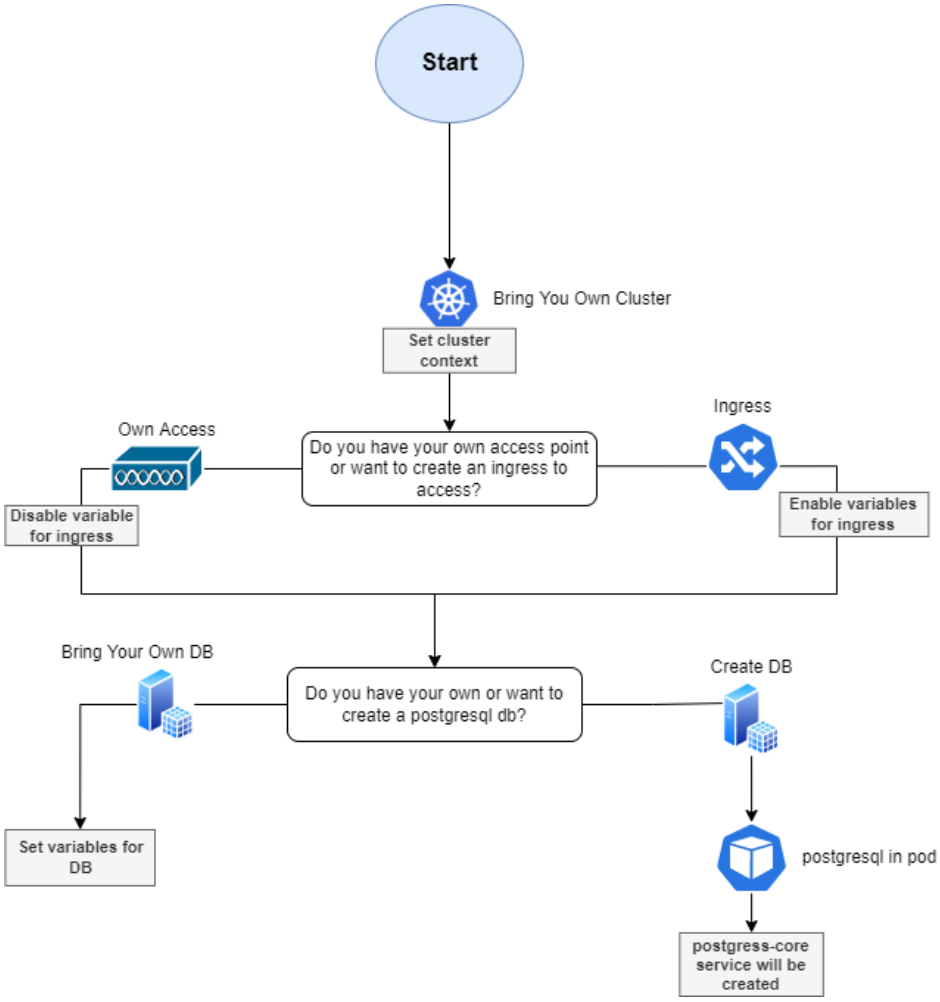
```
env:
  # MD Core proxy settings
  PROXY: false              # Enable the proxy settings in MD Core
  PROXY_HOST: ""            # Proxy server host address
  PROXY_PORT: 8080          # Proxy server port
  PROXY_USER: ""            # (optional) Username for proxy authentication
  PROXY_PWD: ""             # (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.
    a. **An Ingress** will be create per each product flag added as parameter to the script
    b. **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**
    a. **Own database,** will be asked if you want either
        i. the script to set up the credentials and database host url for you
        ii. the script will just indicate the secrets to edit, later on by you, for connecting the MetaDefender Core with your database.
    b. 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
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

```
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

Licensing Configuration

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

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

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:

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
```

High Availability on K8S

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:

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:

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.

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:

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:

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).

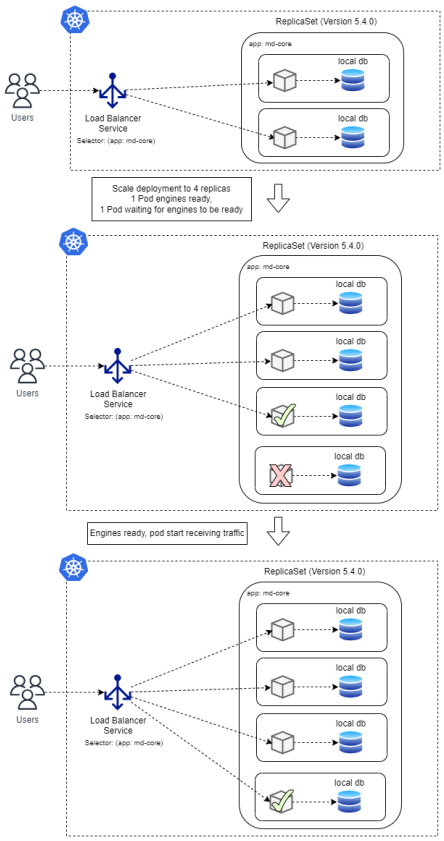
For redundancy, the pods can also be set to run on different nodes by using the `nodeSelector` or `nodeAffinity` Kubernetes features according to the cluster architecture.

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

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

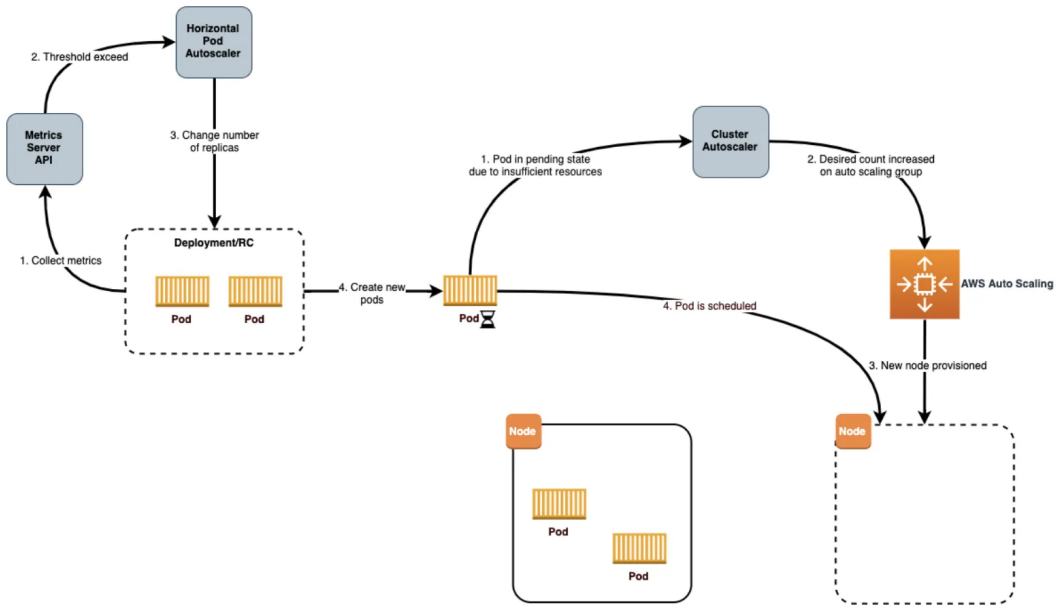
- Kubernetes Cluster already deployed
- Kubernetes Metrics Server installed - Installing the Kubernetes Metrics Server - Amazon EKS
- Kubectl client installed

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

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>

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

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
(<suppor_package_name>)
kubectl exec <md-core_pod_name> -- bash -c "cd /opt/ometascan/core_data/ &
/opt/ometascan/usr/bin/ometascan-collect-support-data.sh"

#export support package, replace <namespace>, <md-core_pod_name> &
<support_package_name>
kubectl cp <namespace>/<md-
core_pod_name>:/opt/ometascan/core_data/<support_package_name>
<support_package_name>
```

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
    a. Check the sidecar container *activation-manager* logs to see if there is any error on deactivation
    b. 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

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:

yaml

```
spec:
  volumes:
  - name: dshm
    emptyDir:
      medium: Memory
  containers:
  - volumeMounts:
      - mountPath: /dev/shm
        name: dshm
```

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:

YAML

```yaml
spec:
  volumes:
  - name: dshm
    emptyDir:
      medium: Memory
  containers:
  - volumeMounts:
      - mountPath: /dev/shm
        name: dshm
```

[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 characters: `*/-+?<>:{}[]()*&^%$#@~`;.,`  Not support `"` (double quotation) and `'` (quotation).
MD_INSTANCE_NAME	optional	instance name
MD_USER	optional	username to create the first admin user
MD_PWD	optional	password to create the first admin user
MD_EMAIL	optional	email to create the first admin user
APIKEY	optional	the API key will be assigned to the admin user for license auto deactivation, activation, HTTPS enabling, proxy enabling and health check enabling.
LICENSE_KEY	optional	an license key for license auto activation
STORAGE_PATH	optional	a full path of storage for sanitized, DLP processed and quarantined files
DATA_DIR	optional	a full path of MetaDefender Core working data directory
REST_ADDRESS	optional	REST binding address for MetaDefender Core's Nginx to be allowed
REST_PORT	optional	REST binding port for MetaDefender Core's Nginx to be allowed

Environmental variable	Necessity	
IGNITION_JSON	optional	The ignition file settings, only JSON format is accepted. For example: `IGNITION_JSON='{"user/name": "admin", "user/password": "admin", "user/email": "admin@local"}'` For database password, this support some specific characters: `*/-+?<>:{}[]()*&^%$#@~`;.,`  Not support `"` (double quotation) and `'` (quotation).
CORE_CONF_JSON	optional	MetaDefender Core configuration file settings, only JSON format is accepted. For example: `CORE_CONF_JSON='{"global/restport": "8009", "logger/loglevel": "info"}'`
CORE_DATA_PATH	optional	a full path to folder (in container) storing all writable data (engine data, logs, runtime data, etc.). Default is */opt/ometascan/core_data* Make sure to mount a volume to this folder to run with the policy.
CERT_PATH	optional	A full path to folder containing certificate file and private key file. There files must have the same filename meanwhile its extensions must be **.crt** and **.key** After being added, the filename without extension will be the name of the certificate in MetaDefender Core
MDCORE_CERT_PASSPHRASE	optional	A passphrase for encrypted private key when adding new certificate via CERT_PATH environment variable. Passphrase must be less than or equal 1023 characters Passphrase does not support single quote character `'`
MDCORE_SSL_PROTOCOLS	optional	Specify the TLS version. Accepted values (case-insensitive): `TLSv1.2,TLSv1.3`

Environmental variable	Necessity	
MDCORE_SSL_ADVANCED_CONFIGS	optional	Configure advanced SSL settings such as cipher suites, session timeout, and caching. Accepted values (case-insensitive): `ssl_verify_depth 10; ssl_session_cache none;` *In case of using docker-compose, they should not input single quote `'` and double quote `"`
MDCORE_DB_NAME	optional	Used for customizing the PostgreSQL database name of MetaDefender Core. By default, in shared DB mode, it is `metadefender_core` Name must begin with a letter (`a-z`) or underscore (`_`). Subsequent characters in a name can be letters, digits (`0-9`), or underscores. Maximum length is 63.
MDCORE_DB_PRIVATE_USERNAME	optional	PostgreSQL server's internal user created for MetaDefender Core own operational purpose. If not specified, then MetaDefender Core will auto generate this user. Details: Customize Internal PostgreSQL User
MDCORE_DB_PRIVATE_PASSWORD	optional	PostgreSQL server's internal user password created for MetaDefender Core own operational purpose. If not specified, then MetaDefender Core will auto generate this user. Details: Customize Internal PostgreSQL User
PROXY	require	**true:** Enable proxy feature. **false:** Disable proxy feature.
PROXY_HOST	require	Proxy server
PROXY_PORT	require	Proxy port
PROXY_USER	optional	Username for proxy authentication
PROXY_PWD	optional	Password for proxy authentication
PROXY_EXCLUSION	optional	List of IPs to be added to proxy exception

Environmental variable	Necessity	
MDCORE_HEALTH_CHECK	optional	**true** • Enable health check feature. • Set 3 health check criterias if customized. **false** • Disable health check feature. • Ignore customized health check criteria. `not defined` • Enable health check feature with • Queue threshold: 100 • Minimum number of AVs: 0 • List of requires engines: empty.
MDCORE_QUEUE_THRESHOLD	optional	Health check criteria. Threshold for scan queue. Unit is %. Default: 100 (%). When allocated slots percentage exceeds the threshold, MetaDefender Core will set itself unhealthy.
MDCORE_MINIMUM_NUMBER_AV	optional	Health check criteria. Minimum number of AV engines. Default: 0. When number of active AV engines is less than or equal to the predefined value, MetaDefender Core will set itself unhealthy.
MDCORE_REQUIRED_ENGINES	optional	Health check criteria. A list of mandatory engines. Default: all the engines of a license. When any predefined engines are not active, MetaDefender Core will set itself unhealthy.

Environmental variable	Necessity	
UPGRADE_DB	optional	**true** • Enable database upgrade process. This migrates Workflow Rules and other settings. • Require `MDCORE_UPGRADE_FROM_DB_NAME` variable. • Require super-user permission for `DB_USER`. • Only support shared-database mode `DB_MODE=4` and remote PostgreSQL `DB_TYPE=remote` . **false** / `not defined` • Database upgrade process will not run.
MDCORE_UPGRADE_FROM_DB_NAME	optional	**Required** when UPGRADE_DB = true If this variable does not exist, upgrade process will exit with code 1.
MIGRATE_HISTORY	optional	**Required** when UPGRADE_DB = true **true** • Database upgrade process migrates processing history, quarantine history, module update and configuration history. **false** / `not defined` • Database upgrade process will not migrate these kinds of data.
MDCORE_LICENSE_COMMENT	optional	**Default**: empty string Add the comment for the license when activating.
OLMS_HOST_URL	optional	The OLMS server endpoint. If users want to use OLMS, this variable is required. MetaDefender Core will treat OLMS higher priority than the normal way in license activation.
OLMS_TOKEN	optional	Token created from OLMS server. If users want to use OLMS, this variable is required.

Environmental variable	Necessity	
OLMS_RULE	optional	Rule containing licenses from OLMS server. If users want to use OLMS, this variable is required.
OLMS_REST_PORT	optional	**Default:** 8040 OLMS port for REST API.
OLMS_SOCKET_PORT	optional	**Default**: 3316 OLMS port for socket interface.
OLMS_CONNECTION_TYPE	optional	**Default:** `production` Can be `production` or `hotbackup`
OLMS_USE_PROXY	optional	**Default**: false If the user want to use proxy for the OLMS activation
OLMS_PROXY_SERVER	optional	**Default**: empty string Proxy server address This field is required if OLMS_USER_PROXY is set to **true**
OLMS_PROXY_PORT	optional	**Default**: empty string Proxy port This field is required if OLMS_USER_PROXY is set to **true**
OLMS_PROXY_PROXY_TYPE	optional	**Default**: empty string Proxy type. Accepted values: • socks4 • socks5 • http This field is required if OLMS_USER_PROXY is set to **true**

Environmental variable	Necessity	
OLMS_PROXY_USERNAME	optional	**Default**: empty string Proxy username
OLMS_PROXY_PASSWORD	optional	**Default**: empty string Proxy password

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=10.40.50.194 \
-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.`

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)

Example

Setup a PostgreSQL version 14 server. After that, we assume that remote PostgreSQL server's information is

- host = **10.40.50.194**
- 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=10.40.50.194 \
  -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
```

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

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.

Anti-malware capabilities

Windows anti-malware

- X = Enabled by default
- O = Supported

Engine name	Vendor name	Country	Signature	Heuristics	PUA/PUP	AI/ML
AhnLab	Harbin AhnLab Technology	South Korea	X		O	
Lionic	Lionic Corp	Taiwan	X			
Antiy	Antiy Labs	China	X	O		
Avira	Avira	Germany	X	X	X	O
BKAV Pro	BKAV Corporation	Vietnam	X			
Bitdefender	Bitdefender	Romania	X	X		
ClamAV	ClamAV	USA	X	X	O	
CMC	CMC Cyber Security	Vietnam	X			
SentinelOne	SentinelOne	USA				X

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
NANOAV	NANO Security	Russia	Signature based	Yes
Zillya!	ALL IT Services	Ukraine	Signature based	Yes*
Anity	Antiy Labs	China	Signature base	Yes*
Comodo	Comodo	USA	Signature based	Yes
McAfee	McAfee	USA	Signature based	Yes
VirusBlokAda	VirusBlokAda	Belarus	Signature based	Yes
Sophos	Sophos	UK	Signature based	No
CrowdStrike Falcon ML	CrowdStrike	USA	AI based	No
RocketCyber	RocketCyber	USA	AI based	No

Engine name	Vendor name	Country	Detection	Heuristic Detection
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
Cylance	BlackBerry	Canada	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.

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, Nano, Comodo, VirusBlockAda and Zillya!
Metadefender 20 Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft, Tachyon, Nano, Comodo, VirusBlockAda, Zillya!, Sophos, Lionic, McAfee and Xvirus
Metadefender MAX Engines	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft, Tachyon, Nano, Comodo, VirusBlockAda, Zillya!, Sophos, Lionic, McAfee, Xvirus, Antiy, CrowdStrike Falcon ML, Microsoft Defender, NETGATE, Systweak, Filseclab, Huorong, Scrutiny, RocketCyber, Webroot SMD, Cylance, CMC and BKAV
Metadefender Public Sector	AhnLab, Avira, Bitdefender, ClamAv, Eset, QuickHeal, K7, Vir.IT, Varist, Ikarus, Emsisoft, Tachyon, Comodo, Sophos, Lionic, McAfee, Xvirus, CrowdStrike Falcon ML, Microsoft Defender, NETGATE, Systweak, Scrutiny, RocketCyber, Webroot SMD and Cylance

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, Nano, Sophos, McAfee, CrowdStrike Falcon ML, Lionic, Xvirus, Webroot SMD, Cylance, RocketCyber and CMC

Public Sector | 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
Cyren	Cyren	Israel	Signature based
Emsisoft	Emsisoft	New Zealand	Signature based
IKARUS	IKARUS Security Software	Austria	Signature based
TACHYON	INCA Internet	South Korea	Signature based
Comodo	Comodo	USA	Signature based
McAfee	McAfee	USA	Signature based
Sophos	Sophos	UK	Signature based
CrowdStrike Falcon ML	CrowdStrike	USA	AI based
RocketCyber	RocketCyber	USA	AI based
Webroot SMD	OpenText	USA	Signature and AI based
Lionic	Lionic Corp	Taiwan	Signature based
Lavasoft	Adaware	Canada	Signature based
Microsoft Defender	Microsoft	USA	Signature based
NETGATE	Netgate Technologies	Slovakia	Signature based
Scrutiny	Cyberstanc	USA	AI based
Xvirus Anti-Malware	Xvirus	Portugal	Signature based

Engine name	Vendor name	Country	Detection
Systweak	Systweak	India	Signature based
Cylance	BlackBerry	Canada	AI based

Please check the configuration page in the product for the detail Detection methods (Heuristic, Archive extraction, ...) of each engine

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

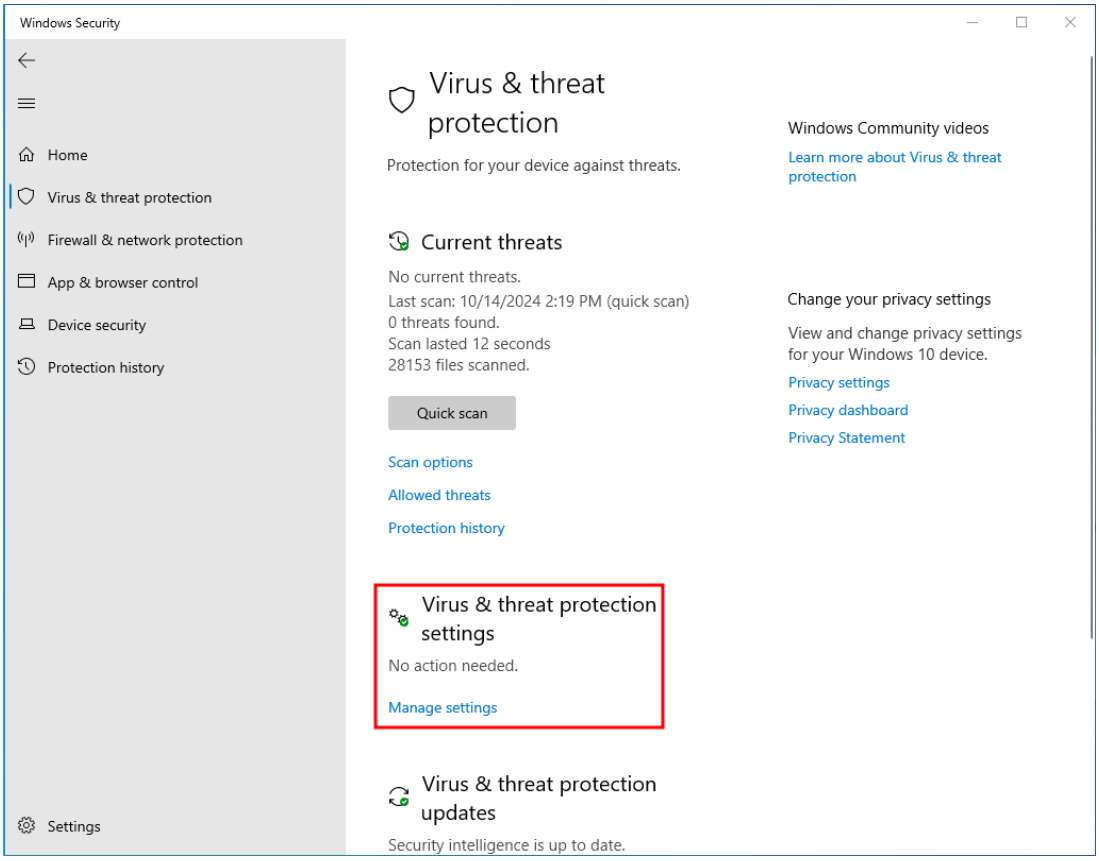
Disclaimer
This information is provided for general informational purposes only. OPSWAT is not responsible for the accuracy or reliability of any information provided. The examples mentioned are for illustration only and do not constitute an endorsement by OPSWAT.

- 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/]

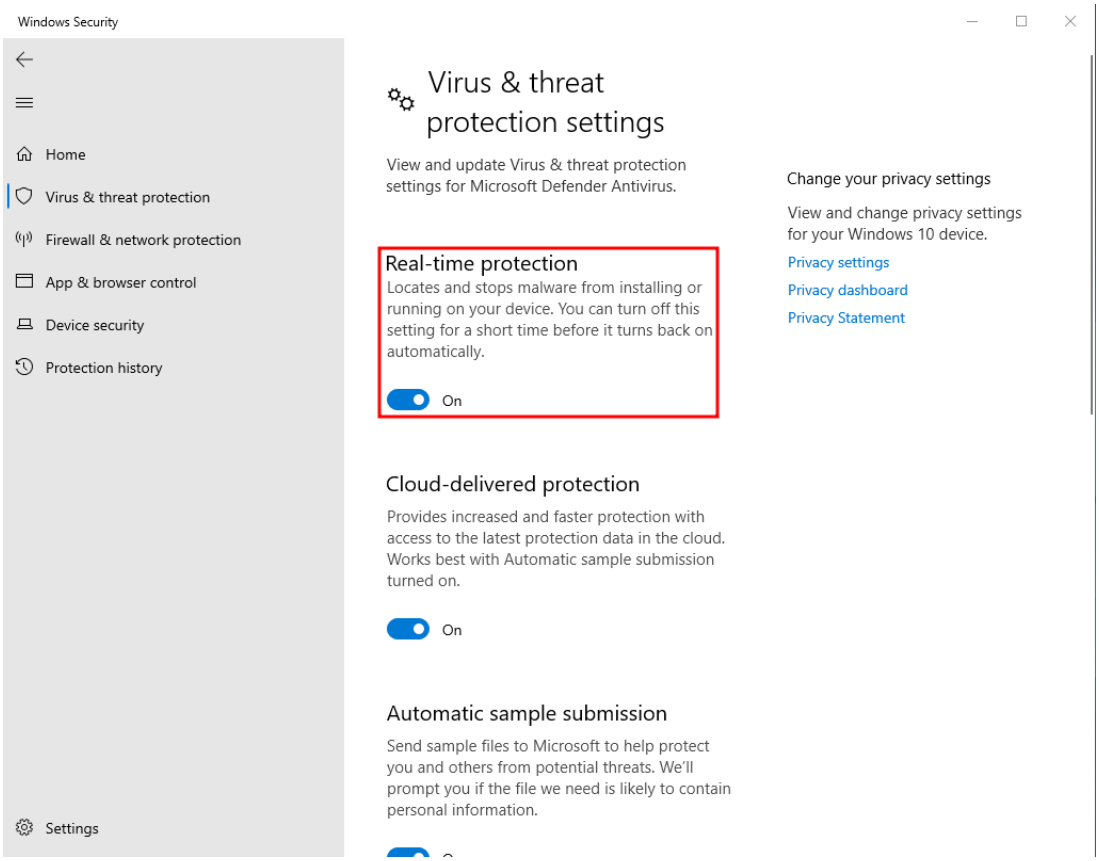
Set up Windows Defender as a custom engine

In order to set up and use Windows Defender as a custom engine in MetaDefender Core, the following conditions must be met:

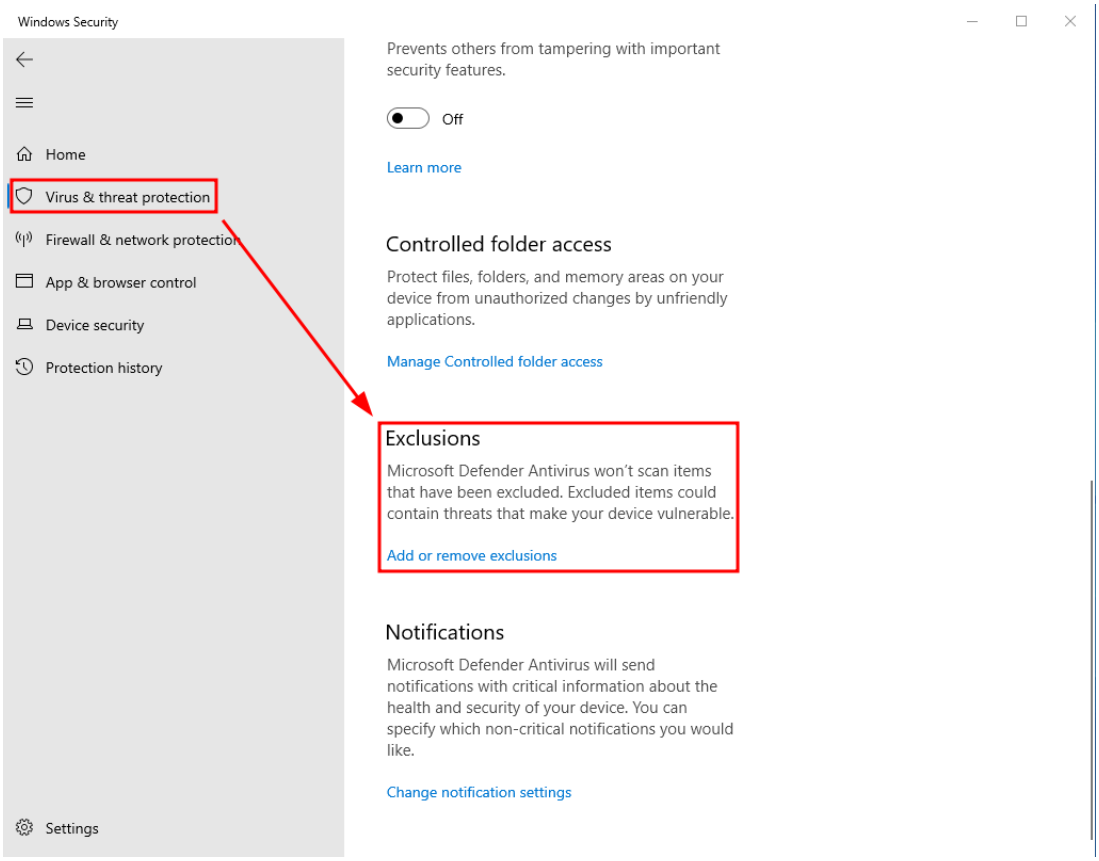
1. The native Windows Defender / Windows Security must exist on the Windows OS.
2. **Real-time protection** must be turned on.



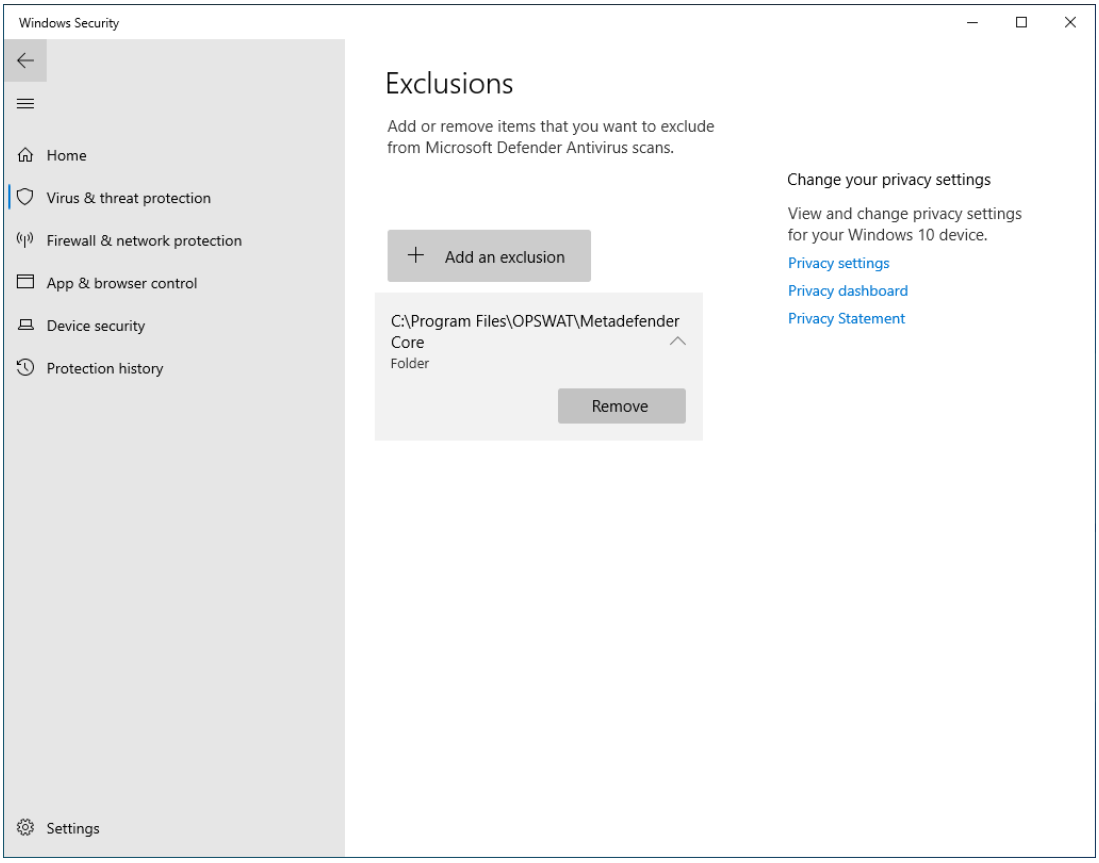
In case of the **Real-time protection** feature is disabled, it can be enabled on the **Manage settings** screen.



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:
    a. download the Windows_Defender_Passive_Mode archive and extract it.
    b. execute the **enable_windows_defender_passive_mode.reg** to automatically add the following two keys to the registry.
    c. 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!

bash

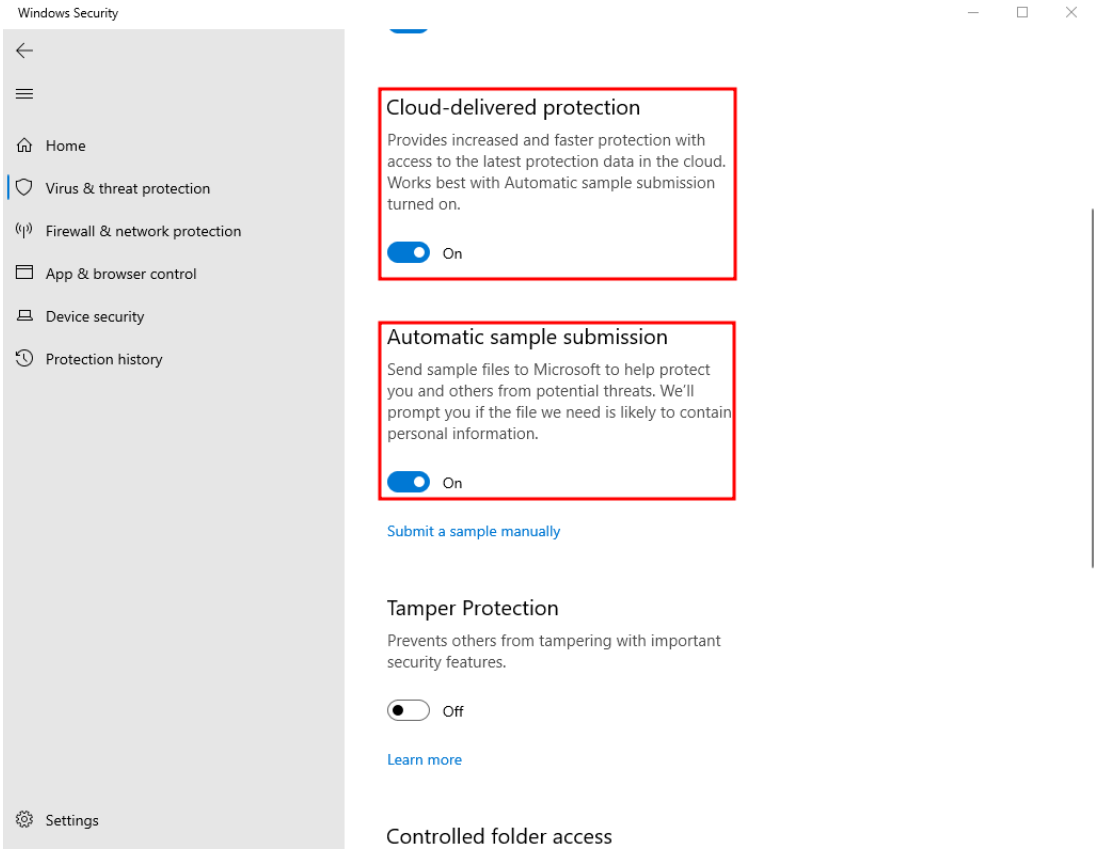
```
[HKEYLOCAL MACHINE\SOFTWARE\Policies\Microsoft\Windows Advanced Threat
Protection]
"DisableRoutinelyTakingAction"=dword:00000001
"ForceDefenderPassiveMode"=dword:00000001
```

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

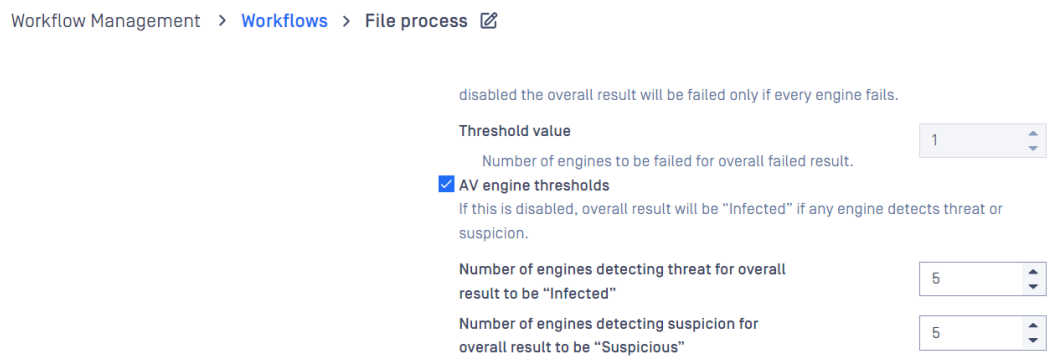
In order to turn off cloud protection or automatic sample submission, turn each feature off in Windows Defender's settings



AV scoring with post action

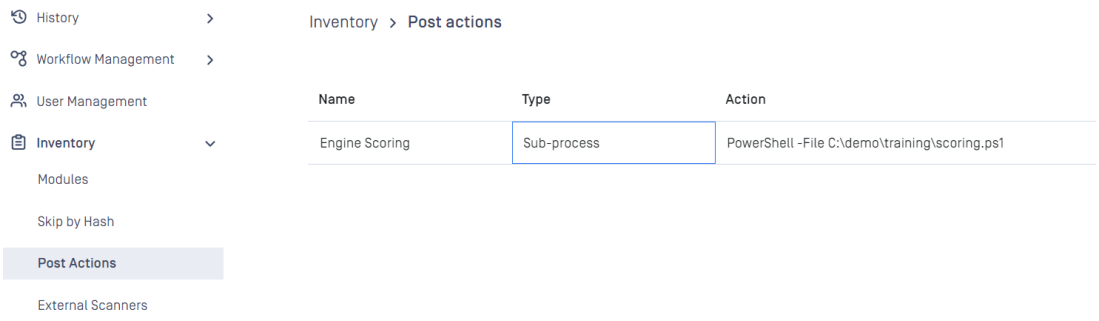
If certain engines frequently produce false positive detections, you can use the "Post Action" feature (available in MetaDefender Core 5.8.0 and later) to execute a script that modifies the final verdict. Within the script, you can implement logic to assign high scores to specific engines and low scores 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.

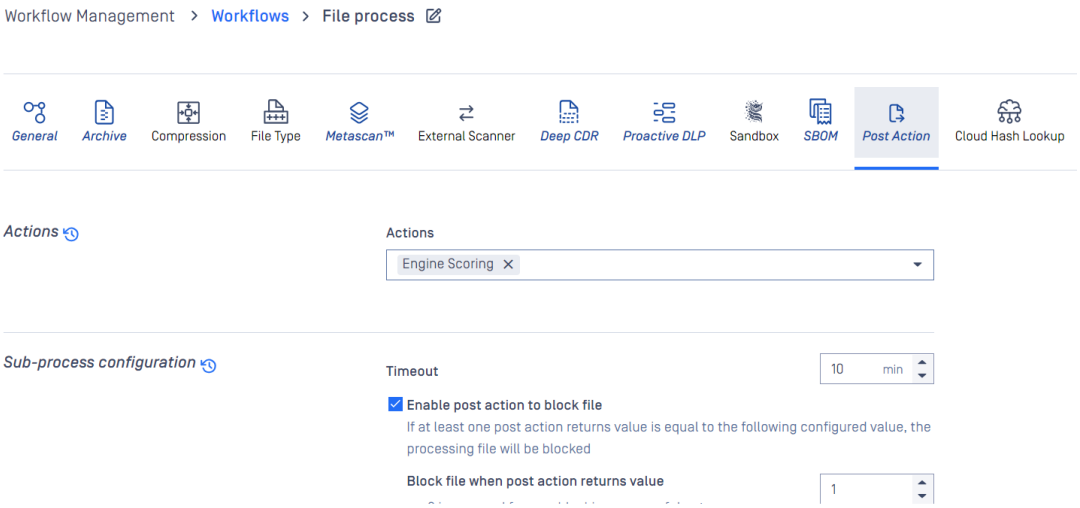


Step 2: Download the scripts at OPSWAT Github and follow the readme (This project is just an example, use and modify it at your own risk)

Technically, you create a post action, for example, on Windows:



Step 3: Configure a workflow to use this post action.


Workflow Management > Workflows > File process
General
Archive
Compression
File Type
Metascan™
External Scanner
Deep CDR
Proactive DLP
Sandbox
SBOM
Post Action
Cloud Hash Lookup
Actions
Actions
Engine Scoring
Sub-process configuration
Timeout
10
min
Enable post action to block file
If at least one post action returns value is equal to the following configured value, the processing file will be blocked
Block file when post action returns value
1

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 201)

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
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 Document	**docm**, docx, txt, html, pdf, ps, jpg, 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**, html, pdf, ps, jpg, bmp, png, tiff*, svg
potx	Microsoft PowerPoint Template	**potx**, pdf, png

Source File Type	Description	Target Sanitized Types
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-Enabled Show	**ppsm**, pdf, png
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

Source File Type	Description	Target Sanitized Types
one	Microsoft OneNote	**one**
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**
pdf	Adobe Portable Document Format	**pdf**, html, svg, jpg, bmp, png, tiff, txt, pptx
ai	Adobe Illustrator	**ai**
ait	Adobe Illustrator Template	**ait**
hwp	Hangul Word Processor	**hwp**
hwt	Hangul Word Template	**hwt**
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**

Source File Type	Description	Target Sanitized Types
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**
vcs	vCalendar	**vcs**
ics	iCalendar	**ics**
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

Source File Type	Description	Target Sanitized Types
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**
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**
dwg	AutoCAD	**dwg**
dwt	AutoCAD Drawing Template	**dwt**
dws	AutoCAD Drawing Standards	**dws**
sfc	SXF Feature Comment	**sfc**
p21	STEP Data Model	**p21**
jww	JW CAD	**jww**
jwc	JW CAD	**jwc**
bfo	V-nas BFO	**bfo**
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
dae	Digital Asset Exchange	**dae**, 3ds, stl, fbx

Source File Type	Description	Target Sanitized Types
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
dcm	Digital Imaging and Communications in Medicine	**dcm**
shp	Shapefile	**shp**
shx	Shapefile	**shx**
dbf	Shapefile	**dbf**
heic	High Efficiency Image File Format	**heic**, jpg, pdf
wsq	Wavelet Scalar Quantization FBI fingerprint format	**wsq**
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**
mpp	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**

Source File Type	Description	Target Sanitized Types
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
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
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**

Source File Type	Description	Target Sanitized Types
pst	Outlook Personal Folder	**pst**
mbx	Outlook Express Mailbox	**mbx,** zip
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
tar	Tape Archive	**tar,** zip, 7z, gz, xz

Source File Type	Description	Target Sanitized Types
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
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	zip
b3dm	Batched 3D Model	zip
asics	Associated Signature Container	**asics**
asice	Associated Signature Container	**asice**
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**
esz	E-szigno	**esz,** zip
base64	Base64 Encoding	**base64**
zstd	Zstandard Compressed	**zstd**

Source File Type	Description	Target Sanitized Types
yz1	Yamazaki Zipper Compressed Archive	**yz1***
scdoc	SpaceClaim Document	**scdoc**
mdzip	MagicDraw	**mdzip**

- (*) Only supported on Windows for now.
- For the archive sanitization, please enable it in the Compression tab

Note

Sanitization is in BETA for these file types:

- ACM, 3GP, M4V, MKV
- SPF
- TWBX
- UTIB
- STORY
- ASICS, ASICE
- ES3, PCK
- ZEI
- EMZ
- MBX, TNEF
- B3ML, GPKG
- JNLP
- TGA
- BASE64, ZSTD, YZ1, SCDOC,
- URL, MDZIP
- WSQ, ODG, HEVC, TTML, MPD

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: 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→HTML, IMG
- XLSX->HTML, CSV, IMG

- PPTX→HTML, IMG

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
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)

File types	Supported Versions
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
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

File types	Supported Versions
DWG, DWT, DWS	DWG 2000, DWG 2004, DWG 2007, DWG 2010, DWG 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

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:

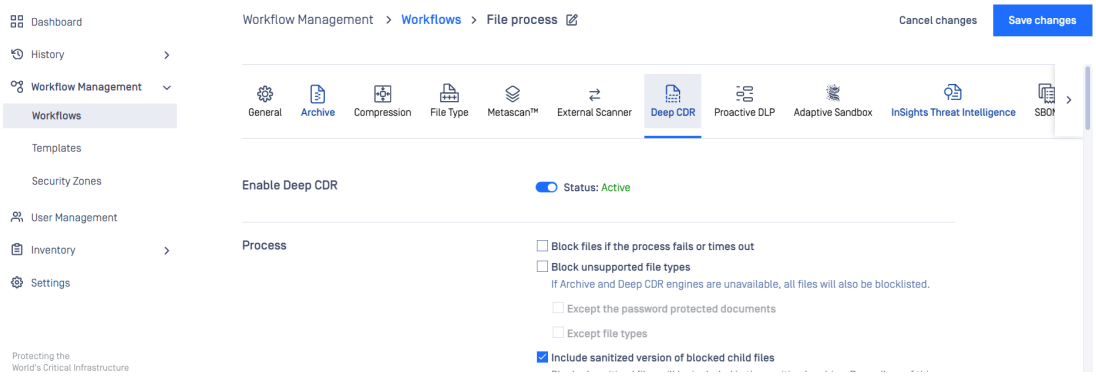
- CPU: Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz (8 CPUs)
- RAM: 32GB
- SSD: 500GB

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	
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	

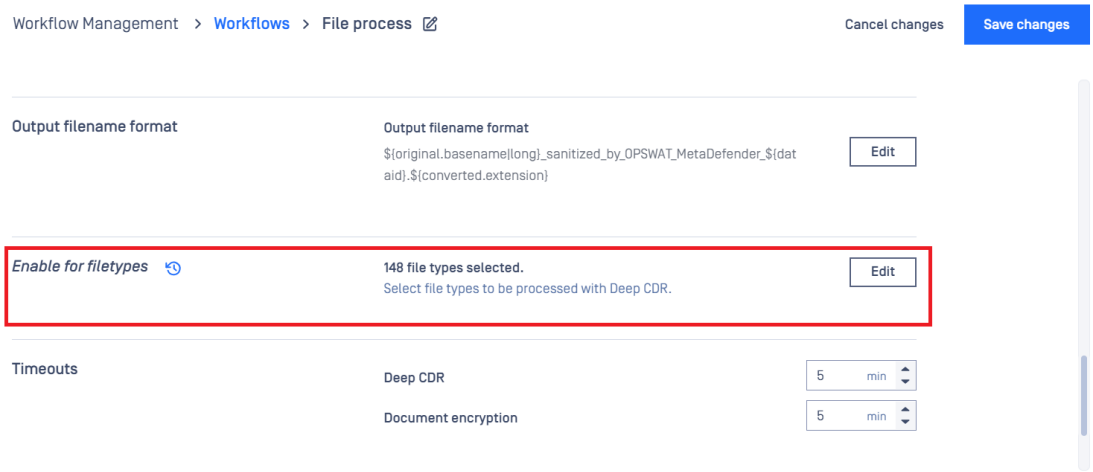
File type	File Size (MB)	Processing time (s)	Notes
WAV	688	35	

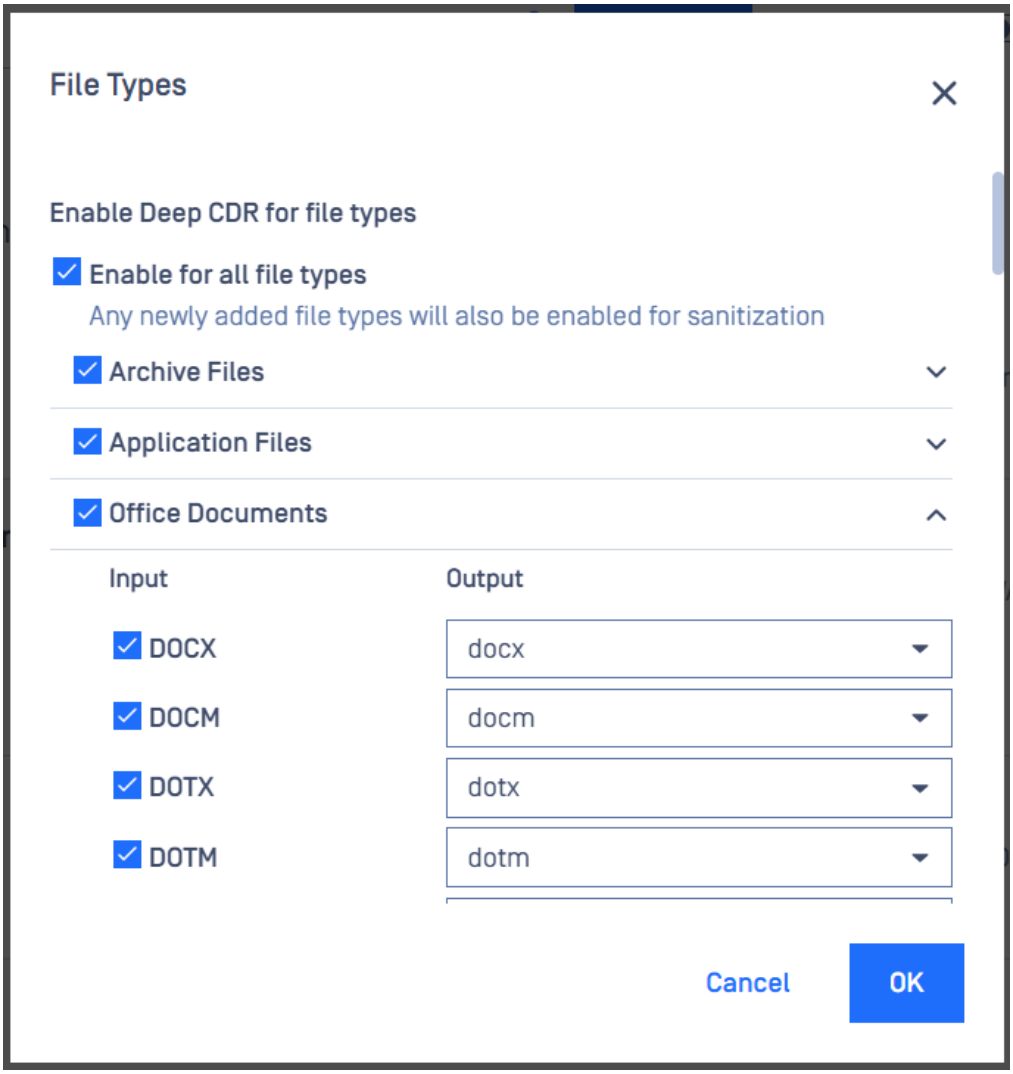
Configurations

To enable Deep CDR, go to **Workflow Managements > Workflows > [Workflow name] > Enable Deep CDR**

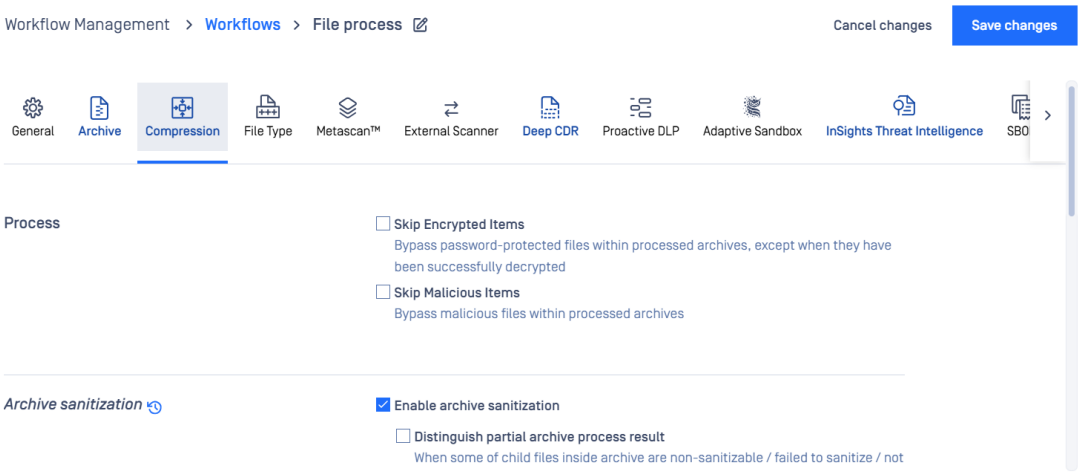


To enable file types to be processed

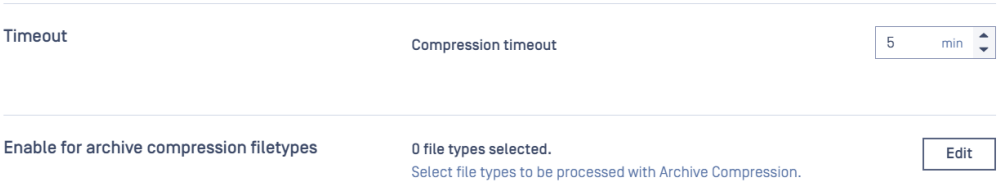


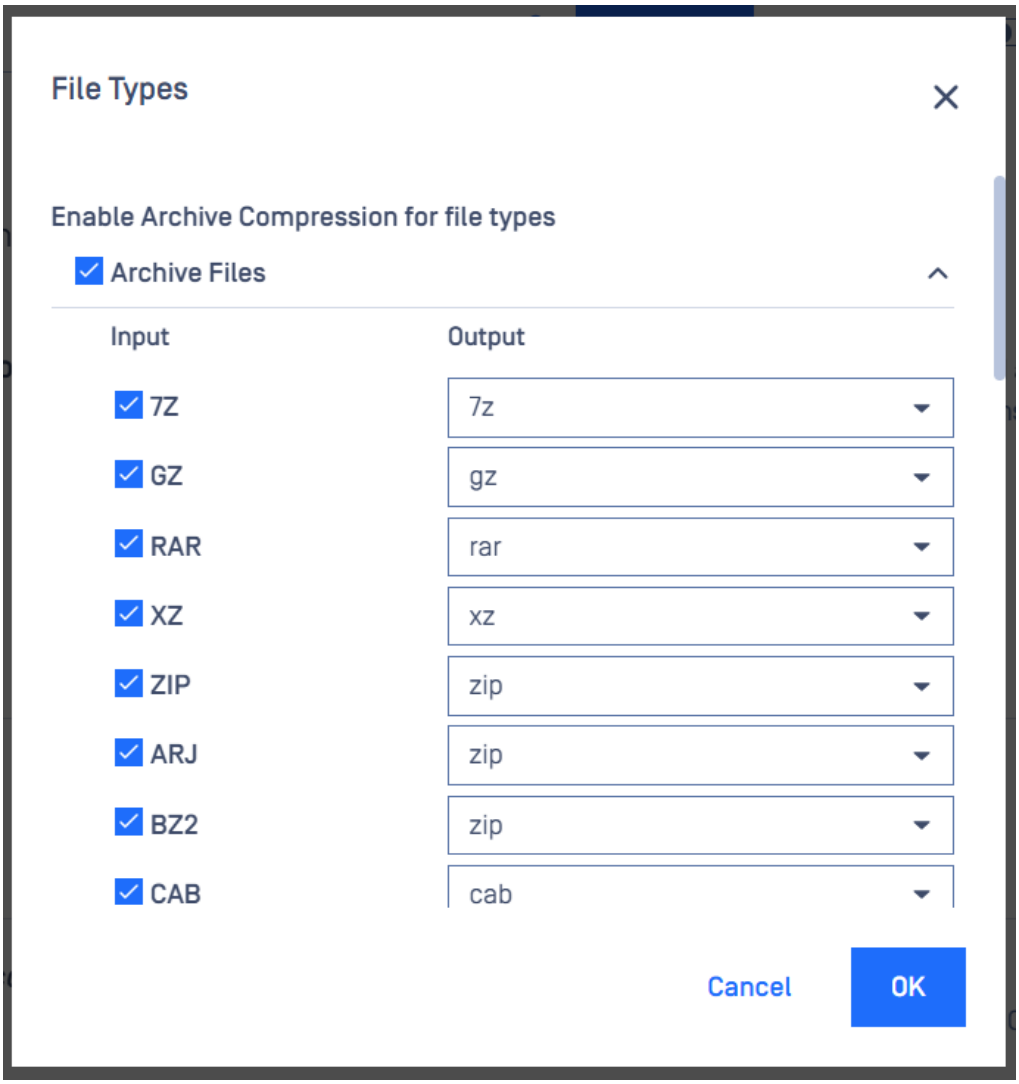


To enable Archive file types sanitization, go to the Compression tab



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.

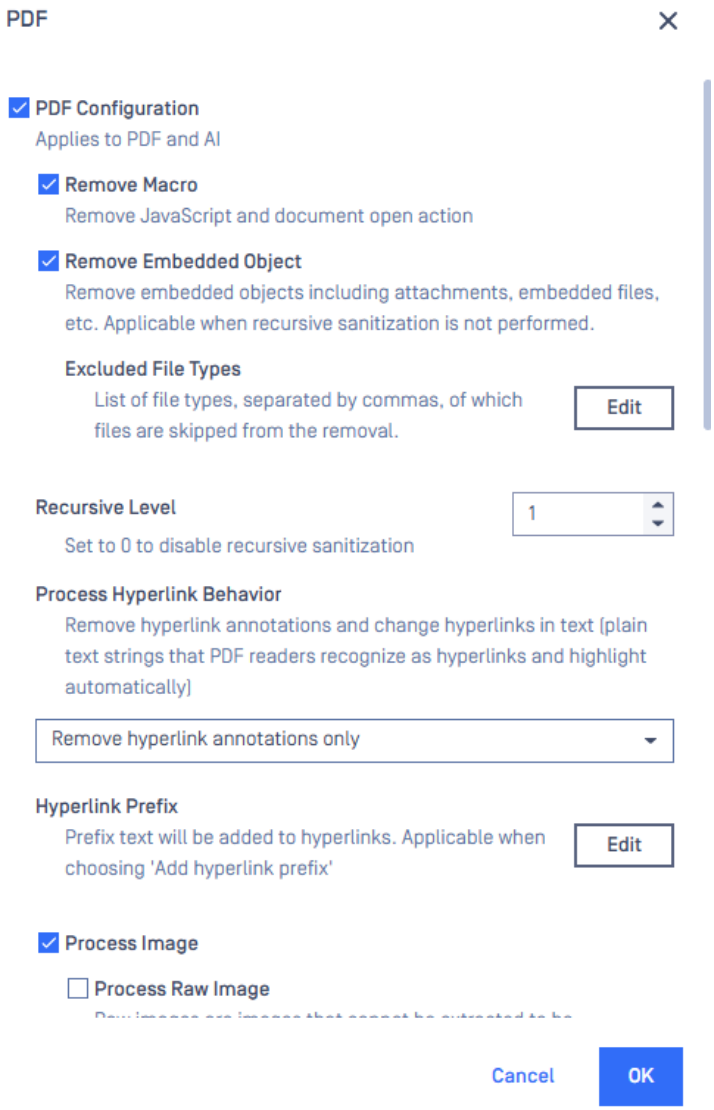
The engine configuration is accessible under **Workflow Management > Workflow > {Workflow name} > Deep CDR > Advanced configuration > File type handling > {File type}**

File type handling

Select a file type to configure how it is handled by Deep CDR

Adobe	
PDF	Portable Document Format
Office Document	
Microsoft Word	Microsoft Word Document
Microsoft Excel	Microsoft Excel Document
Microsoft Powerpoint	Microsoft Powerpoint Document
Microsoft OneNote	Microsoft OneNote Document
RTF	Rich Text Format
Visio	Microsoft Visio Drawing
Image	
Image	Raster Image
SVG	Scalable Vector Graphics
DWF	Design Web Format
DWG	AutoCAD Drawing
Open Office	
Media	
Japanese	
Korean	
Others	

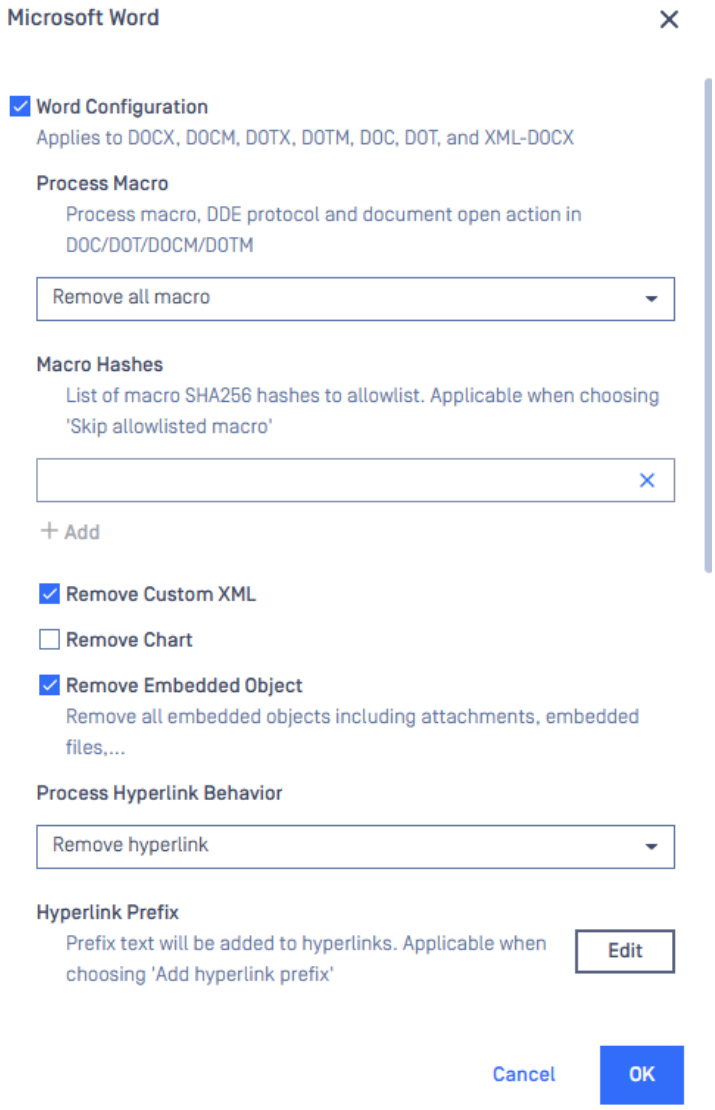
PDF



- **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.
- **Recursive Level:** Enable/disable recursive sanitization. Default is 1 level.
- **Process Hyperlink Behavior:** Remove hyperlink annotations and change hyperlinks in text (plain text strings that PDF readers recognize as hyperlinks and highlight automatically).
    - Remove hyperlink annotations only.
    - Remove hyperlink annotations and text links.
    - Remove hyperlink annotations and prevent clickable links auto-tuning.
    - Remove hyperlink annotations and leave them in text form.
    - Add hyperlink prefix.
    - Return list of hyperlinks.
    - Do nothing (Keep as is).
- **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.
- **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:** Strictly report standard compliance issues or ignore slight error in file structure.
- **Skip Signed File:** Skip processing PDF file with digital signature.
    - **Deep CDR only validates the file integrity and the signature, it can't recognize if a signature is a self-signed signature or not.**
- **Validate digital signature**
- **Process 3D object**
- **Process PostScript Object:** PostScript object data are set to empty and related streams are removed.
- **Remove User-Defined Named Action:** Named actions are predefined actions that can force the PDF viewer application to invoke an action that might adversely impact the system. User-Defined Named Actions object data are set to empty and related streams are removed.
- **Allowed Named Actions:** List of Named actions should be allowed.
    - Note: NextPage, PrevPage, FirstPage, and LastPage are always allowed as they just navigate around a document.

Office Document



Microsoft Word/Excel/Powerpoint document

- **Process Macro:** Process macro, DDE protocol and document open action in DOC/DOT/DOCM/DOTM/XLS/XLT/XLSM/XLTM/XLSB/XLAM/PPT/PPS/POT/PPTM/PPSM/SLDX/SLDM.
  - Remove all macro.
  - Skip allowlisted macro.
  - Do nothing (Keep as is).
- **Skip Signed Macro:** Signed macro is detected and validated against its signature and excluded from removal if valid. Applicable to DOCM, DOTM, PPTM, PPSM, POTM, PPAM, SLDM, XLSM, XLTM and XLAM.
- **Remove Custom XML**
- **Remove Chart**
- **Remove Embedded Object:** Remove all embedded objects including attachments, embedded files, etc.
- **Process Hyperlink Behavior:**
  - Remove hyperlink.

    - Add hyperlink prefix.
    - Return list of hyperlinks.
    - Do nothing (Keep as is).
- **Process Image**
- **Sanitize QR Code:** Hyperlink in QR code is detected and processed based on Image Configuration.
- **Remove Comment:** is applied to DOCX family (DOCX, DOTX, DOCM, etc.), XLSX family (XLSX, XLSM, XLSB, etc.), PPTX family (PPTX, PPTM, PPSX, etc.).
- **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.
- **Cleanup Unused Resources:** Remove unused styles and lists from DOC/DOT/XLS/XLT files.
- **Recursive Level:** Enable/disable recursive sanitization. Default is 1 level.
- **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, XLSX family and PPTX family.
- **Skip Signed File:** Skip processing file with digital signature.
    - **Deep CDR only validates the file integrity and the signature, it can't recognize if a signature is a self-signed signature or not.**
- **Validate Document Properties:** Enable validation on Document Properties. Applicable to DOCX family, XLSX family, PPTX family, XML-DOCX and XML-PPTX.
    - **Validation Effect on Sanitization Result:** Choose how failure result of validation affects sanitization result. Default is Success.

Microsoft OneNote document

- **Remove Embedded Object:** Remove all embedded objects including attachments, embedded files, etc.
- **Process Image**
- **Process Hyperlink Behavior:**
    - Remove hyperlink.
    - Add hyperlink prefix.
    - Do nothing (Keep as is).

RTF

- **Remove Embedded Object:** Remove all embedded objects including attachments, embedded files, etc.
- **Process Hyperlink Behavior:**
    - Remove hyperlink.
    - Add hyperlink prefix.
    - Do nothing (Keep as is).
- **Remove Embedded HTML:** Remove HTML tags containing malicious HTML nodes.
    - Remove HTML tags containing malicious nodes.
    - Remove all HTML tags.
    - Do nothing.

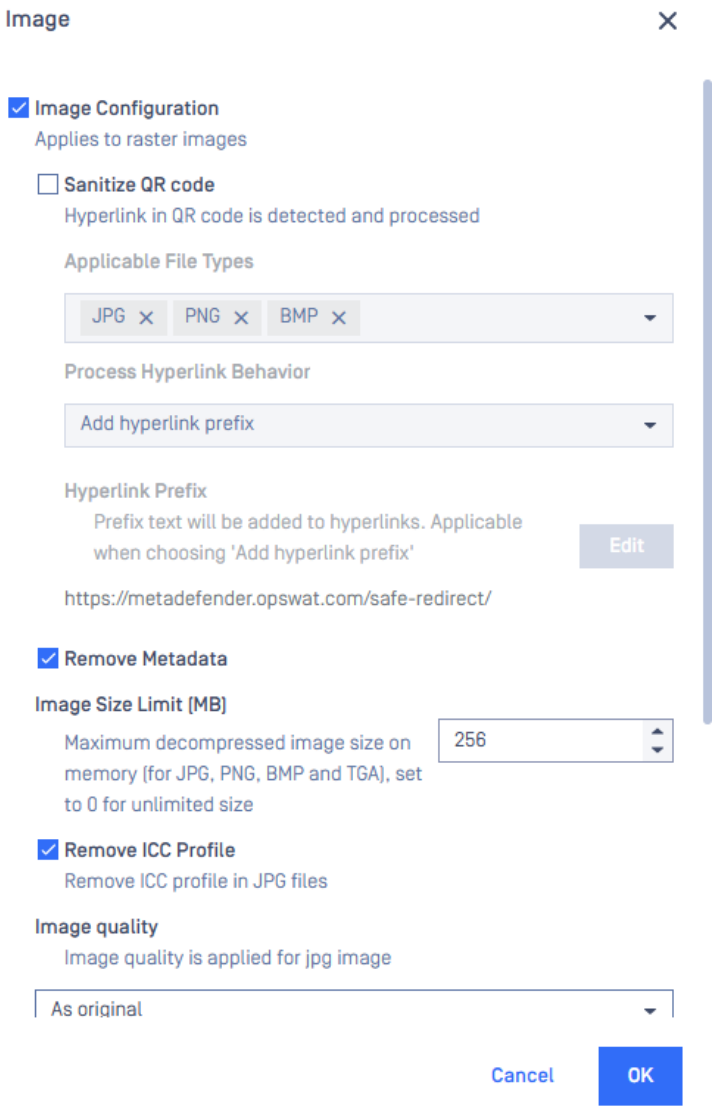
- **Process Image**
- **Process font table:** Remove embedded font table. Font table exceeding a font count setting will be removed.

Microsoft Visio Drawing

- **Process Macro**
- **Remove Embedded Object:** Remove all embedded objects including attachments, embedded files, etc.
- **Process Hyperlink Behavior:**
  - Remove hyperlink.
  - Add hyperlink prefix.
  - Do nothing (Keep as is).
- **Process Image**
- **Remove Metadata**
- **Recursive Level:** Enable/disable recursive sanitization. Default is 1 level.

Image

Deep CDR supports sanitization for AutoCAD, raster and vector image files.



Raster image files

- **Sanitize QR code:** Hyperlink in QR code is detected and processed. Process Hyperlink Behavior:
  - Add hyperlink prefix.
  - Return list of hyperlinks.
- **Remove Metadata**
- **Remove ICC Profile:** Remove ICC profile in JPG files.
- **Remove GeoTIFF tags:** Remove or preserve private TIFF tags which store georeferencing information.
- **Image quality:** is applied for JPG image (as original or 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:** Maximum decompressed image size on memory for TIFF, set to 0 for unlimited size.

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.

DWF

- **Remove Embedded Font**
- **Process Hyperlink Behavior**: Remove all hyperlinks, Add hyperlink prefix or Do nothing (Keep as is).
- **Process Image**: Process raster images included inside SVG file.

DWG

- **Remove Macro**

Text

TXT

- **Remove Invisible Character**: Remove non-printable UTF-8 characters in these categories:
  - Format character (cf): U+00AD (Soft Hyphen - SHY), U+200E (Left-to-right mark), U+200F (Right-to-left mark), ...
  - Control character (cc): U+0000 (NULL), U+0001 (Start of heading), U+0002 (Start of text)
  - Private use character (co): U+E000 to U+F8FF, U+F0000 to U+FFFFD, U+100000 to U+10FFFD
  - Unassigned character (cn): code points that have not been assigned characters by the Unicode standard.

Markup

HTML

- **Remove Script:** Remove all script.
- **Remove Object**
- **Remove Applet**
- **Process Form:** Remove actions in forms
- **Remove Comment:** Remove all comment from HTML file.
  - **Remove Conditional Comment:** Remove conditional comment from HTML file.
- **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.
        - **Process Text Link:** If enabled, add prefix to text links besides hyperlinks. Only applicable when choosing 'Add hyperlink prefix'
        - **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 Tags**: Accept documents with multiple separate HTML tags
- **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.

JSON

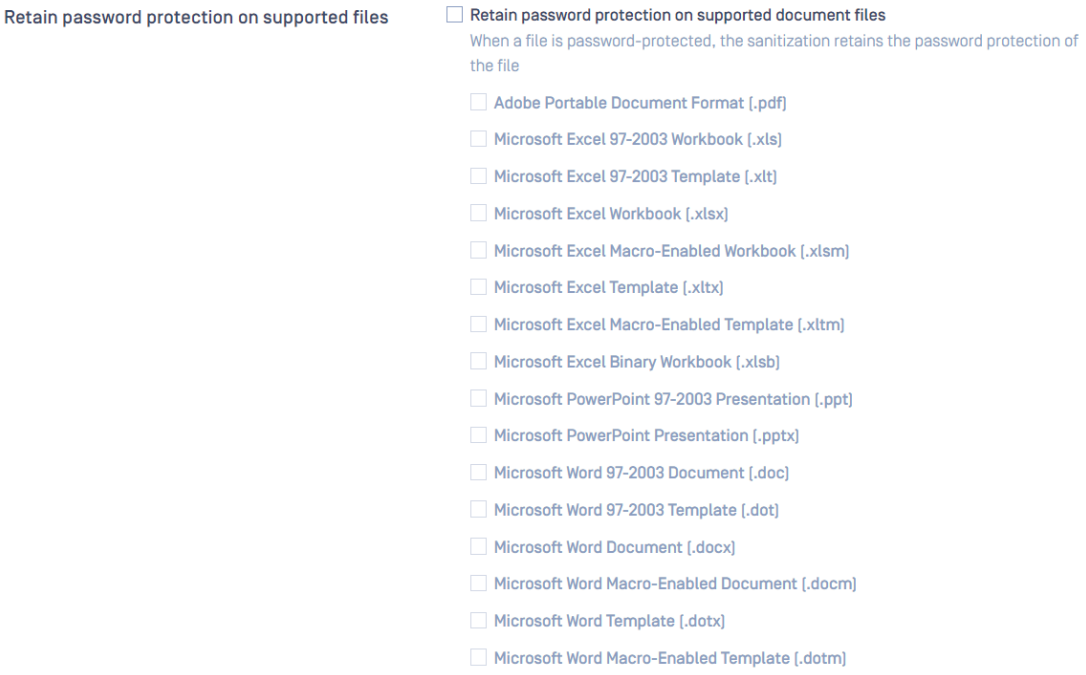
- **Preserve Format:** Preserve formats like whitespaces, newlines, etc.
- **Support Newline Delimited:** Support Newline Delimited JSON
- **Process Base64 encoded data**
    - **Data following "data" URL scheme**
    - **Data at certain JSON path**
    - **Action on data when sanitization not applied**

Retain password protection on sanitized files

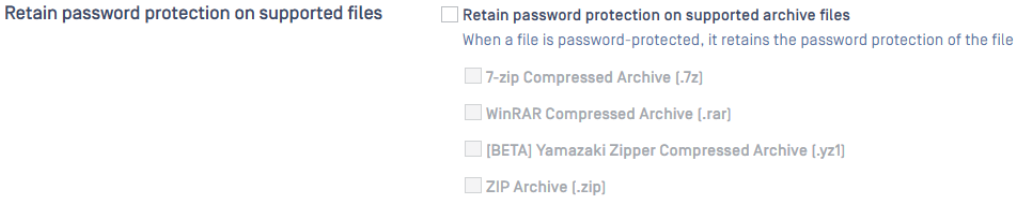
When having a password-protected encrypted file processed by Deep CDR, you can decide whether to encrypt the sanitized file with the same password or no password.

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:

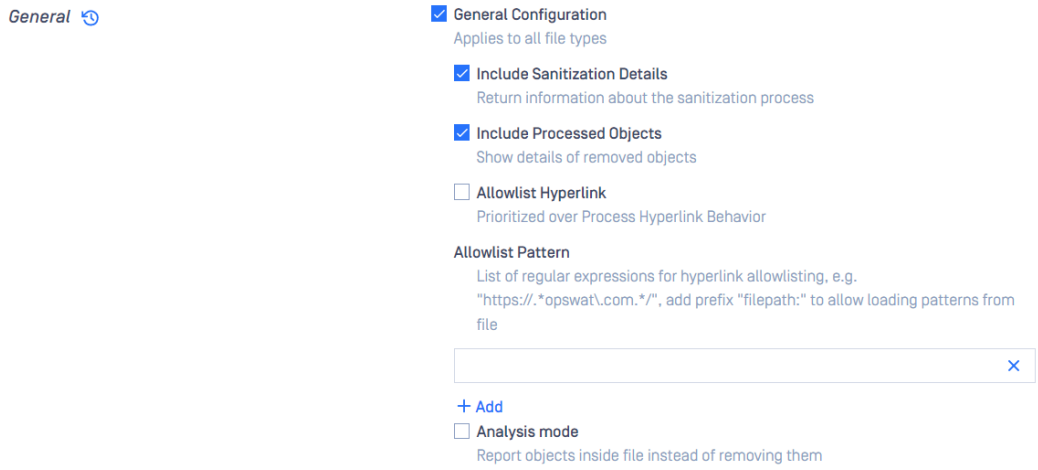
- If you enable "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing" on Windows, the encryption will not work with PPTX file.

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.

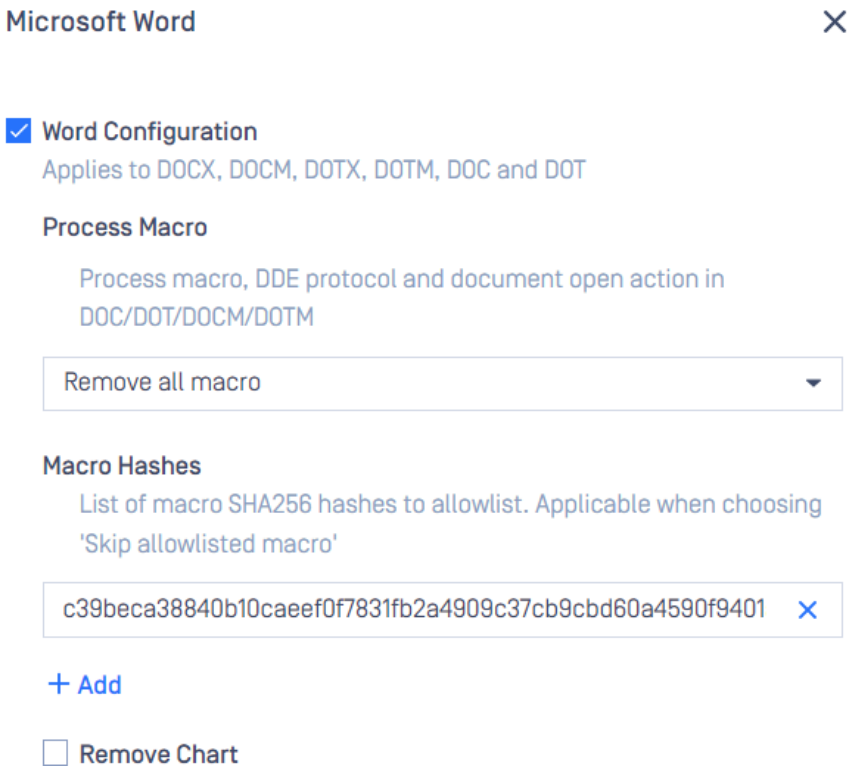
- Enable “INCLUDE PROCESSED OBJECTS” in Workflow Deep CDR Setting



- 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

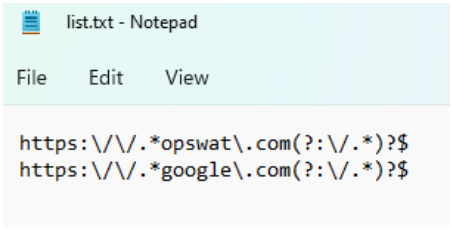
Allow Hyperlinks

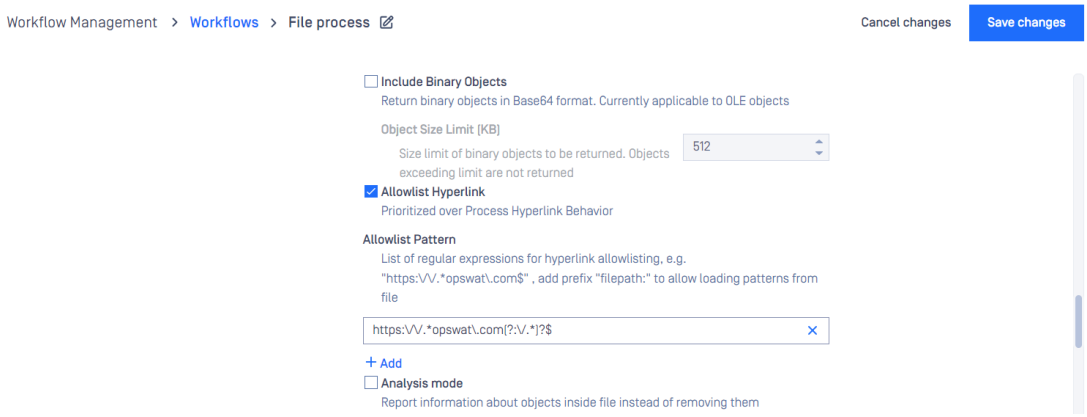
Despite hyperlinks' centrality to the user experience, connecting pages and sites with a single click, they could be a high-risk attack vector. To ensure that all potential threats in files were eliminated before reaching the users, hyperlinks should be removed during Deep CDR process by default. However, 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:

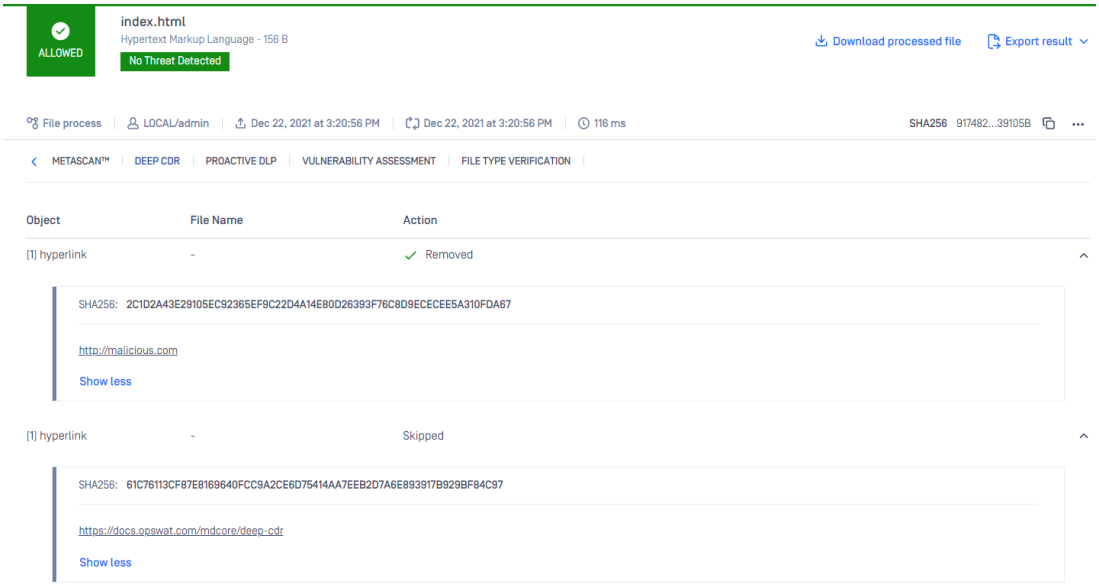
- Regular expression
- or **filepath**:<file path to a file which contains a regular expression list, line by line>, e.g.,





The above example allows all links from **opswat.com** domain/sub domain.

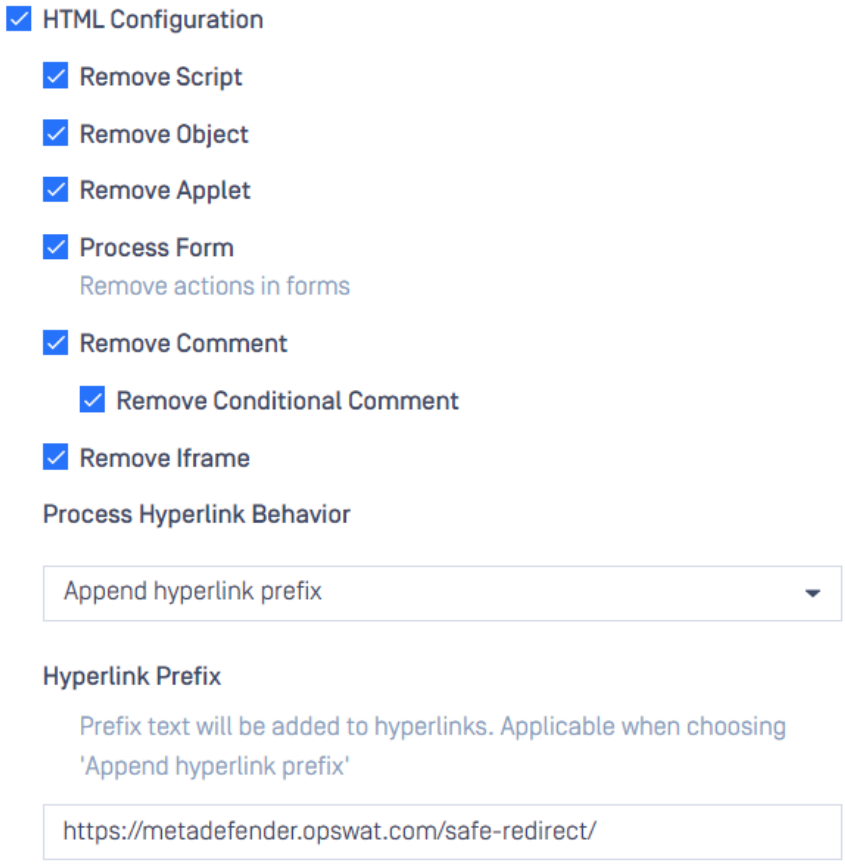
An example output when processing a file with two hyperlinks (http://malicious.com, https://docs.opswat.com/mdcore/deep-cdr) on it.



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"

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.

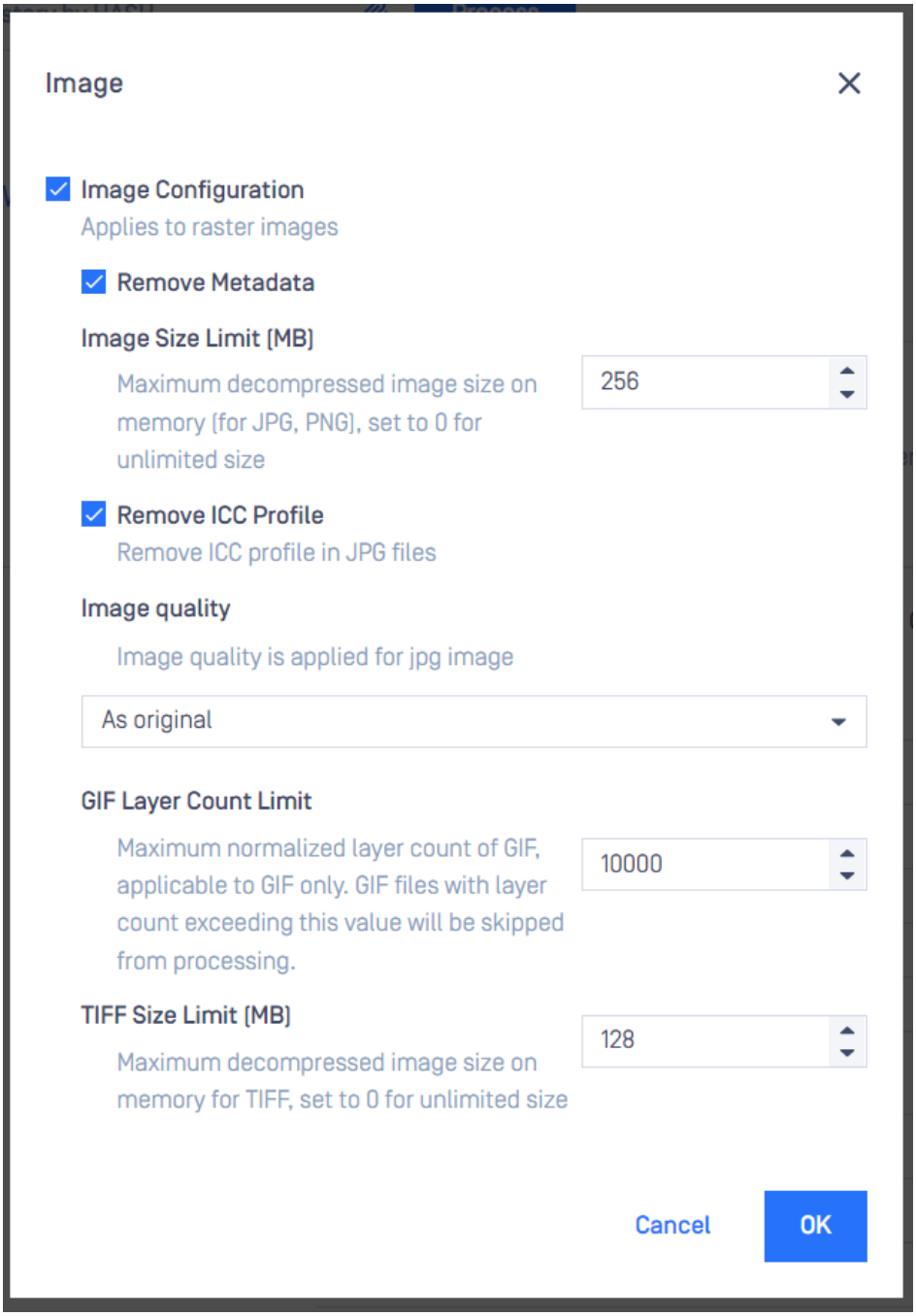


Image Size Limit (MB): A JPG/PNG file with a few byte in file size can be decompressed to become a few to a hundred MB. Therefore, this configuration is applied for the **decompressed raw data size**.

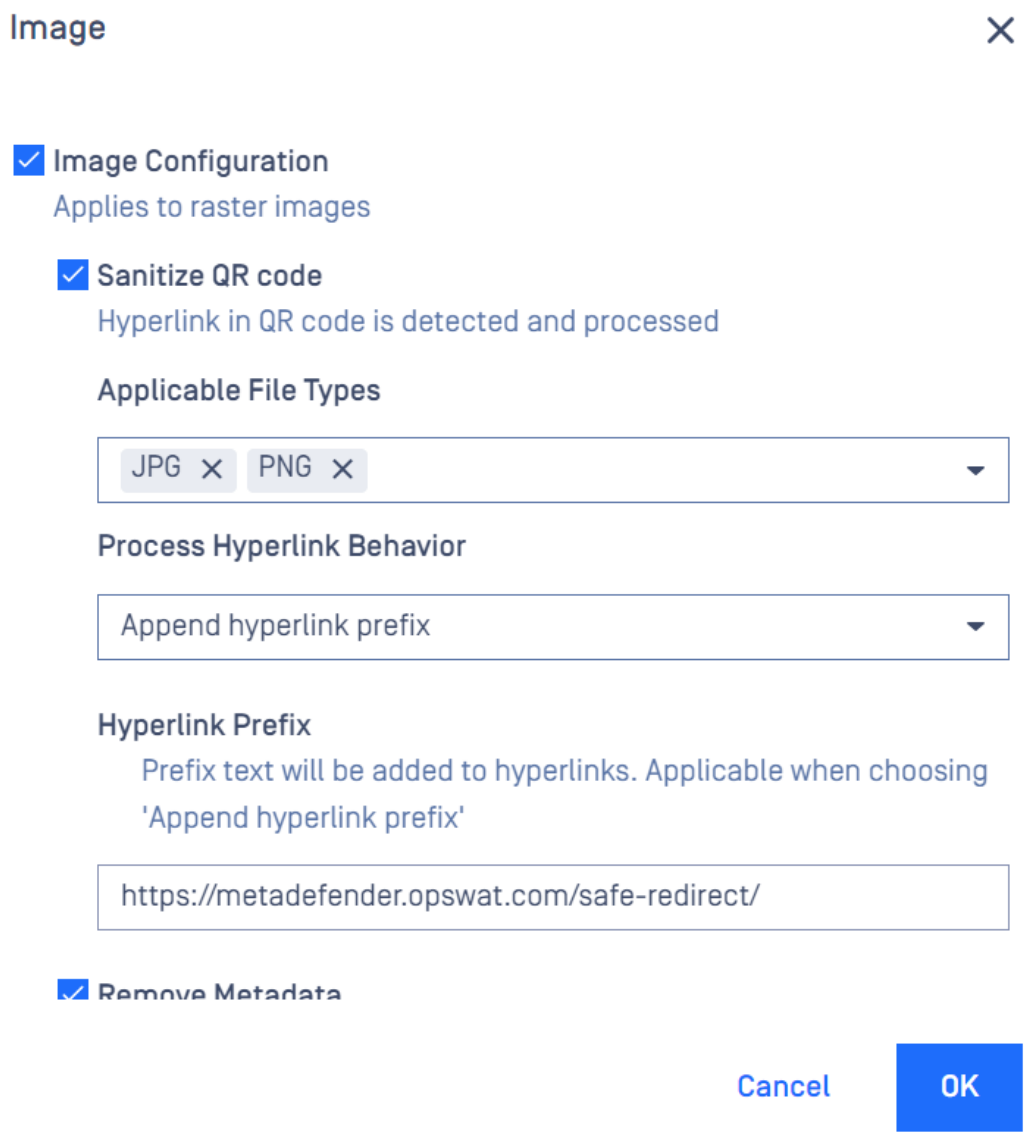
TIFF Size Limit (MB): why do we separate PNG/JPG size limit from the TIFF size limit? The TIFF file processing is expensive when it uses Group 4 compression but not in the other format. Also, the TIFF raw data size is calculated in a different way from PNG/JPG. Therefore, using the same values with JPG/PNG 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

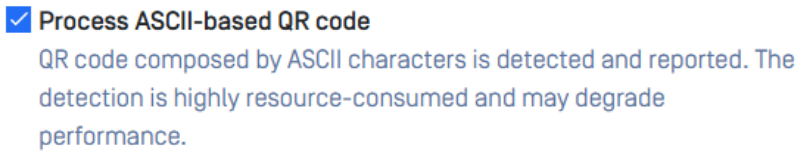
QR code can be created in a document by a combination of ASCII block characters (e.g., ▄). This tactic is designed to prevent security software from extracting malicious URLs from QR codes. Attackers can embed malicious links into the QR code and encourage users to scan the code, which would then take 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 data can contain base64-encoded data within certain attributes. For example:

Json json

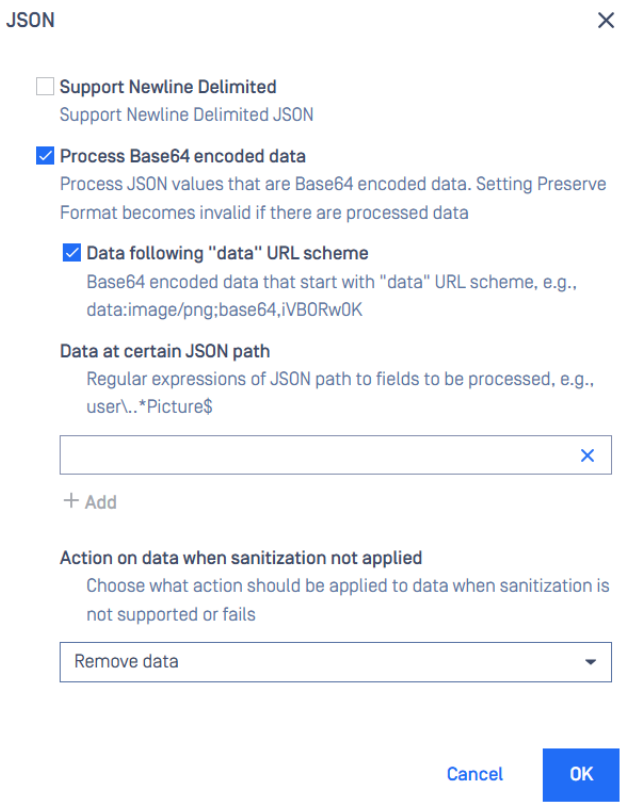
```
{
  "user": {
    "id": "12345",
    "name": "John Doe",
    "profilePicture": "...",
    "photos": [
      {
        "id": "photo1",
        "description": "A scenic view",
        "image": "..."
      },
      {
        "id": "photo2",
        "description": "A portrait",
        "image": "..."
      }
    ]
  },
  "metadata": {
    "timestamp": "2024-12-24T12:00:00Z",
    "source": "Generated Example"
  },
  "data": {
    "image1": "R0lGODlhAQABAIAAAISwxwAAACwAAAAAAQABAAACAkQBADs=",
    "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.

json

```
{
  "data": {
    "image1": "R0lGODlhAQABAIAAAISwxwAAACwAAAAAAQABAAACAkQBADs=",
    "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.

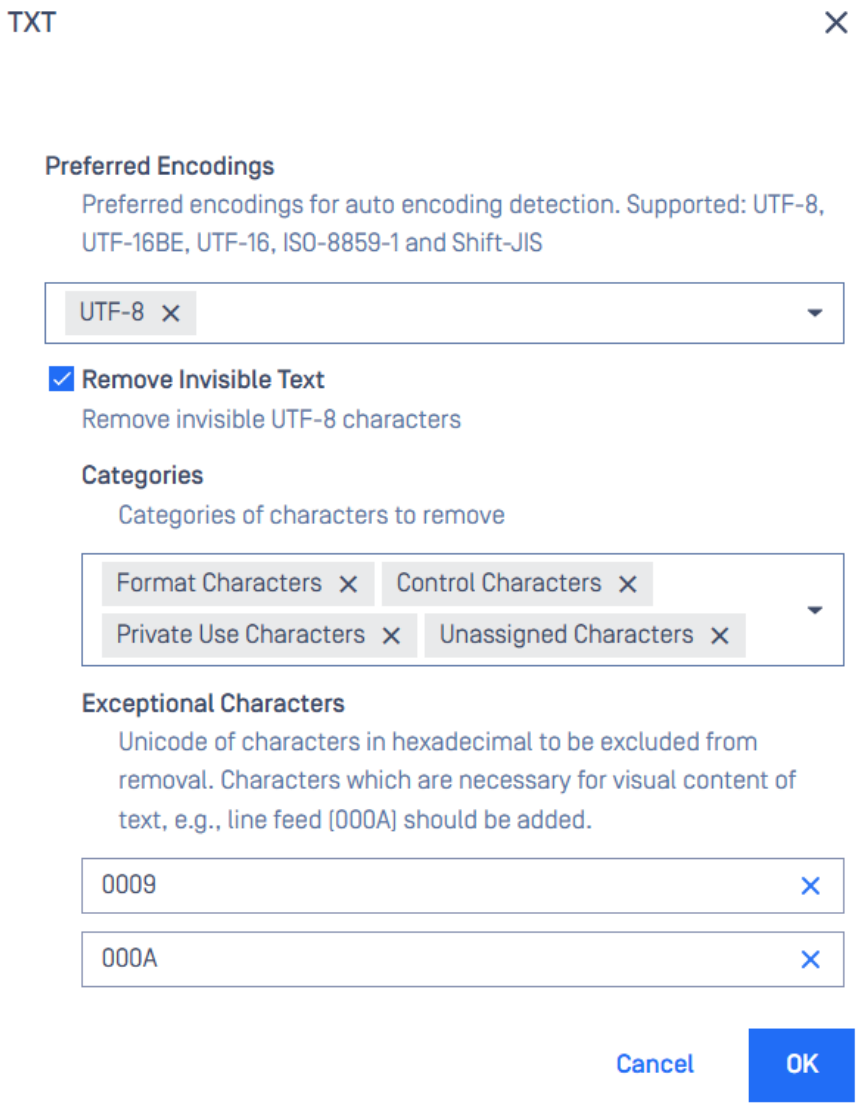
Remove invisible characters in TXT

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**.



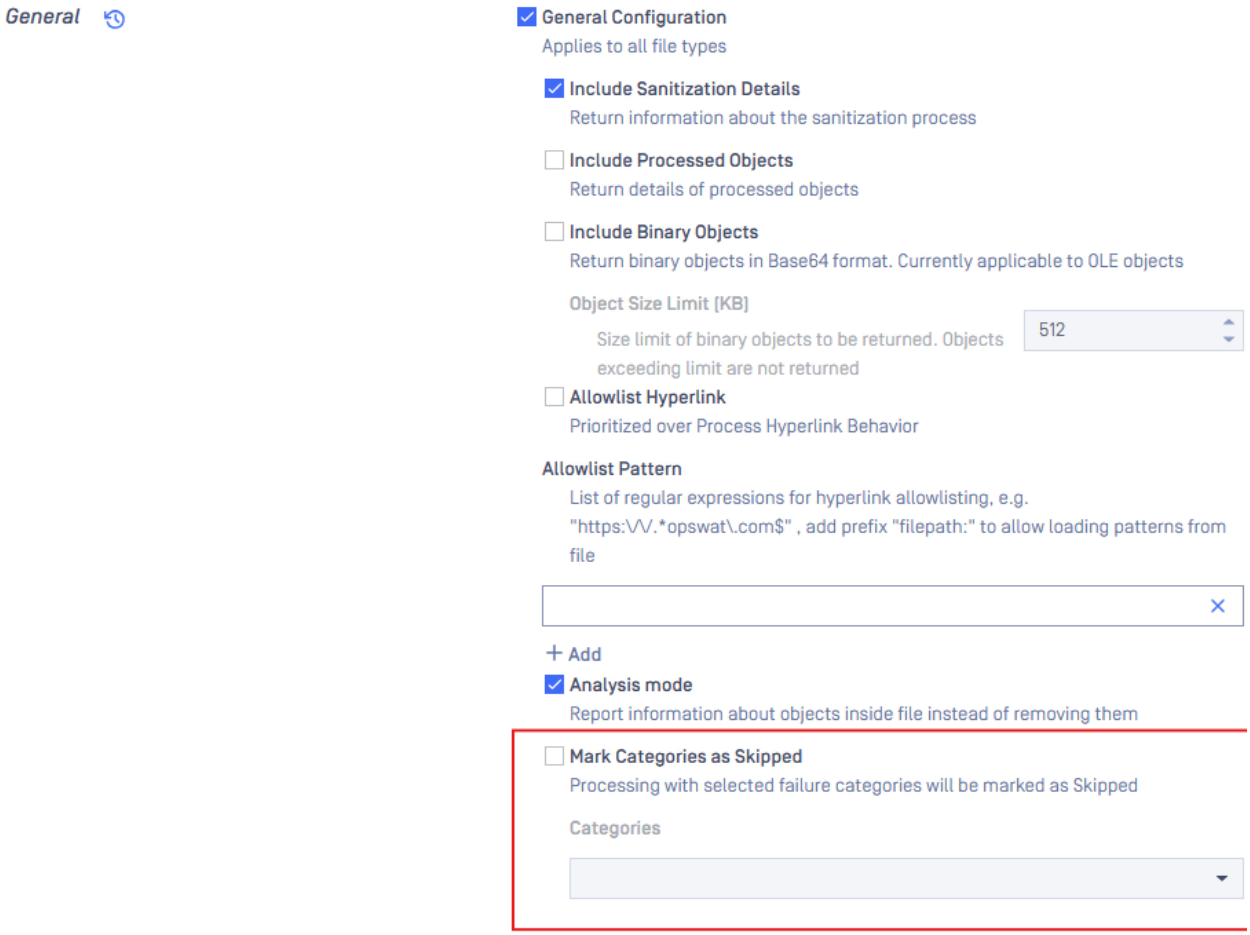
Mark Sanitization Result as Skipped

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:

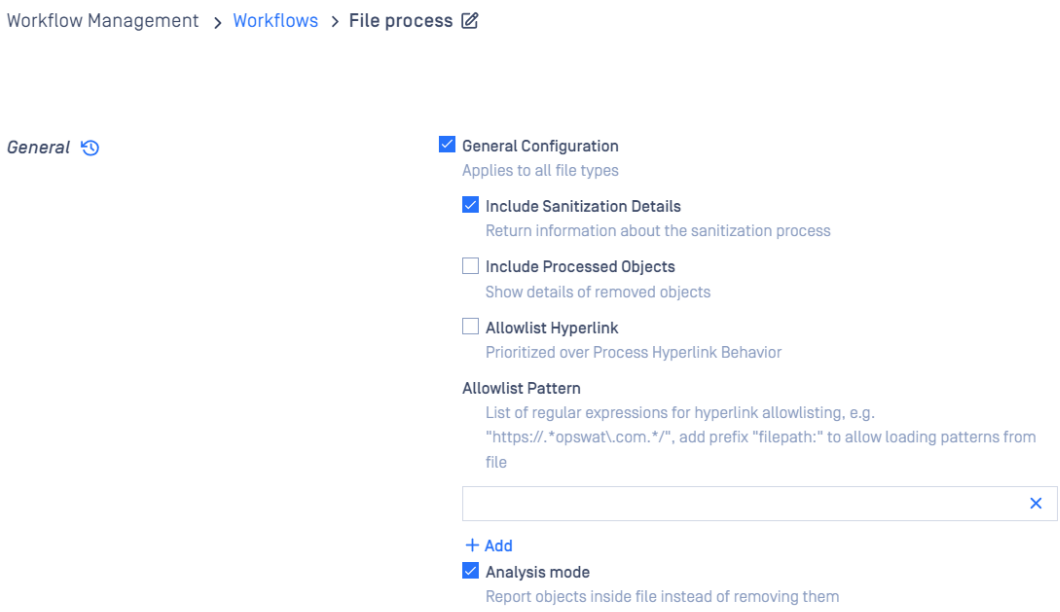
- Unsupported Version.
- Exceeded File Size.
- Password Protected.
- Invalid File Structure.



Analysis mode

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

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

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

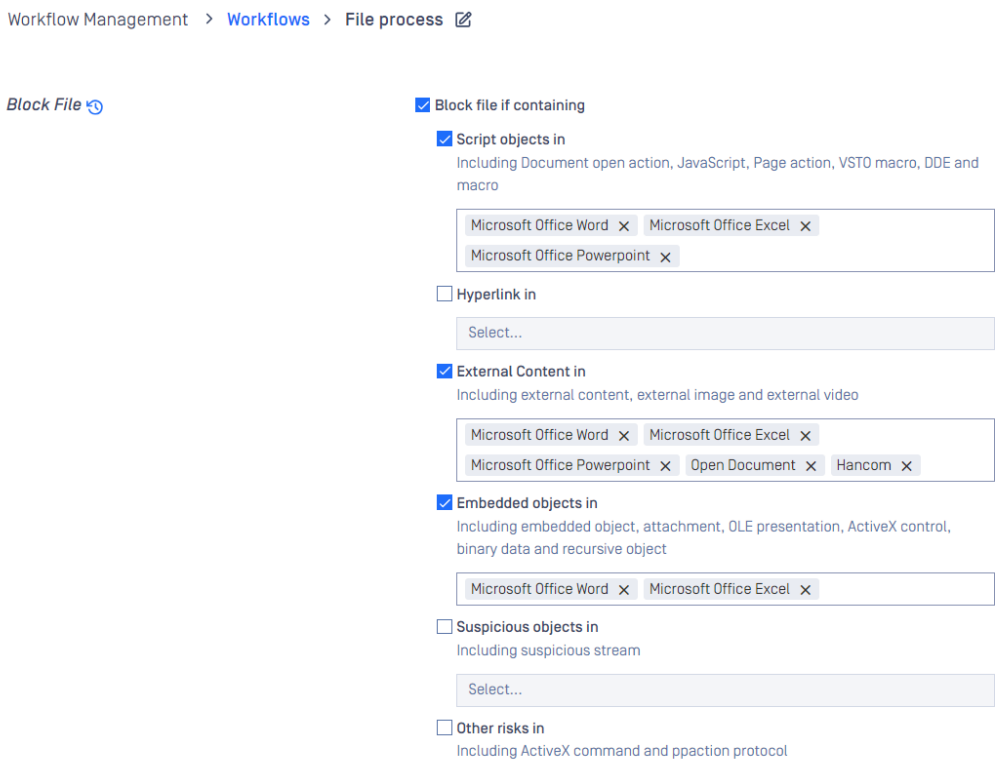
Performance

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
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

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
		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

Object category	Included Object	Applicable file families
Embedded Object	Embedded object	PDF
		Markup
		Visio
		RTF
		Ichitaro
		Hancom
		Open Document
		Microsoft Office Word
		Microsoft Office Excel
		Microsoft Office Powerpoint
	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

Object category	Included Object	Applicable file families
	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
		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

Object category	Included Object	Applicable file families
	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 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

Object category	Included Object	Applicable file families
	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
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
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

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
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
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

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
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
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

Conversion type	Total file	Average file size (KB)	Average Time (s) Windows	Average Time (s) Linux
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
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.

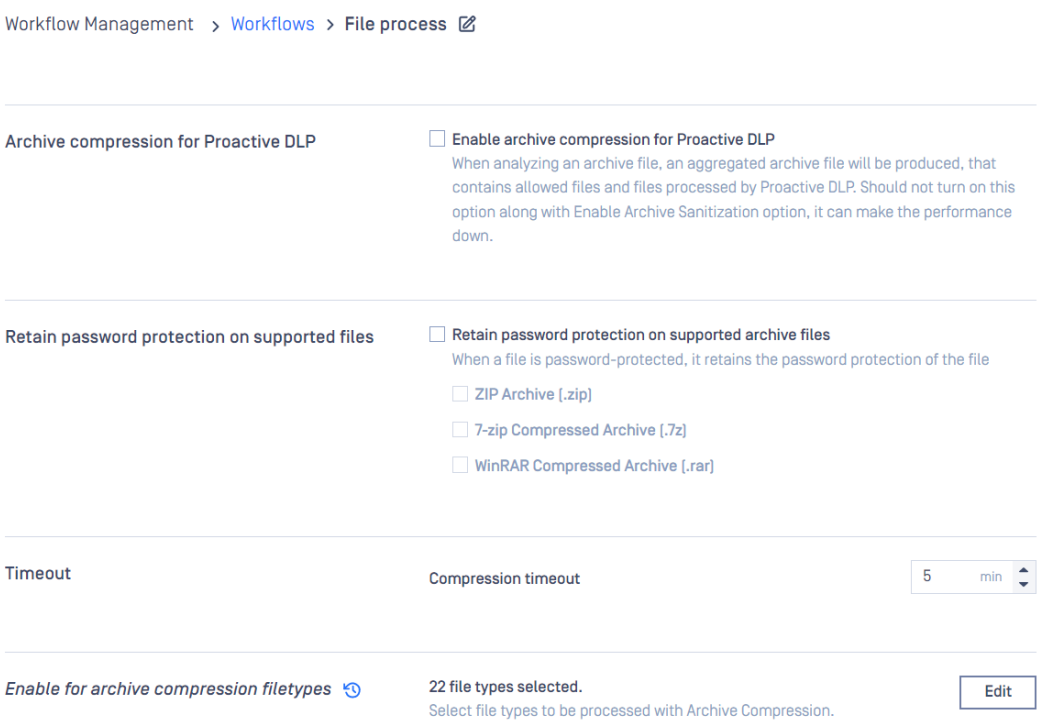
Usually this means that a sanitized archive contains:

- All allowed child files.
  - Or their sanitized versions (if available).
- Sanitized versions of blocked child files (if blocked child files are sanitized successfully).

Operation

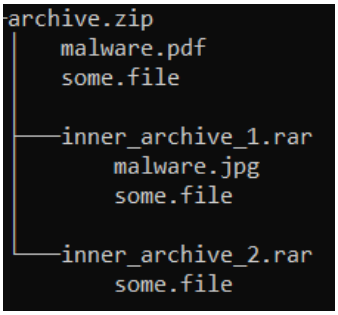
If archive sanitization is enabled for a type, then all archives within the original with the same type get sanitized as well. If there are other archives inside the original for which the sanitization is disabled and their result is not allowed then those archives won't get into the sanitized archive.

To enable the archive sanitization: **Workflow Management > Workflows > Workflow name > Compression**

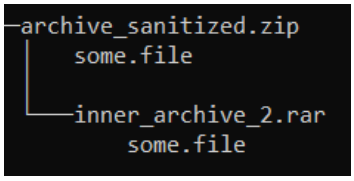


Use cases

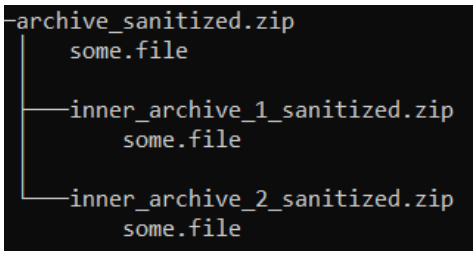
Given the following file as example:



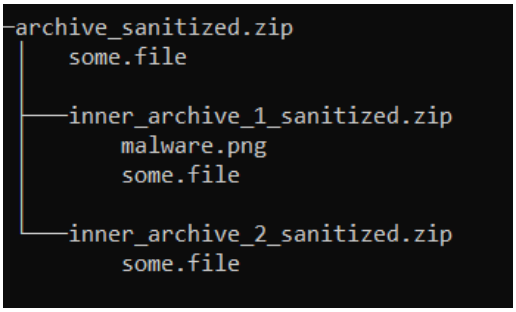
Zip to zip is enabled



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`

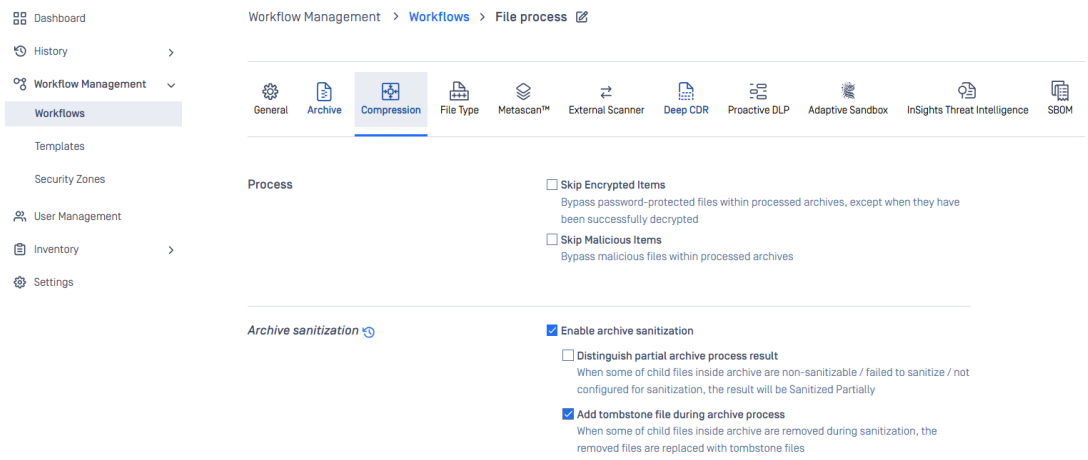
json

```
{
    "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

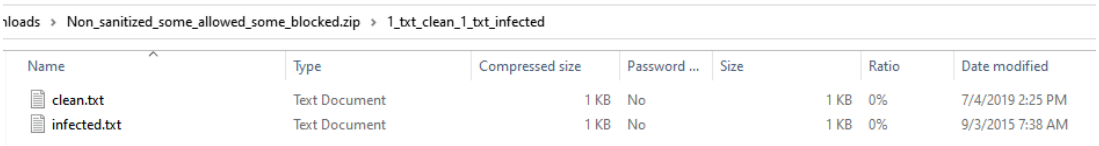
Please note that the tombstone functionality for the PST file type is supported starting from Core version 5.13.0, in combination with Archive Compression engine version 7.3.0.

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:

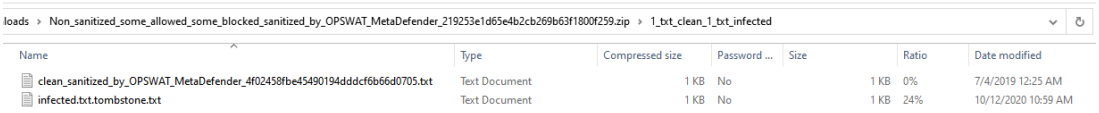


For example:

Original archive file content:



Downloading archived sanitization file and check its content:



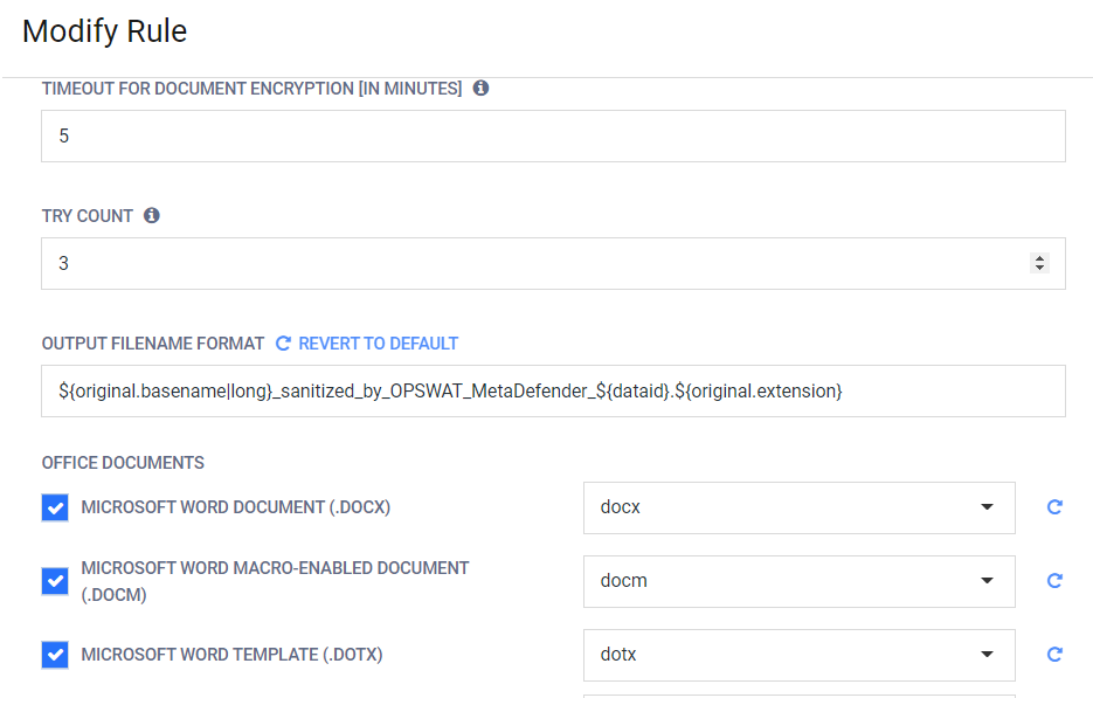
whereas a tombstone file called `infected.txt.tombstone.txt`

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:

none

```
<>:"\|/?*$
```

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

none

```
${original.basename|long}_sanitized_${dataid}.${converted.extension}
```

Example output:

none

```
testfile_sanitized_db3761f43e4545ab886f5930dbb037f3.pdf
```

Variables

Data ID

none

```
${dataid}
```

The result is the dataid of the sanitized file.

Original filename

none

```
${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

none

```
${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

none

```
${converted.extension}
```

The result is the selected target extension used for sanitization.

Date and time

none

```
${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.
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)
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

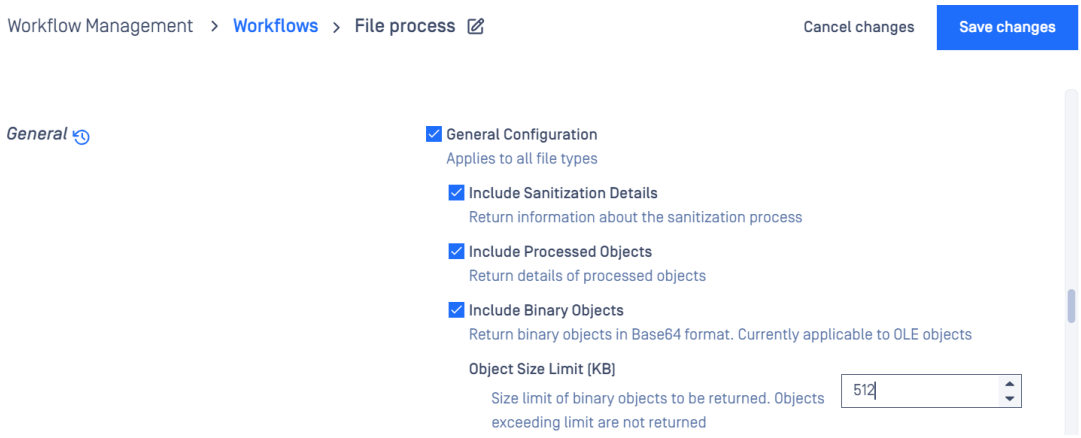
```
...
"post_processing": {
        "actions_failed": "",
        "actions_ran": "Sanitized",
        "converted_destination": "",
        "converted_to": "",
        "copy_move_destination": "",
        "sanitization_details": {
                "description": "Sanitized successfully.",
                "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".



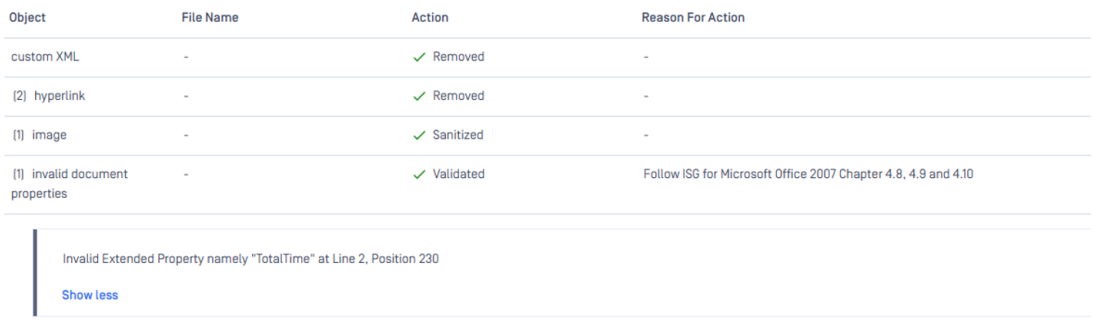
The length of object details is limited to 5000 characters. If it exceeds the limit, it will be truncated.

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.
Sanitization Skipped	File is skipped because of the configurations.
Sanitization Timeout	Santization process is timeout

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.



Object	File Name	Action	Reason For Action
custom XML	-	✓ Removed	-
(2) hyperlink	-	✓ Removed	-
(1) image	-	✓ Sanitized	-
(1) invalid document properties	-	✓ Validated	Follow ISG for Microsoft Office 2007 Chapter 4.8, 4.9 and 4.10

> Invalid Extended Property namely "TotalTime" at Line 2, Position 230
>
> Show less

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. 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

json

```
{
  "deep_cdr": {
    "charset": "ISO-2022-JP"
  }
}
```

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.

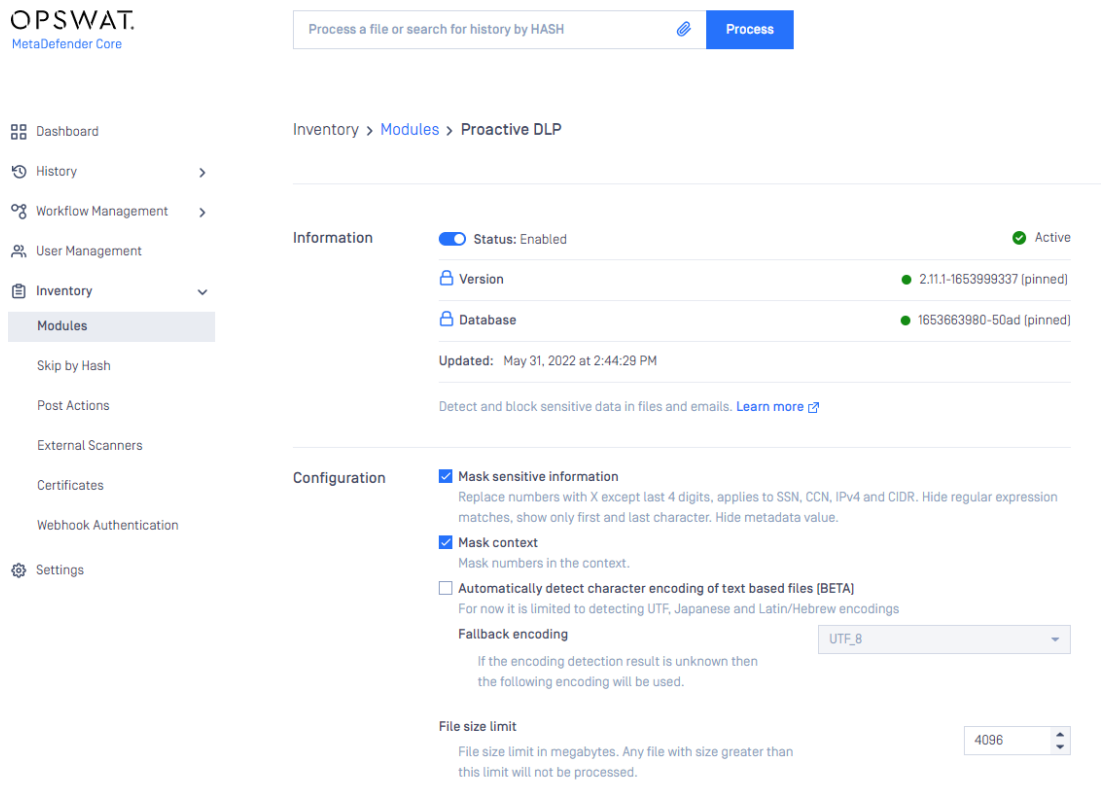
Detect and then block approach

- 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 6 dependencies
  - CPU must support AVX2 and SSE4.1 instruction set to use OCR feature
- Linux version:
  - libgdiplus, mscorefonts
  - .NET 6 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
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

	File Type	Description	Supported Features
18	.fdf	Adobe Forms Data Format	Redact | Scan | Secret detection
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
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

	File Type	Description	Supported Features
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 Presentation	Scan
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

	File Type	Description	Supported Features
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 | 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 metadata | Scan OCR
71	.prm	Parameter File	Scan | Secret detection
72	.ps1	PowerShell Script	Scan | Secret detection

	File Type	Description	Supported Features
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 Document Template	Scan
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

	File Type	Description	Supported Features
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 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

	File Type	Description	Supported Features
111	.xlsx	Microsoft Excel Workbook	Embedded image crop | 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 Spreadsheet 2003	Metadata removal | Redact | Scan | Scan metadata
118	.xps	Microsoft XML Paper Specification	Scan

	File Type	Description	Supported Features
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

- Social Security Number (SSN, US and JP)
- Credit Card Number (CCN)
- IPv4
- Classless Inter-Domain Routing (CIDR)
- Any specific data pattern using the regular expression
- **_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)

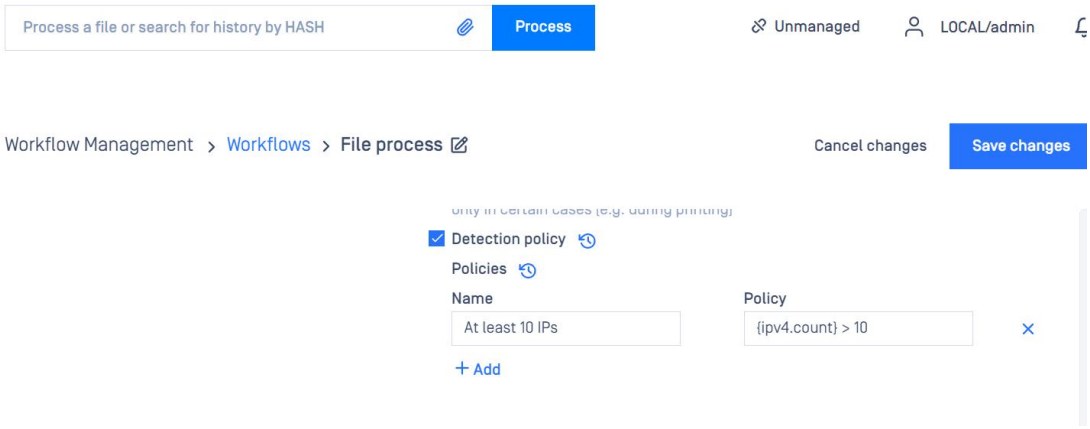
Detection Policy

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

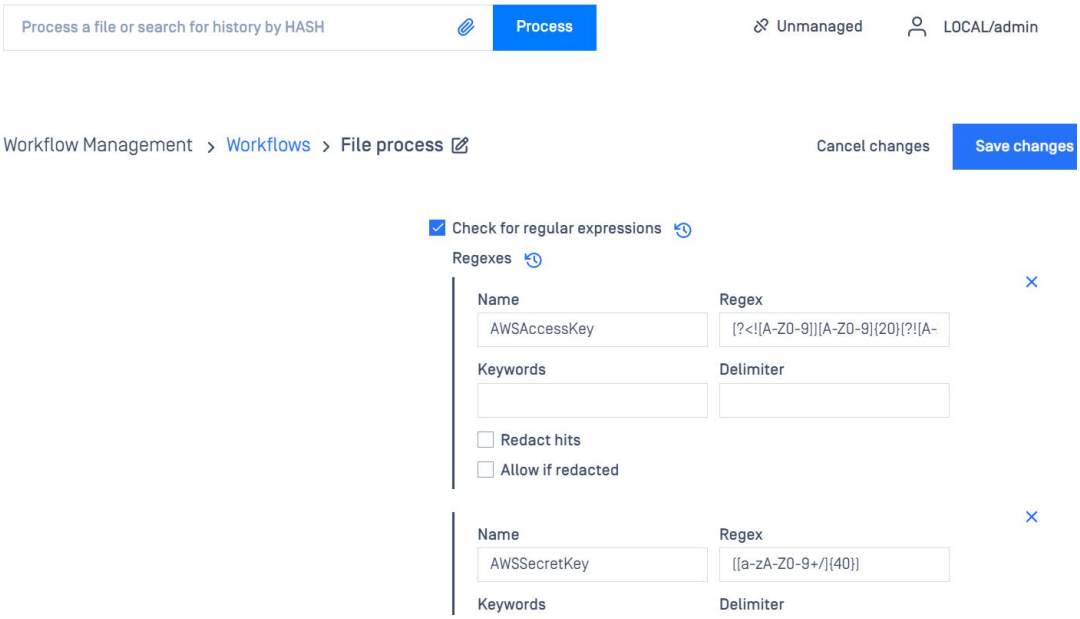


2. For Regular expression: {regex.name.atrribute} > X . The example below shows you how to block a file if it has both AWS Key ID and AWS Secret Key

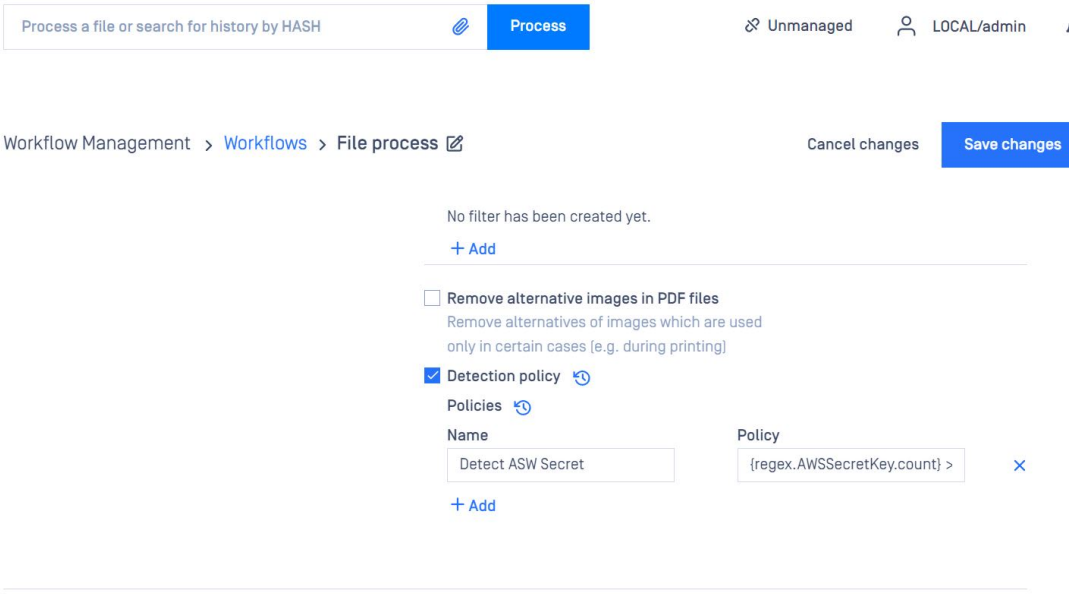
Create regular expressions to block AWS Access keys and AWS secret key

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
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

Sensitive Type Name	Sensitive Type ID
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
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

Sensitive Type Name	Sensitive Type ID
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.

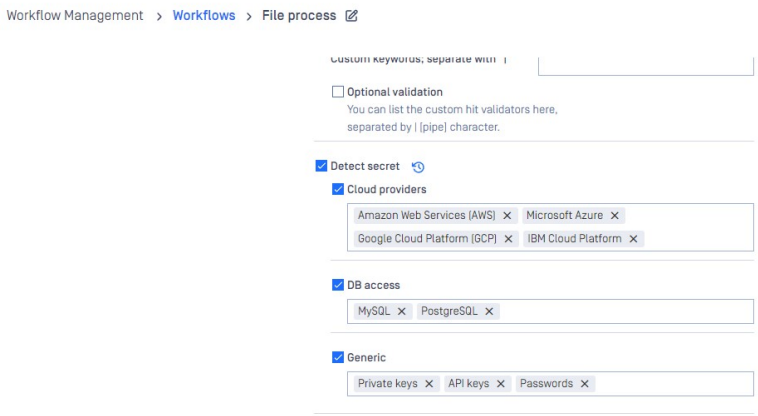
Supported Secrets

- 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'	**Amazon Web Services**
'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=https&sig=FictionalSignature%3D'	**Microsoft Azure**
'Azure Access Key' = 'Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsu Fq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw=='	
'Azure Access Token' = 'eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik1u Q19WWmNBVGZNNXBPWWlKSE1iYTlnb0VLWSIsImtpZCI6Ik1uQ19WWmNBVG ZNNXBPWWlKSE1iYTlnb0VLWSJ9.eyJhdWQiOiJodHRwczovL21hbmFnZW1lbn QuY29yZS53aW5kb3dzLm5ldC8iLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3d zLm5ldC8wMTIzNDU2Nzg5L2I4YmY0M2YtNmVlOC00ZmUxLWEzOTEtMGRiOG I1ZDM3Y2Y2LyIsImlhdCI6MTU5Mzg5OTQ2NCwibmJmIjoxNTkzODk5NDY0LCJl eHAiOjE1OTM5MDMzNjQsImFjciI6IjEiLCJhaW8iOiJBU1FBRG8vOEhBQUFBQVFB Q2dlY0FBQUY1YmVyM0l1dCIsImFtciI6WyJwd2QiXSwiYXBwaWQiOiI5OTljZDI2M C0zZjVhLTQ1NzctYmEwZC0xMjg2MmFmMjMxODgiLCJhcHBpZGFjciI6IjAiLCJmY W1pbHIfbmFtZSI6IkJyb3duIiwgImdpdmVuX25hbWUiOiJKb2huIiwiaXBhZGRyIjoiMT 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 Cloud Platform**

Example	
'Google OAuth Client ID' = '123456789012-abcdefghijklmno1pqrstuvwxyz.apps.googleusercontent.com'	
'Google OAuth Client Secret' = 'Xyz1A2B3C4d5e6F7g8H9i0'	
-----BEGIN RSA PRIVATE KEY----- MIIEowIBAAKCAQEAzC0EXAMPLEKEYCONTENTVwIDAQAB -----END RSA PRIVATE KEY-----	**Private keys** **(PEM, PPK)**
'IBM Cloud API Key' = 'fYgHijKlmn123_OPqrst456-UvWx789zABCDEF01'	**IBM keys**
' 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	**PostgreSQL credentials**
Host: db.example.com Port: 5432 Database: example _ db Username: example _ user Password: P@ssw0rd!123	**MYSQL credentials**
'Generic Password' = 'S3cureP@55w0rd!2023'	**Generic passwords**
'Generic API Key' = '1234abcd5678efgh9012ijkl3456mnop'	**Generic API tokens**

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

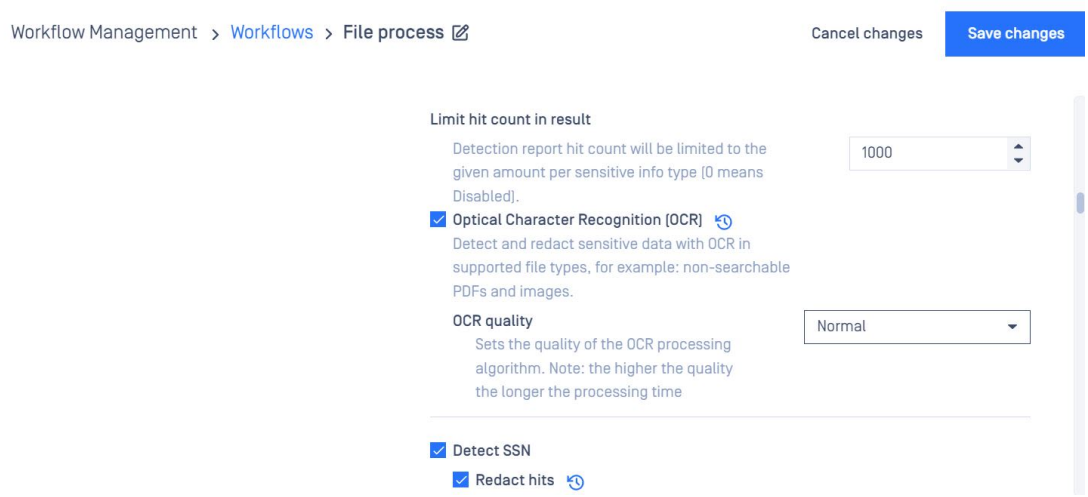
- Portable Document Format: PDF
- Microsoft Office: doc, docx, MS Word XML, xls, xlsx, ppt, pptx, rtf,
- Standalone image: jpg, png, tiff, bmp, .jp2, .jpg2, .jpf, .jpx, .mj2, .mjp2, .jpm, .jpgm

Supported languages

- English

Enabling OCR

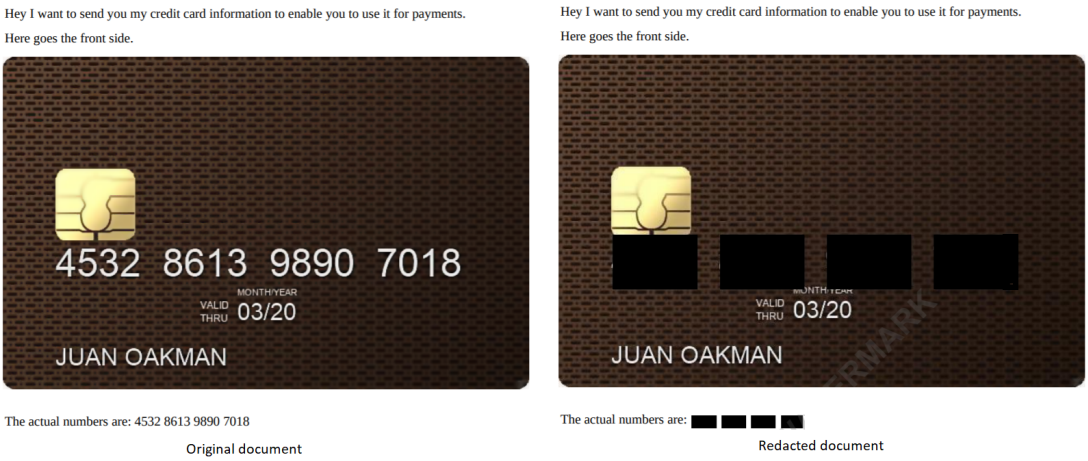
Policies > Workflow rules > "Workflow name" > Proactive DLP > Optical character recognition (OCR)



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

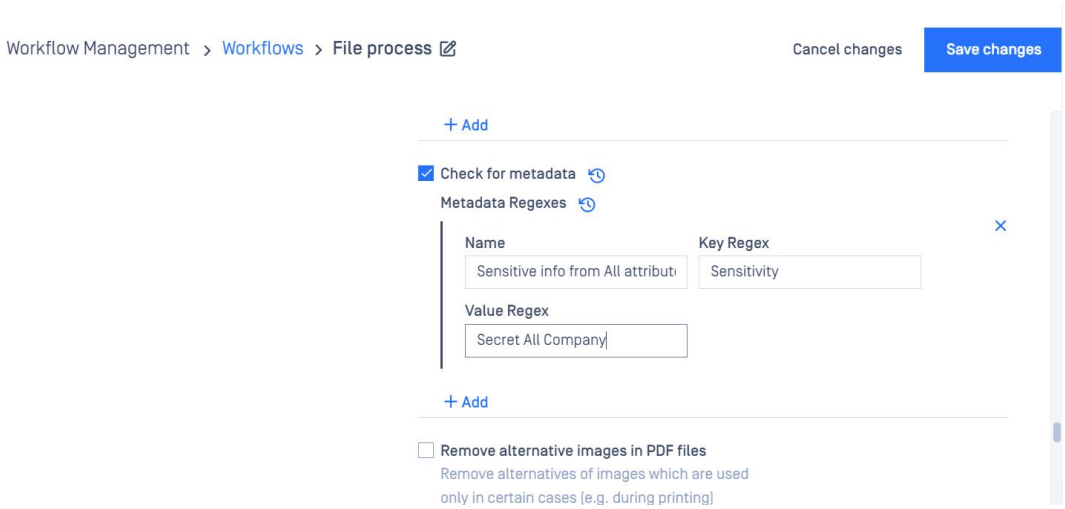
> This feature uses AI functionality for Optical Character Recognition (OCR) to extract text from images and scanned documents, converting it into machine-readable text. You have the option to enable or disable the AI-powered feature within the configuration settings. By default, this feature is disabled 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

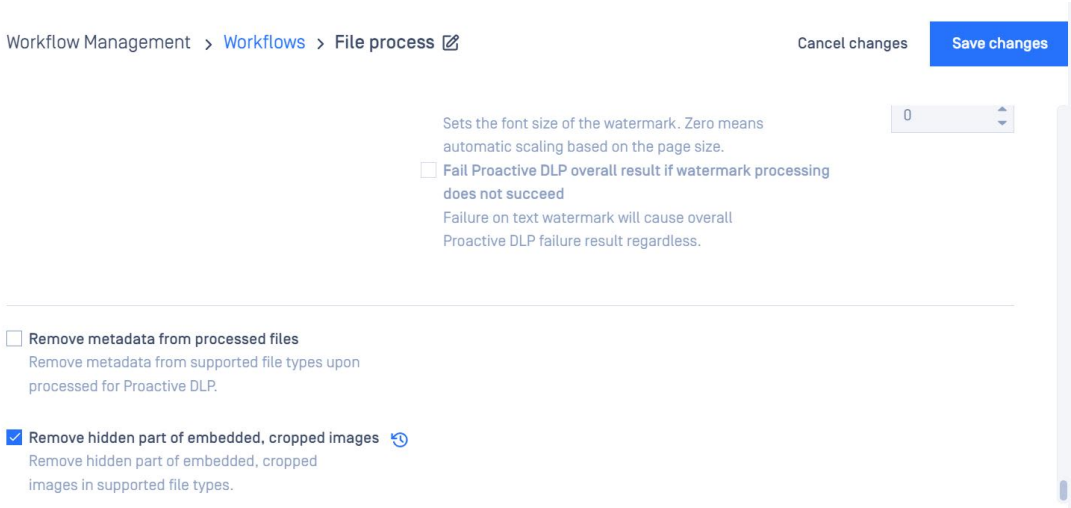
Process hidden parts in a cropped image

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)

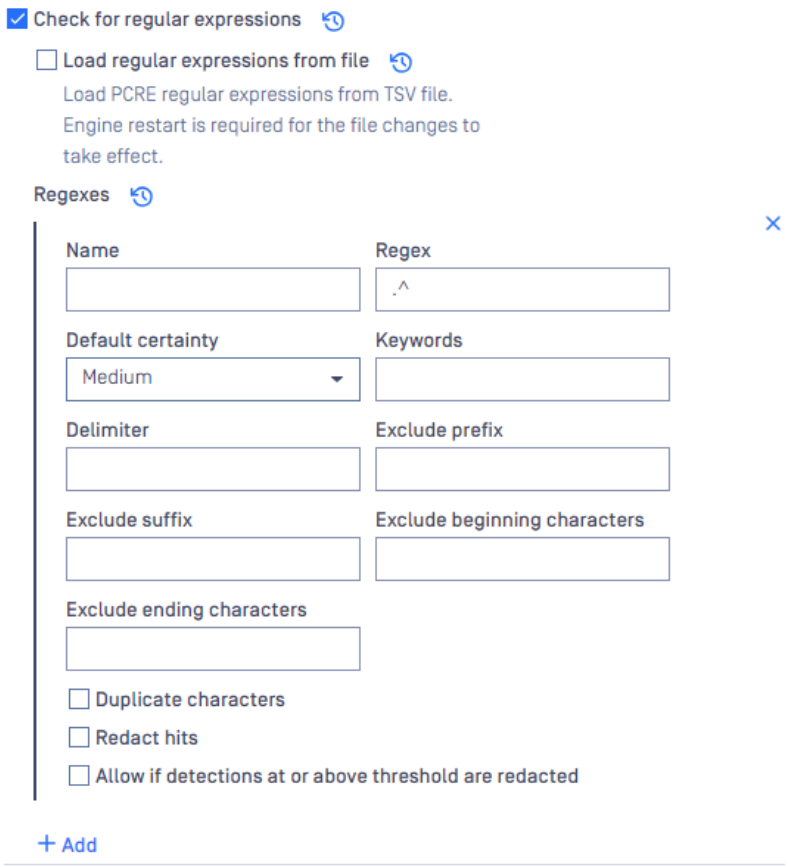
The feature is available from MetaDefender Core 4.19.2 or newer

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`
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`

Name	Description	Regular Expression
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). For example: • Date of birth: 1900/12/01 • Date of birth: 2019.01.25 • Birthday: 2020-10-30	`(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
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

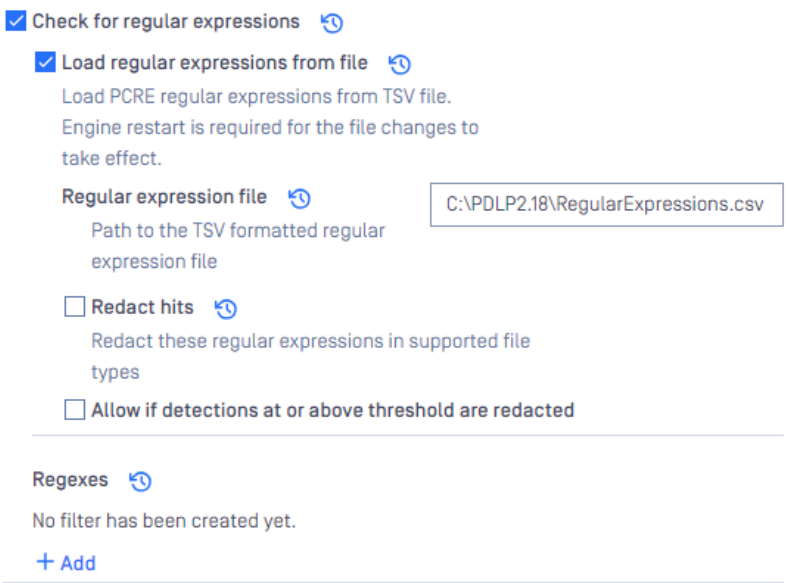
In this section, we will guide you through the process of importing regular expressions using a TSV (Tab-Separated Values) file and integrating them into the PDLP. We'll cover the basics of preparing your TSV file for regular expressions and the steps to import them effectively into the PDLP.

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

none

```
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
```

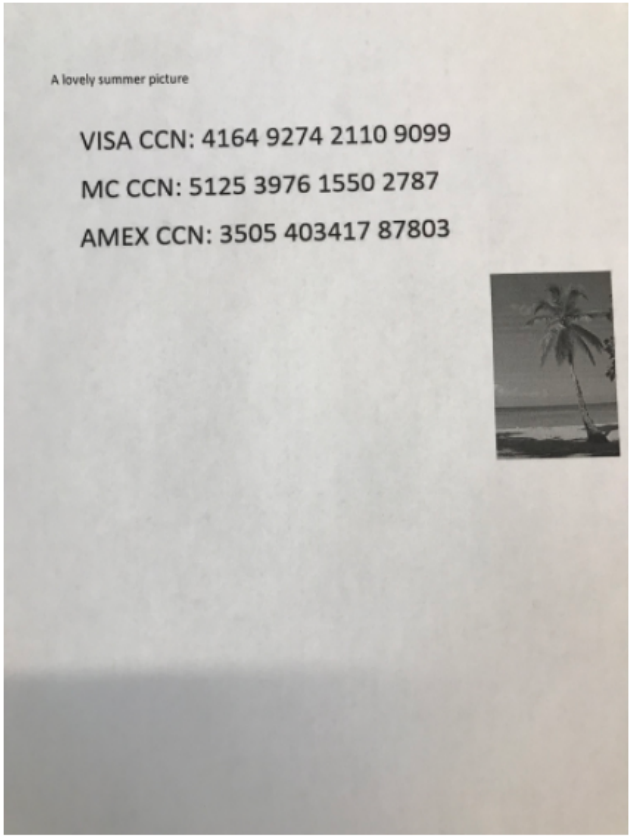
Remove alternative images in PDF

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

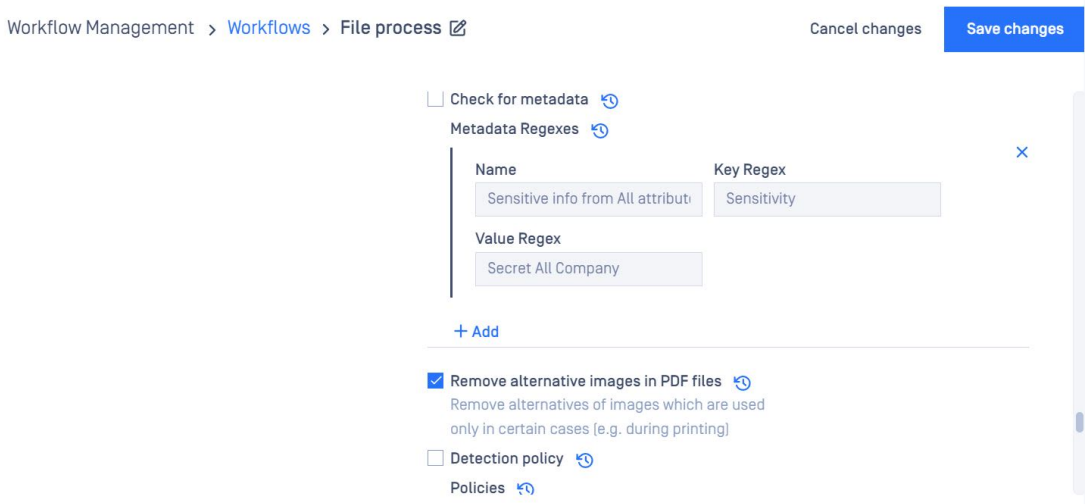


PDF reader



Print version

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
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 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

	File Type	Metadata
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 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** and **substitution**:

- **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.

Supported File Types

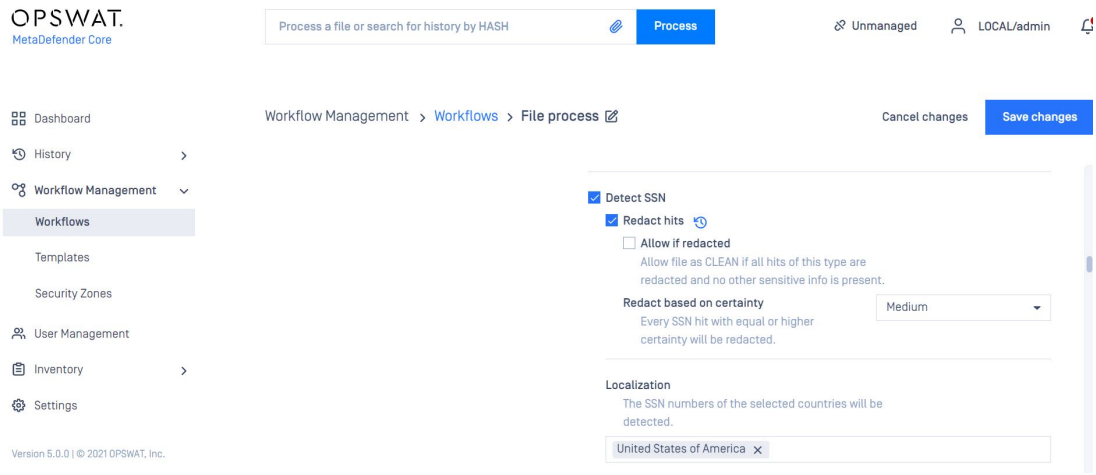
- Portable Document Format (PDF)
- Microsoft Office Word (DOC/DOCX)
- Microsoft Office Word XML (XML)
- Microsoft Office Excel (XLS/XLSX)
- Microsoft Office Excel XML (XML)
- Microsoft Office PowerPoint (PPT/PPTX)
- Microsoft Rich Text Format (RTF)
- Extensible Markup Language (XML)
- Comma Separated 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

- Social Security Number (SSN - U.S. and JP)
- Credit Card Number (CCN)
- IPv4
- Classless Inter-Domain Routing (CIDR)
- 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]

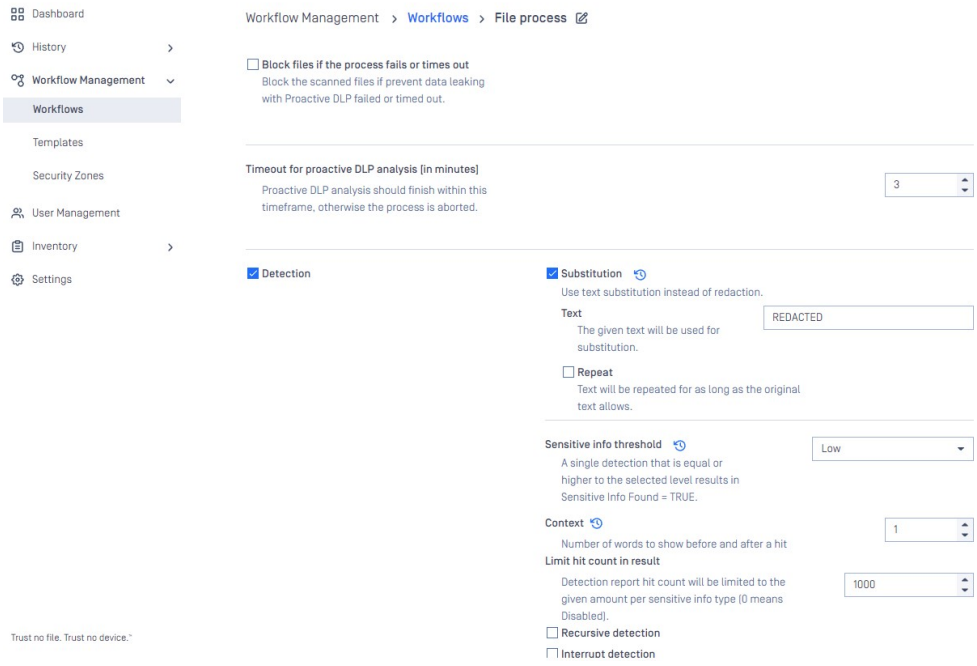
Expiration Month: 5 Expiration Year: 2020

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

Watermark files

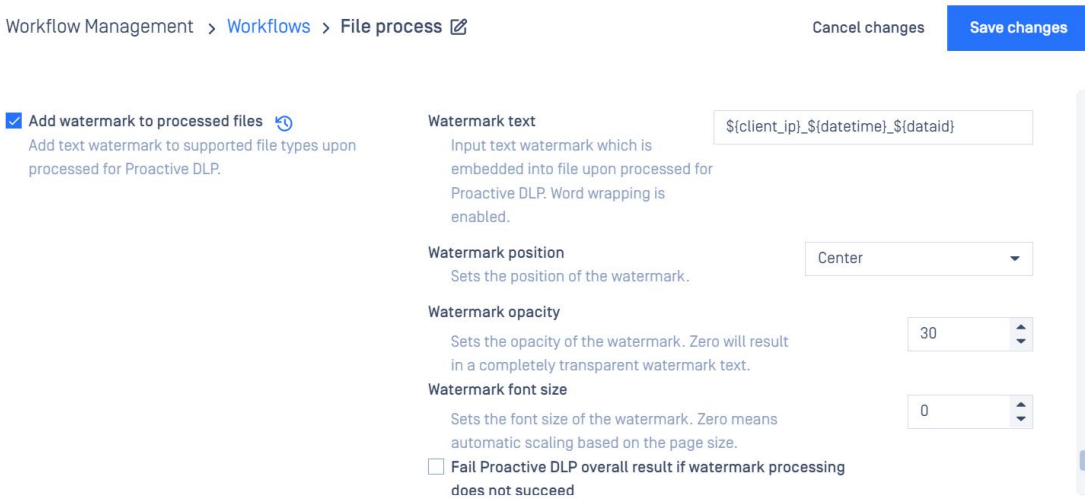
Nowadays, data leak becomes more and more popular, once the data went outside the organization, there is no way to track who caused the leak. Adding watermark to sensitive data is one of the solutions.

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}

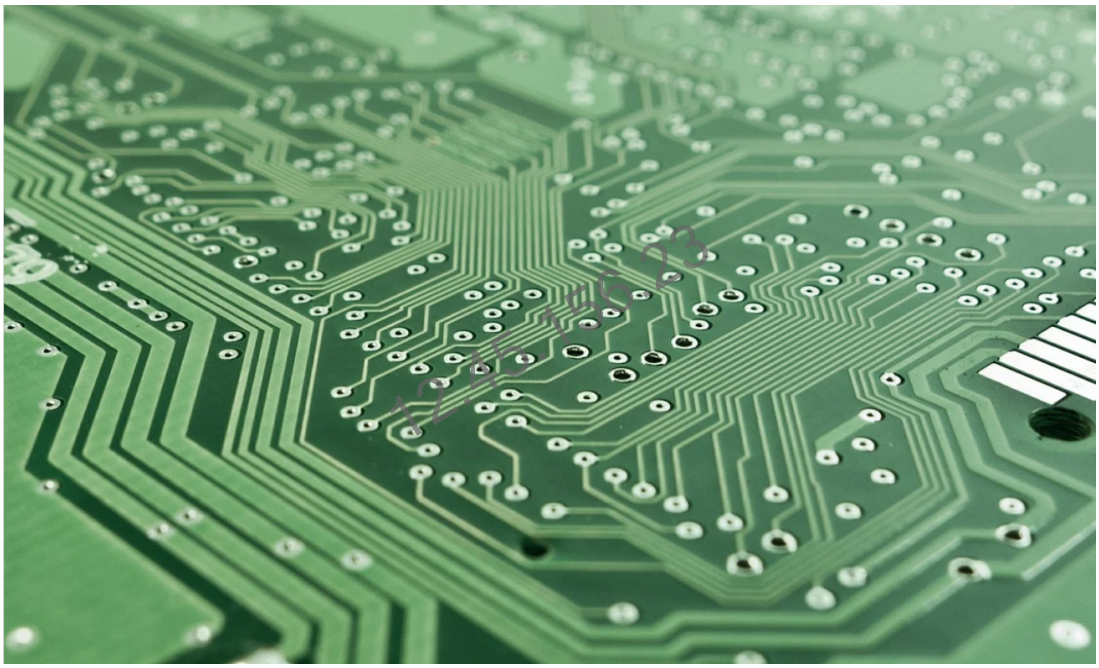
Watermark position

- 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

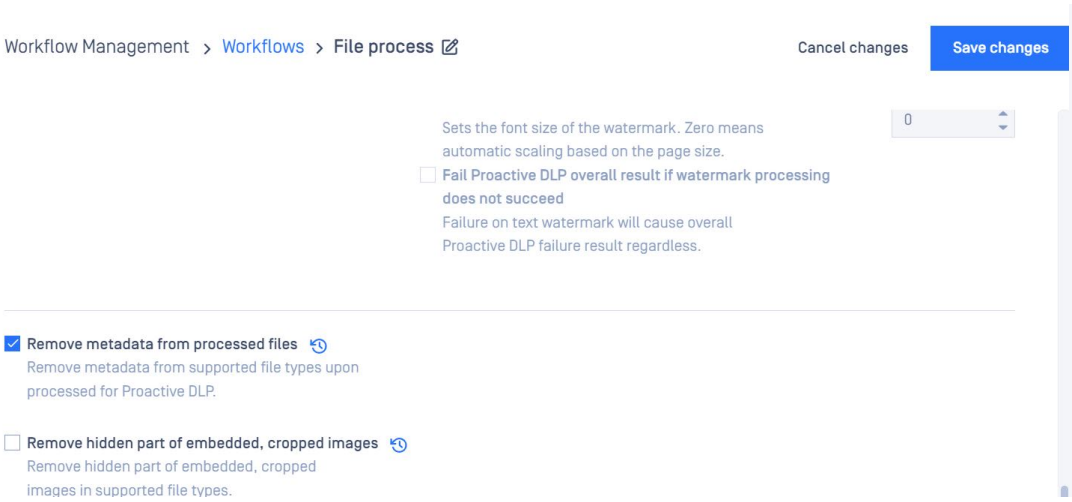
Most of the files have some kind of metadata which could contain sensitive information like the author, GPS coordination, etc. which can expose the organizations to risk. Proactive DLP can remove metadata contained within the file.

Supported File Types

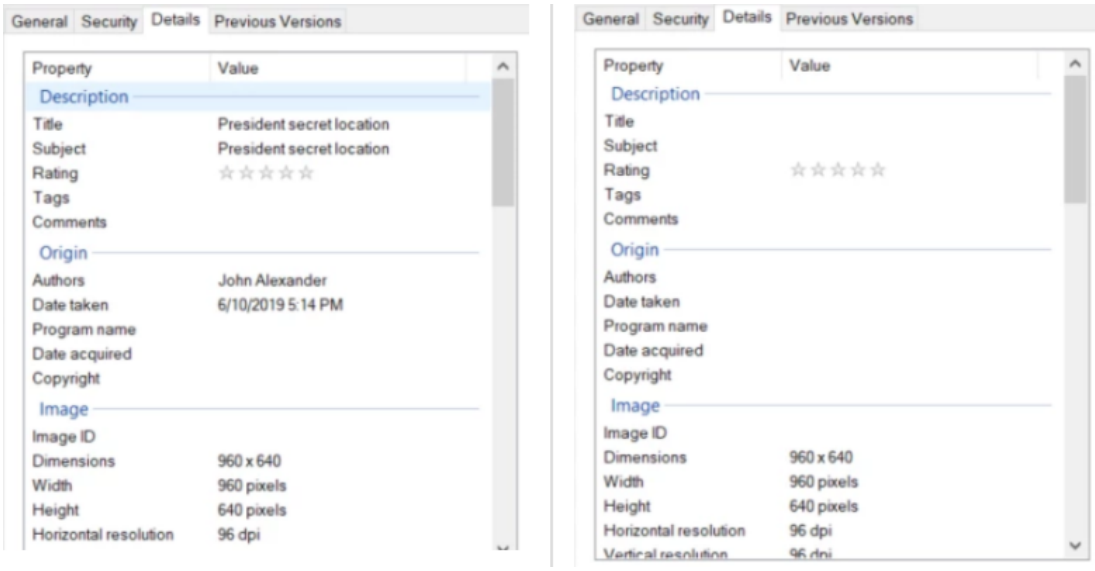
- Images: JPG, PNG, TIFF, GIF
- Documents: PDF, MS Office, RTF
- OpenOffice: ODT, ODS

To set remove metadata

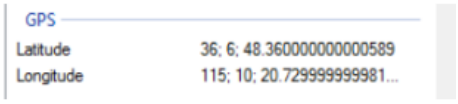
- Workflow Management > Workflows > "Workflow name" > Proactive DLP



Sample Exif info before and after processing


General
Security
Details
Previous Versions
Property
Value
Description
Title
President secret location
Subject
President secret location
Rating
Tags
Comments
Origin
Authors
John Alexander
Date taken
6/10/2019 5:14 PM
Program name
Date acquired
Copyright
Image
Image ID
Dimensions
960 x 640
Width
960 pixels
Height
640 pixels
Horizontal resolution
96 dpi
General
Security
Details
Previous Versions
Property
Value
Description
Title
Subject
Rating
Tags
Comments
Origin
Authors
Date taken
Program name
Date acquired
Copyright
Image
Image ID
Dimensions
960 x 640
Width
960 pixels
Height
640 pixels
Horizontal resolution
96 dpi

GPS info

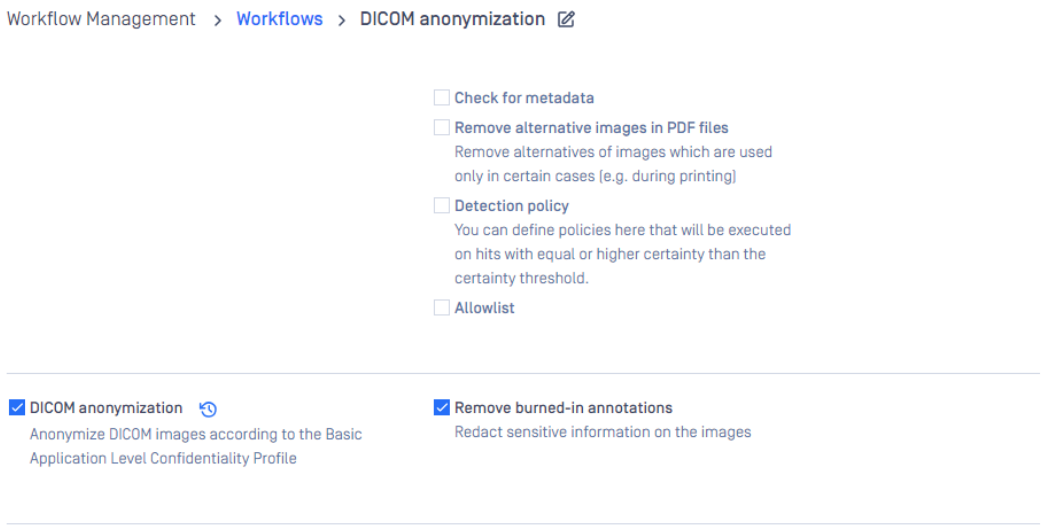

GPS
Latitude
36; 6; 48.360000000000589
Longitude
115; 10; 20.729999999981...

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

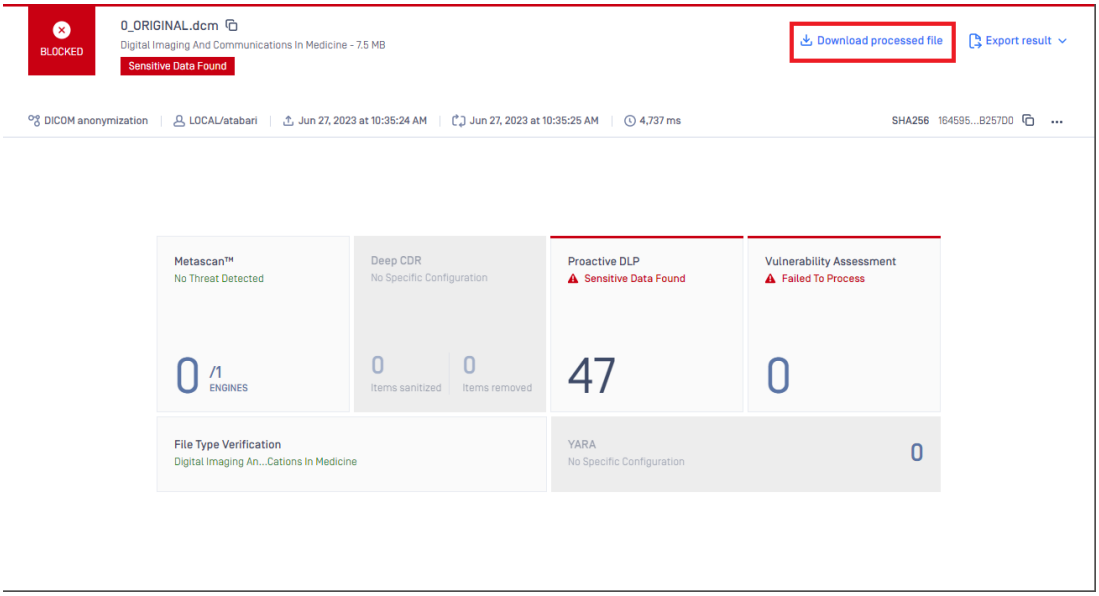
Policies > Workflow rules > Workflow name > Proactive DLP > DICOM anonymization



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.

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:

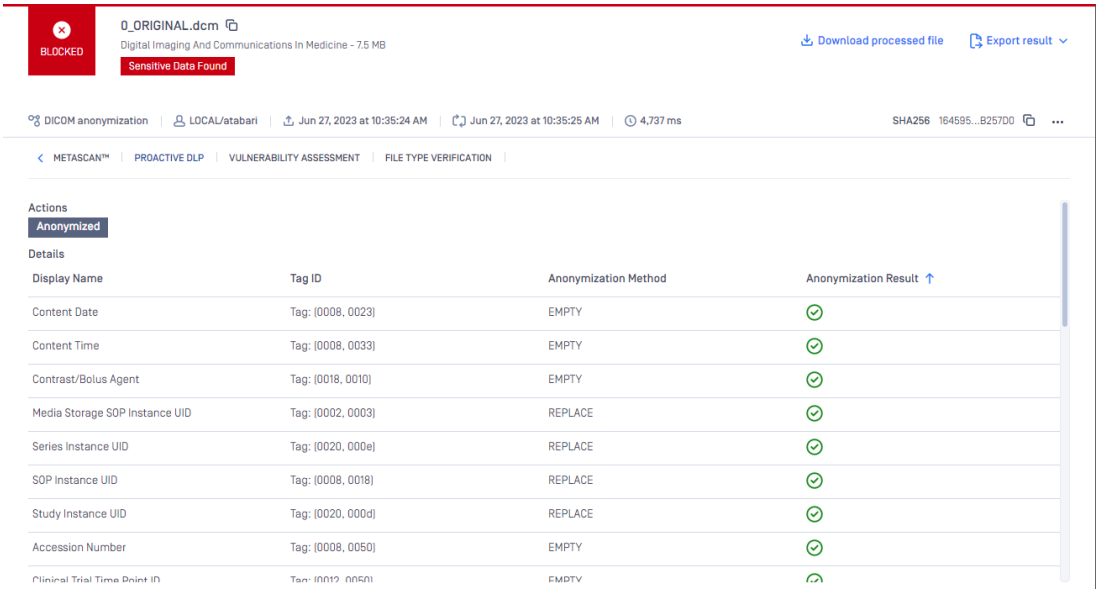
Groups	Actions
D_TAGS	Replace with a non-zero length value that may be a dummy value
Z_TAGS	Replace with a zero length value, or a non-zero length value that may be a dummy value
X_TAGS	Completely remove the tag
U_TAGS	Replace all UID's random ones. Same UID will have the same replaced value
Z___D___TAGS	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:

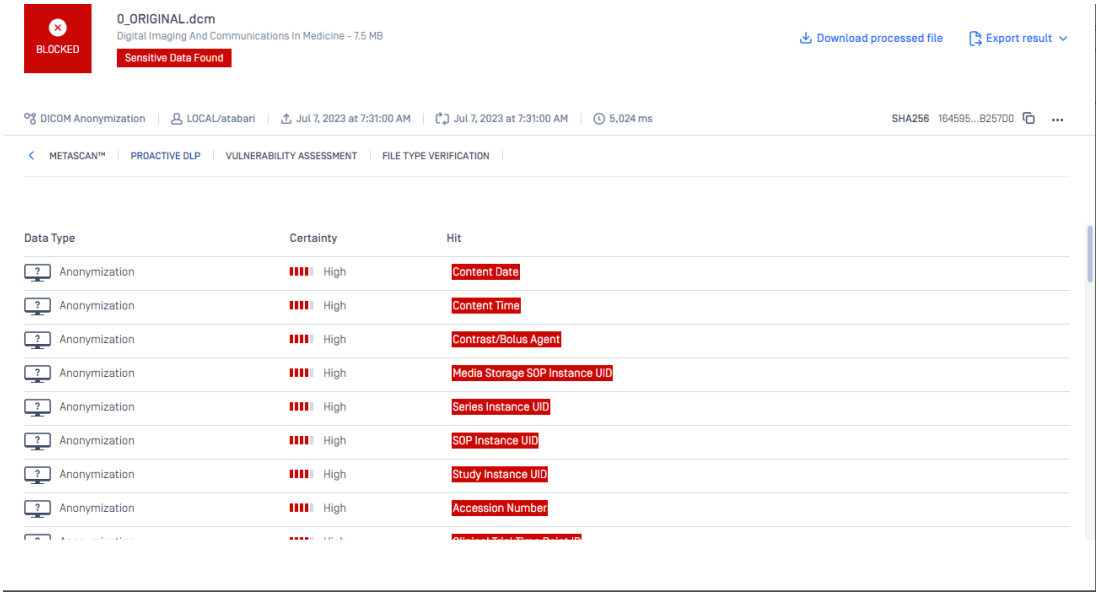
- https://dicom.nema.org/medical/dicom/current/output/chtml/part15/chapter_E.html

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

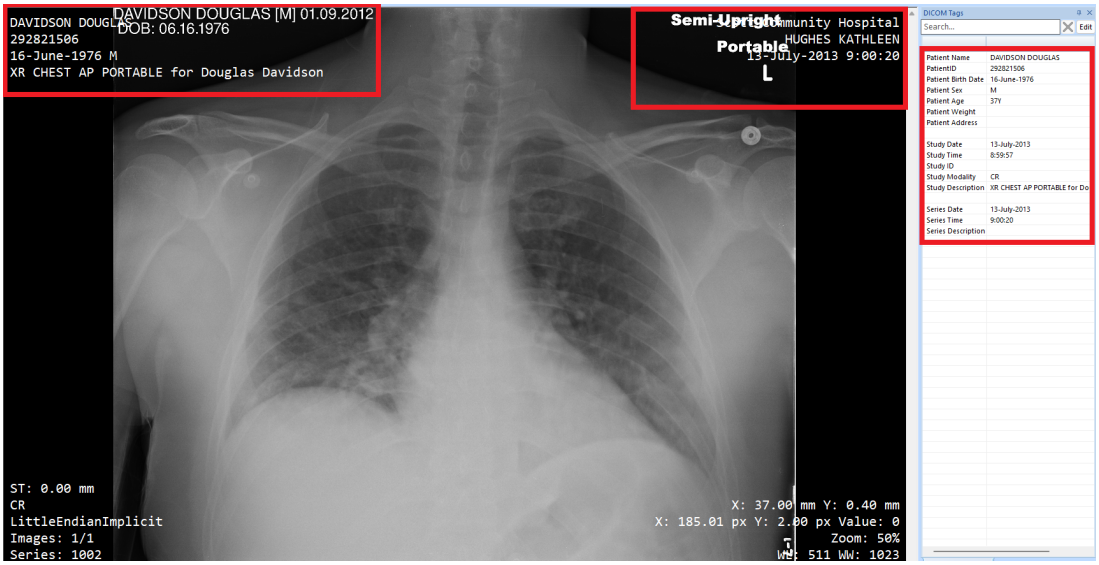


DICOM image anonymization

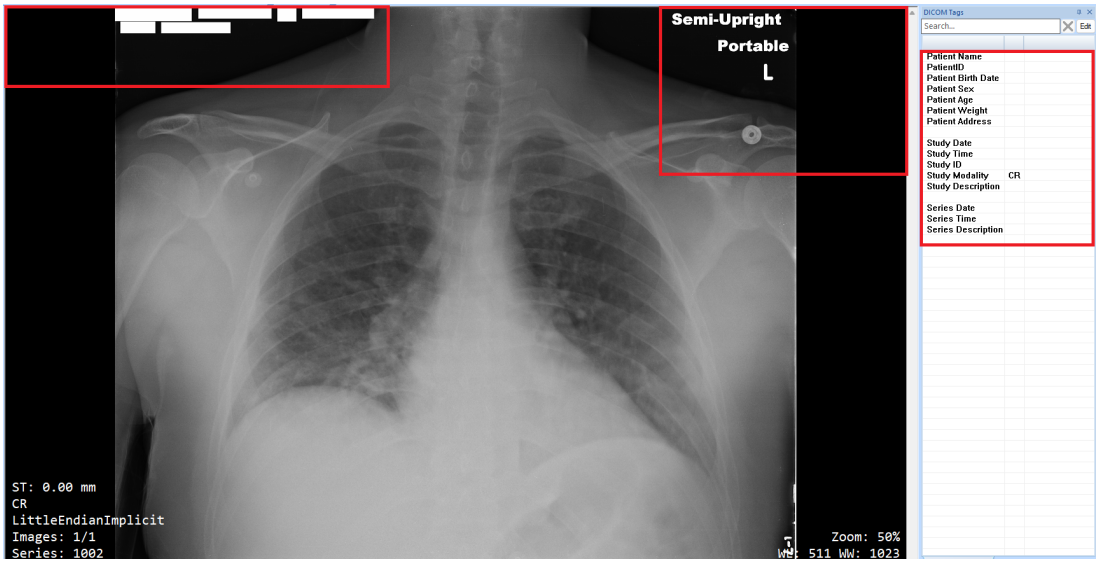
Additionally, we are also capable of redacting (removing) sensitive content from DICOM images. With the powerful AI-based identification engine that has been added to the OPSWAT PDLP, we have been able to identify and redact the PII and PHI that is burned onto the medical images.

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

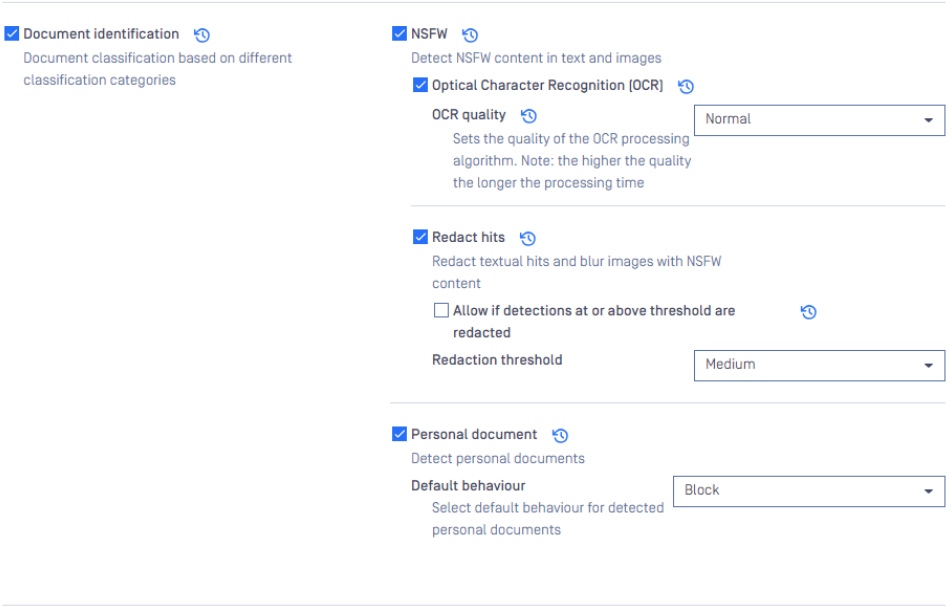
Supported NLP entities

- **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:

- **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.

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.

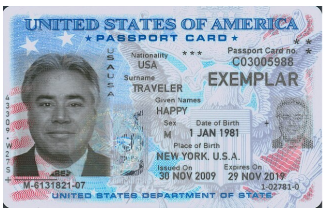
II. Personal Document Configuration

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.

Example





Personal Document Identification, the default behavior is set up to *Allow* for personal documents:
United States of America Passport Card: ***Allowed***
United States of America Dog Passport card: ***Blocked***

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.

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
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 "Filescan" and "Sandbox" are terms both referring to the same product.

What is Adaptive Sandbox?

The OPSWAT Sandbox engine 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.

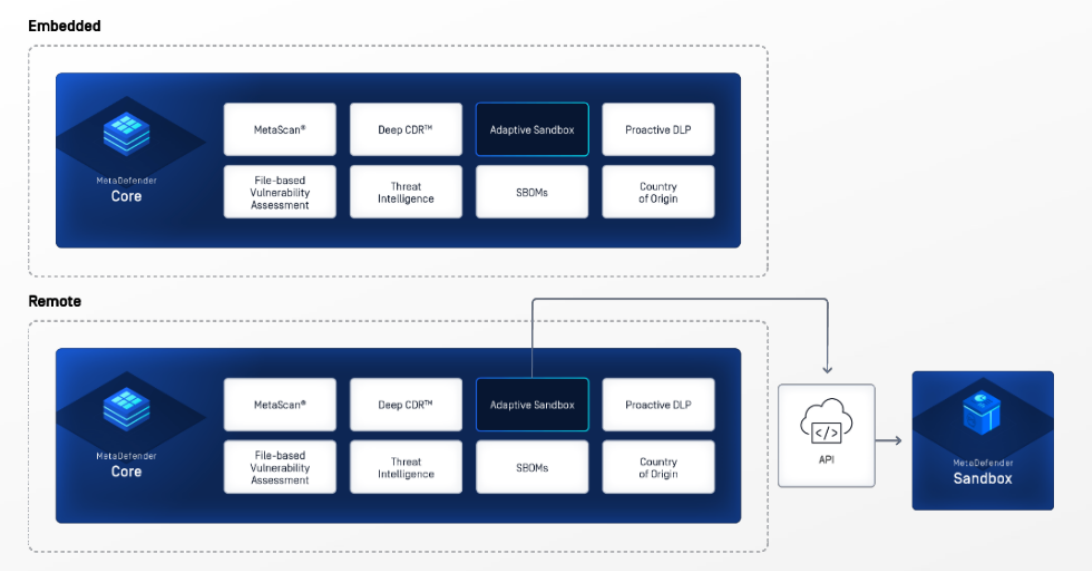
Sandbox Engines in MetaDefender Core

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.

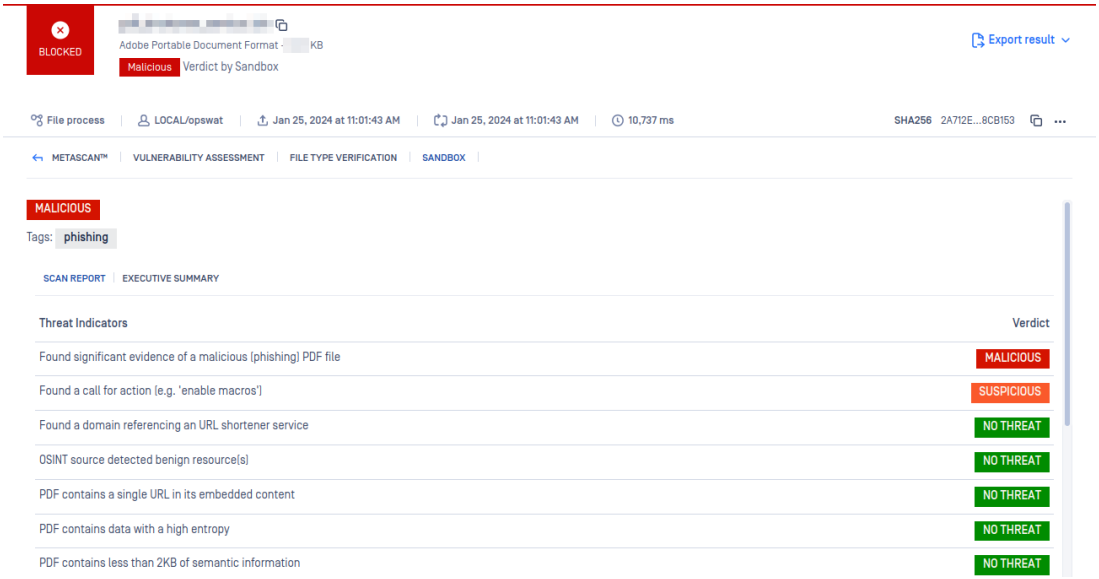
The following diagram illustrates the different engine setups:



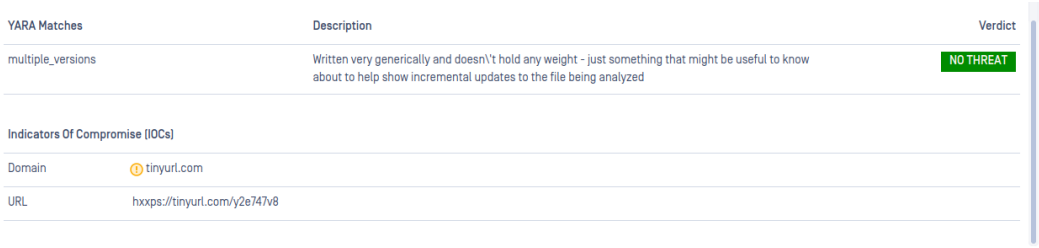
More details on the exact feature sets can be found here.

Example Sandbox Engine results

Verdict and Threat Indicators



YARA matches and detected IOCs

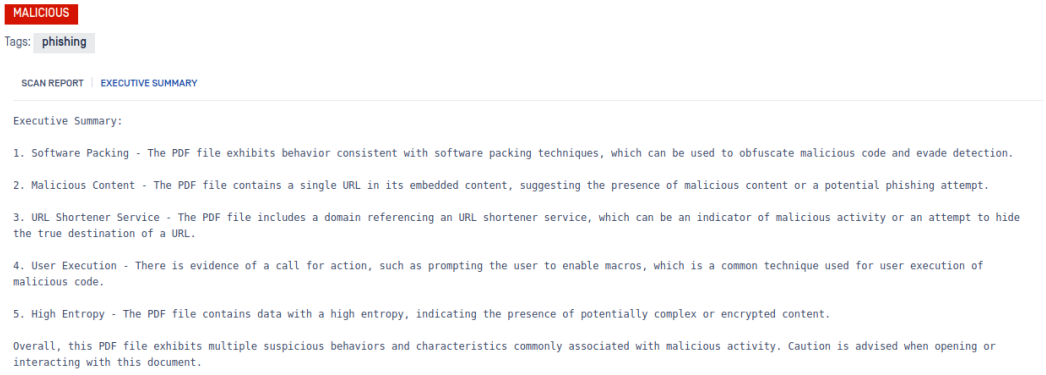


YARA Matches	Description	Verdict
multiple_versions	Written very generically and doesn\'t hold any weight - just something that might be useful to know about to help show incremental updates to the file being analyzed	NO THREAT

Indicators Of Compromise (IOCs)	
Domain	tinyurl.com
URL	hxxps://tinyurl.com/y2e747v8

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. The list of supported types is provided below:

File Type	OS	Comment
A3X	Windows	
APK	Android	
ASF	Windows	Windows Media Video
AU3	Windows	
BAT	Windows	
CPL	Windows	
DOC	Windows	
DOCM	Windows	
DOCX	Windows	
DOT	Windows	
DOTM	Windows	
DOTX	Windows	
DWG	Windows	VBA macro extraction
ELF	Unix	
EML	Windows	
HTA	Windows	
HTML	Windows	
HWP	Windows	Hangul Word Processor
Java	Multi-OS	Java Decompilation & re-analysis
JScript	Windows	
JSE	Windows	
LNK	Windows	
MBOX	Windows	
MSC	Windows	

File Type	OS	Comment
MSI	Windows	
ODS	Windows	
ODT	Windows	
OLE	Windows	
ONE	Windows	Microsoft OneNote exported section
PDF	Multi-OS	
PE (EXE/DLL)	Windows	PE unpacking & re-analysis
PE (dotnet_pe)	Windows	PE.NET decompliation & re-analysis
PE (other)	Windows	Other specific PE types (SFX, Golang, Rust, etc.)
POT	Windows	
POTM	Windows	
POTX	Windows	
Powershell	Windows	
PPAM	Windows	
PPSX	Windows	
PPT	Windows	
PPTM	Windows	
PPTX	Windows	
PUB	Windows	
RFC822	Windows	
RTF	Windows	
SCT	Windows	
SVG	Windows	
TXT	Windows	

File Type	OS	Comment
VBScript	Windows	
WSF	Windows	
XLS	Windows	
XLSM	Windows	
XLSX	Windows	
XLTM	Windows	
XLTX	Windows	

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
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 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

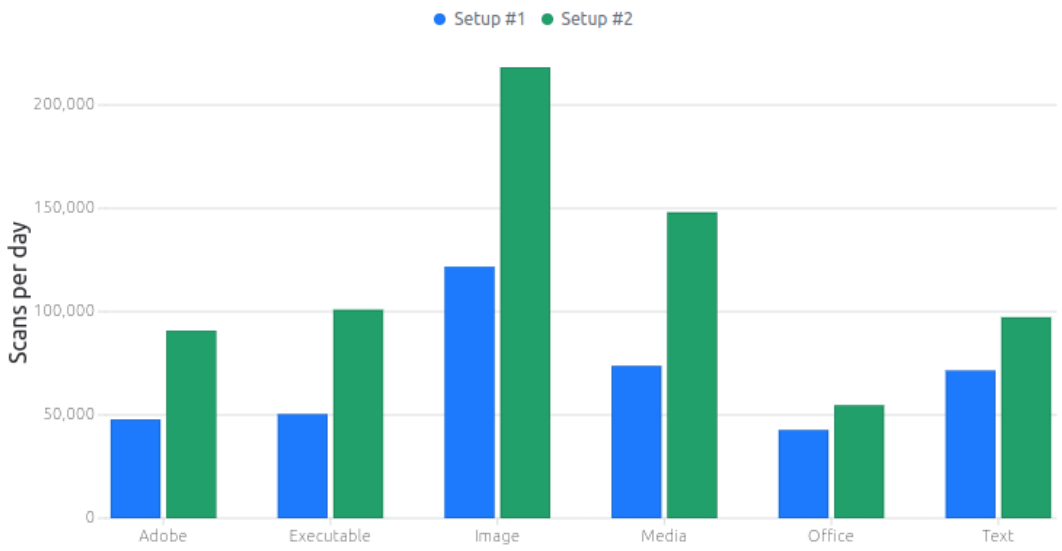
Example Configurations

	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 Sandbox Engine

For MetaDefender Sandbox (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 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
```

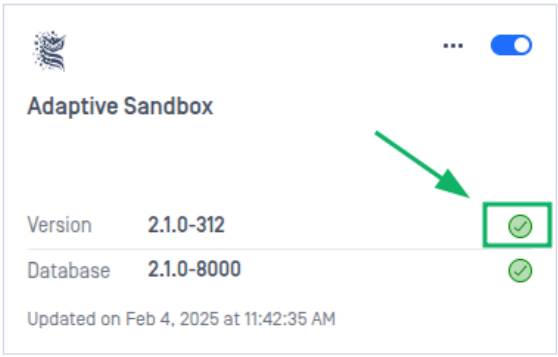
Verify the Java version

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.


Adaptive Sandbox
Version
2.1.0-312
Database
2.1.0-8000
Updated on Feb 4, 2025 at 11:42:35 AM

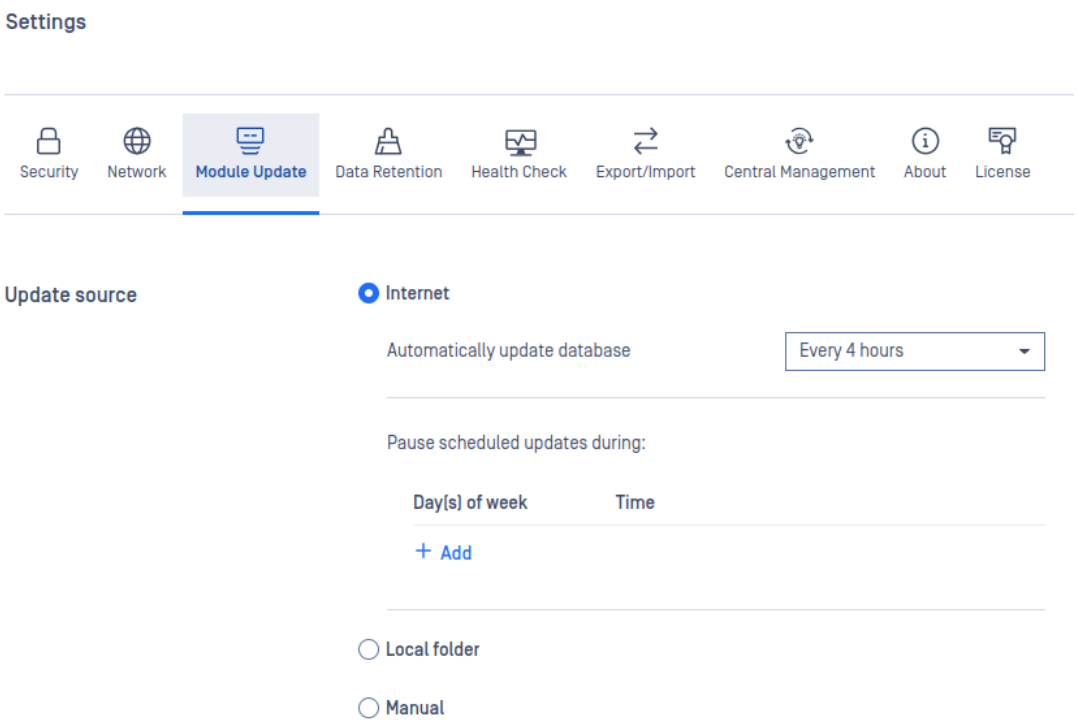
Installation

Pro Tip
The 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 Sandbox environment

You might want to use your Remote Sandbox Engine with a standalone MetaDefender Sandbox instance running in an offline environment. For more details on how to use your MetaDefender Sandbox instance offline, please see the link below.

How do I use MetaDefender Sandbox in an offline environment

Licensing

> **Note**
> This page applies to the **Embedded Adaptive Sandbox** engine version **2.2.0** and above.

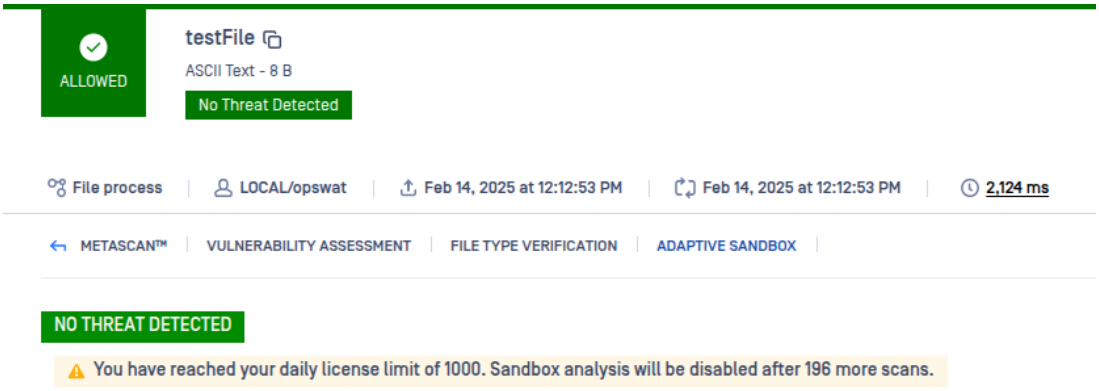
License Usage and Execution Limits

Your MetaDefender Core license includes a **daily execution limit** for the **Embedded 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.

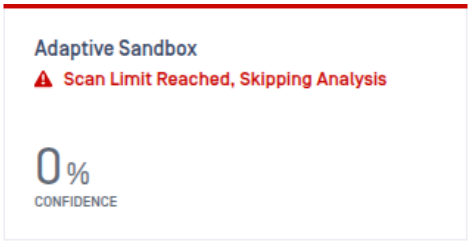
Soft Limit (Warning)

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 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 Sandbox 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

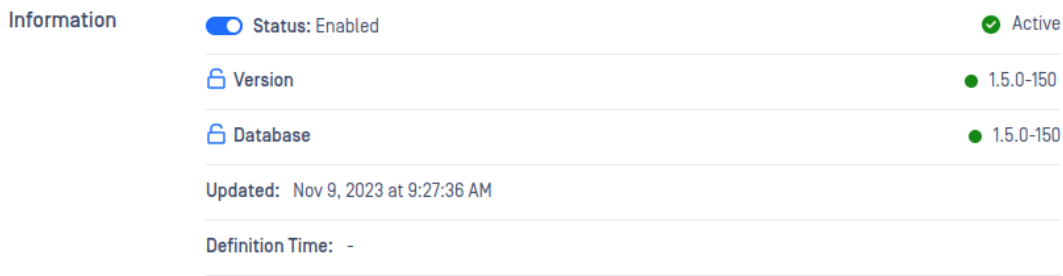
Go to **Inventory > Modules** and select the **Sandbox** module

> **Pro Tip**
> Use your OPSWAT Reputation secret to further enhance your 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

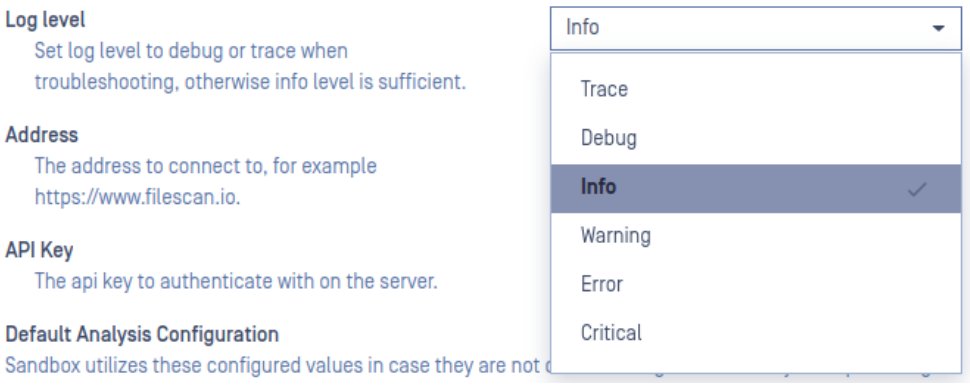
See details like engine version and status.



Configuration

Log level

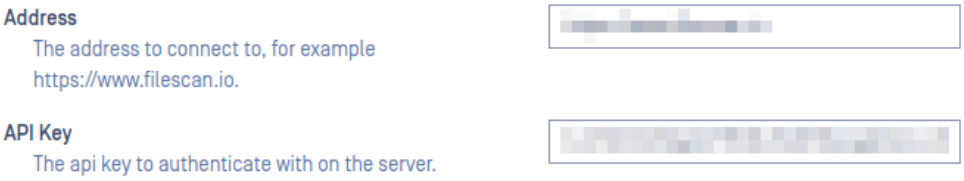
Configure the log level of the Sandbox Engine. In case of issues a lower log level (**Debug**) might help the support team.



Remote server address

Configure the address and secret of your standalone MetaDefender Sandbox instance.

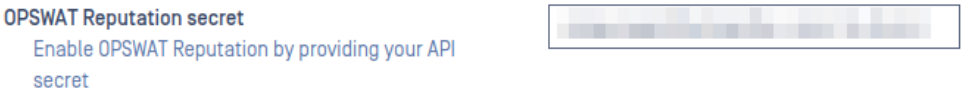
> **Info**
> Only available for the Remote Sandbox Engine



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

Default Analysis configuration

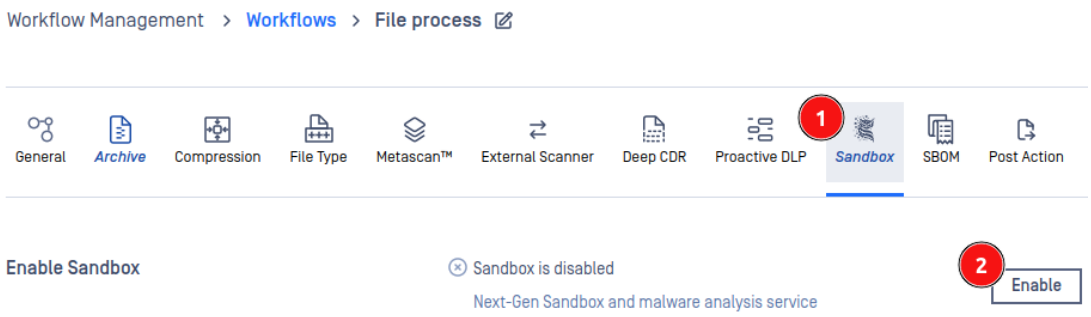
The global configuration stores the default configuration settings for default workflow and quarantine processing. For more details see the workflow configuration section below on this page.

Workflow configuration

Go to **Workflow Management > Workflows**, select your workflow and select the **Sandbox** module

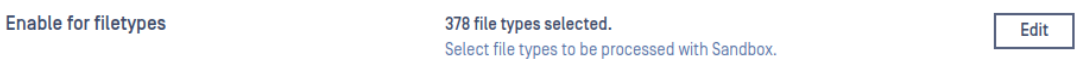
> **Pro Tip**
> Customize your workflows based on your specific usage scenario to enhance performance. For example turn off *file types* and update *engine results filtering*.
>
> Setup several workflows tailored to different use cases, if required.
>
> Activate `Rapid mode` when there's a need for higher throughput.

Enable the Sandbox Engine in your workflow



Filetype filtering

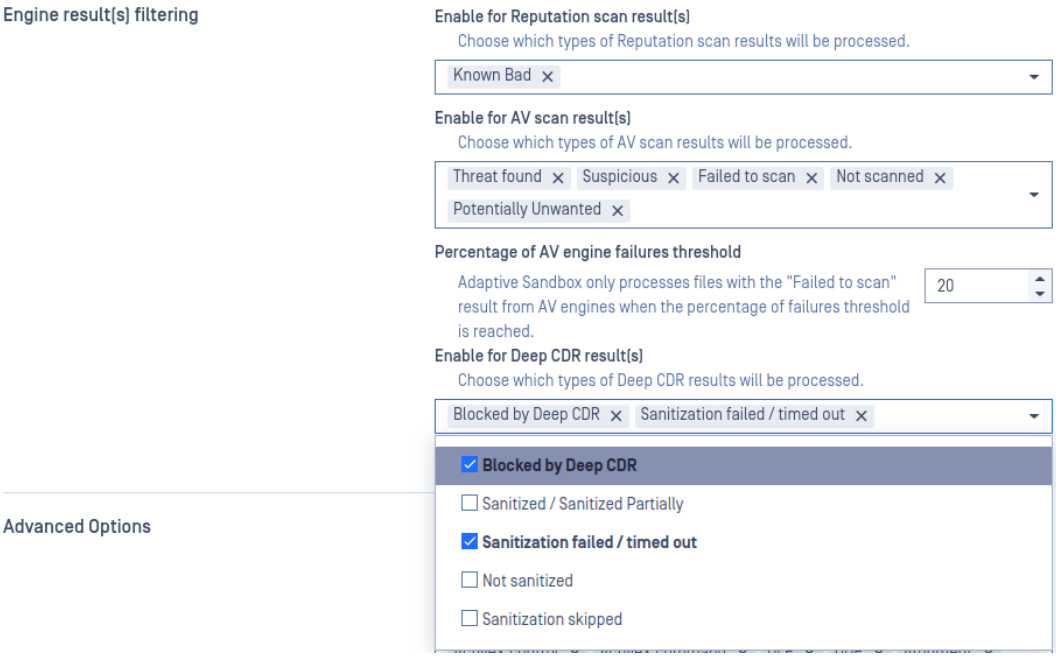
Configure the Sandbox Engine to run only for the selected file types. All the supported file types are selected by default.



> **Warning**
> It's not recommended to choose file types that aren't included in the default configuration.

Engine result filtering

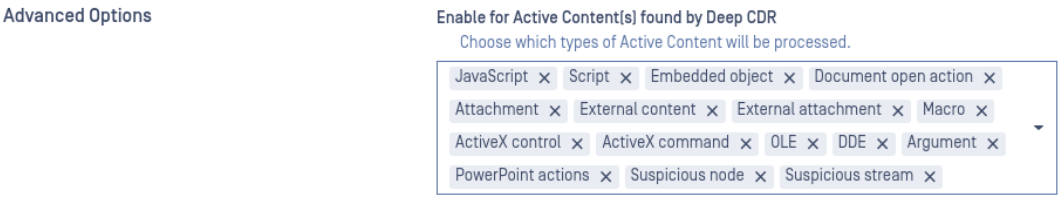
Configure the Sandbox Engine to run conditionally based on **Reputation** engine results, **Metascan** AV engine results or **Deep CDR** sanitization results.



Advanced options

Deep CDR filtering

Configure the Sandbox Engine to run conditionally based on Active Content(s) found with Deep CDR processing.



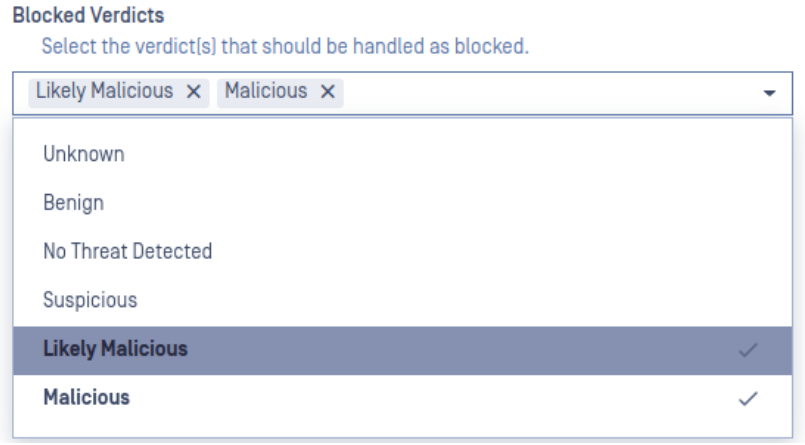
Scan timeout

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.

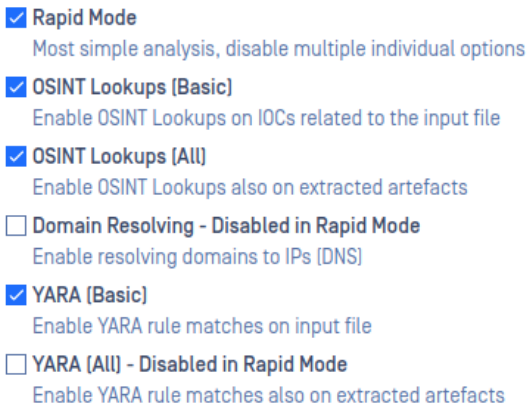


Scan tasks

Configure which tasks should be executed and configure **Rapid mode**. Different configuration settings available for the Embedded and Remote Sandbox Engine.

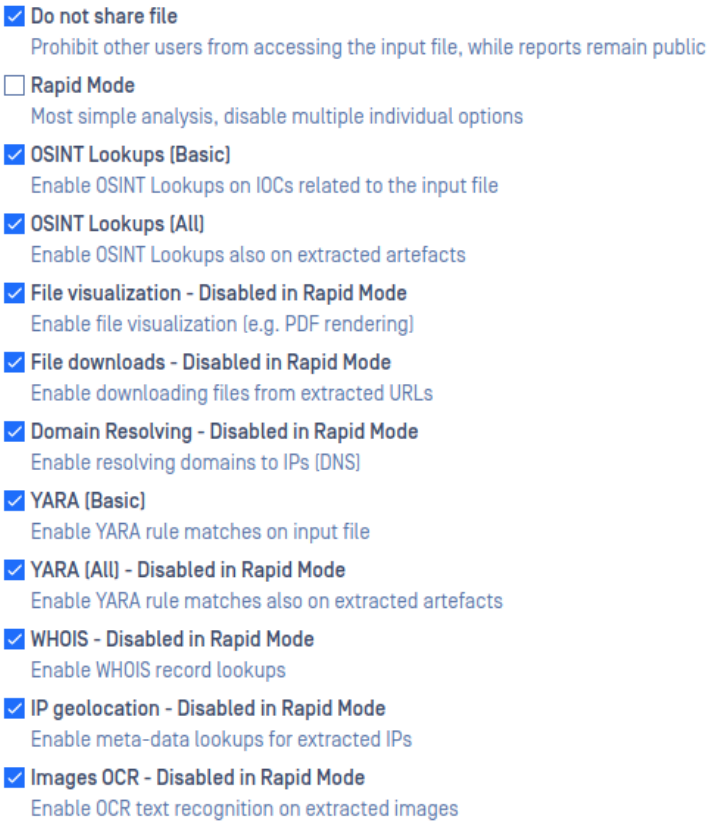
> **Info**
> Rapid mode disables some scan tasks which is indicated in the task description and default settings.
>
> If Rapid mode is activated, the scan process will focus on the input sample and it will spend less time with extracted resources (additionally downloaded and extracted files).

Embedded Engine



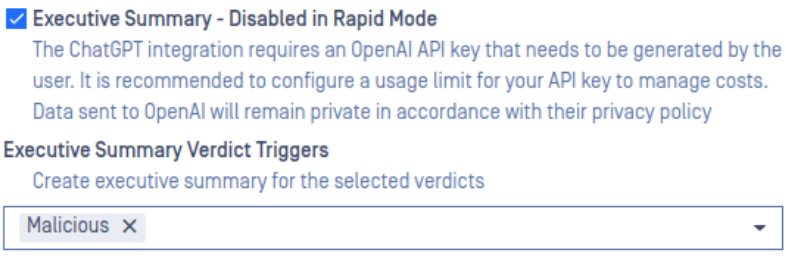
Remote Engine

> **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.



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 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 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 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
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

Release notes

Our release notes were moved here.

Knowledge Base

A collection of information regarding the Adaptive Sandbox module in MetaDefender Core. For an FAQ of MetaDefender Sandbox, 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 & 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.

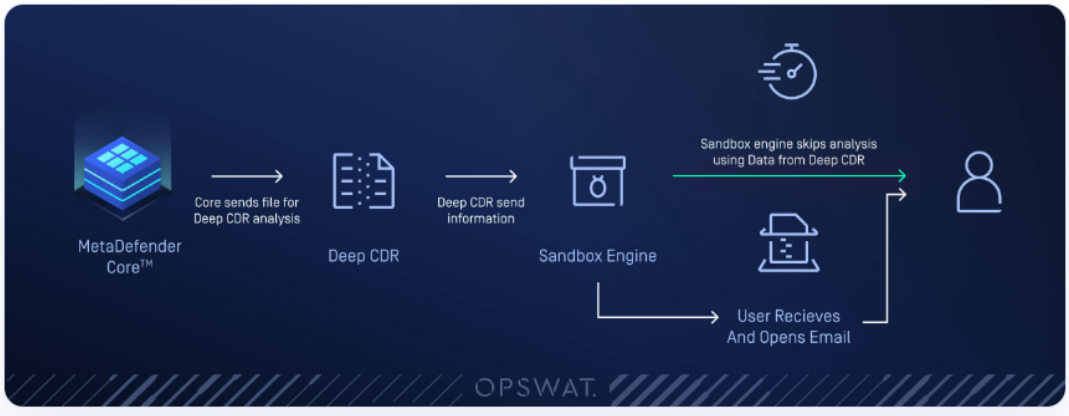
Integrating Deep CDR Analysis Mode with Adaptive Sandbox provides a layered approach to security, combining content disarmament (CDR) for immediate risk reduction with sandbox analysis for advanced zero-day detection.

- **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.

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.**



Setup Deep CDR Analysis mode

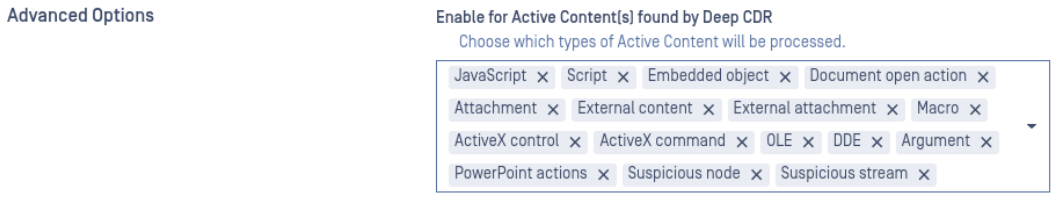
If you prefer to retain your files without modifications while still leveraging the benefits of CDR integration, enable the **CDR Analysis Mode**. For more information, visit the Deep CDR analysis mode page.

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.

Setup Adaptive Sandbox Deep CDR 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":



By default, **every supported Active Content is selected,** but it can be configured to best suit the current application.

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

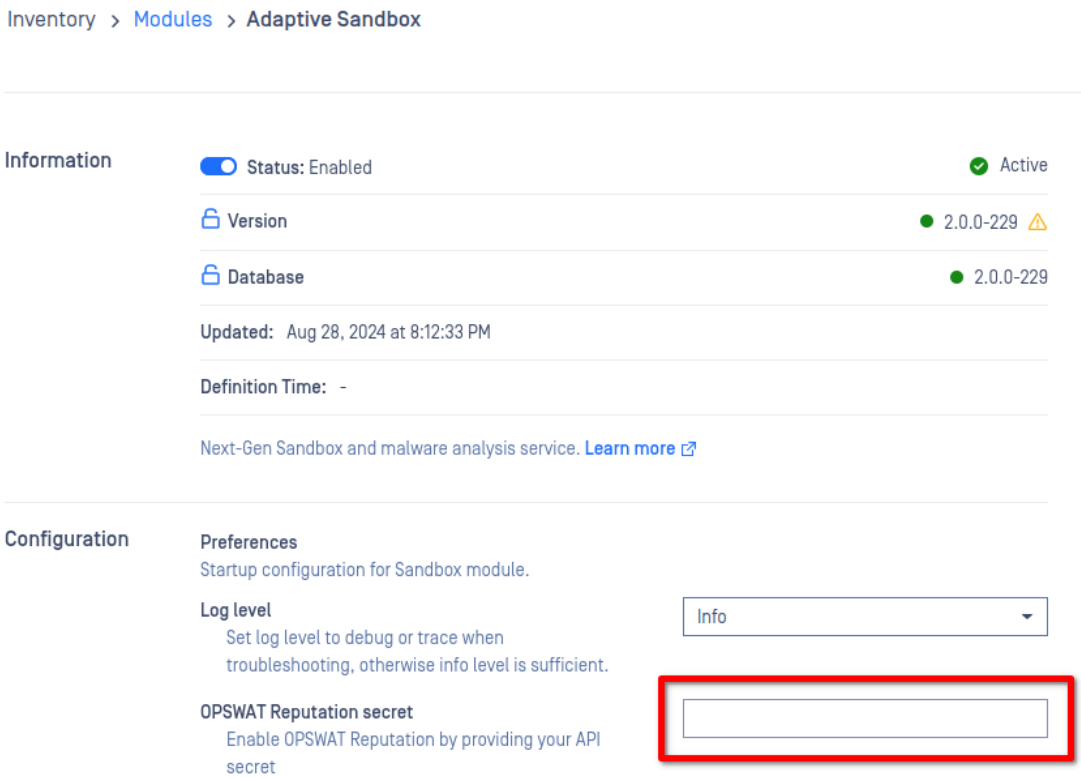
Step 2: Locate the API Configuration Section

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


Inventory > Modules > Adaptive Sandbox
Information
Status: Enabled
Active
Version
2.0.0-229
Database
2.0.0-229
Updated: Aug 28, 2024 at 8:12:33 PM
Definition Time: -
Next-Gen Sandbox and malware analysis service. Learn more
Configuration
Preferences
Startup configuration for Sandbox module.
Log level
Set log level to debug or trace when troubleshooting, otherwise info level is sufficient.
Info
OPSWAT Reputation secret
Enable OPSWAT Reputation by providing your API secret

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

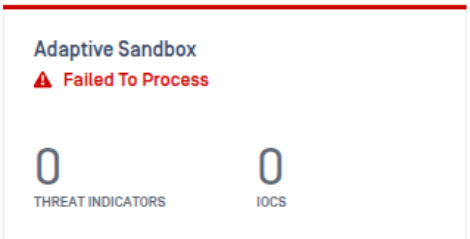
For successful scans, the tile displays the Adaptive Sandbox verdict, the number of threat indicators, and the IOCs (Indicators of Compromise) identified during the scan. The tile's color corresponds to the verdict, providing a quick visual summary.



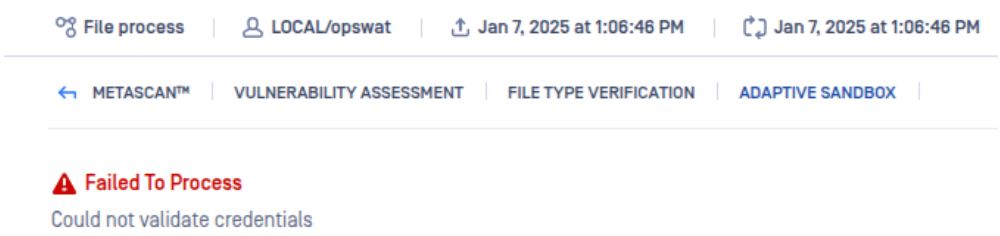
Clicking on the tile opens the detailed report, allowing users to review in-depth information about the scan results.

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.

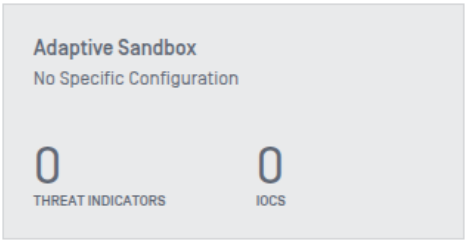
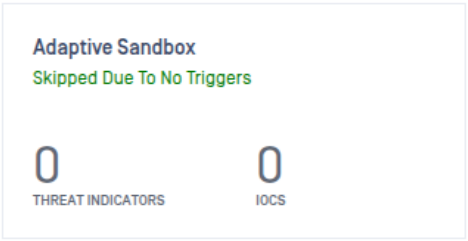


Clicking on the tile opens the error report, where users can review the error details to understand the cause of the failure.



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 & 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.

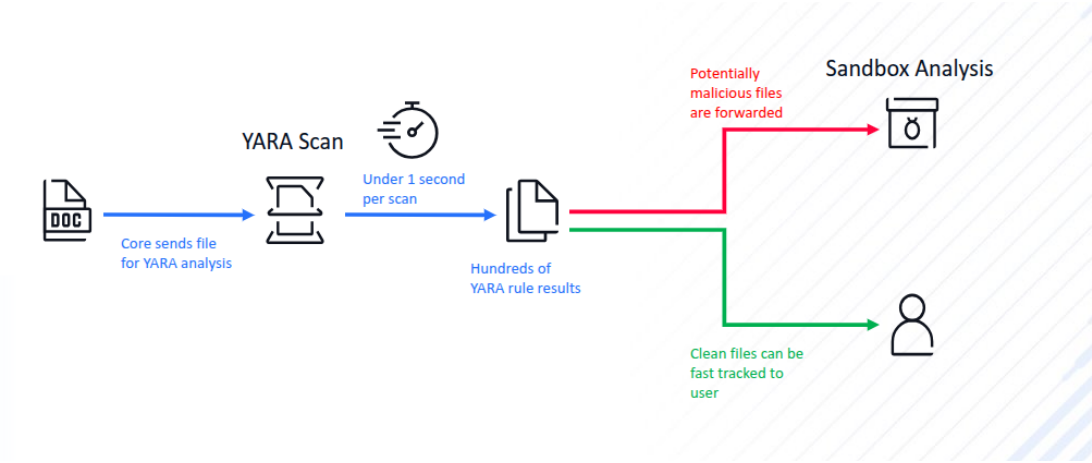
Integrating YARA Scanning with Sandboxing provides a layered approach to threat detection, combining rule-based static analysis for rapid identification of known threats with dynamic sandbox analysis for uncovering advanced, evasive, or zero-day malware.

- **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

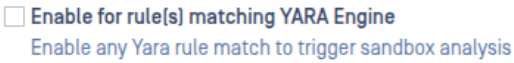
YARA rules are powerful for identifying specific patterns and characteristics of malware. By using YARA, you can detect known threats and suspicious behaviors within files or processes. This initial detection helps in identifying potential attack vectors that might be missed by traditional antivirus 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.

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

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.

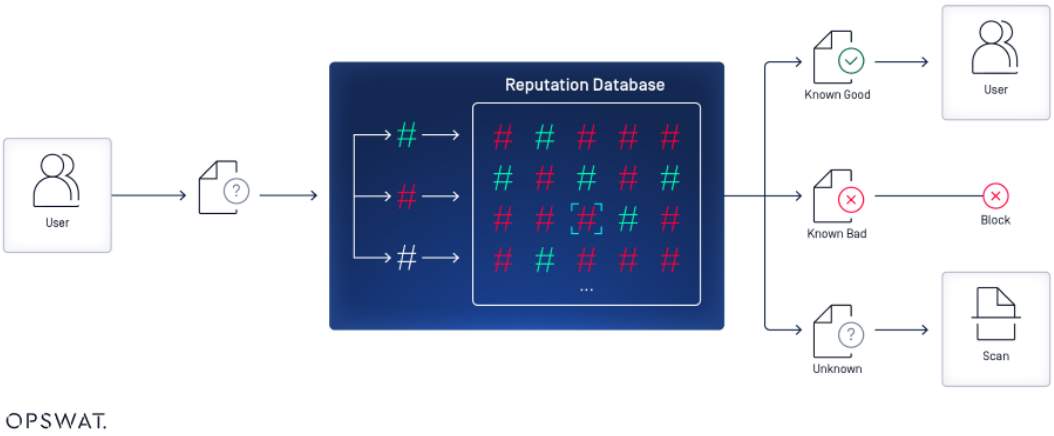
By combining these elements, MetaDefender Core offers comprehensive protection that addresses multiple vectors of attack, ensuring a more resilient defense against cyber threats.

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.



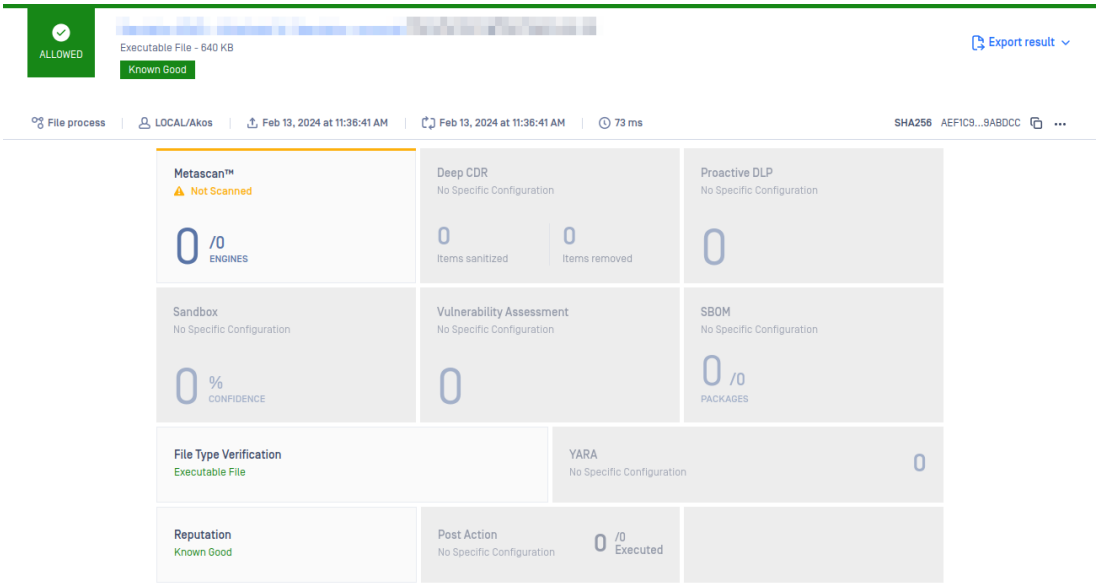
Reputation Engine Mechanism

Displaying engine results

Info

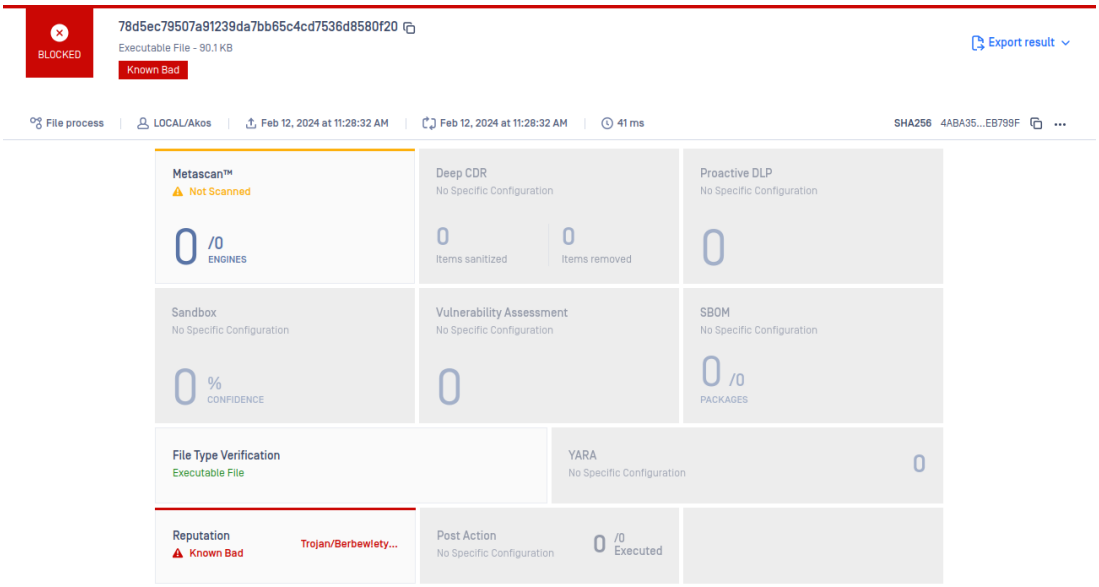
Below you will be shown examples of how the three types of file reputation results are displayed inside MetaDefender Core

Known Good: The hash matches an entry of known good files in OPSWAT's the real world database.

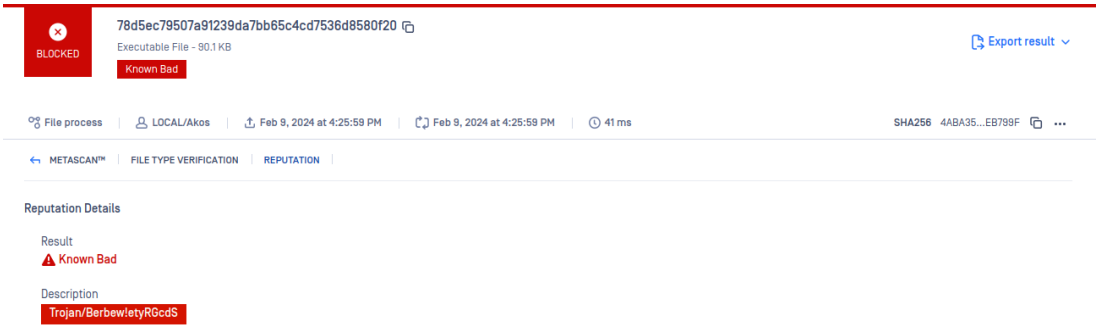


Known Good result example

Known Bad: The hash matches an entry of known malicious files in OPSWAT's real-world database.

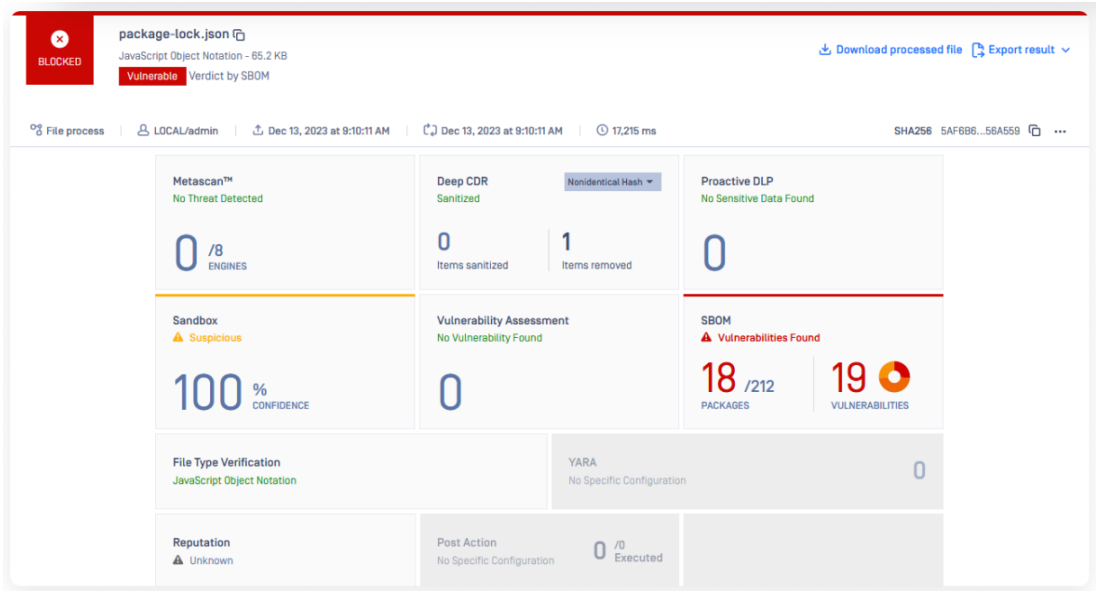


Known Bad result example



Known Bad detailed result example

Unknown: There aren't any matches on this hash in OPSWAT's real-world database.



Unknown result example

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

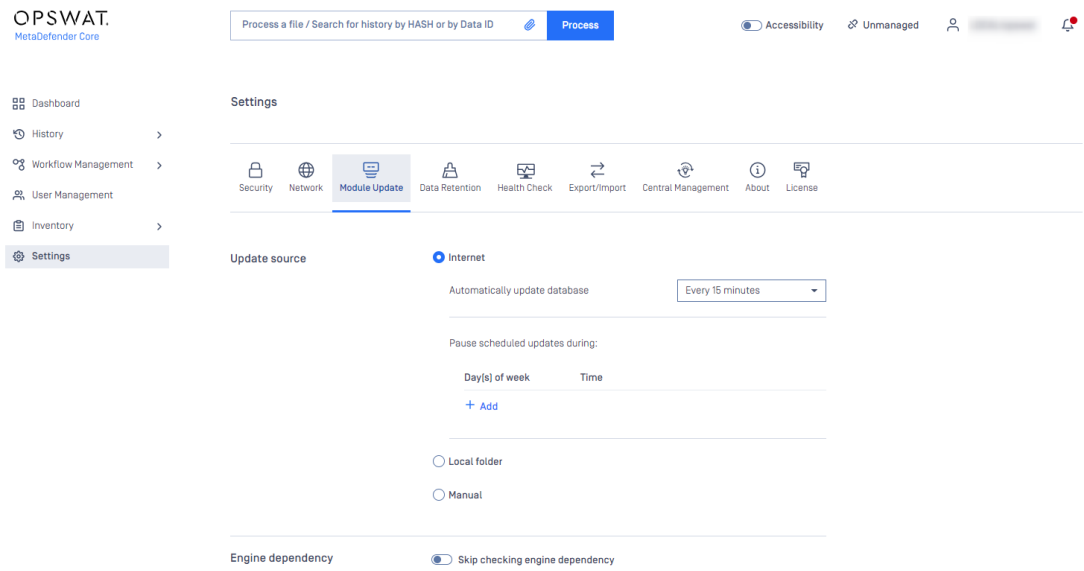
> **Info**
> **Reputation engine is officially supported starting from Core 5.9.0.**

> **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:

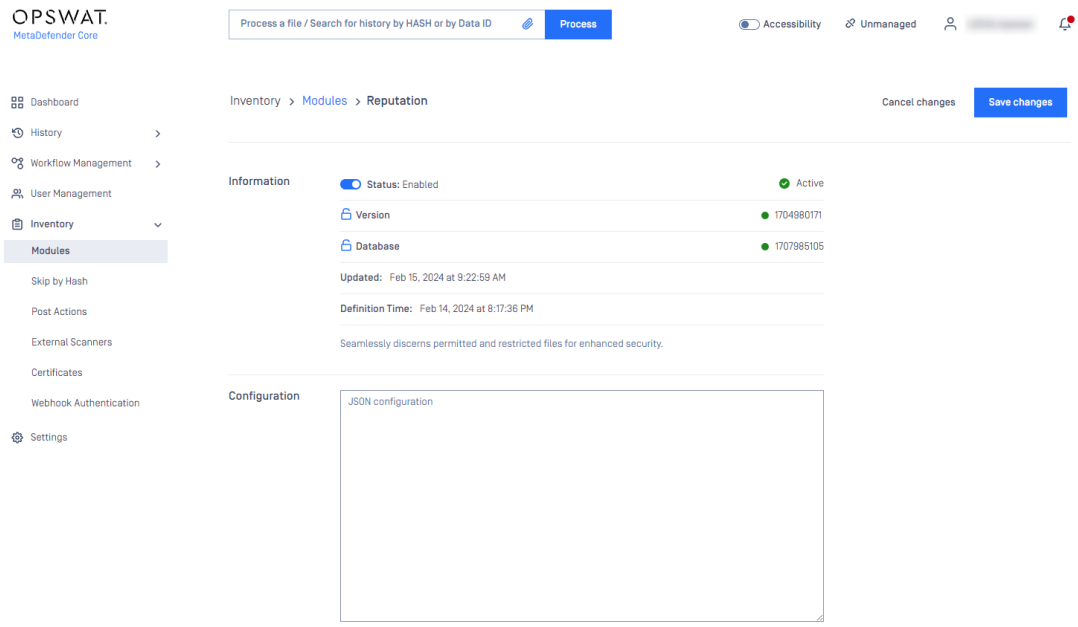
Inline: working as a part of MetaDefender Core processing workflow (real-time processing). Allowing users to block entire processing based on the outcome and decision of the Reputation Engine.

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:

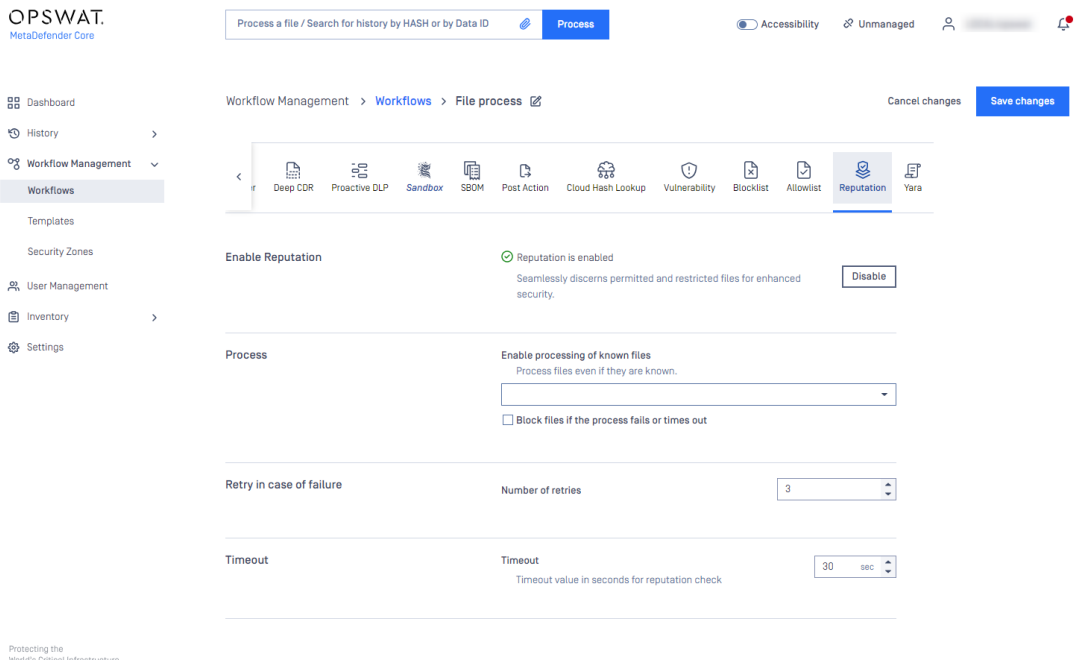
Configuration steps

- 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



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**

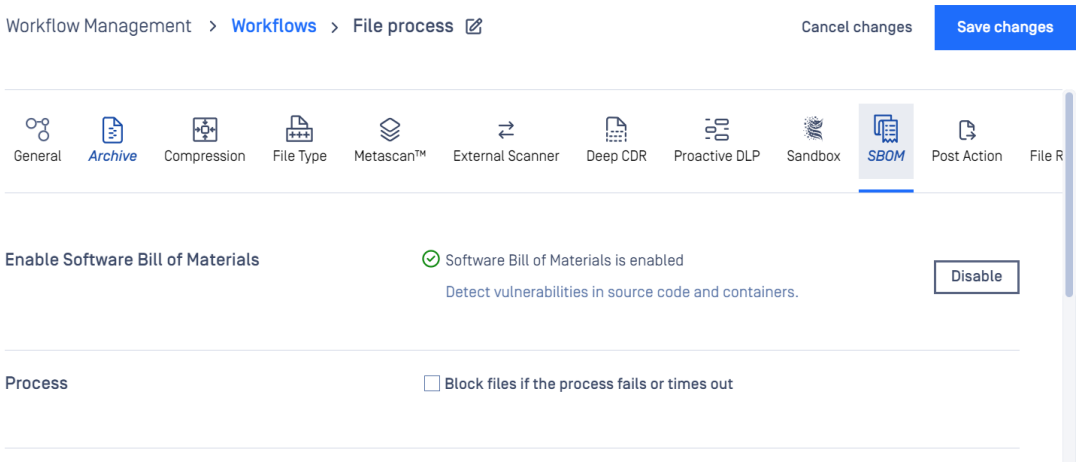
How will the false detection submission impact the AV vendor's verdict?

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 (Software Bill of Materials) analyzes the third-party packages in source code and containers, identifies known vulnerabilities, then generates inventories of these software components. With SBOM, users can scan track dependencies in large codebases to understand the risks associated with open-source software (OSS) and secure their applications.

To enable this engine, please go to Workflow > Workflow name > SBOM



Support:

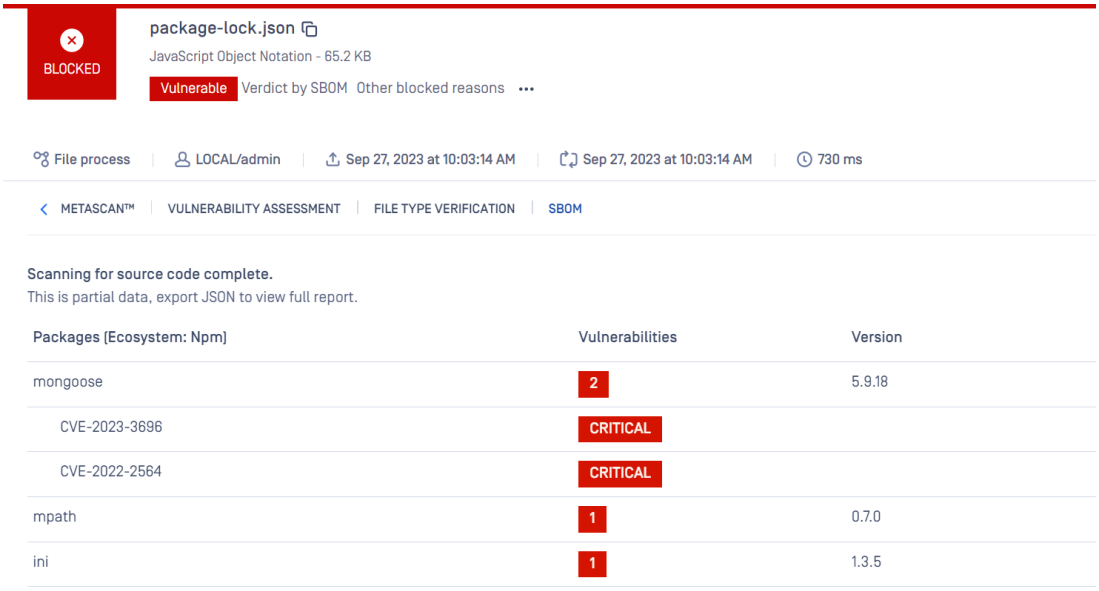
- Source code and container

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	File to check
Ruby	Gemfile.lock lib package in tar.gz, gem format
Python	Pipfile.lock poetry.lock requirements*.txt setup.py pyproject.toml lib package in tar.gz, egg, whl, zip format
PHP	composer.lock lib package in zip format
NodeJS	package-lock.json yarn.lock pnpm-lock.yaml lib package in tgz format
Java	pom.xml pom.properties gradle.lockfile *.jar lib package in *.zip, *-src.zip, *-sources.zip, *.tar.gz, *-src.tar.gz, *-sources.tar.gz format
Go	go.mod
Rust	Cargo.lock
Dart	pubspec.lock

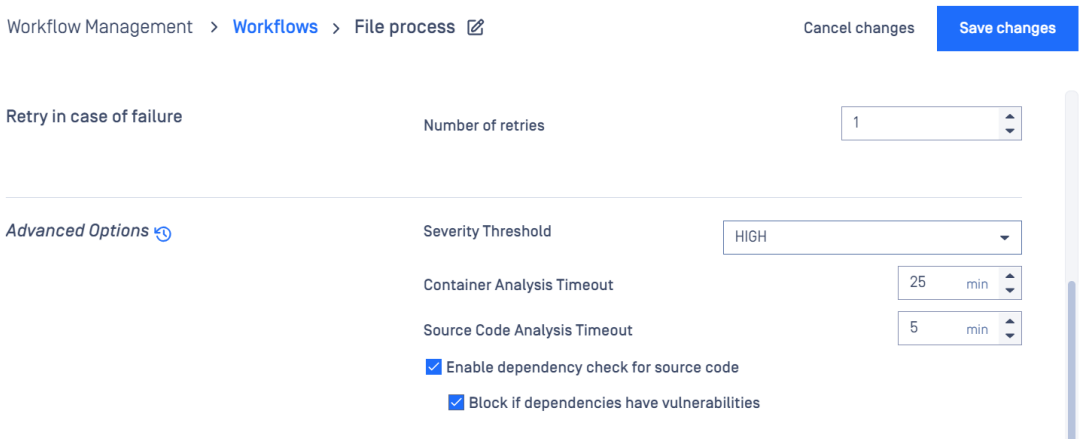
Programming language	File to check
.NET	packages.lock.json packages.config .deps.json .nuspec dll library in *.nupkg
Elixir	mix.lock
Swift	Podfile.lock
C/C++ package manager	conan.lock



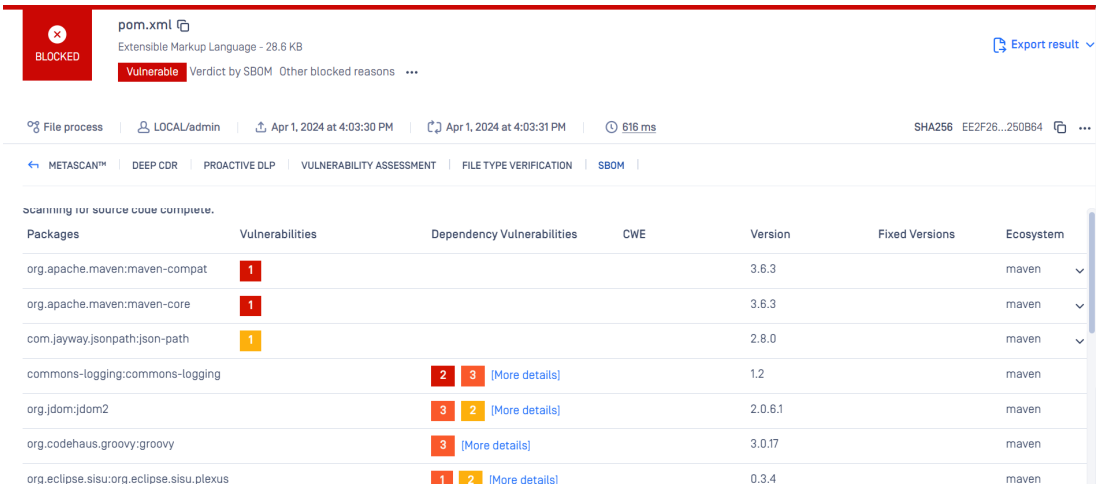
Recursive check

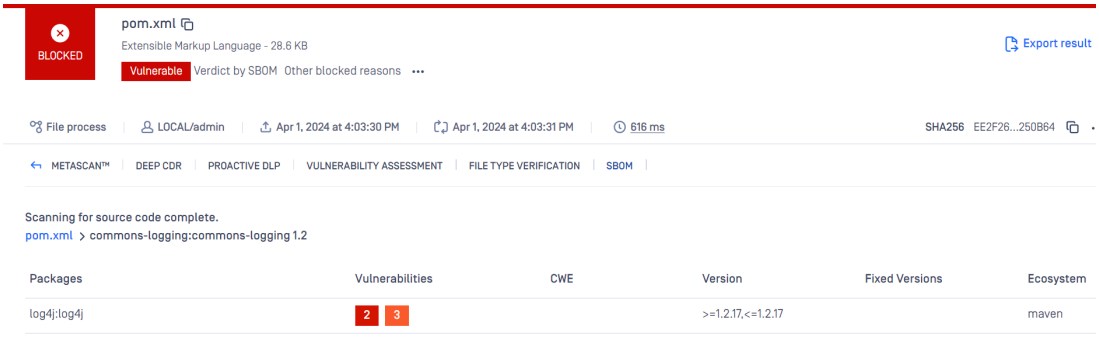
SBOM scans (nested) dependencies used in open source libraries to detect known vulnerabilities from potentially harmful packages.

To enable recursive check: Workflow > Workflow name > SBOM > Enable dependency check for source code.



Example scan result



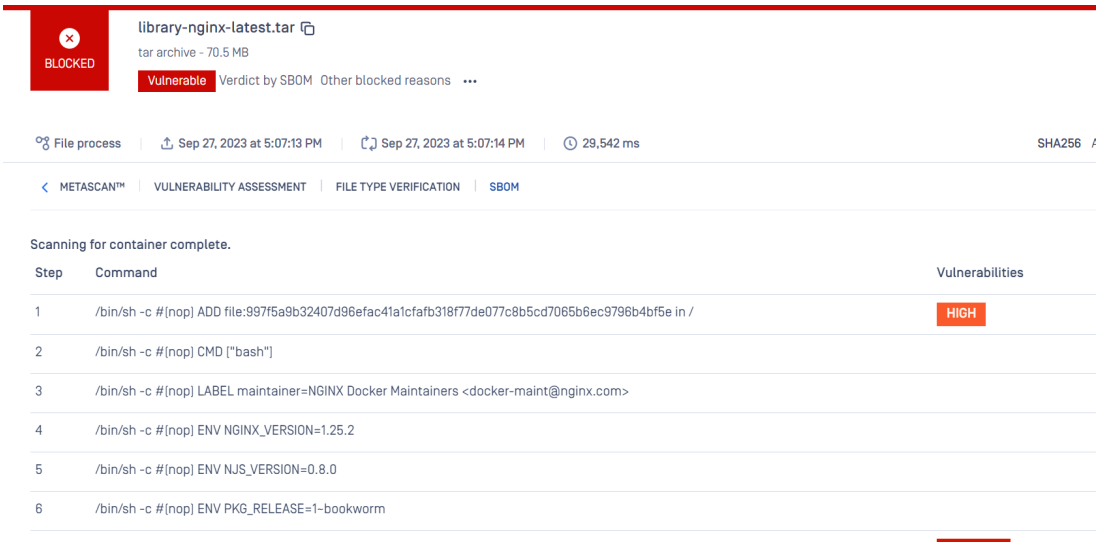


- Supported programming languages: python, nodeJS, .net, ruby, maven, php
- Supported recursive level: 1

Note: in the container, all dependencies are on the container image, the SBOM engine checks all of them without a configuration.

Container image

Support a .tar input file which is generated from **docker save** command



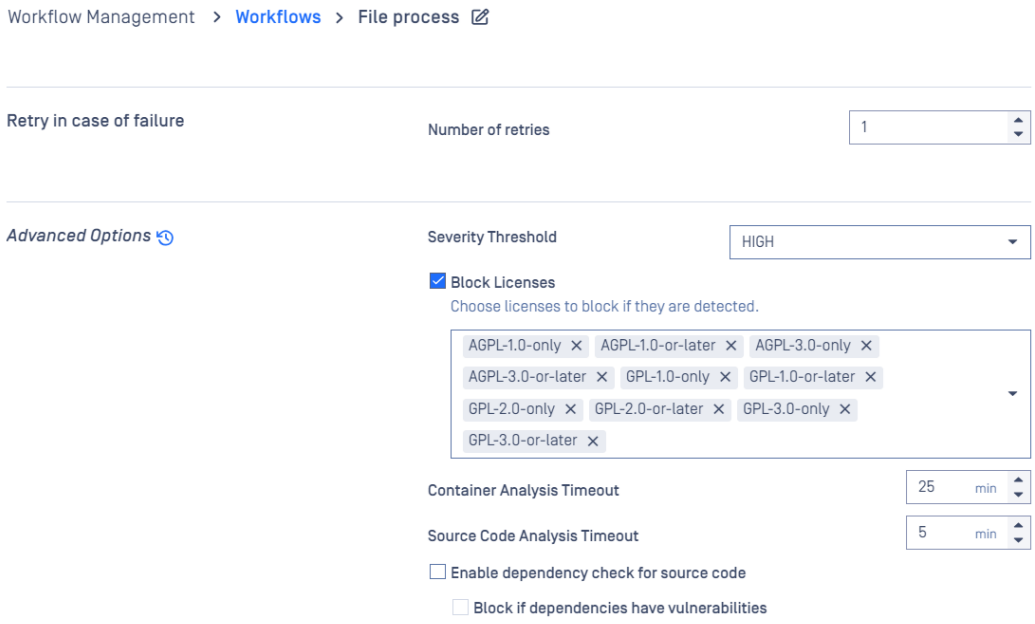
Support OS libraries in

- Alpine
- RedHat
- Centos
- Ubuntu
- Debian
- SUSE

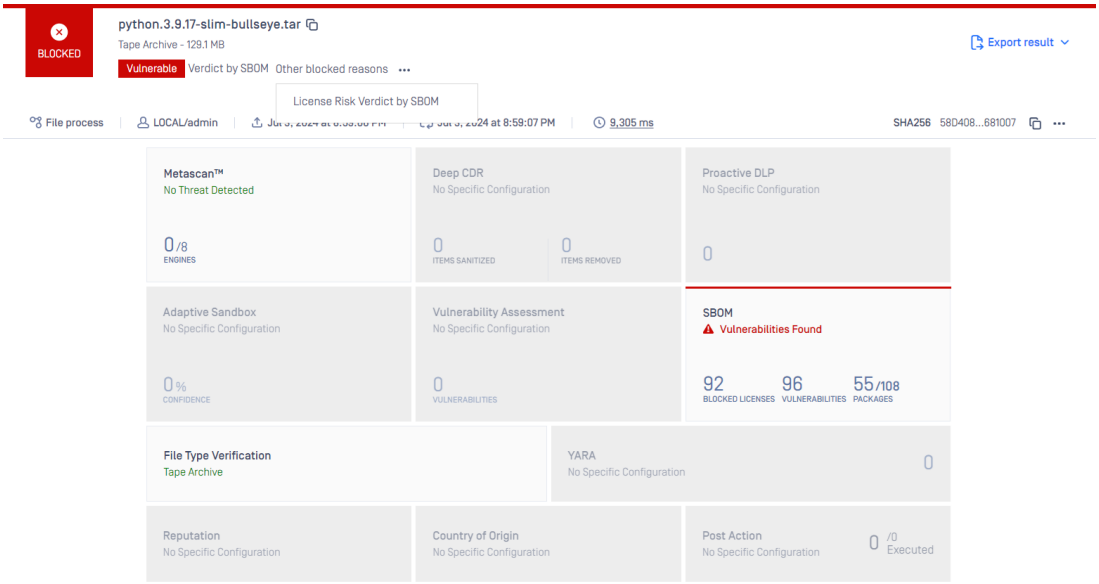
License risk detection

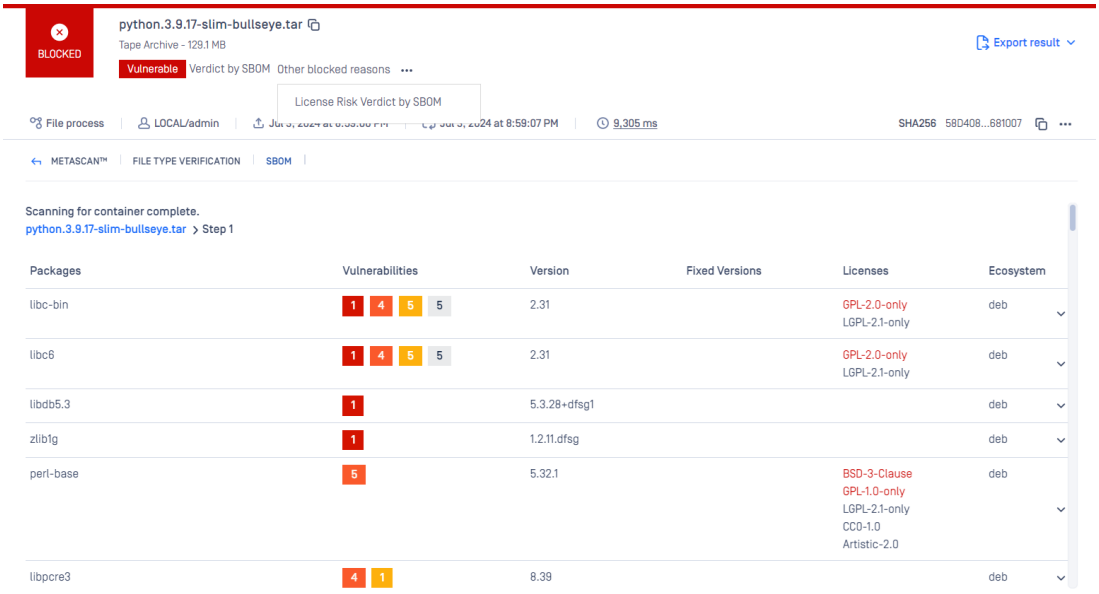
Licenses can include clauses that require you to open-source your code. Ensuring you use licenses that do not jeopardize your company's value is crucial. OPSWAT SBOM offers a robust license detection feature that thoroughly examines each open-source components. This functionality goes beyond merely identifying the license type, offering a detailed view of your open-source dependencies.

Configure to block specific license types in the workflow:



Process result:





Supported programing languages: Python, Ruby, PHP, Maven, Npm.

JSON response

Source code:

- If the "Enable dependency check for source code" option is not enabled, please ignore all "dependencies" attributes

json

```
{
  "final_verdict":{//the aggregated section for both direct and dependent libs
     "blocked": <boolean> //true: found CVEs that have severities are higher than
the configured threshold, otherwise it is false
     "verdict": <string>
        <"No Vulnerabilities Found" | "Vulnerabilities Found"> //explaination
     "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
    "dependencies_severity": <string>, //show if enable dependency detection
  },
        "overview_report": {
    "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": <number> //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
```

```
    {
      "ecosystem": <string>,
      "group": <string> ["lang_pkg"],
      "package_name": <string>,
      "package_version": <string>,
      "target": <string>,//in case of source code, it is the input file name such
as pom.xml, requirement.txt
      "vulnerabilities": [
        {
          "id": <string>,
          "severity": <string>,
          "source": <string>,
          "cwes": [<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>
      },
      "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:

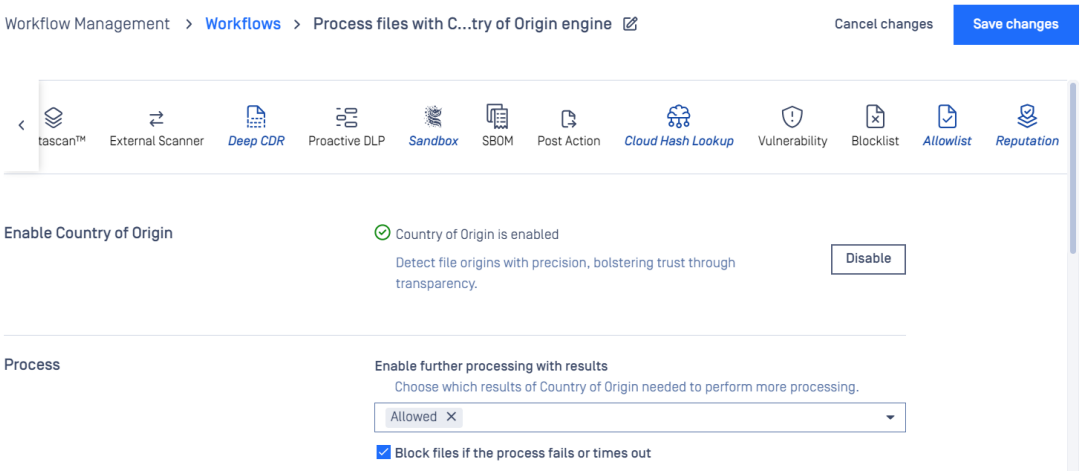
json

```
{
  "final_verdict":{
     "blocked": <boolean>
     "verdict": <string>
        ["No Vulnerabilities Found" | "Found Vulnerabilities"]
     "vulnerable_package_count": <int>
     "total_package_count": <int>
     "vulnerability_counts":{
      "critical": <int>
      "high": <int>
      "medium": <int>
      "low": <int>
      "unknown": <int>
    },
    "severity": <string>
  }
  "scan_mode": "container",
  "os_info": {
    "os_name": <string>,
    "os_version": <string>
  },
  "steps": [
    {
      "step": <int>,
      "command": <string>,
      "sha256": <string>,
      "severity": <string>
      "vulnerability_count": <number>
      "packages": [
        {
          "arch": <string>,
          "ecosystem": <string>,
          "group": <string> ["os_pkg"],
          "os_name": <string>,
          "os_version": <string>,
          "package_name": <string>,
          "package_version": <string>,
          "vulnerabilities": [
            {
                "id": <string>,
                "severity": <string>,
                "source": <string>,
                "fixed_versions": <string>
            }
          ],
          "vulnerability_counts": {
            "critical": <number>,
            "high": <number>,
            "low": <number>,
            "medium": <number>,
            "unknown": <number>
          }
        },
```

```
        {
          "ecosystem": <string>,
          "group": <string> ["lang_pkg"],
          "package_name": <string>,
          "package_version": <string>,
        }
      ]
    }
  ]
}
```

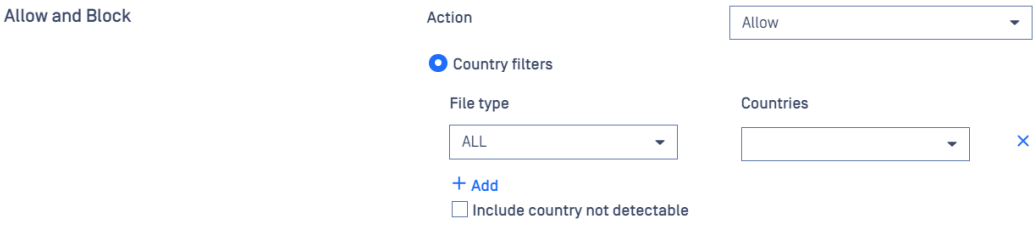
Country of Origin

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.



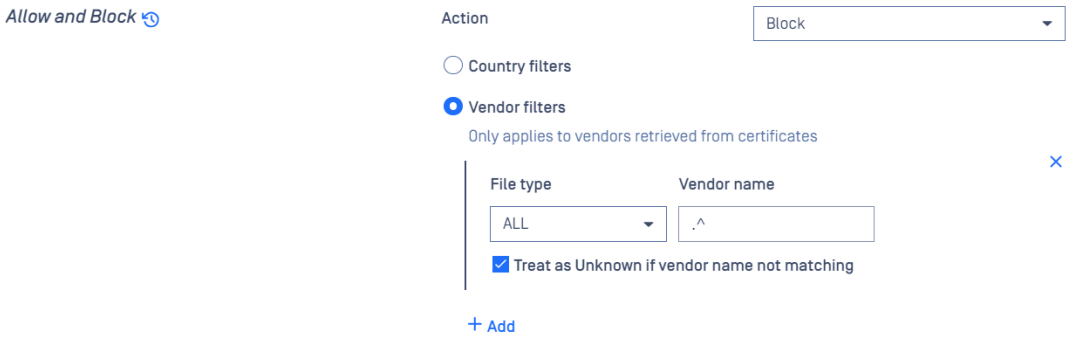
Detect and block by country

You can configure to block or allow from specific countries for each file types

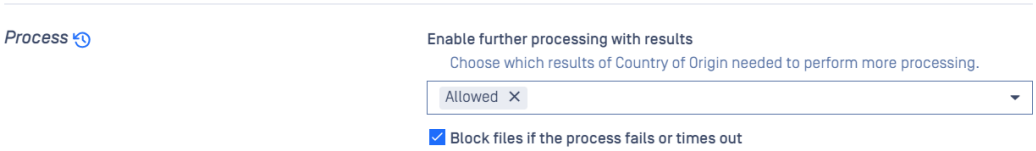


Detect and block by vendor

Similar to country, you can configure to block or allow from specific vendors.



Important: if the Allow action is chosen, we highly recommend you to let the files go to a regular scan process (with AV, ...) by choose this option

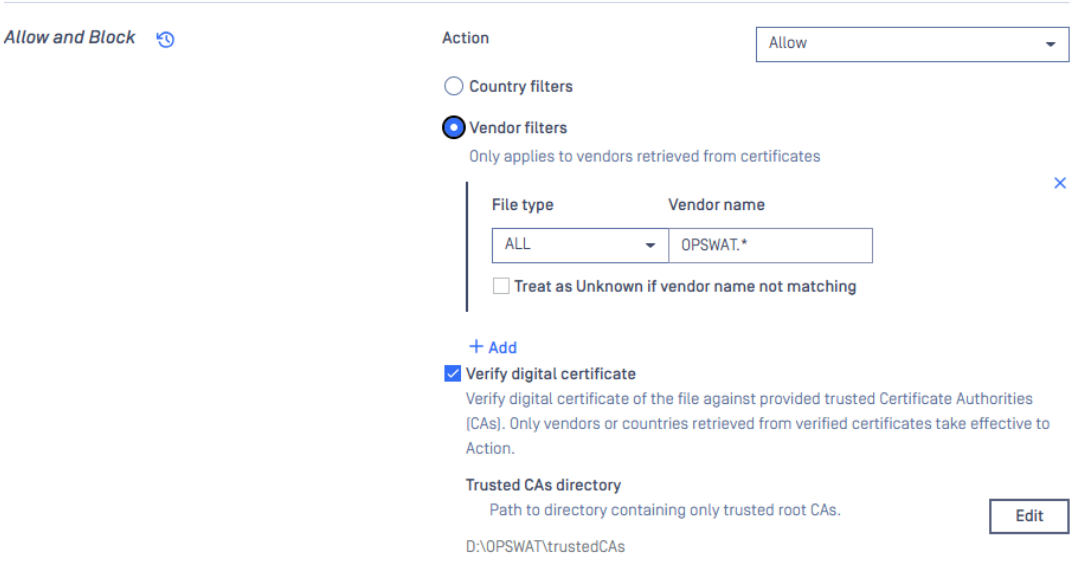


Support file types: DLL, EXE, EXE_SFX, EFI, MSI

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 EXE, DLL, and MSI file types.

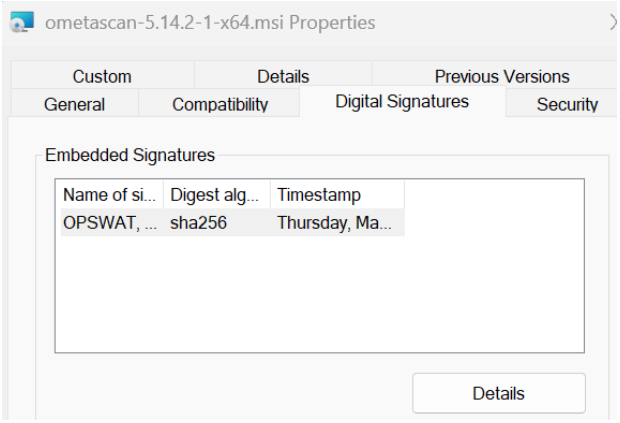
Example: Allowing Files Signed by a Specific Vendor

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.

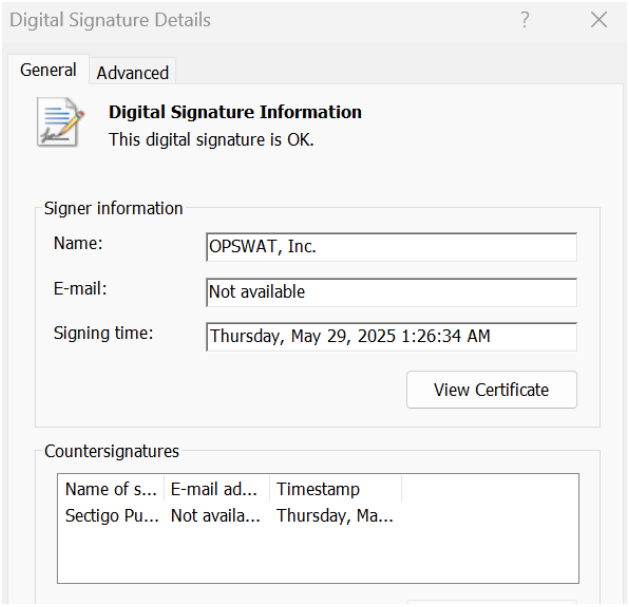
1. Collect Trusted CA files:

- Created a folder to store Trusted CA files, e.g.: C:\OPSWAT\TrustedCA

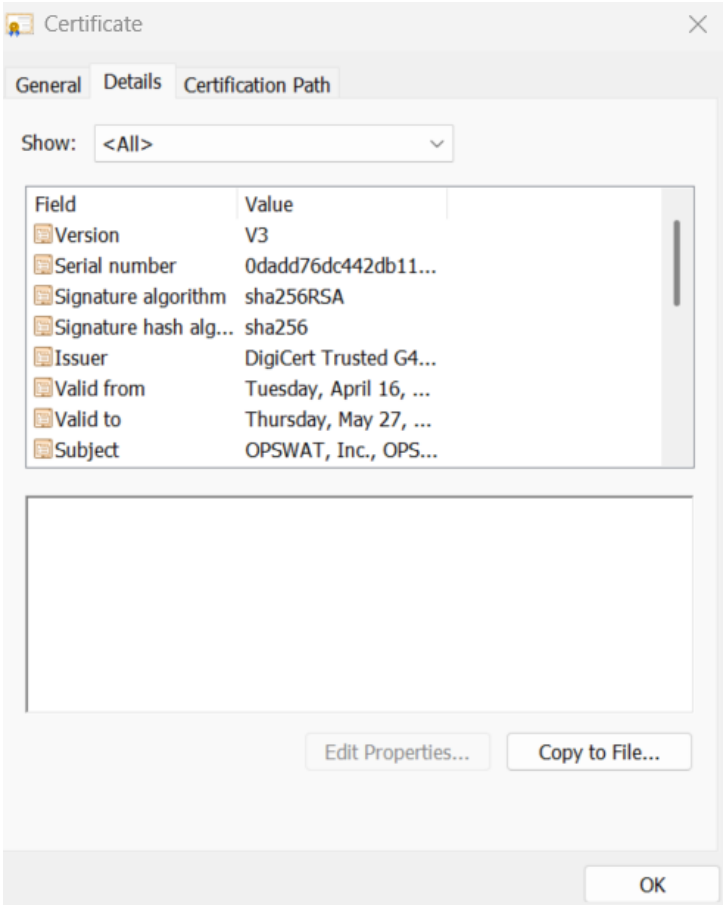
- Choose a file that is digitally signed by OPSWAT and that you want to allow.
- Right-click the file and select **Properties**. Go to the **Digital Signature** section, click **Details**



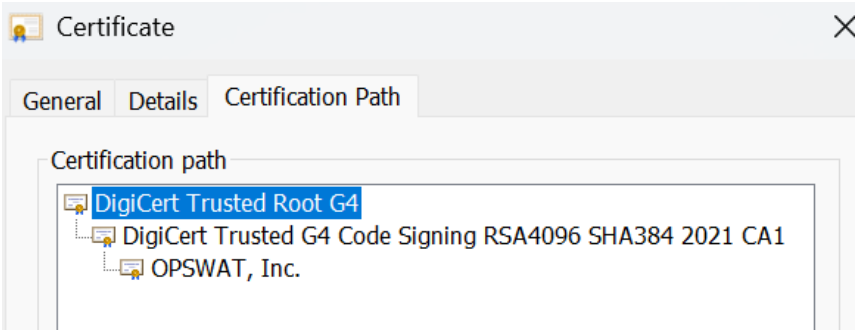
- In the **Signer Information** section, click **View Certificate** to open the certificate details window.



- 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



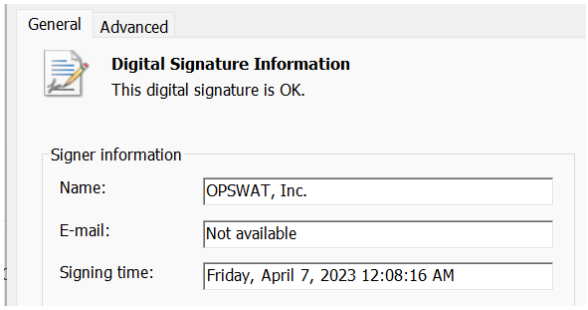
Notes: If a customer exports a root certificate (e.g., "**DigiCert Trusted Root G4**"), the COO engine will validate the entire certificate chain — including intermediate certificates like "**DigiCert Trusted G4 Code ...**" and the leaf certificate (e.g., "**OPSWAT, Inc.**") — to ensure both integrity and trust.

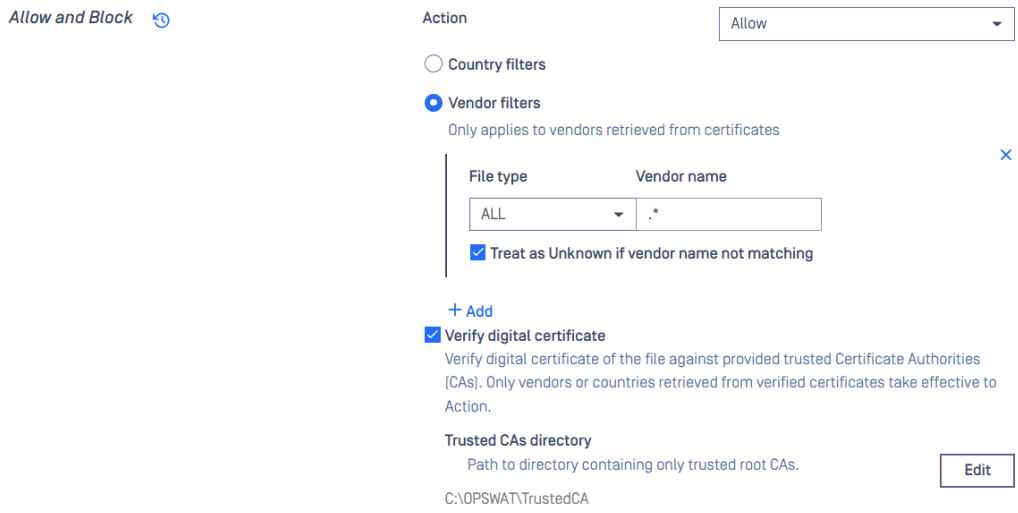


2. Configurations:

- 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**).**

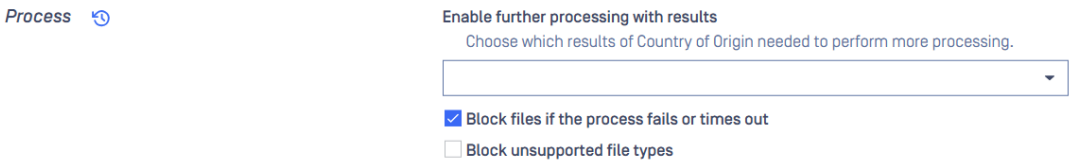
Info
Note: This value is taken from the Signer Name in the digital certificate.





Note:

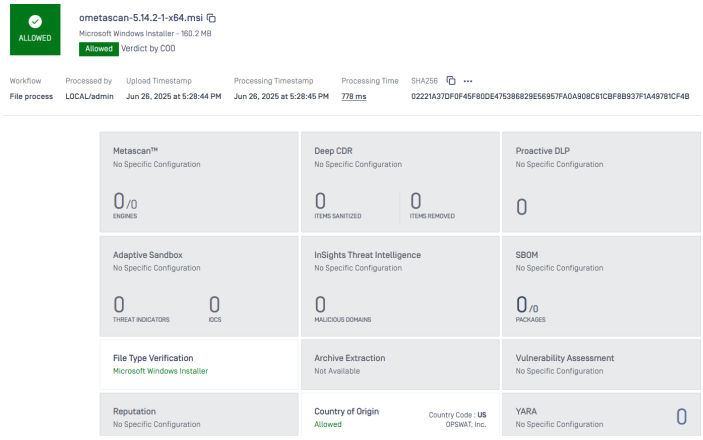
- Enable the option "**Treat as Unknown if vendor name does not match**". This ensures that any file failing signature validation will be scanned instead of being blocked.
- Do not select any option under "**Enable further processing with results**" to avoid unintended scans.

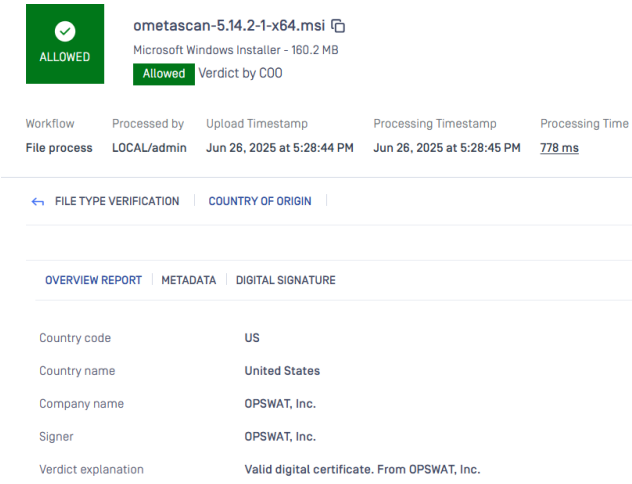


When processing files from OPSWAT with a valid digital signature:

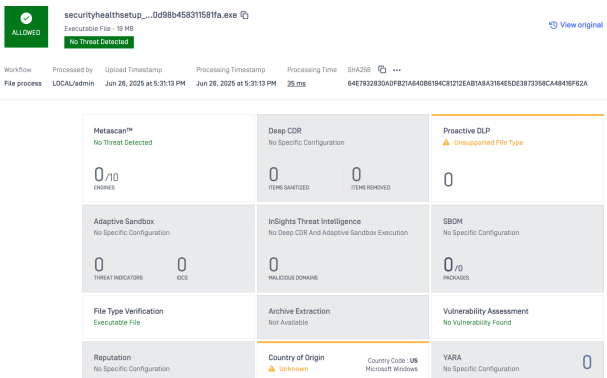
- The file will match the trusted vendor filter ("OPSWAT, Inc.").
- The configured action **Allow** will be applied.

- 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.



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**.

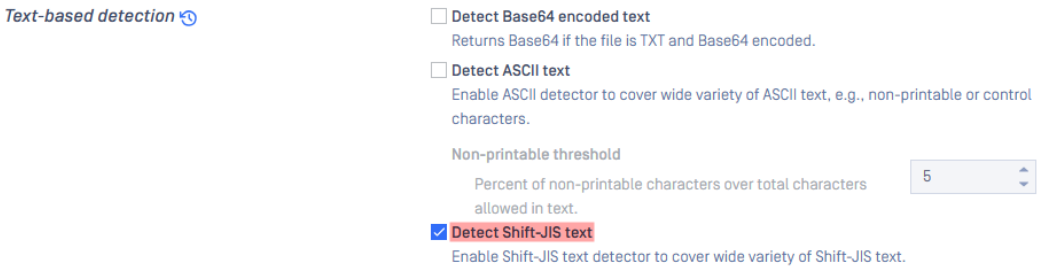
Issue: When processing plain text files containing Shift-JIS encoded characters (e.g., Japanese text), the files are detected as **"data" (application/octet-stream)** instead of **"ASCII text" (text/plain)**.

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 Engines 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**, 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 auto$

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.

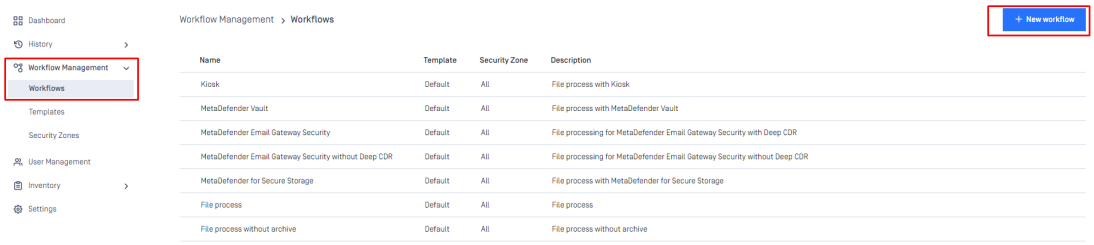
To contact us, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

Can I create new workflows?

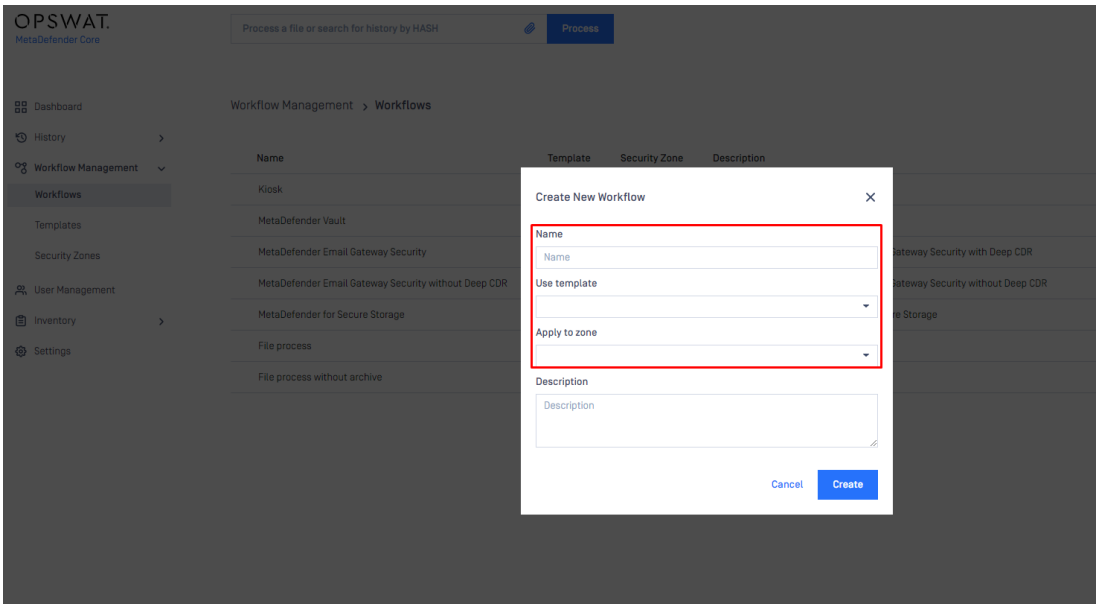
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

In order to create new custom workflow rules on the MetaDefender Core product, you will need to perform the following steps:

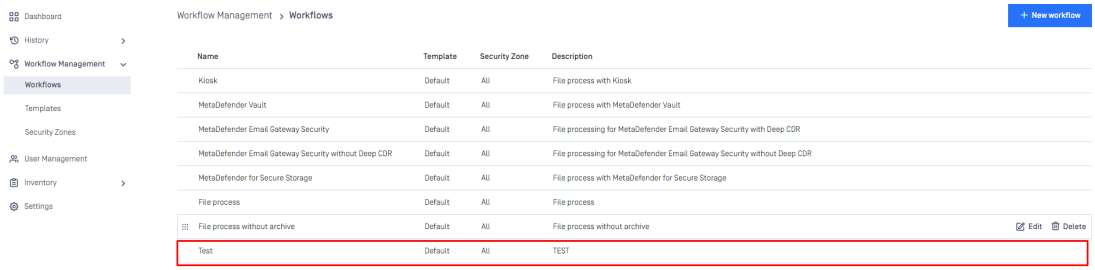
- Navigate to the Core console and access the Workflows tab.



- 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 auto$, then proceed to create a support case or chat with our support engineer.

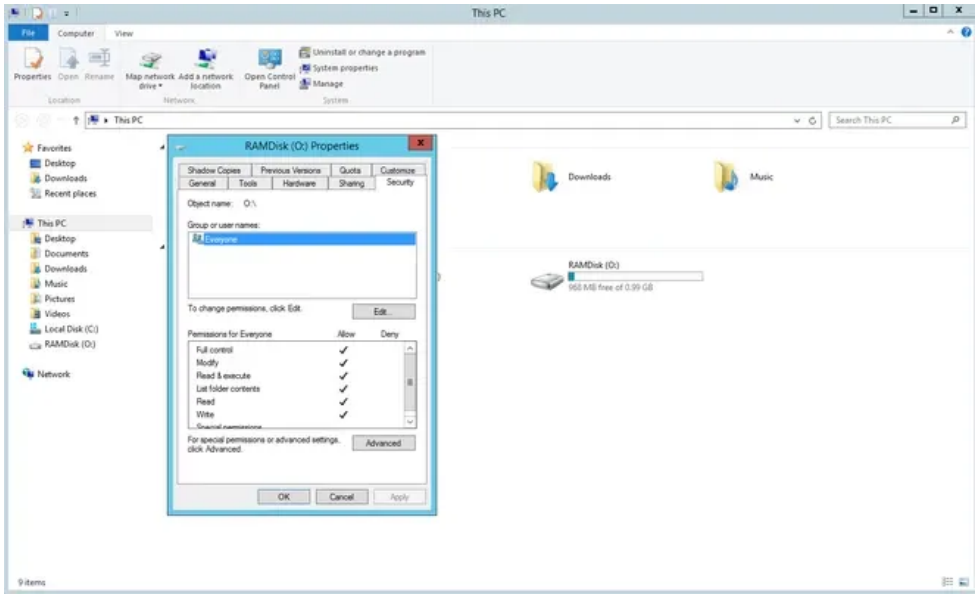
Can I control access to the RAM disk in MetaDefender Core v5?

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.

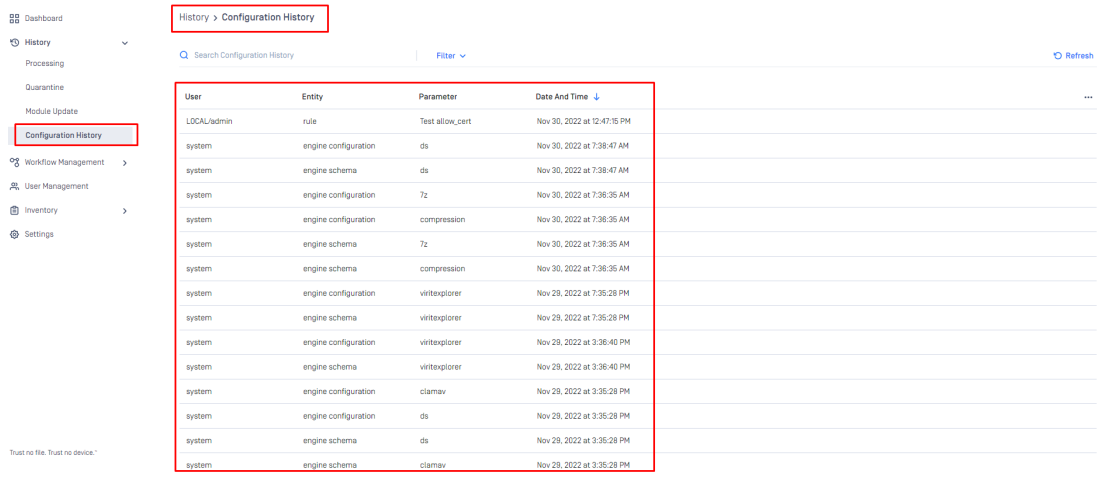


Support:
If you have followed all of the steps above but are still unable to set the desired **RAM Drive** security permissions, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

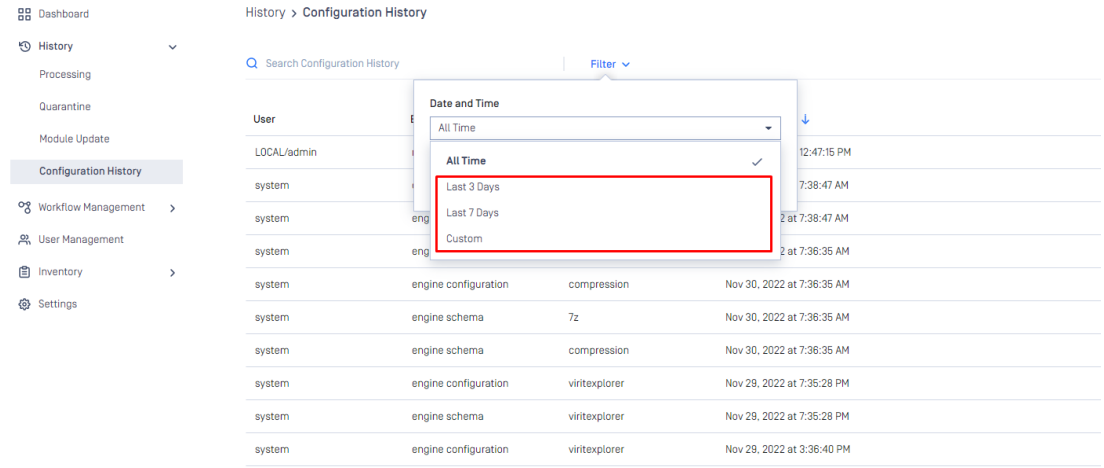
Can I view configuration changes on the Core side (audit)?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

Admin users are able to view all configurations that have taken place on the core side via the History\Configuration History tab, this can be found on the Core console.



There is also a filter option on the configuration page to allow for a more granular search regarding the time frame.



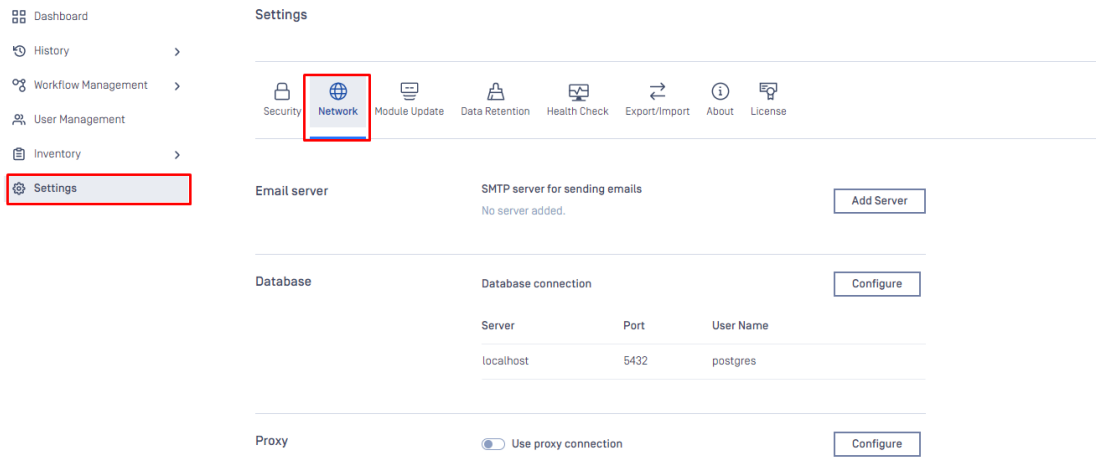
Support:
If **Further Assistance** is required, please follow the instructions on auto$, then proceed to create a support case or chat with our support engineer.

How can I add a mail server to the Core deployment?

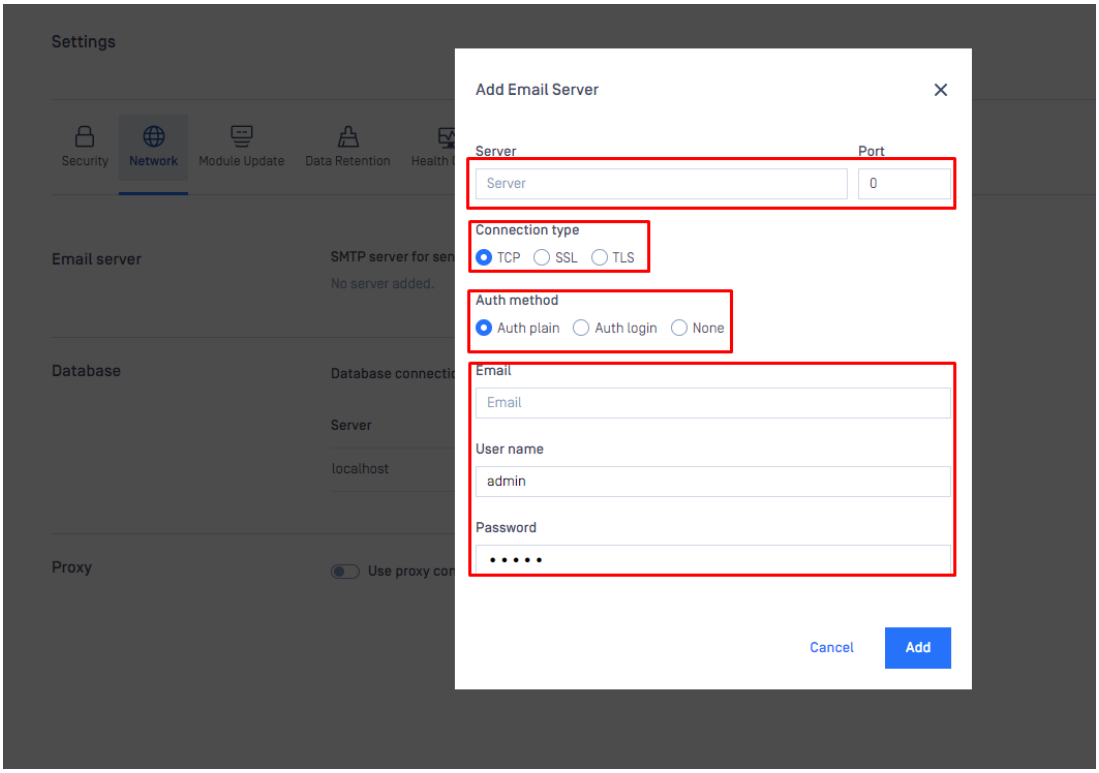
Check Your Version:
This article applies to all MetaDefender V5 releases deployed on Windows and Linux systems.

An email server is added to the MetaDefender Core deployment by performing the following steps:

- Log into the Core console and navigate to the **Settings\Network** tab.



- Open the "**Add server**" tab and complete the requested information fields, regarding Server URL, Port, Connection Type, Auth method, Admin user email and User name and password.



- After adding the requested information, simply click on the “add” button and the integration is complete.

Support:
If **Further Assistance** is required, please follow the instructions on auto$, then proceed to log a support case or chat 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.

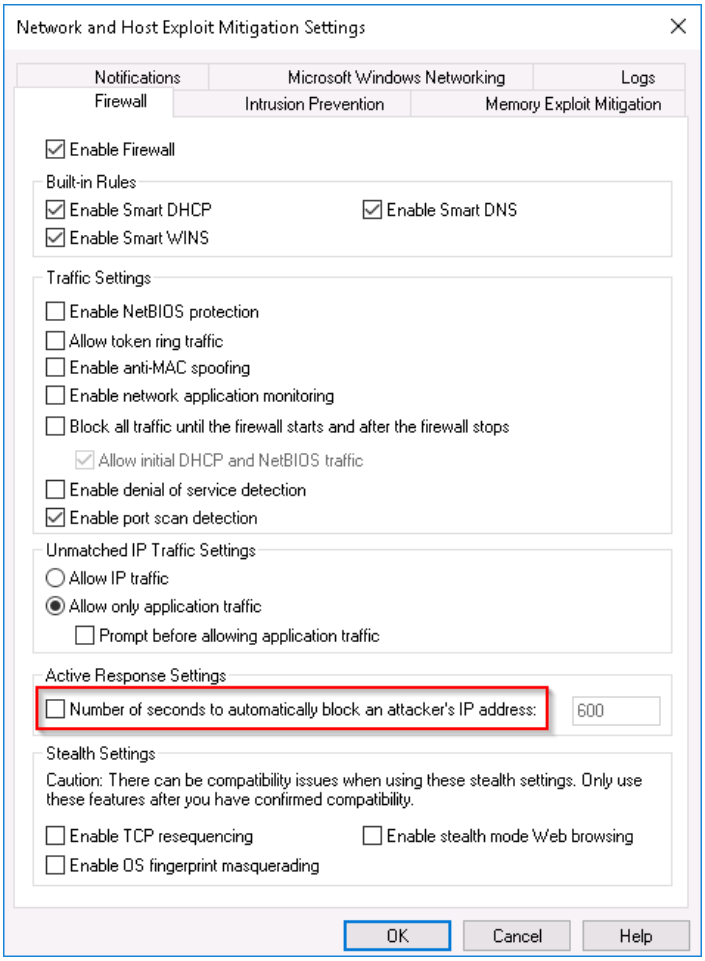
Update your SEP settings as follows to ensure optimal functioning of MetaDefender Core:

1. Exclude the folder where MetaDefender Core is installed from the SEP real-time protection:

- Add security risk exception **> Folder > C:\Program Files\OPSWAT**

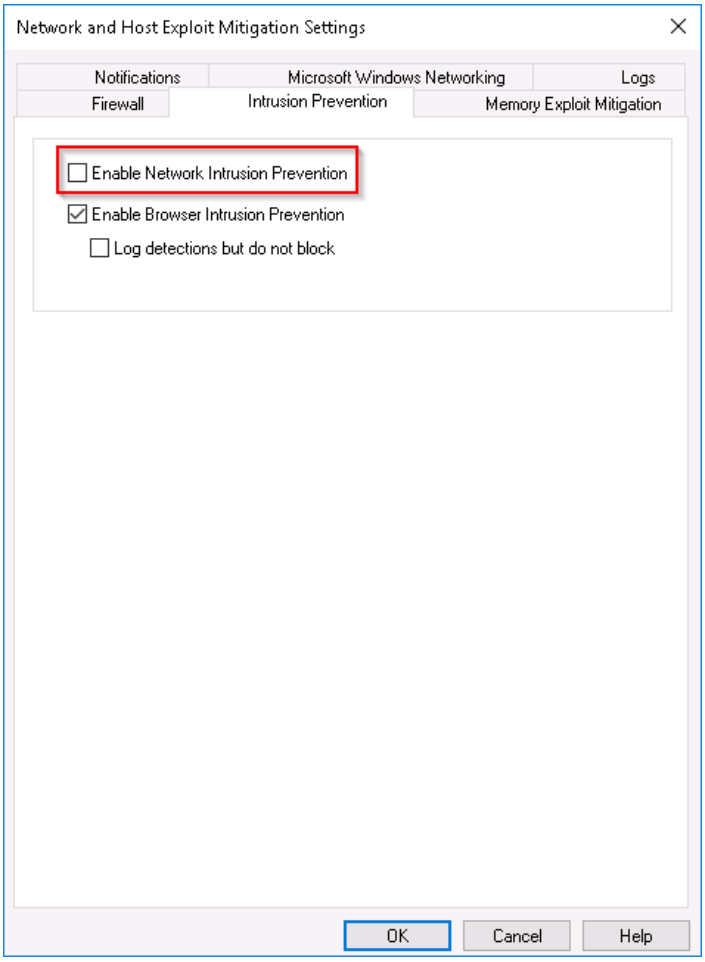
2. Go to **Network and Host Exploit Mitigation Settings > Firewall**:

- 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 auto$, before creating a support case or chatting with our support engineer.

How can I find a sanitized file scanned with MetaDefender Core v5?

Check Your Version:
This article applies to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

Once a file is scanned by MetaDefender Core and then sanitized, it can be downloaded from the following link:

URL javascript

```
http://<MetaDefender>:8008/file/converted/<dataid>?apikey=<apikeyset>
```

- <MetaDefender> needs to be set to your MetaDefender I.P. location or name
- <dataid> needs to be inserted as per data-id of the file
- <apikey> Only required if REST API keys have been defined

Support:
If **Further Assistance** is required then please proceed to create a support case or chat 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 https://regex101.com/ 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:

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

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

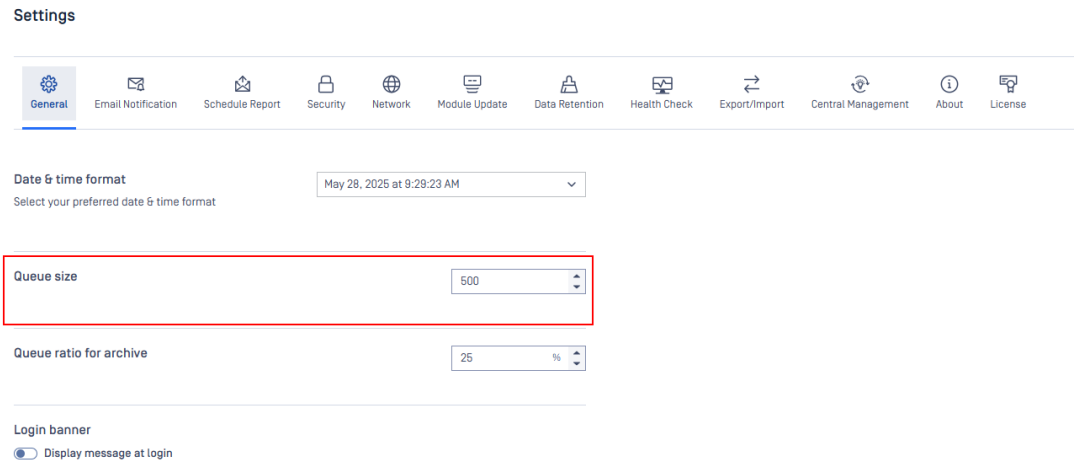
HTTP status code: **500**

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.

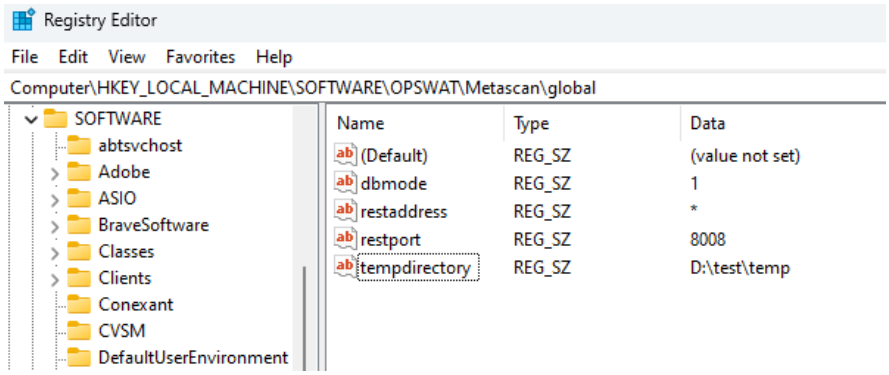
It can be changed by configuring your MetaDefender Core settings as below:

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.

The only reason to increase the scaling up performance is if you have 32-48 CPU cores and require your file to be processed very fast.

In order to increase the scaling up performance, there are two common approaches, as follows:

Approach A

When the MetaDefender Core queue is full of incoming scan requests, they will be denied until there is a free slot. To change the queue size, you can use the following API call:

API json

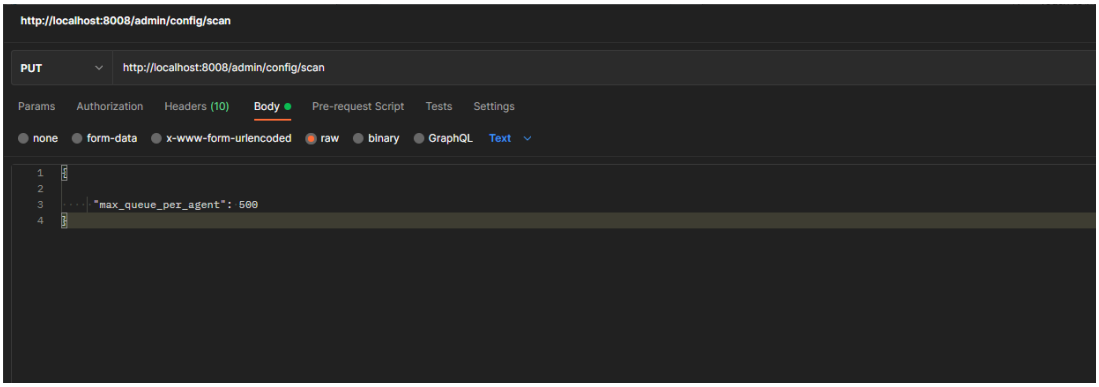
```
PUT /admin/config/scan
```

With the following body:

json

```
{
     "max_queue_per_agent": 500
}
```

Postman:



API key required
Note that you will need the API key from the MetaDefender Core deployment to perform this call.

To learn more about configuring queue size, How can I configure the maximum queue size in MetaDefender Core v5 ?.

Approach B

Another way to better utilize your CPU is to increase the parallel number of scans running per engine.

This way you can tell the Core how many scans should run simultaneously per engine. This applies to engines supporting multi-threaded scan.

To change this, you have to use the **parallelcount** setting key, explore the details at MetaDefender Configuration.

Pro Tip:
Once you have configured the parallel scan count, be sure to restart the MetaDefender Core service for the changes to take effect.

Support:
If you have any issues **Increasing the Scaling Up Performance** on MetaDefender Core, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

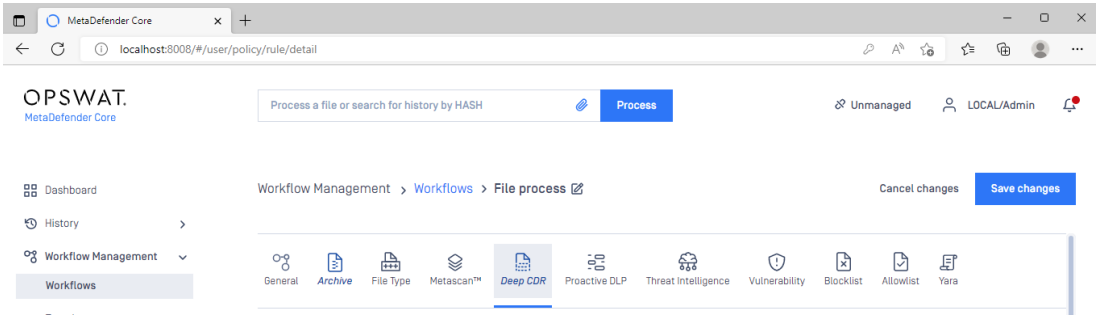
How can I change MetaDefender Core Deep CDR's timeout settings?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

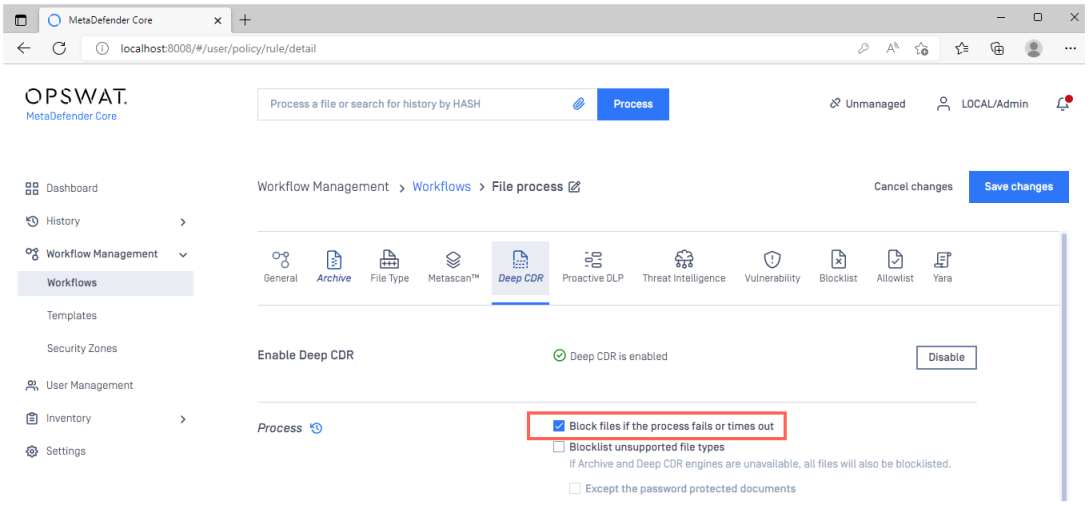
Pro Tip:
When processing complex files with MetaDefender Core's Deep CDR engine, it's not uncommon for the sanitization to fail due to exceeded timeout settings, particularly in the case of larger, encrypted archive files that take time to extract, process and sanitize.

However, Deep CDR's timeout setting can be easily configured to allow time for processing your anticipated datasets, so that no such failures occur.

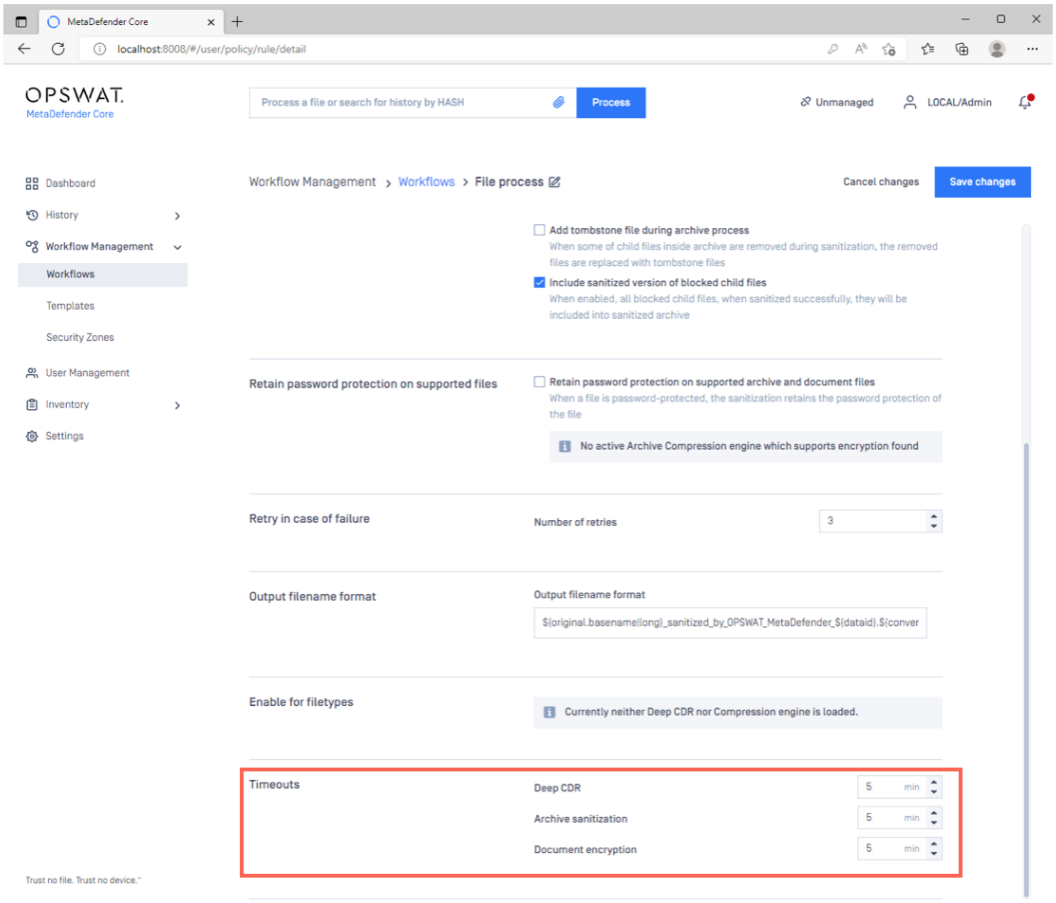
1. To access these settings, please go to your **MetaDefender Core Management Console>Workflow Management>{Your Chosen Workflow)>Deep CDR**, as illustrated in the screenshot below.



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 auto$, 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.

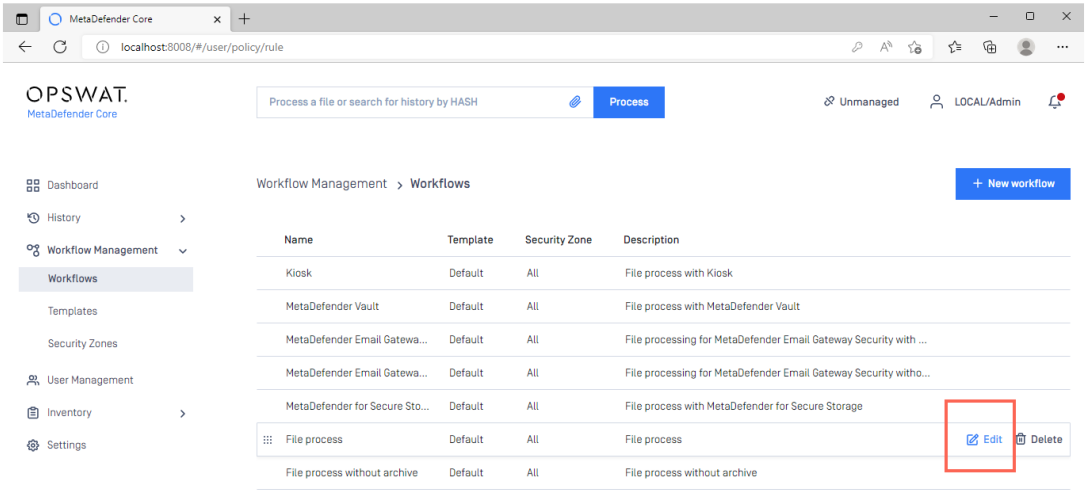
Pure archive formats	*Zip, 7z, JAR, RAR, RAR5, TAR, ISO, CAB, ARJ, LHA, LZH, RPM, DEB, LZMA, WIM, DMG, XAR, MSM, SFX, XZ, VDI, VHD, CPIO, HFS, APK, GZ, MSI, TAZ, TGZ, TBZ, BZ2, VIB, AR, ALZ, TSE, TSEC, TSEZ, ACE, PST, OST, OFT, IPA, JMOD, MSIX, APPX, EGG, DLL,* and *PGP* (with password only)
Self-extracting 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.

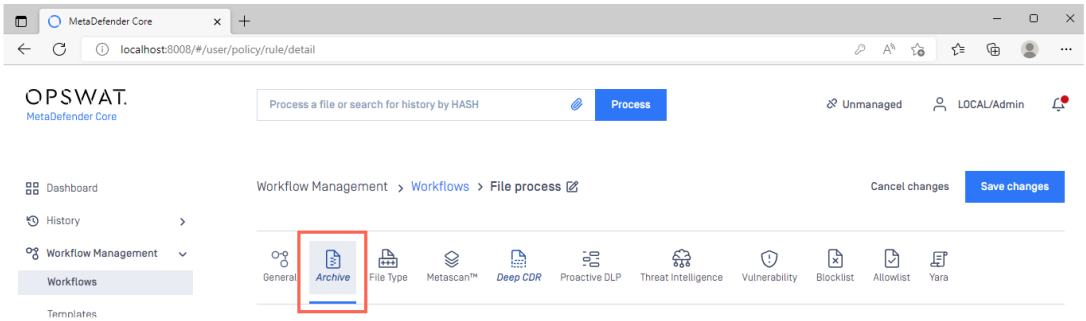
Unfortunately, there's no quick-fix in terms of estimating the number of child-files or folders within a given archive, so it is recommended that you test and adjust your settings against a file that is a fair, general representation of the larger files you're expecting to scan.

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.

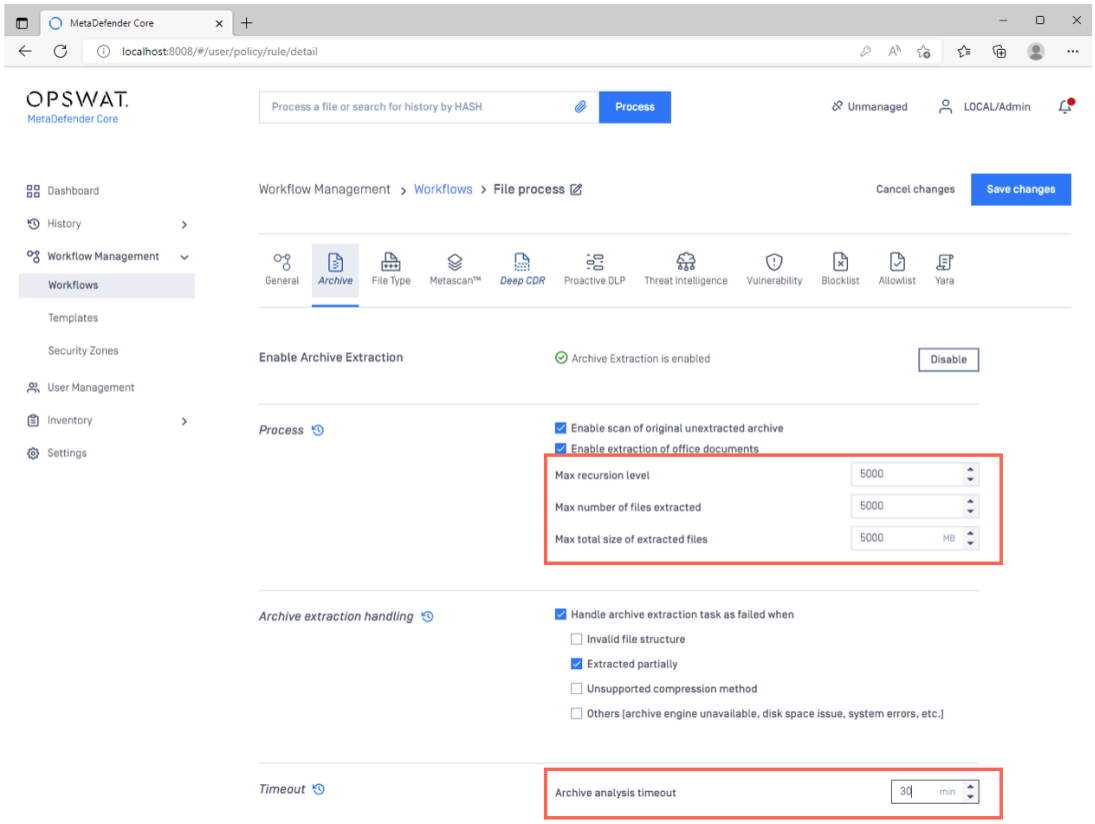


2. Inside your chosen workflow, click on the **Archive** tab, as shown below, where you can configure a range of archive-related rules.



3. Now, adjust the following values to a level that lets your test-file scan to completion. Then allow some room for error:
    a. **Max recursion level**
    b. **Max number of files extracted**
    c. **Max total size of extracted files**
    d. **Archive analysis timeout**

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 auto$, 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.

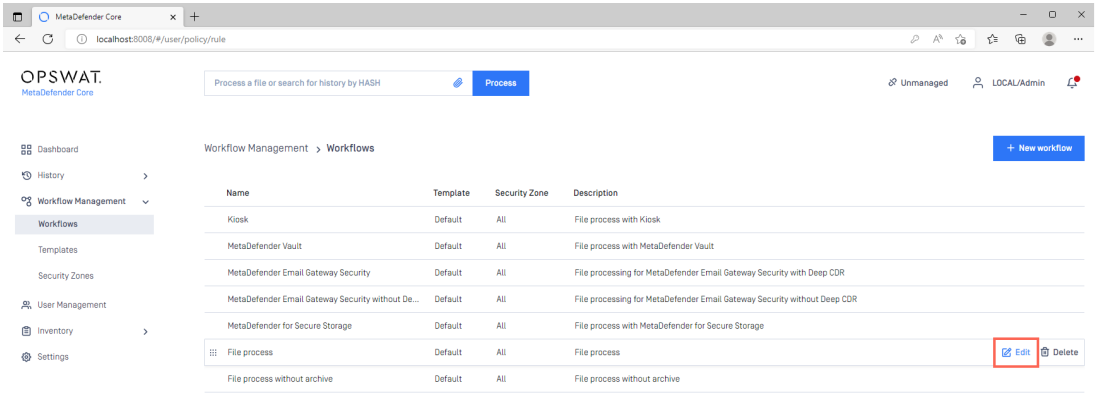
We advise customers to rescan the password protected file via both Core UI and REST API

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:

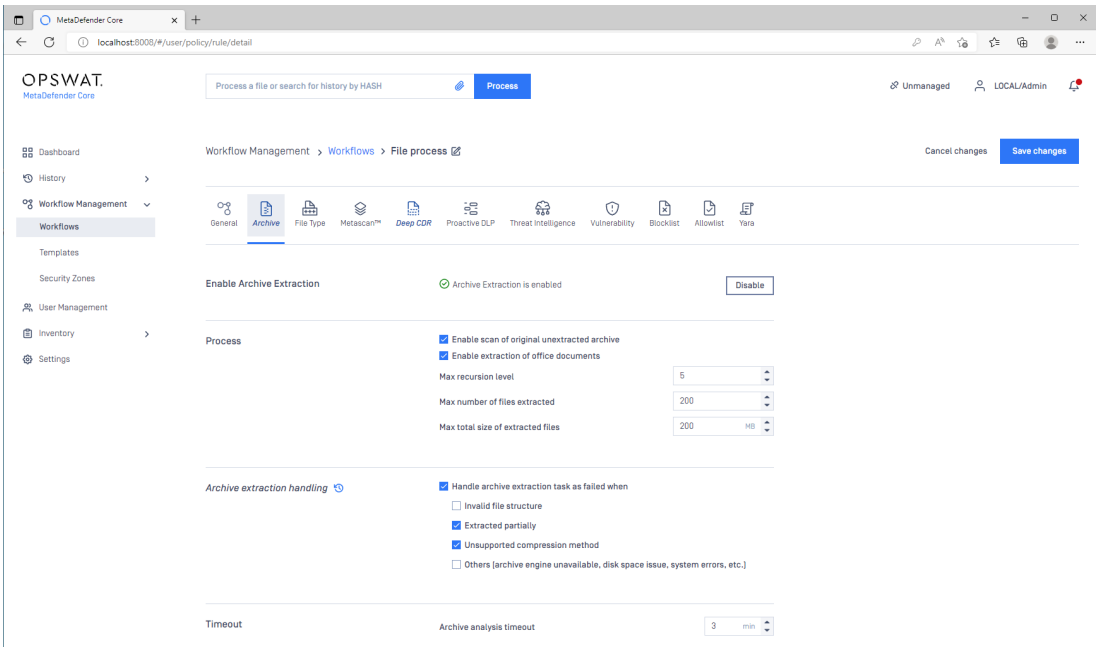
1. Go to **MetaDefender Core Management Console>Workflow Management>Workflow**s, and click **Edit** to adjust the relevant workflow rules, as illustrated below.



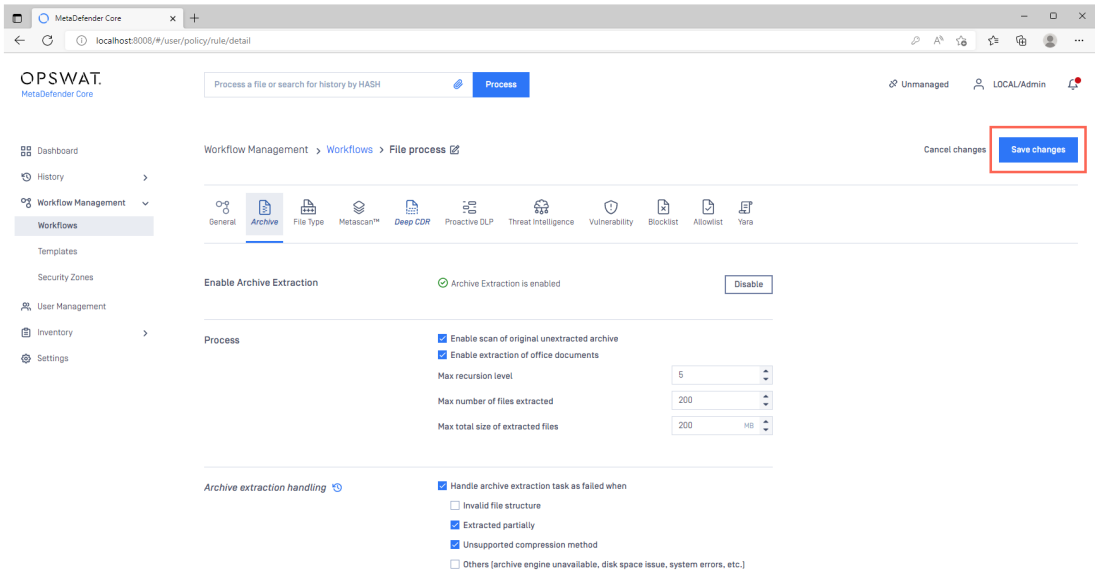
2. Once in your chosen workflow, go to the **Archive** tab.
3. Here, you should ensure that the following actions are enabled:
    a. Archive extraction
    b. Scan of original unextracted archive
    c. Extraction of office documents

You can also ensure that tasks are handled as **Failed** when they are/in the case of:

- Extracted partially
- Unsupported compression method.

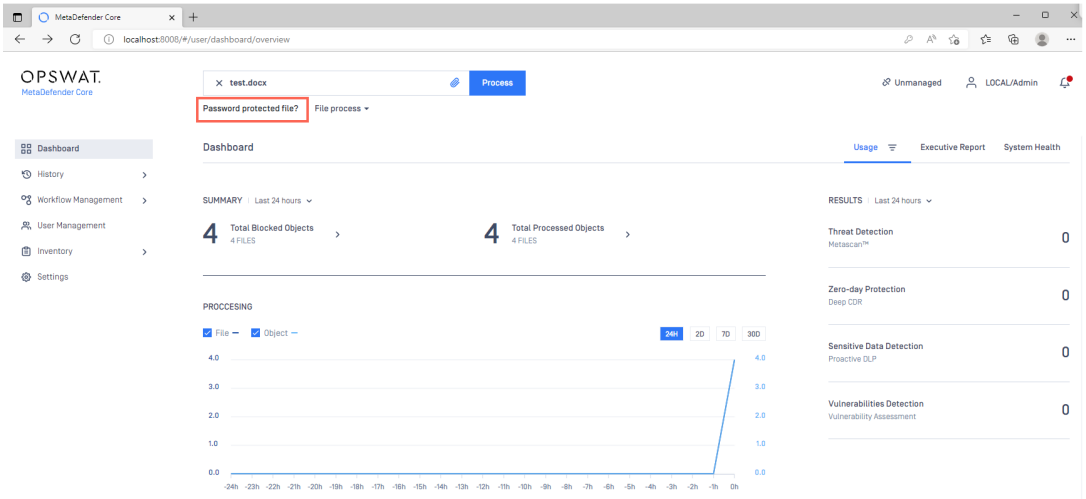


4. Finally, click **Save changes** to apply your settings.

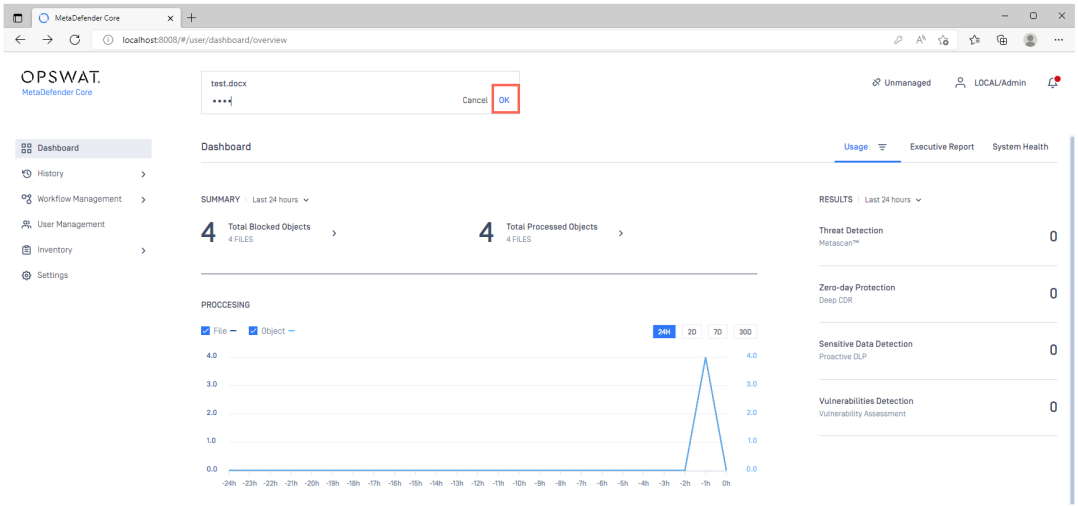


To unlock supported password-protected archives for scanning (once you have configured your chosen workflow), please do the following:

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**.



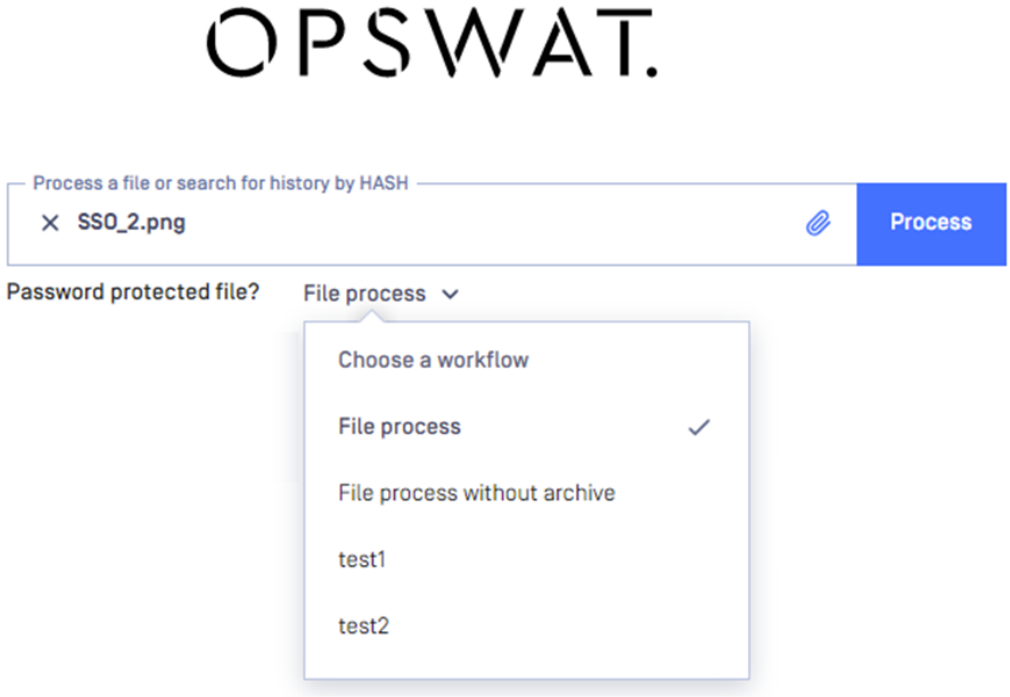
4. Your file is now ready for processing and, provided the encrypted file type is supported and your archive settings are configured, the file should no longer return a **Blocked** result. (if no other restriction settings are in place on the workflow)

Support:
If you have followed the instructions above but are still **Unable To Unblock Password Protected Archives**, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

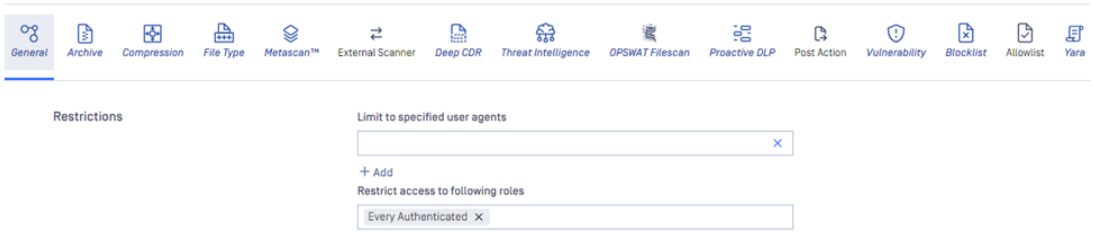
How to disable WEB UI file scan without user authentication?

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:

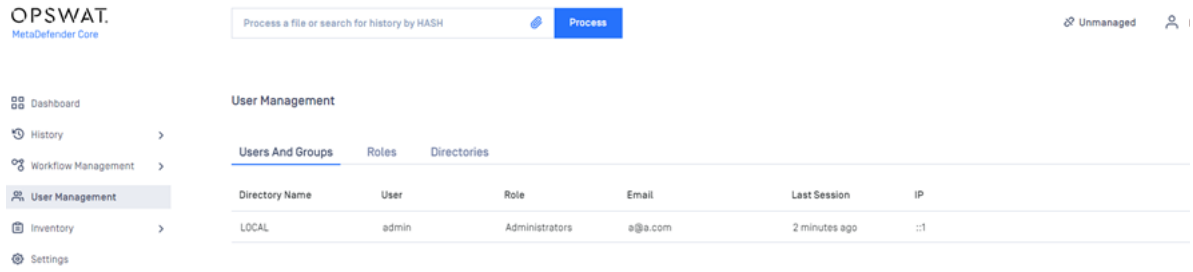
- For more information on using the MetaDefender Core Blocklist/Allowlist feature, refer to the following documentation: How to use MetaDefender Core V5 Blocklist/Allowlist feature?
- To learn about other workflow configuration options in MetaDefender Core, including the Blocklist and Allowlist, consult 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 auto$, before creating a support case or chatting with our support engineer.

How to generate an API key on the Core deployment?

In order to be able to generate an API key on the Core console the following steps are necessary:

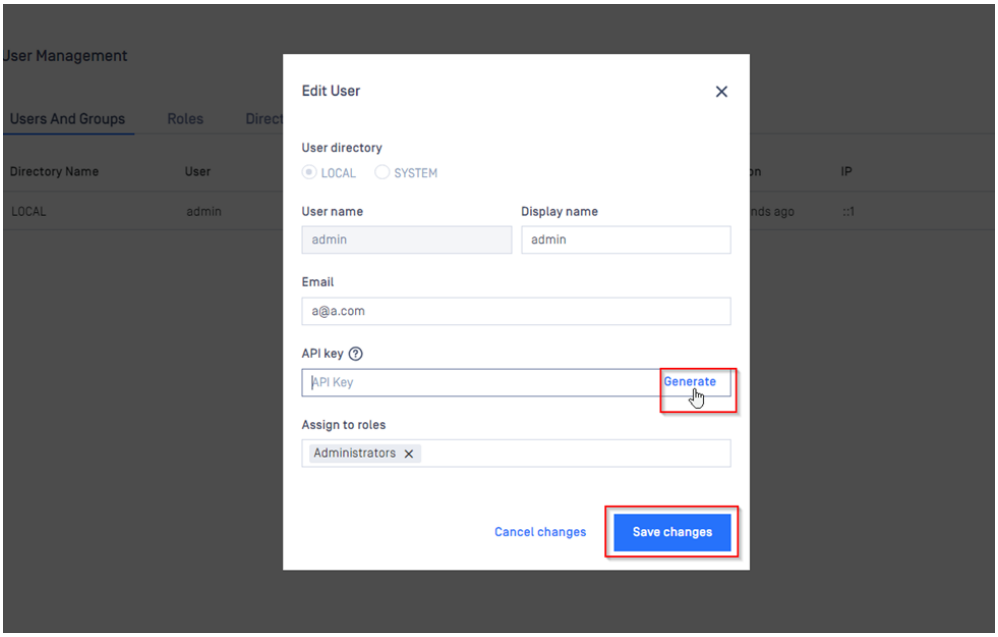
1. Navigate to the core console (Console\User Management\Users and Groups)



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.


User Management
Users And Groups
Roles
Directory Name
User
LOCAL
admin
Edit User
User directory
LOCAL
SYSTEM
User name
admin
Display name
admin
Email
a@a.com
API key
API Key
Generate
Assign to roles
Administrators
Cancel changes
Save changes

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 auto$, before logging a support case or chat with our support engineer.

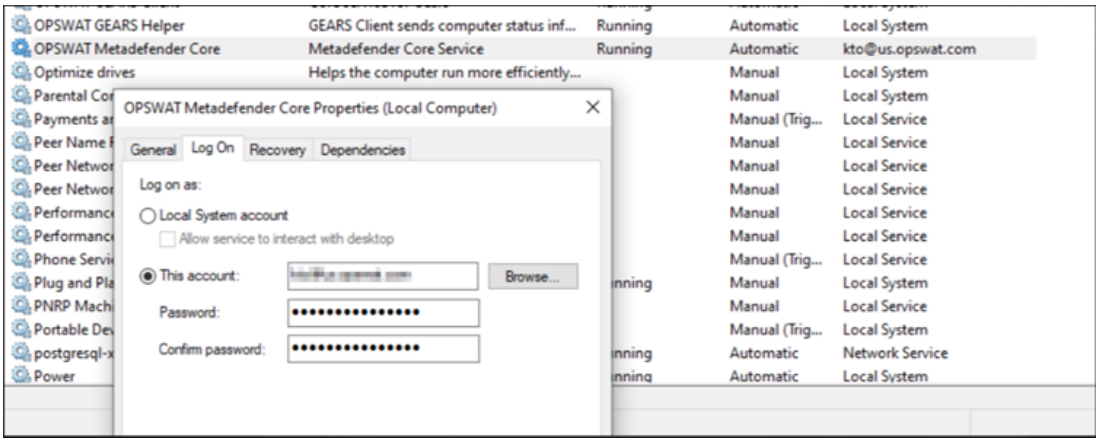
What are the permissions on the shared folder for the temp directory?

Check Your Version
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

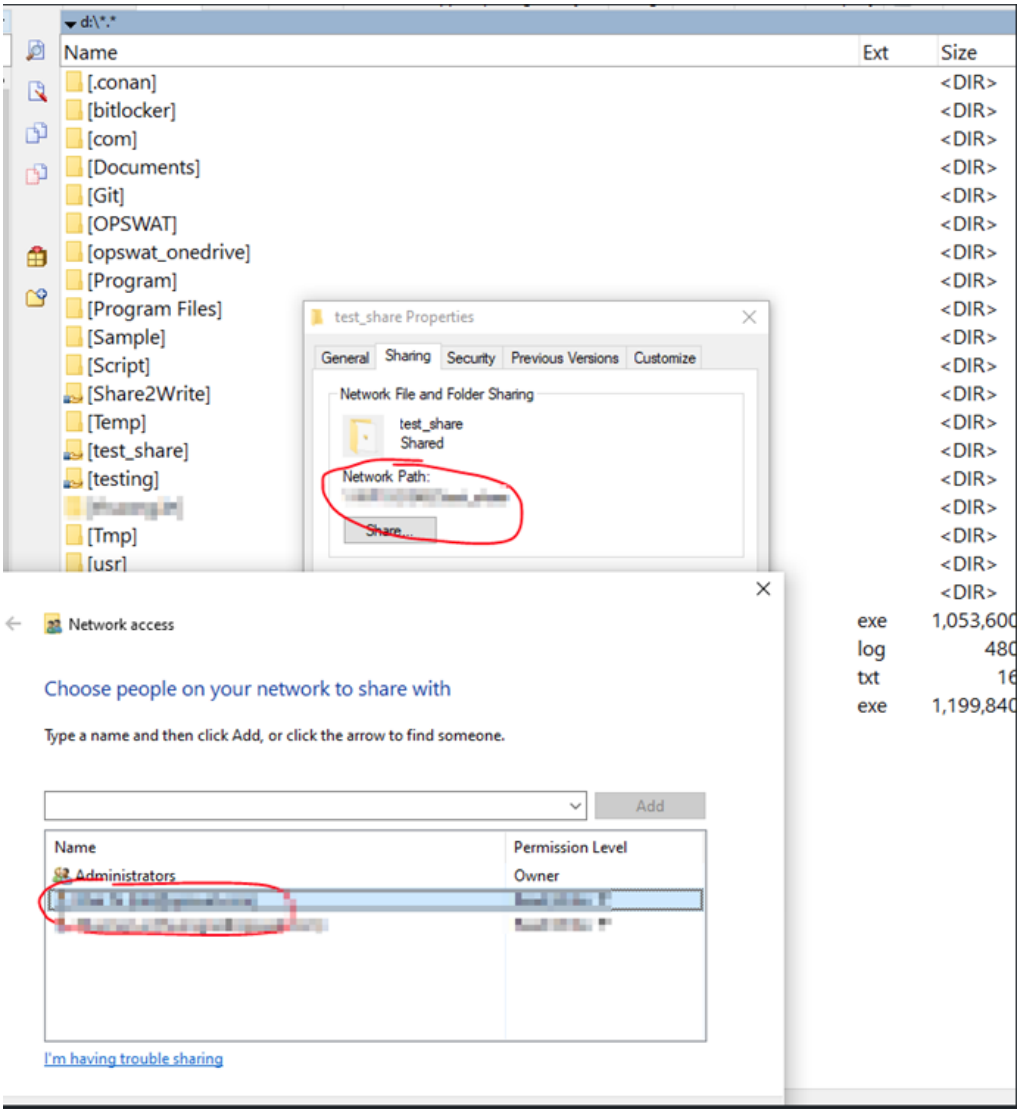
Kindly adhere to the following steps to ensure the appropriate permissions are granted for the Shared folder:

- 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:

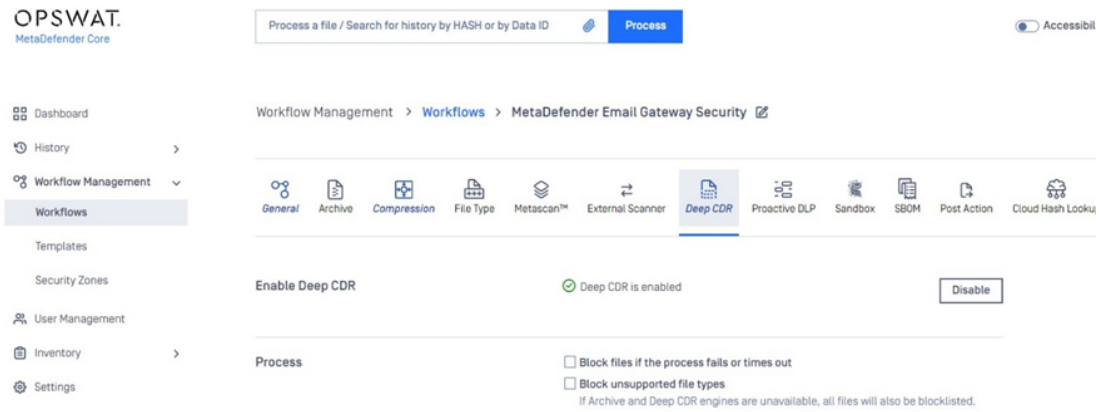
`’`

Original Email: Please find the summary of today's meeting

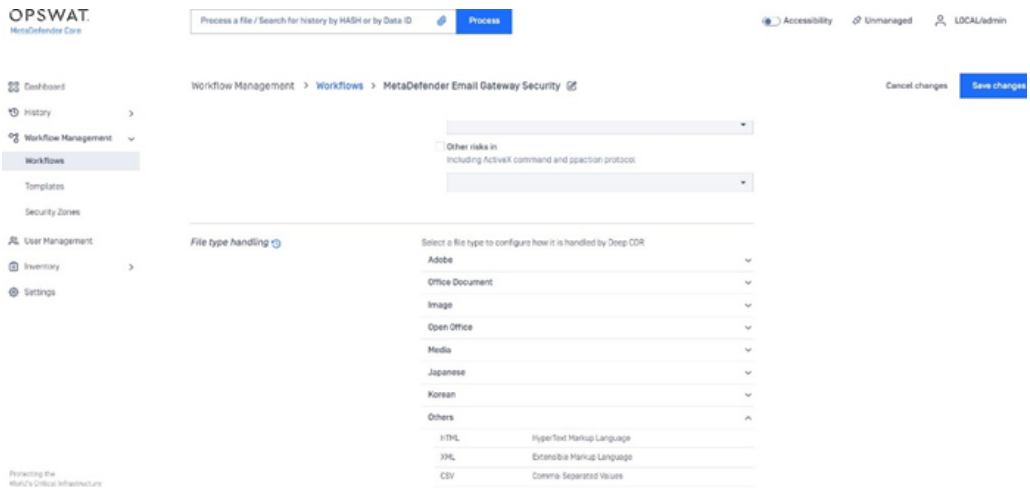
Processed email: Please find the summary of today `’` s meeting

> **Note**
> The fix is available starting with CDR engine version 6.8.0

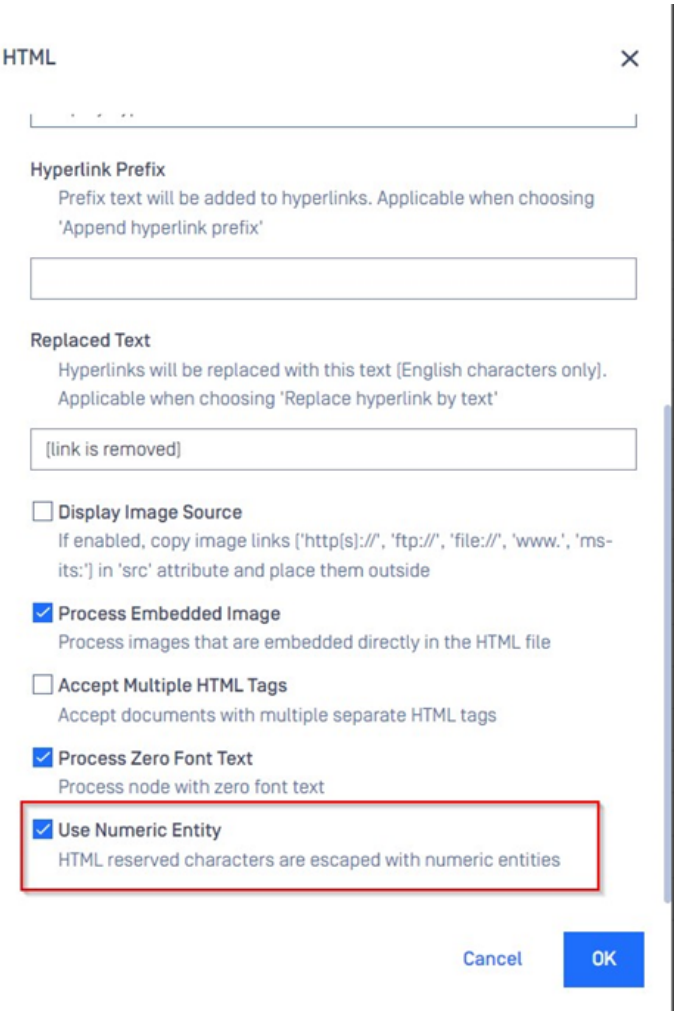
1. Navigate to the MD Core Web UI.
2. Locate and choose the workflow associated with the EGS instance.
3. Access the DeepCDR tab and proceed to the Advanced section.



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.



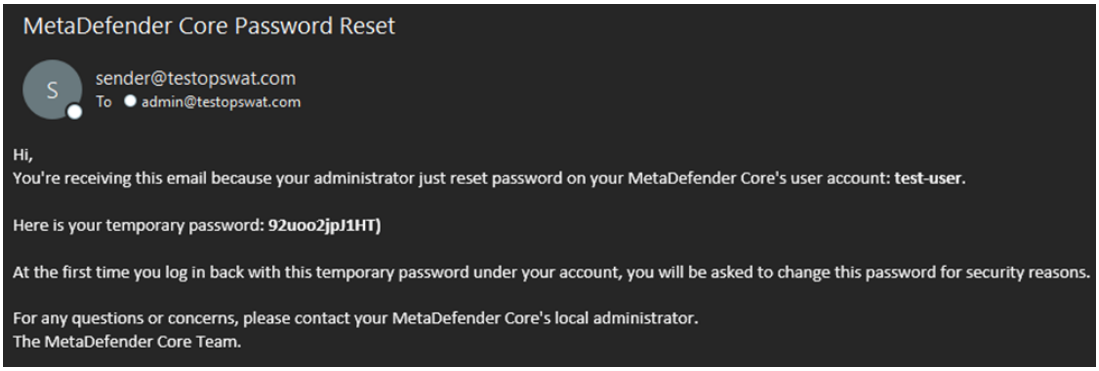
6. Confirm your changes by clicking on the "OK" button, followed by "Save Changes."

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.

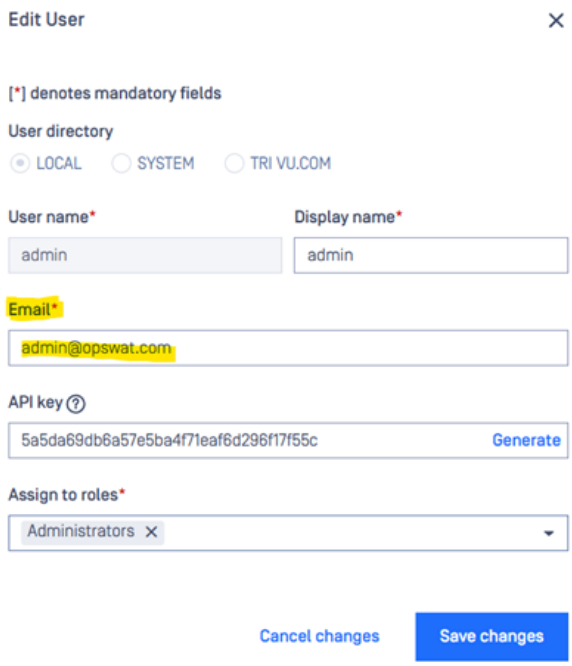
How to reset password for users?

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:

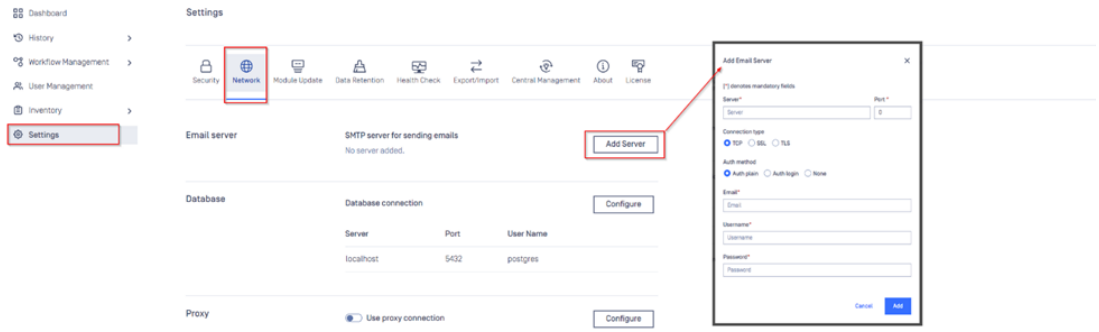


The user's Email Address should also be configured correctly for the SMTP Server to send the email to the recipient successfully.



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.

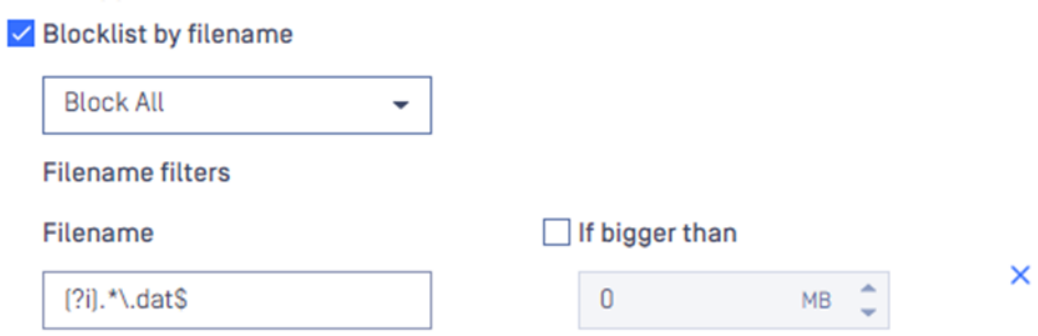
How to Block/Allow Custom Extensions using RegEx?

> **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
$	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.

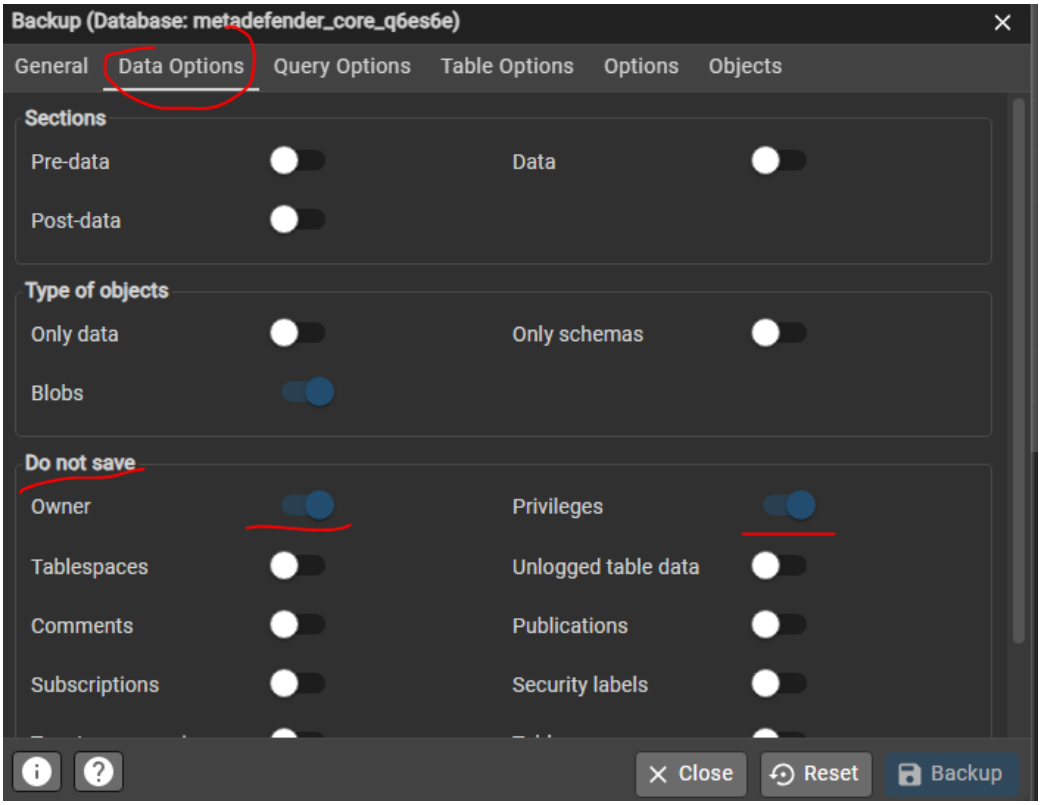
How to migrate MetaDefender Core from PostgreSQL local to PostgreSQL remote?

> **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 auto$, before logging a support case or chatting with our support engineer.

How to change Core Web Management Interface Address Port?

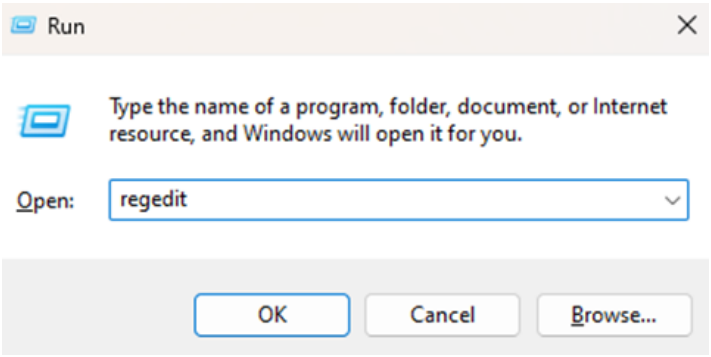
By default, MetaDefender Core's Web Management Interface is available via Port 8008.

This is customizable via Registry Editor in Windows and /etc/ometascan/ometascan.conf file in Linux.

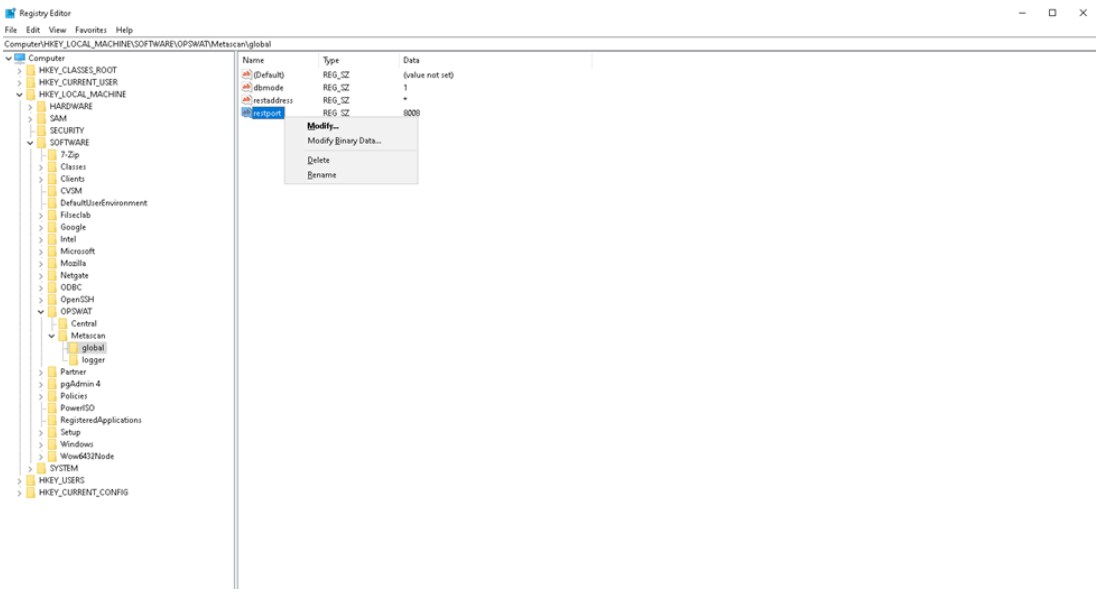
To change the port, you can follow the steps below:

Windows:

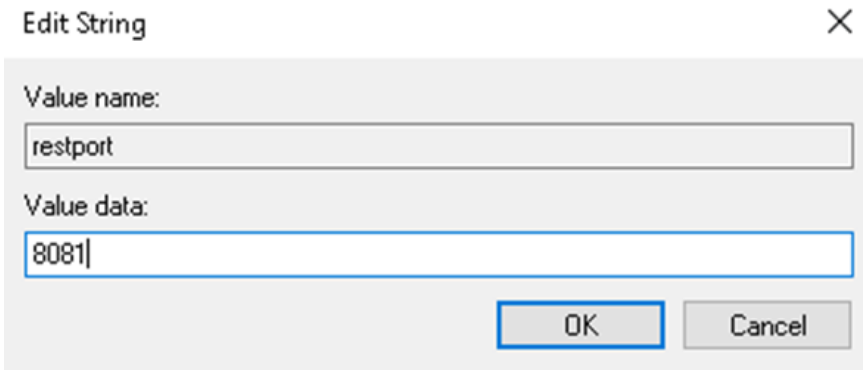
1. Press **Win+R**, type **regedit** and press Enter to open **Registry Editor**:



2. Navigate to **HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global**, select **restport** Key and select **Modify...**



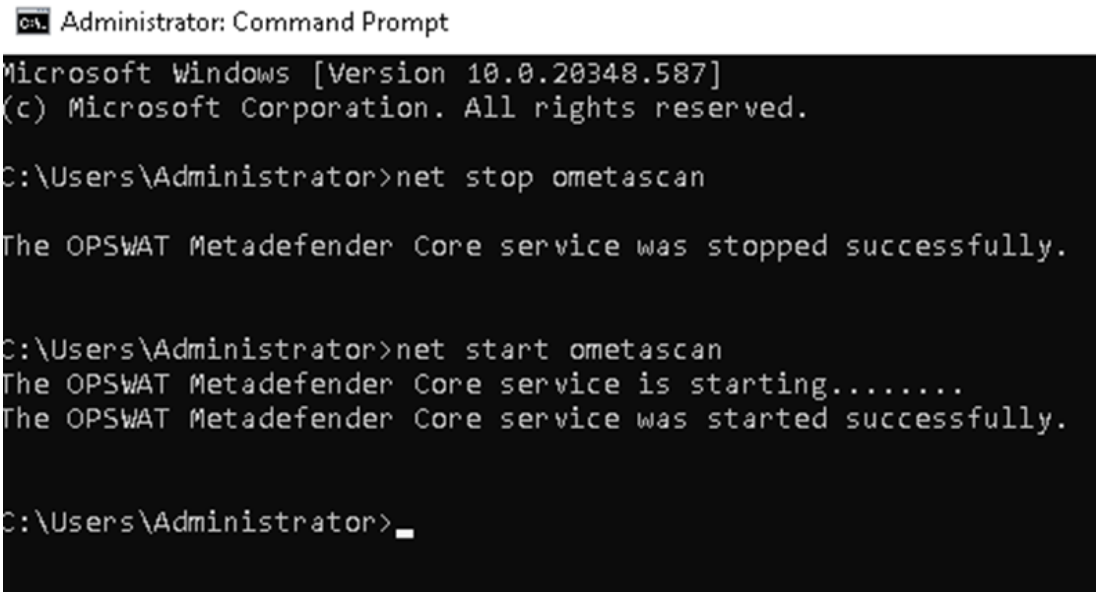
3. Enter the desired port and click the **OK** Button



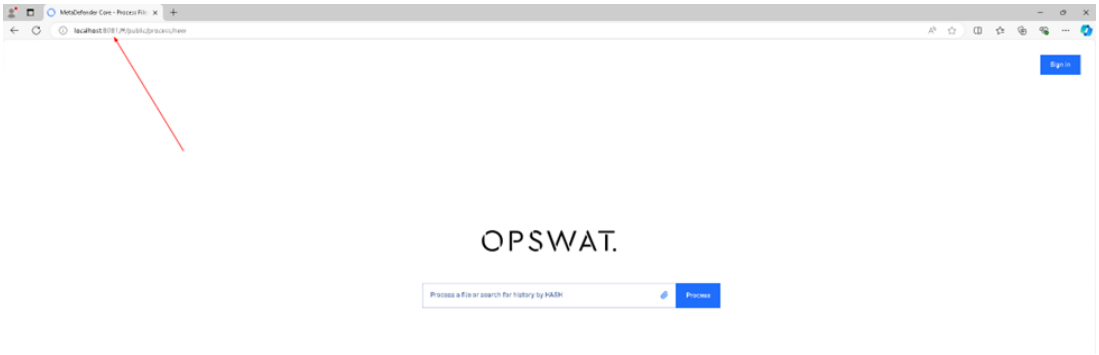
4. Open a **Command Prompt** as an **Administrator** and run the following commands to restart **MetaDefender Core Service**:

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:

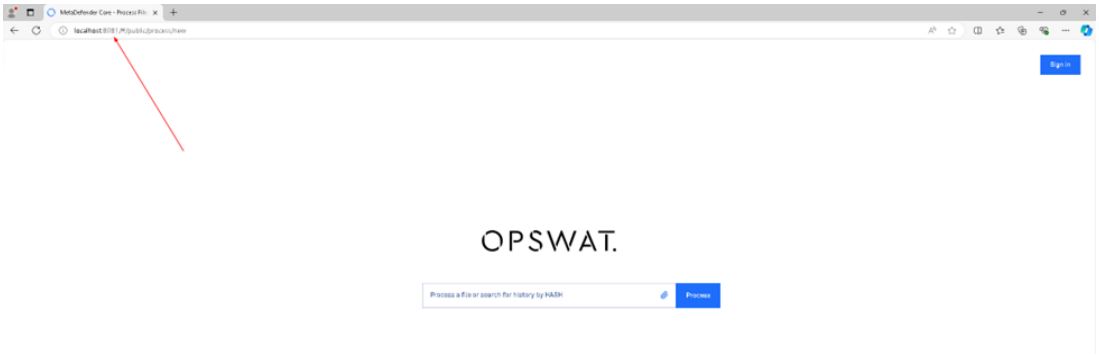


3. Save the changes and restart the **MetaDefender Core Service** using the following command:

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.)

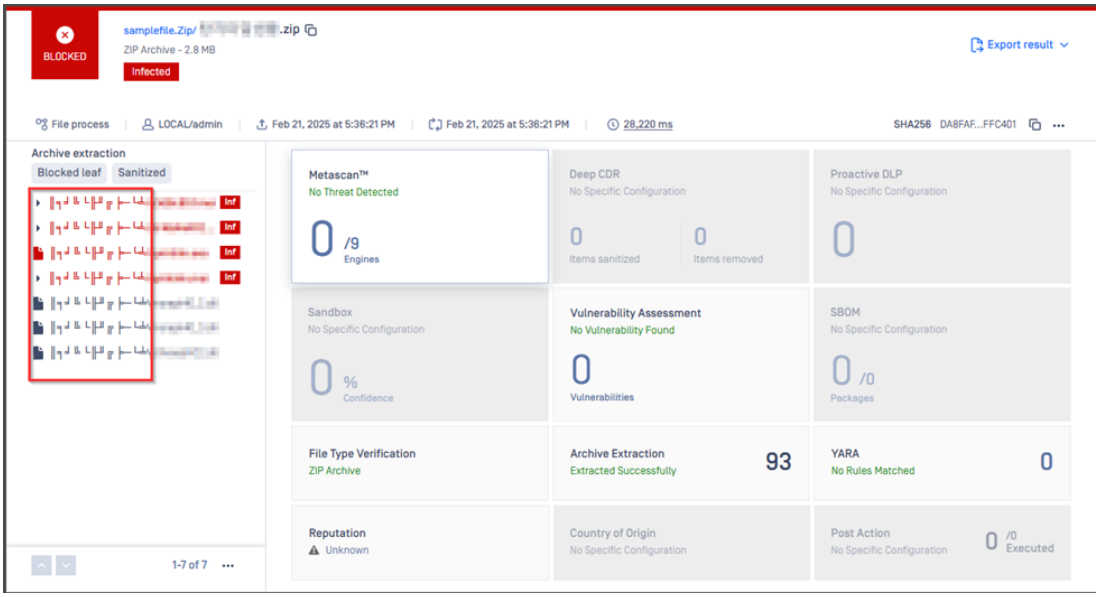
To change the temporary folder location, please refer to this article for more information How can the Temp folder be changed?

> **Support:**
> If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

Why Japanese characters are not recognized or displayed correctly in archives?

Warning
This knowledge base article provides details about MetaDefender Core functionality with specific **SHIFT-JIS** charset for ZIP, TAR and CAB archives.

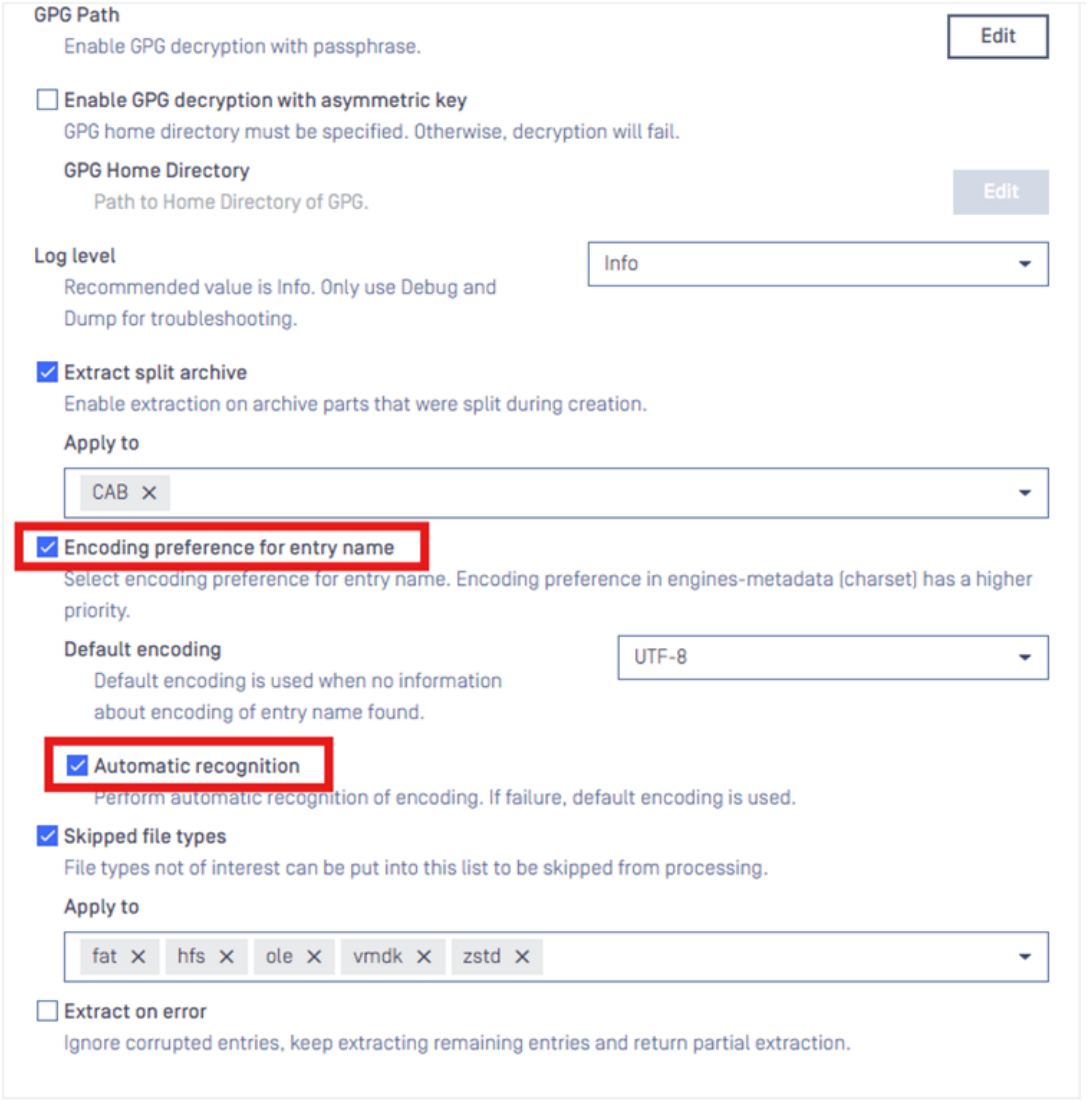
If you have issues with displaying specific characters in MetaDefender Core of child files inside an archive, as shown below:



Please follow the instructions below to resolve the issue:

- Login to MetaDefender Core UI
- Go to Inventory
- Go to Modules
- Go to Utilities
- Go to Archive Extraction
- Enable the option “Encoding preference for entry name”
- Enable the option “Automatic Recognition”

Inventory > Modules > Utilities > Archive Extraction



Support:
If **Further Assistance** is required, please proceed to log 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
subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-rpms
dnf makecache
dnf install -y 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
```

Support:
If you have followed the steps above but are still having trouble installing pre-built **libgdiplus**, please follow these instructions on auto$, 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?

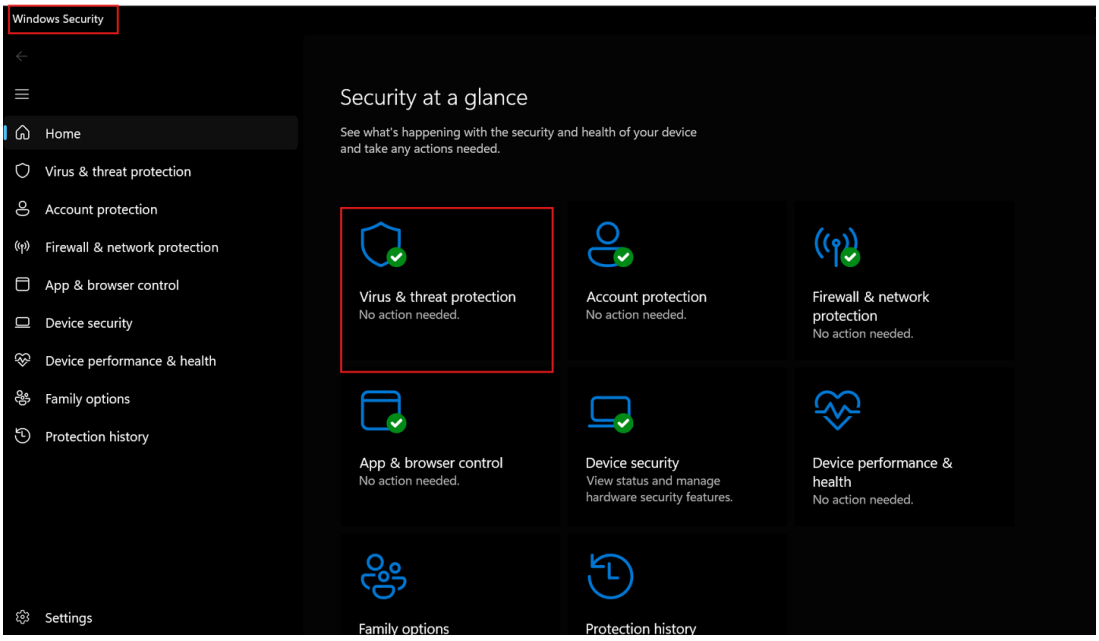
When scanning sample malware files such as the commonly used EICAR from Download Anti Malware Testfile - EICAR it is important to exclude the OPSWAT folder from **all** existing Anti-Virus software programs on the machine. Failure to exclude the OPSWAT folder can lead to “Not Scanned” results, as the 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”.

Please Note:
Common Anti-Virus softwares to perform exclusions, including but not limited to: McAfee, Norton, Bitdefender, Malwarebytes, SentinelOne

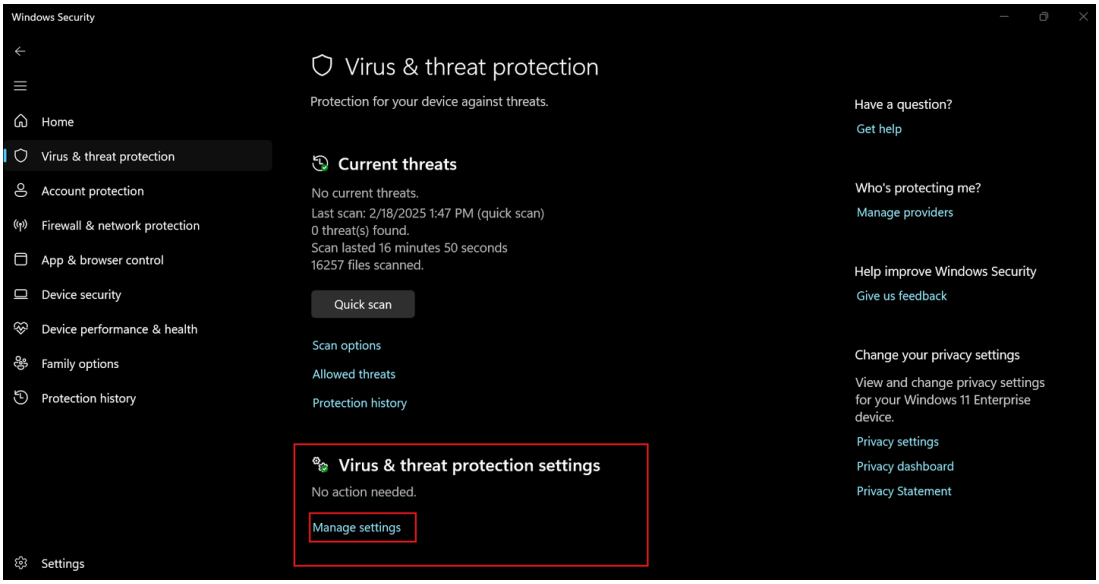
How to exclude on Windows Defender

Please ensure you have the proper permissions to add exclusions, such as administrator rights.

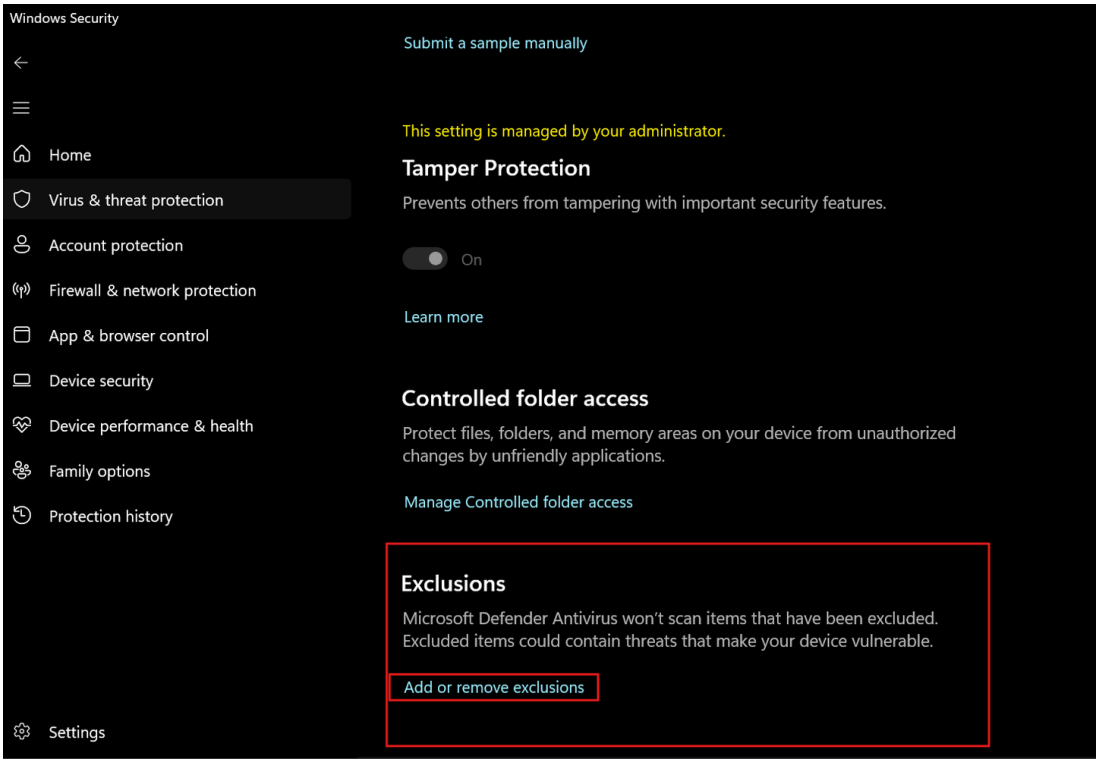
1. Navigate to “Windows Security” > “Virus & threat protection”



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`.

Please Note:
If you are still receiving “Not scanned” results, please also exclude the MetaDefender Core processes: **engineprocess.exe**, **engineprocess32.exe**, **ometascan.exe**, **postgres.exe** and **nginx.exe**.

Additionally you can exclude the MetaDefender Core Registry path:

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\global

For the following:

`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

none

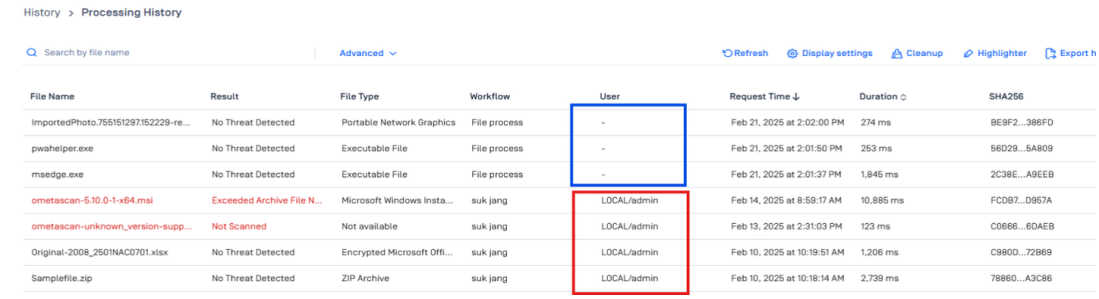
```
Computer\HKEY_CURRENT_USER\Software\OPSWAT\MetaDefender Core

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.



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	

Scan History exported from csv

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.

Support:
If **Further Assistance** is required, please proceed to create a support case or chat with our support engineer.

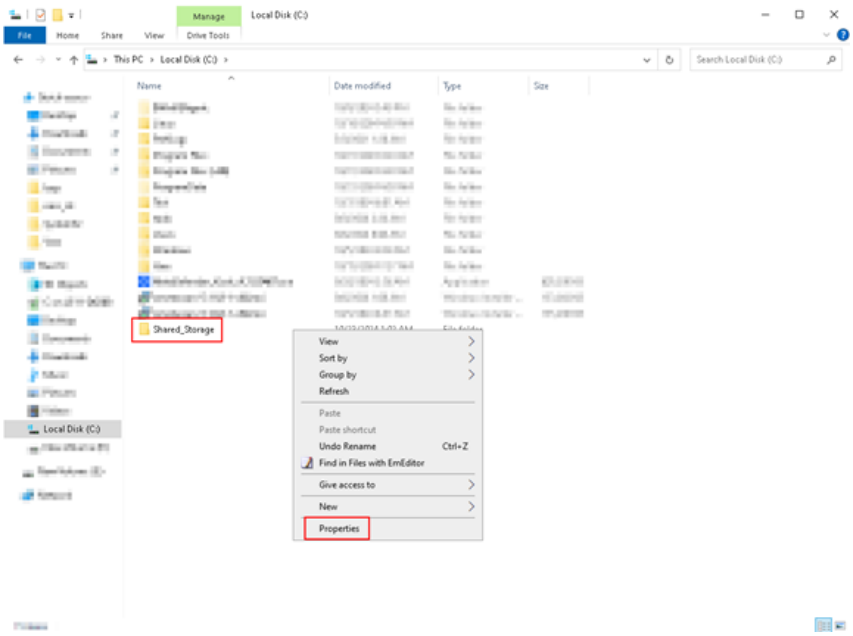
How to send MetaDefender Core logs to NAS (Network Attached Storage) Path?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

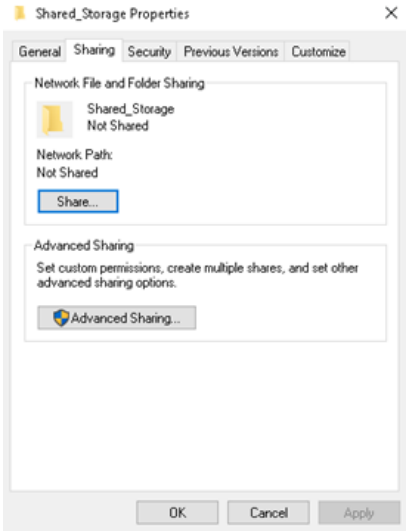
Creating a Network Attached Storage

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.

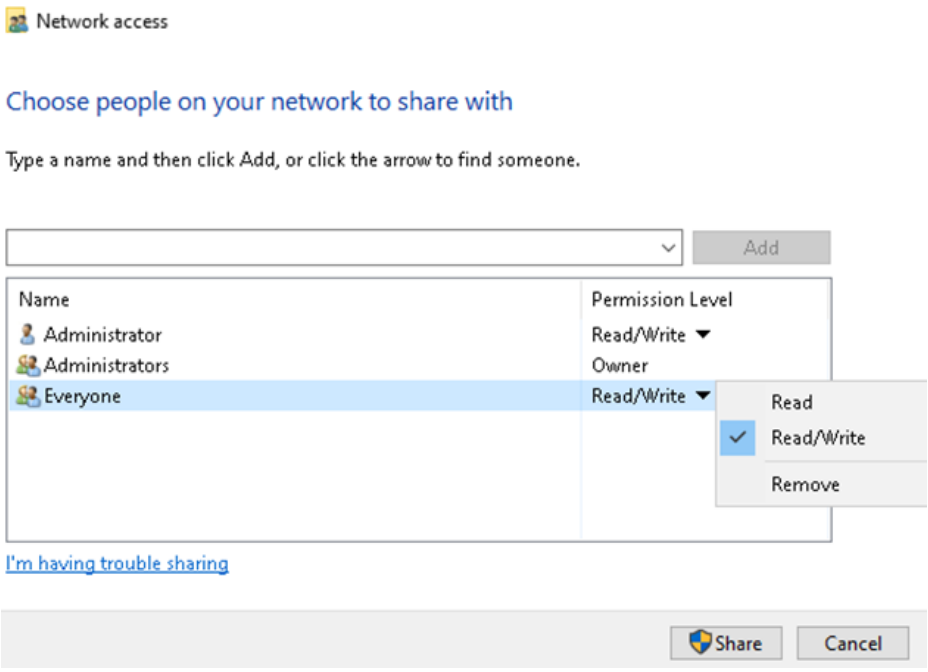
Step 1: Open the **File Explorer** and create a folder that is going to be shared.



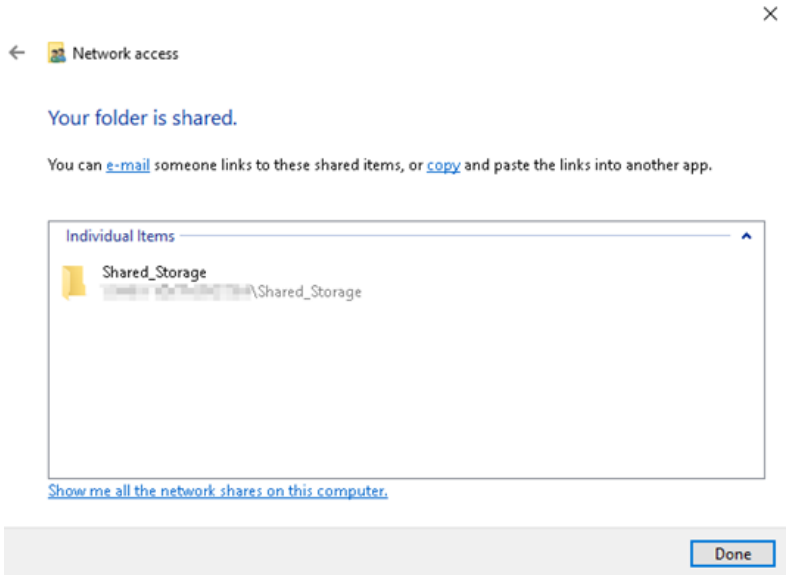
Step 2: *Right click* the created folder, open **Properties** and under **Sharing** tab click button **Share...**



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

bash

```
sudo mount -t cifs //windows_server_ip/Shared_Storage /core_logs -o
username=windows_user,password=windows_password,domain=windows_domain,dir_mode=07
77,file_mode=0777
```

Step 4: Modify /etc/ometascan/ometascan.conf file to send the logs to the desired directory.

a. Open the file:

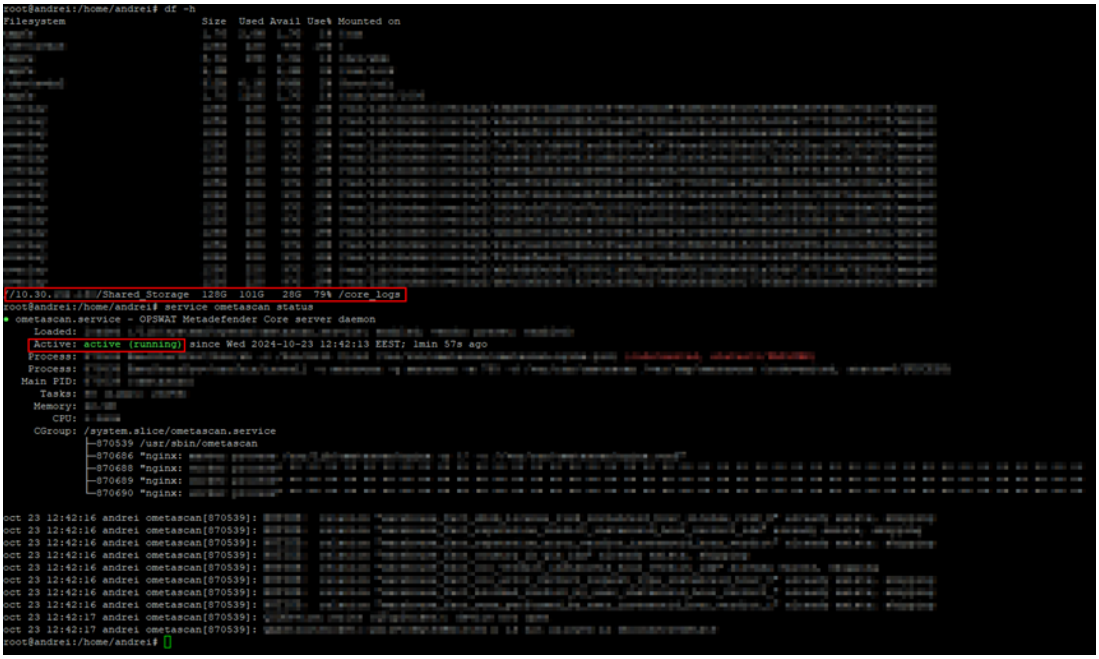
```
sudo vim /etc/ometascan/ometascan.conf
```

b. Change the logfile path to the configured one:

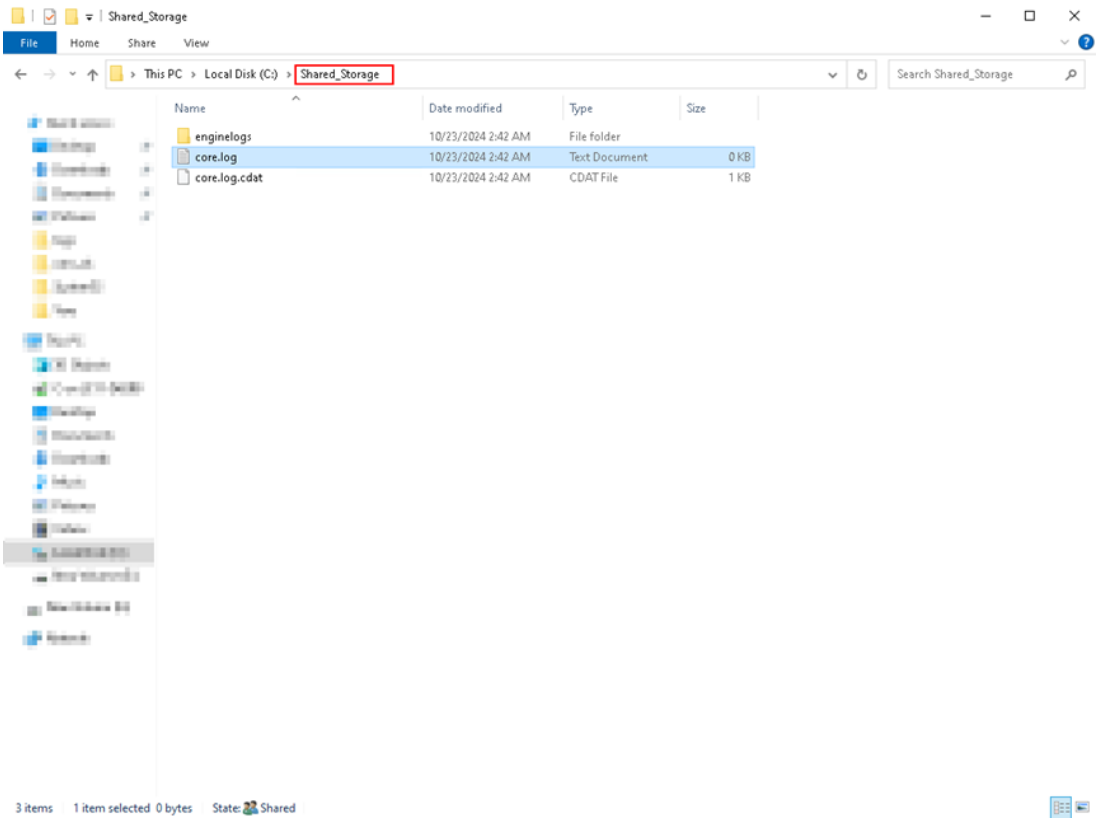
```
logfile=/core_logs/core.log
```

Step 5: Restart the MD Core Service:

```
service ometascan restart
```



Core will now send all the logs to the Network Attached Storage:

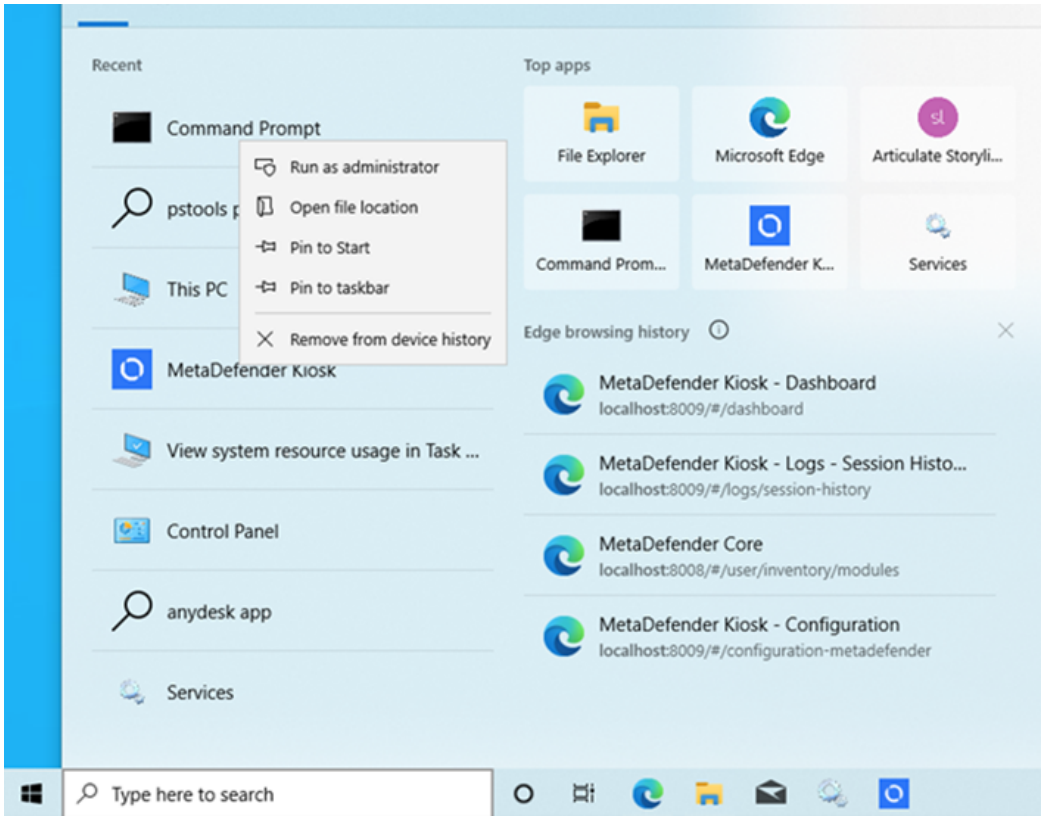


Windows:

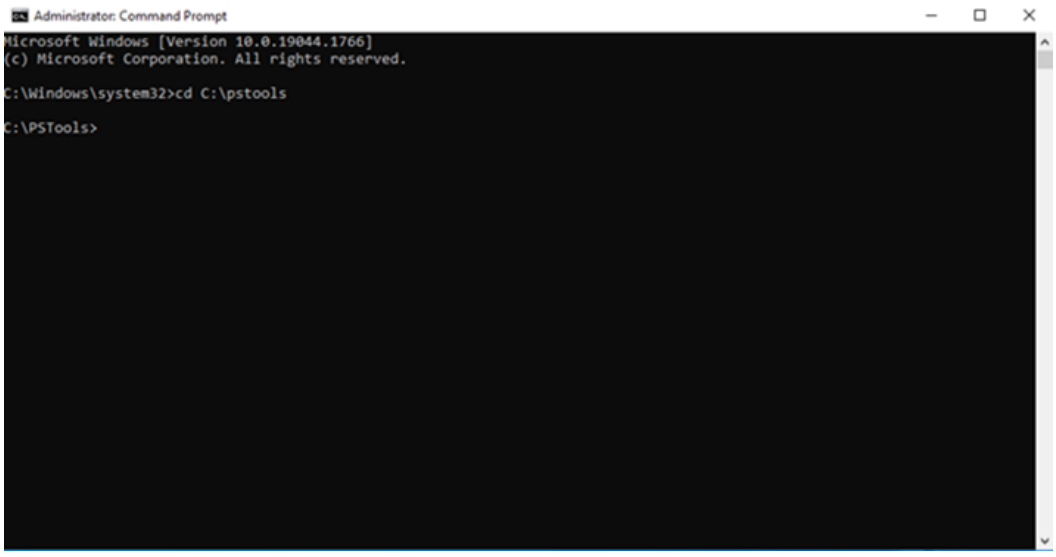
Step 1: In order to save logs to a mapped drive, the mapped drive must first be created under the **SYSTEM** account.

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.**



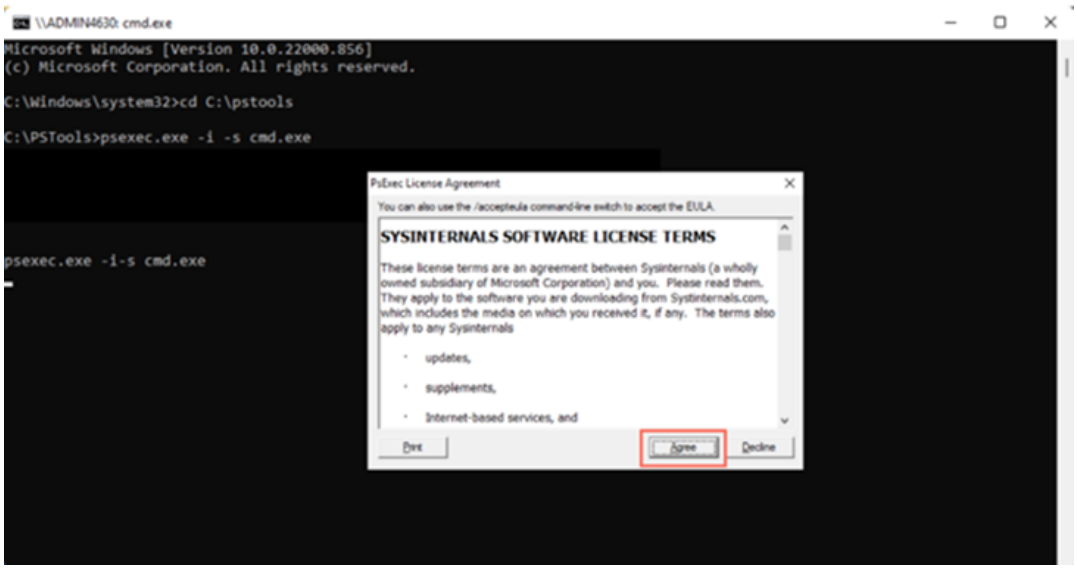
- In the command prompt, navigate to the directory where psexec.exe was copied (e.g. `cd C:\pstools`, as illustrated in the screenshot below).



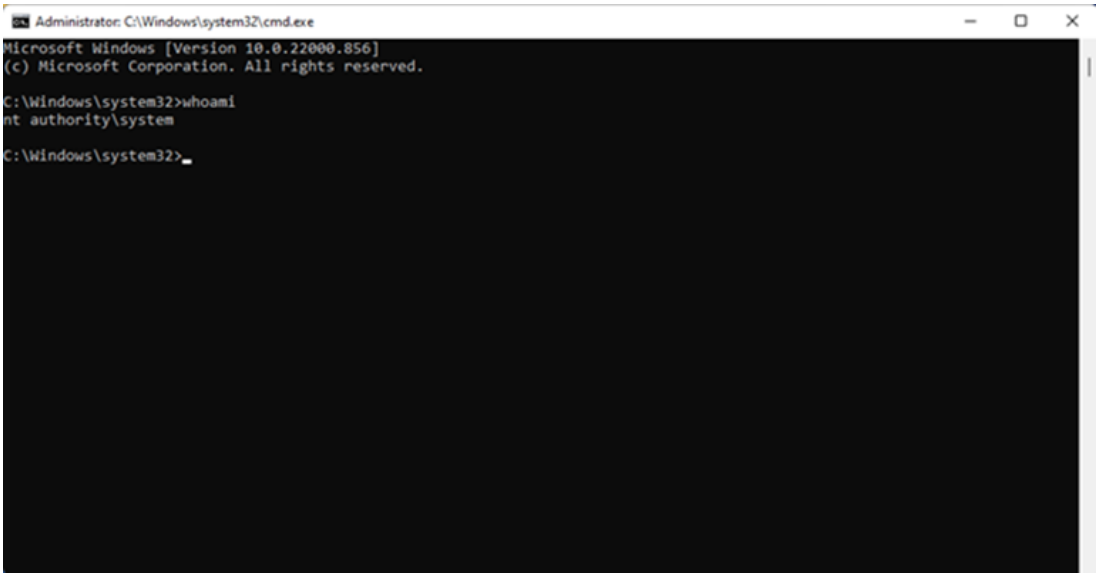
- Enter the following command `psexec.exe -i -s cmd.exe`, as illustrated in the screenshot below:



- If the Sysinternals pop-up appears, click **Agree**, as illustrated in the screenshot below.



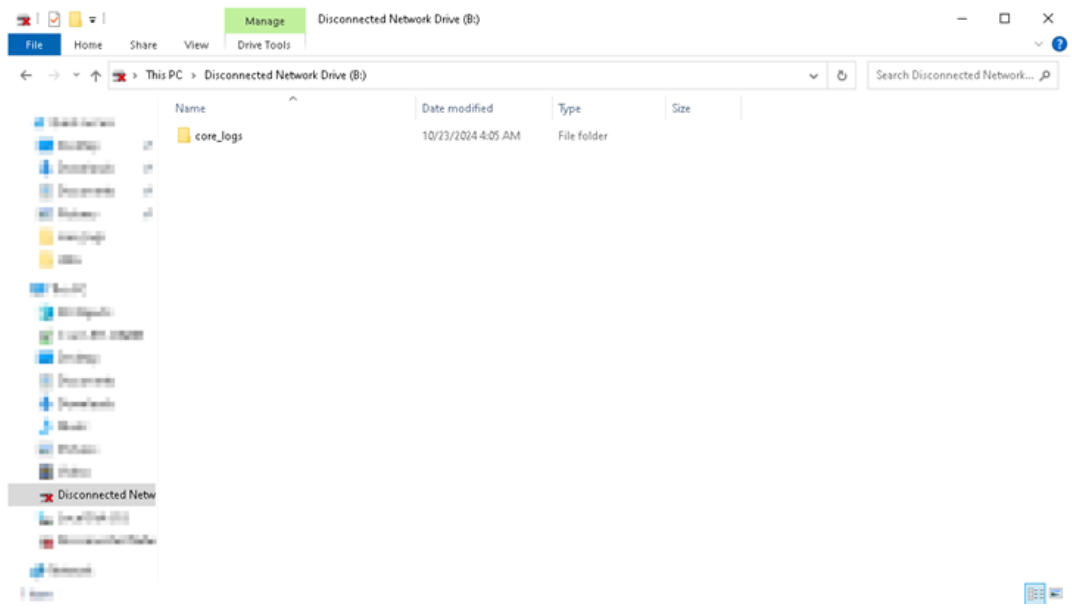
- An elevated command prompt window will open.
- Enter the command: **whoami** to confirm that the new command prompt is running as: **nt**



- 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.
```

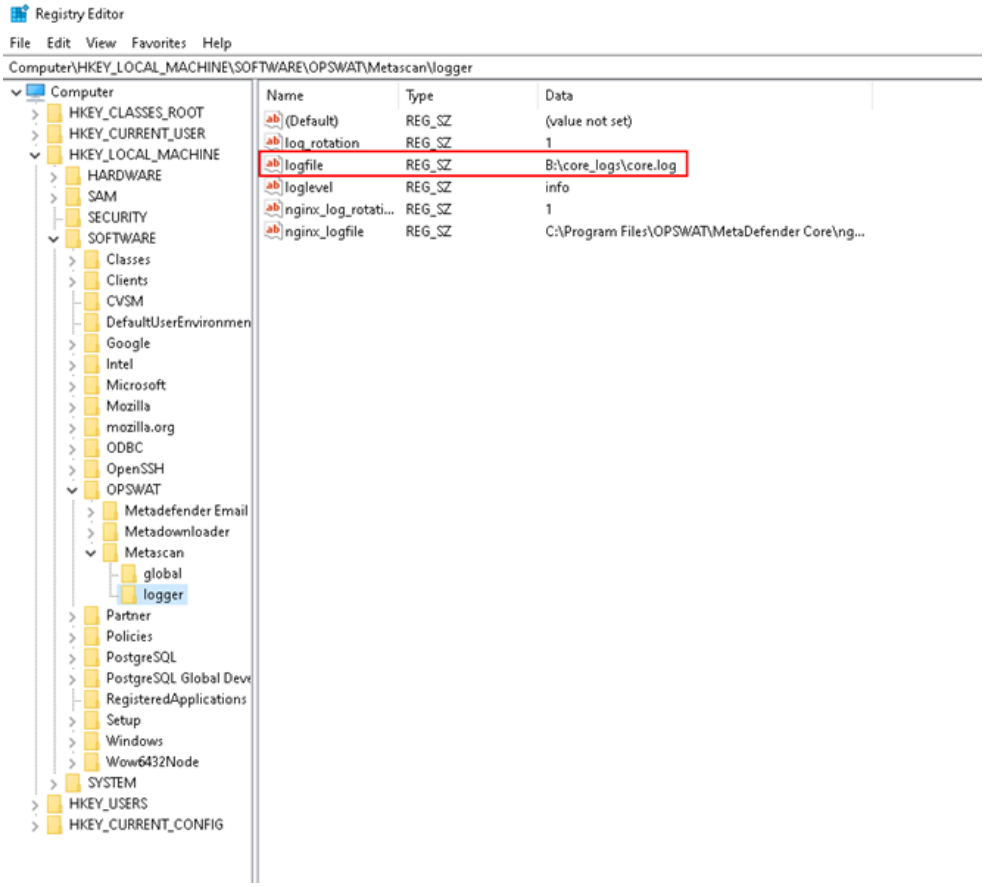
Step 2: Now that we have mapped the **Shared_Storage**, we have created a new **core_logs** folder under which we have created **core.log** file, where we are going to send the logs.



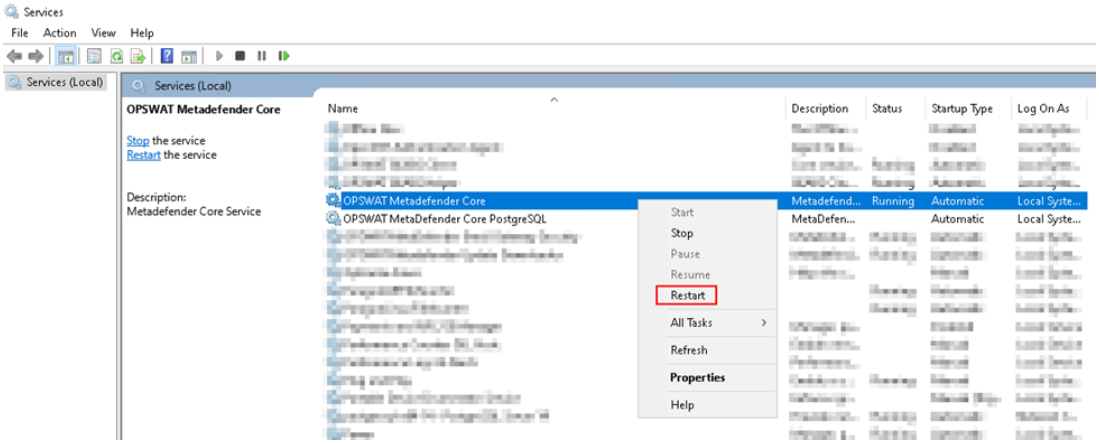
Step 3: Open **Registry Editor** and navigate to the following path:

`Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`

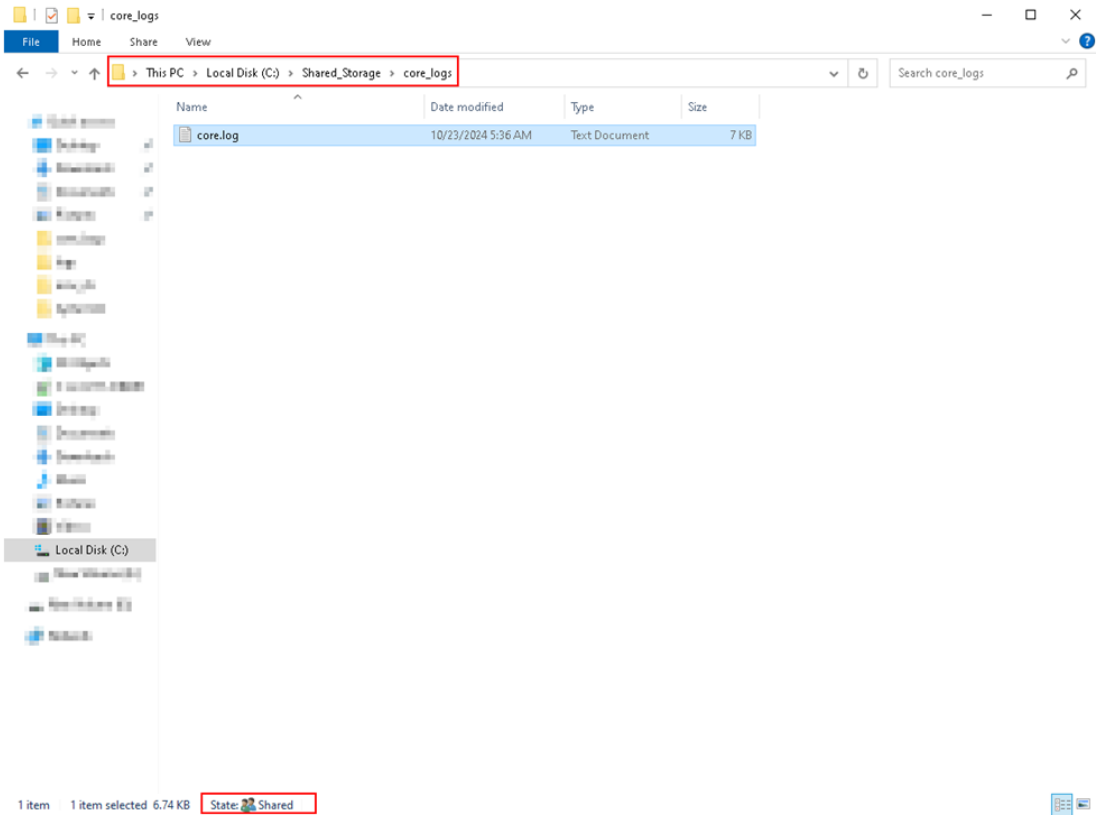
Here you will need to change the **logfile** value to the path we have just configured:



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.

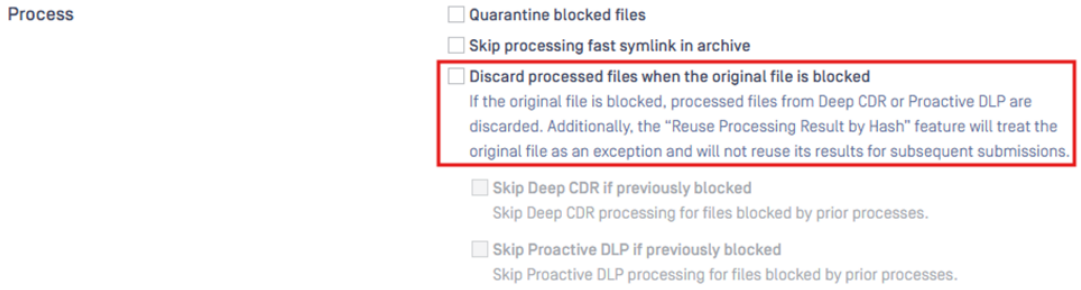
Why is a file blocked by Deep CDR or Proactive DLP but I can still download the sanitized version?

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.

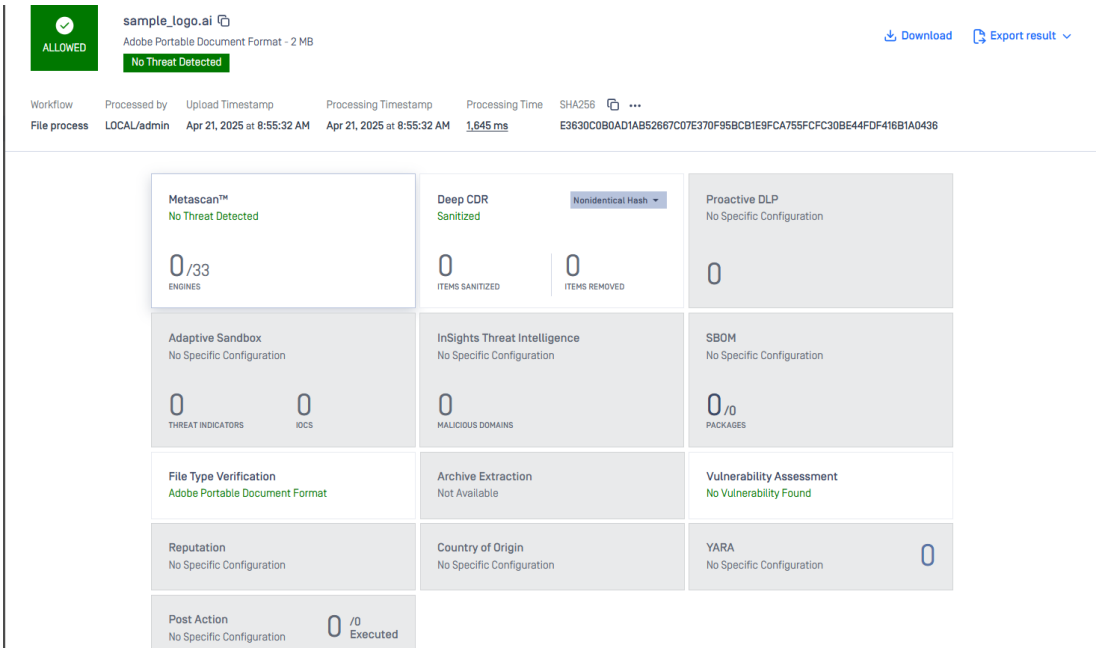
Answer:

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?

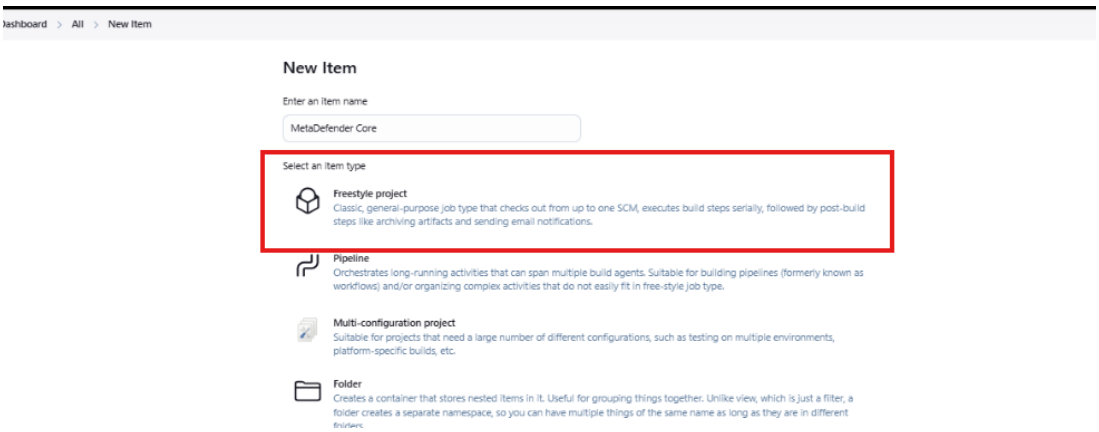
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

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).

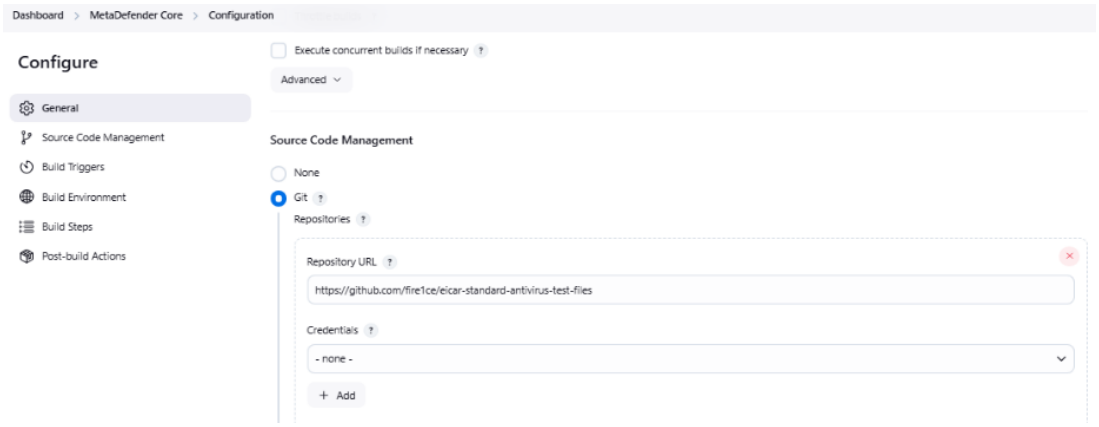
New Pipeline

Create a new pipeline using the 'Freestyle project' option. Use the name 'MD Core' for testing purposes.

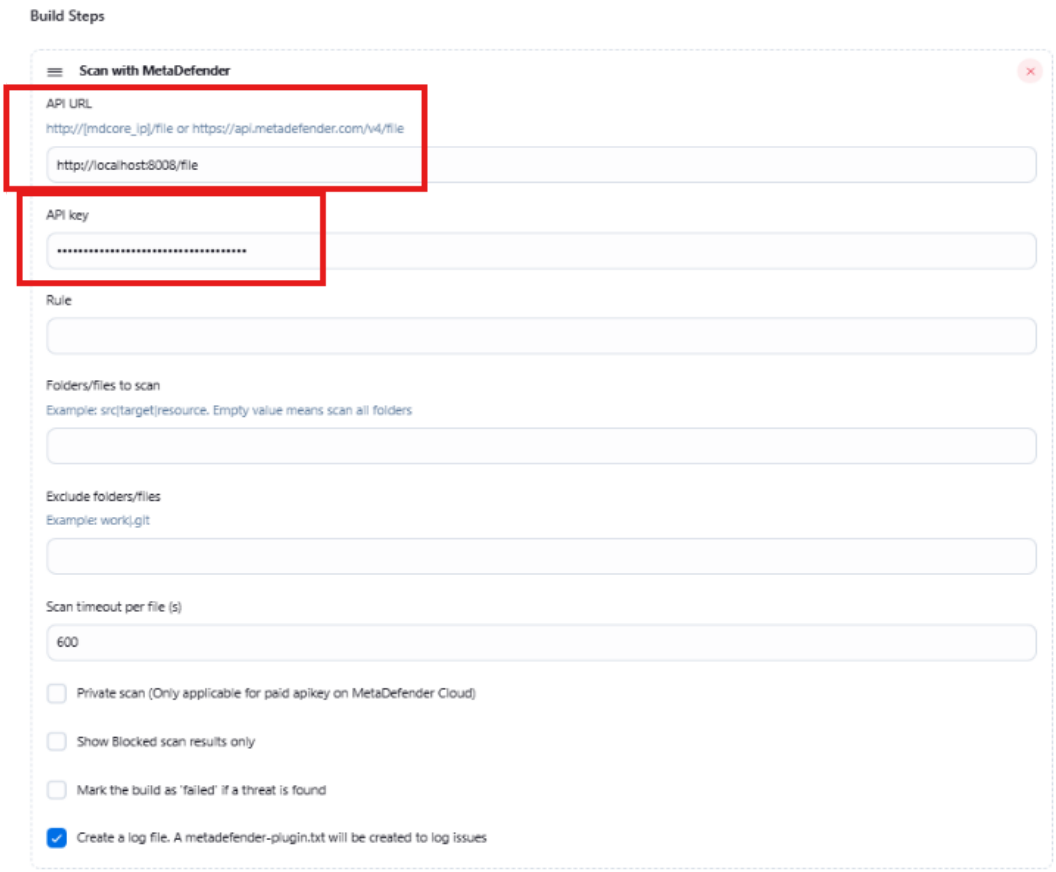


Test files

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

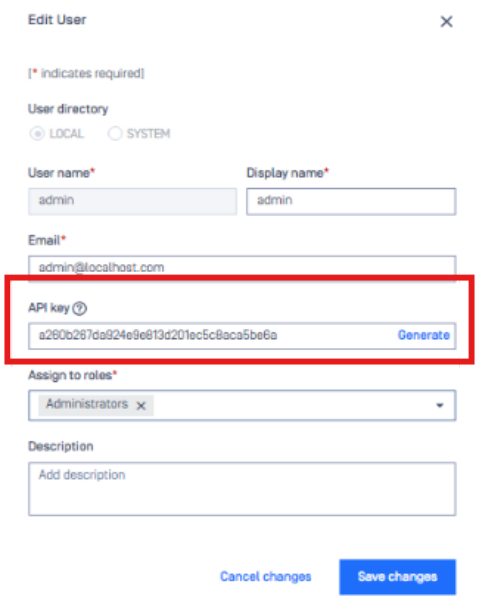


We will then add the steps to build the pipeline with MetaDefender Core



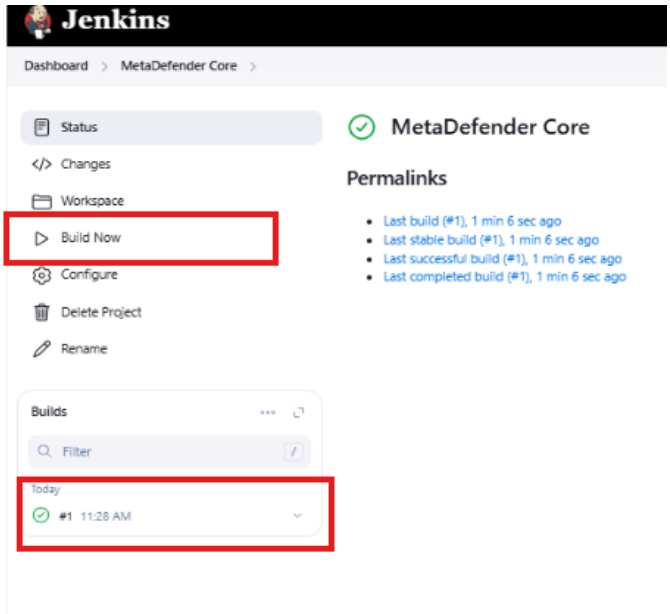
Since this test is performed on a local machine, the API URL is set to http://localhost:8008/file

and the API key will be used from the user management API key.



MetaDefender Core

Now, save the configuration and start testing the pipeline by clicking 'Build Now'.



Start the build

```
[MetaDefender Plugin] [INFO] .git\objects\07\6caa47e105c3a1732298444bc3c28a9313f0dc |
http://localhost:8008/#/public/process/dataId/724e348bd9174c98a7d14400ae38b7c8 | Infected
[MetaDefender Plugin] [INFO] .git\objects\0c\de84f06a5dc24809569e0dce08f998c5820ebc1 |
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
```

Pipe #2

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...	785 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...	667 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: https://www.opswat.com/docs/mdcore/metadefender-core#adminexportv2

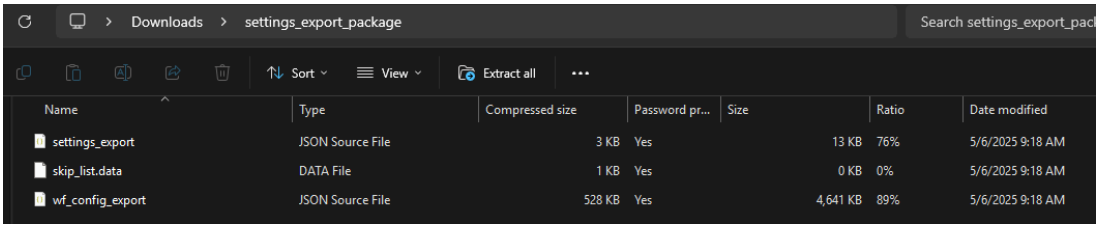
Prerequisites

- A valid **MetaDefender Core API key** (`session_id`), which the user has received from the post request for Login https://www.opswat.com/docs/mdcore/metadefender-core#userlogin

Export settings:



Response example

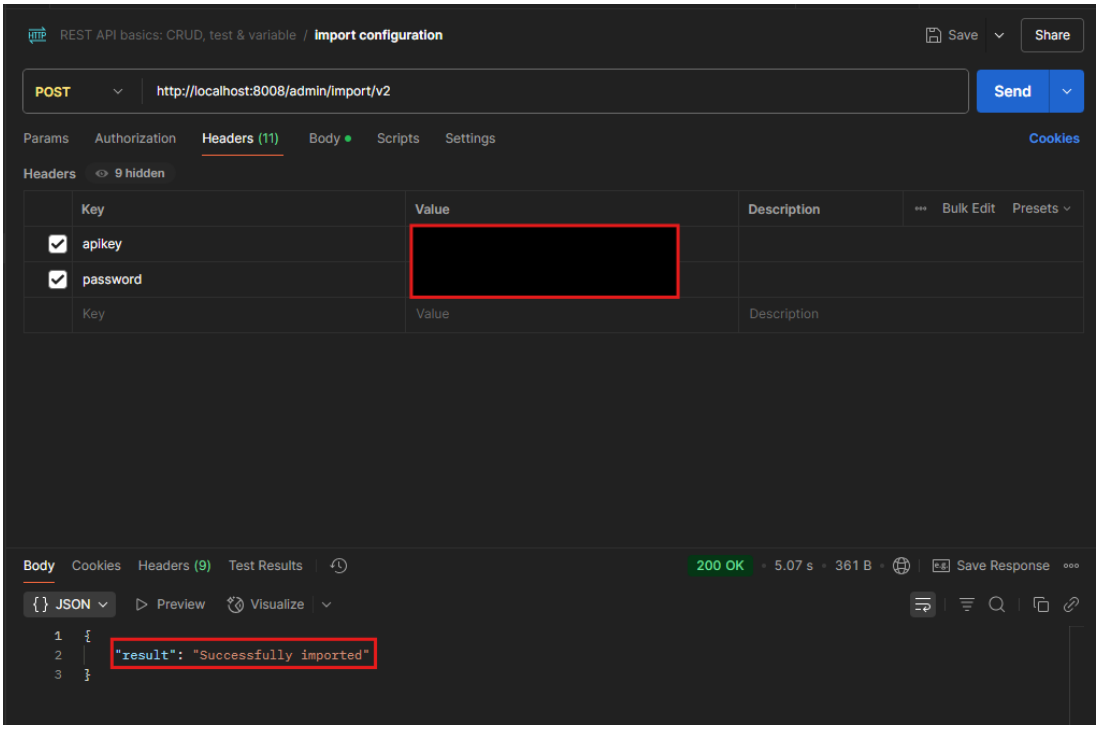


Bash script template:

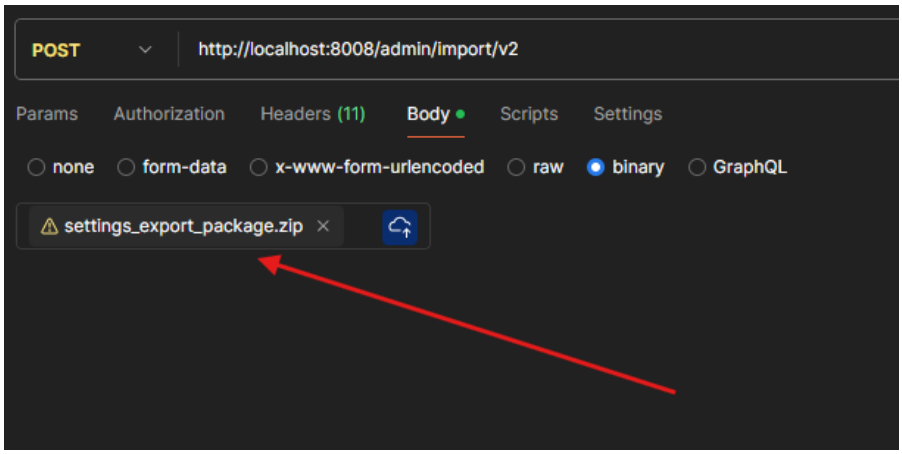
```
#!/bin/bash

# === 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: https://www.opswat.com/docs/mdcore/metadefender-core#adminimportv2



Import request



Include the file

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:

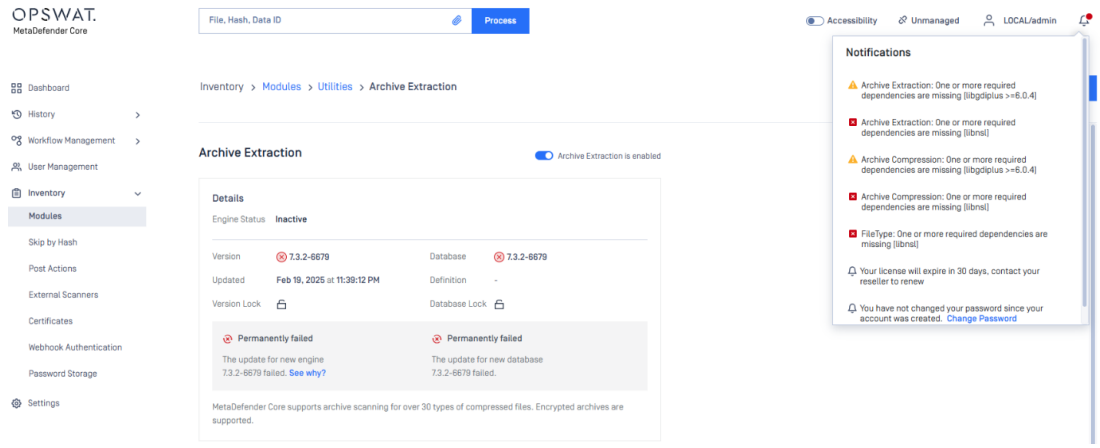
If you have older versions of MD Core connected to OCM that do not have the Password Storage feature and you upgrade to a version that has the feature, you must remove the Core from OCM and re-add the core for the settings to take effect. Otherwise, you will receive the following error when you try to 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:



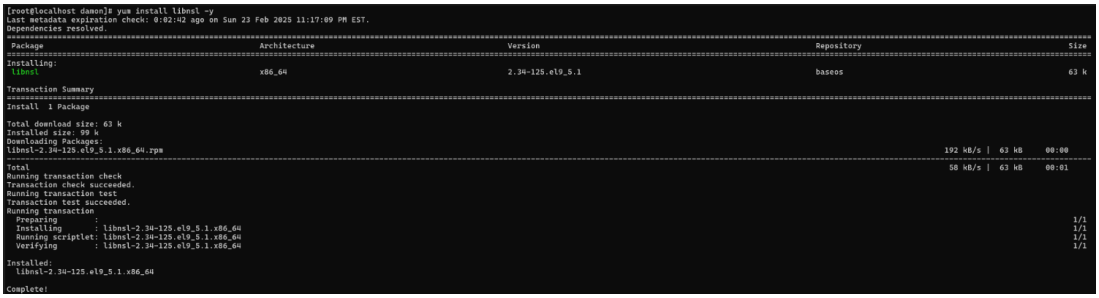
The screenshot shows that the Extract Archive Engine failed to install due to missing libraries: **libgdiplus** and **libnsl**. (these dependency packages are requested)

You can install them using the following command:

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:

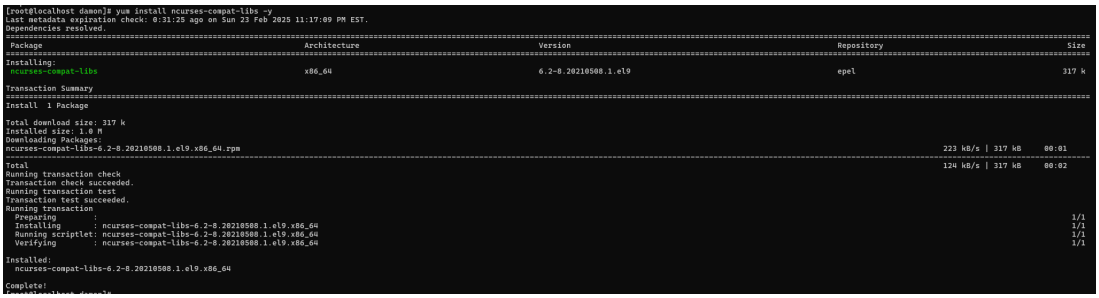


Besides that, the Deep CDR engine requires additional dependency packages (**libgomp**, **ncurses-compat-libs**), which can be installed using the command below:

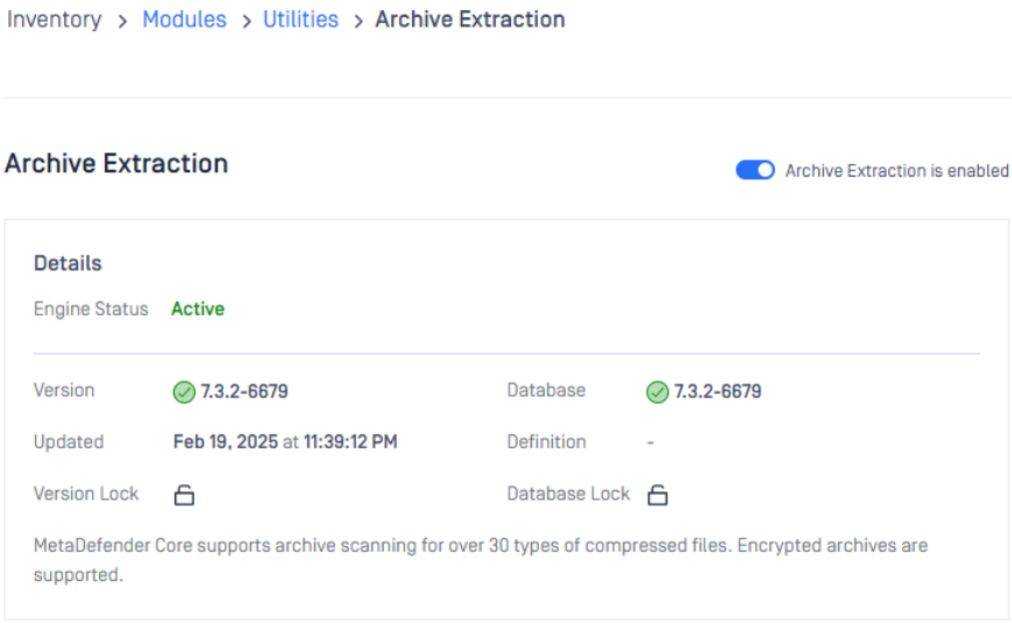
bash

```
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: REST API Endpoint to configure Syslog for Core

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:

https://www.opswat.com/docs/mdcore/configuration/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?

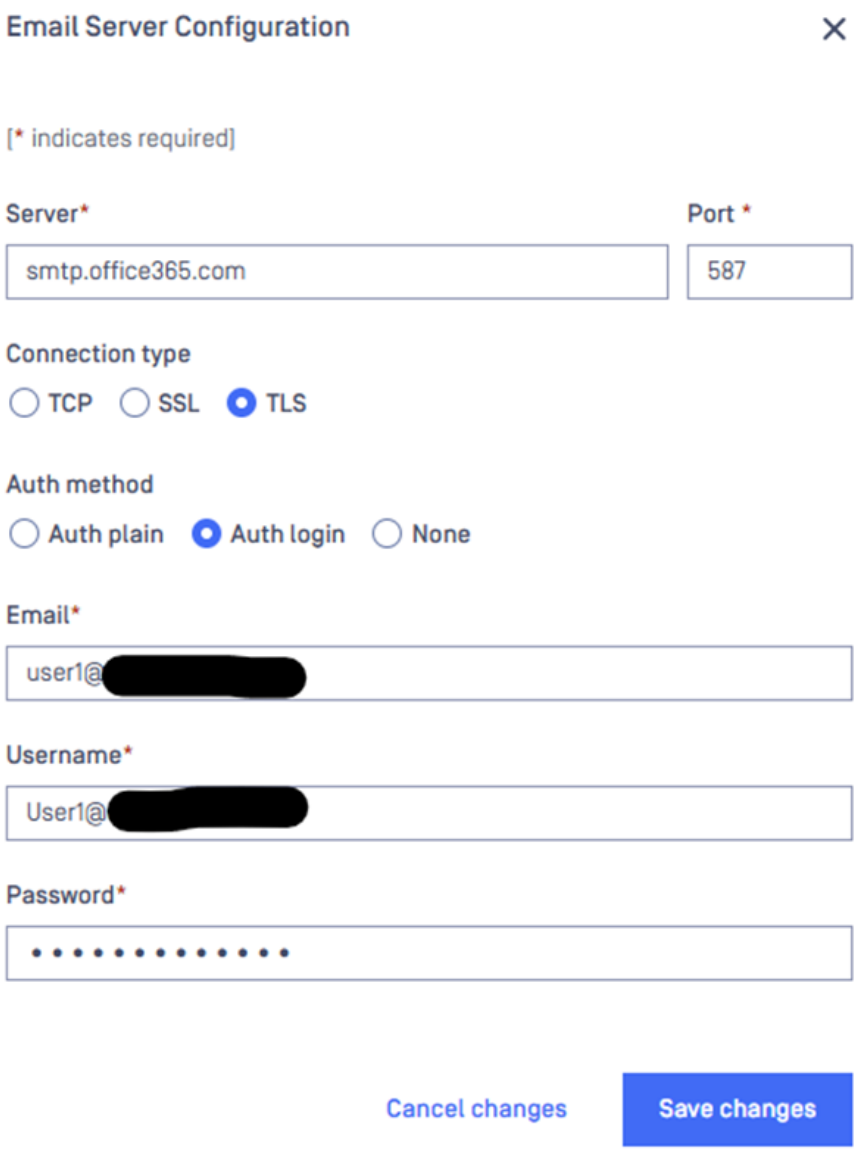
Solution

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.

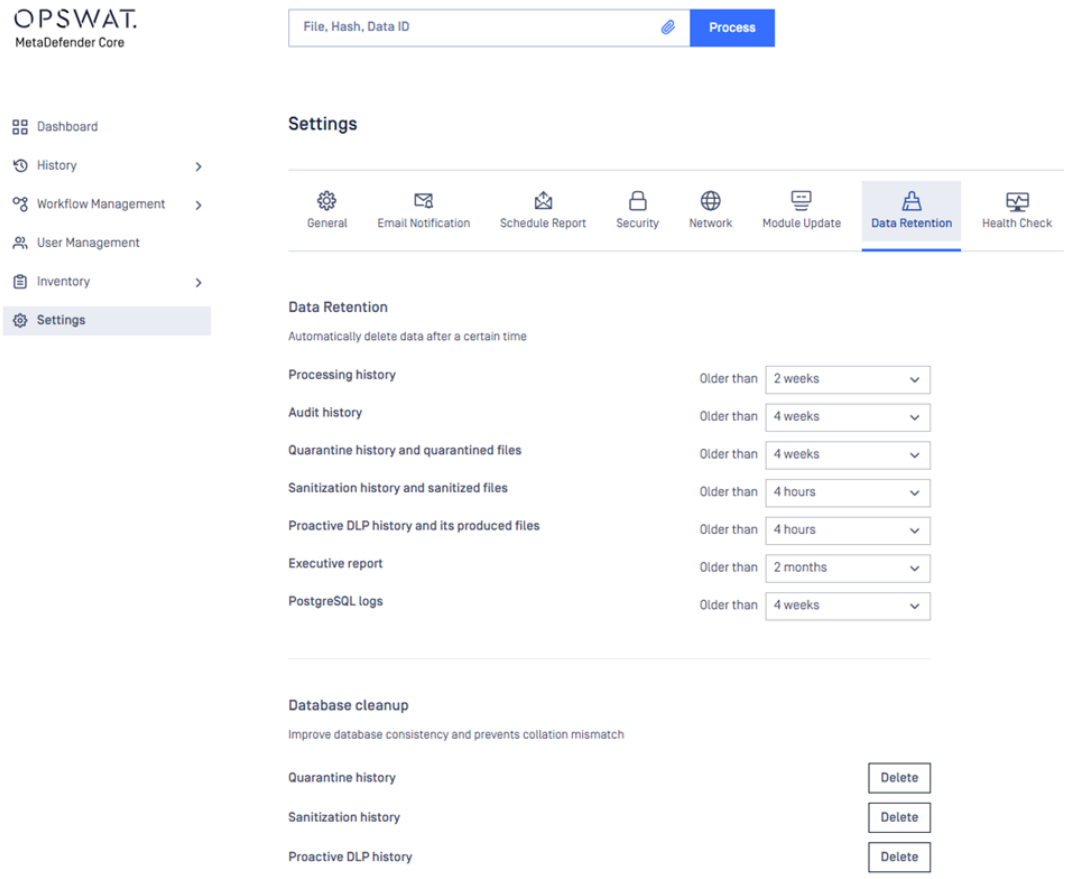
Optional: Immediately Free Up Space

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 - MetaDefender Core)

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

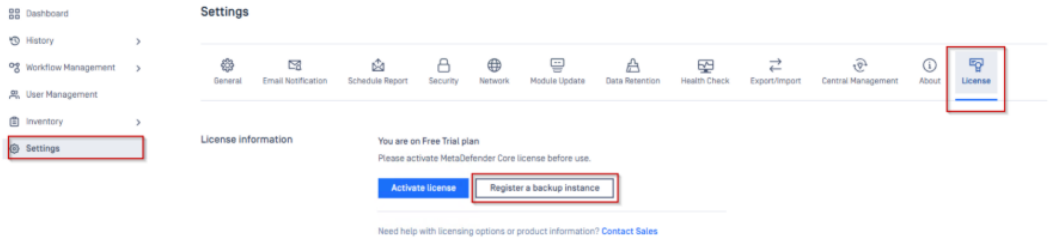
Mapping Google Cloud Storage for MDSS requires careful consideration of the permissions assigned to the user account. While the initial recommendation is to use the Owner role, assessing the operational needs and following best security practices might suggest using more limited roles such as Storage 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**.

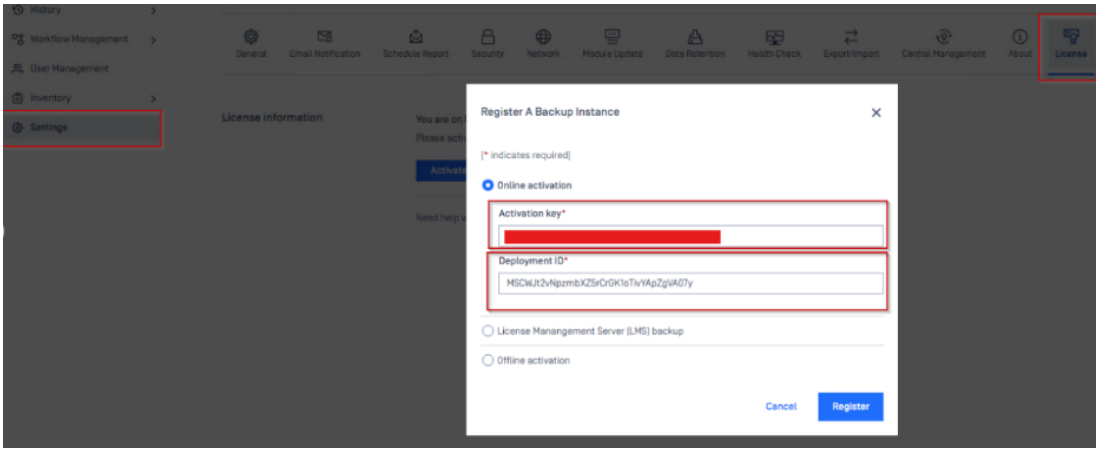
How do I setup Hot Swap Backup Configuration between MetaDefender Core instances?

Online procedure

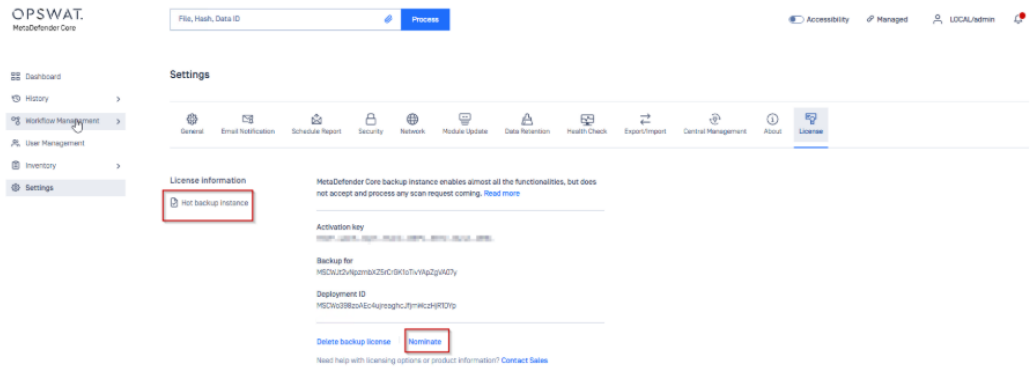
- Make sure the primary MetaDefender Core deployment is activated
- Gather the deployment ID and Activation key from the primary MetaDefender Core deployment
- Navigate to the secondary MetaDefender Core instance and take the following steps
    - Select the Register as back-up instance option on the 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.

bash

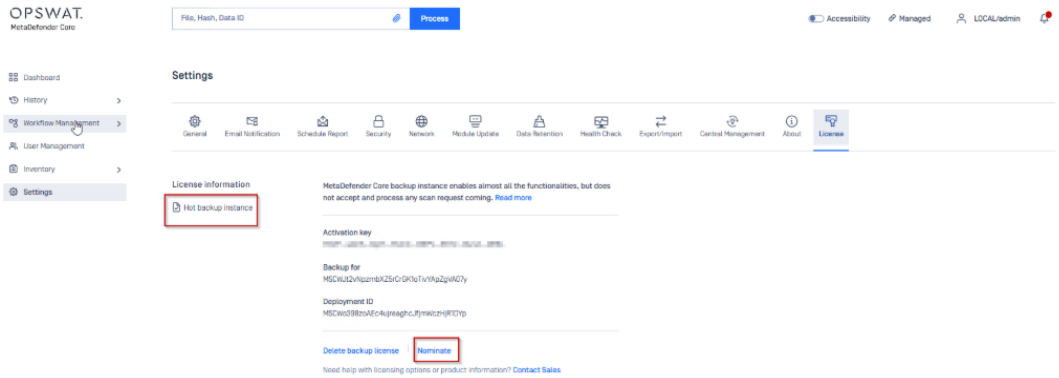
```
curl --request POST \ --url '__http://localhost:8008/admin/license/backup' \ --header 'Content-Type: application/octet-stream' \ --header 'apikey: {apikey}' \ --header 'active-deployment-id: {active-deployment-id}' \ --header 'activation-key: {activation-key}' \ --data-binary '{ "schema": "{string}" }'
```

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:

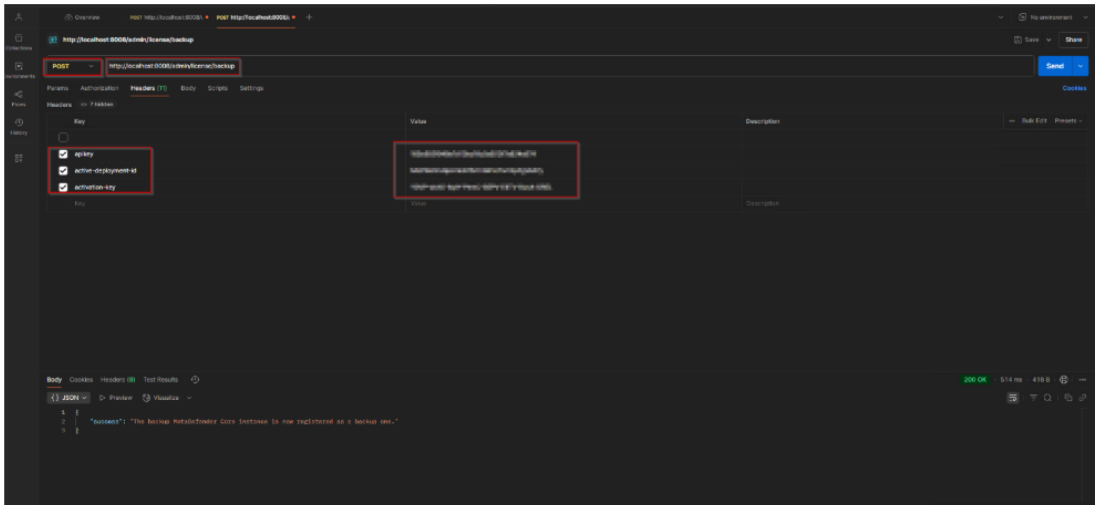
- Make sure the primary MetaDefender Core deployment is activated
- Gather the deployment ID, Activation key, and the offline yml activation file from the primary MetaDefender Core deployment
- Use the API call listed below with any preferred solution.

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:

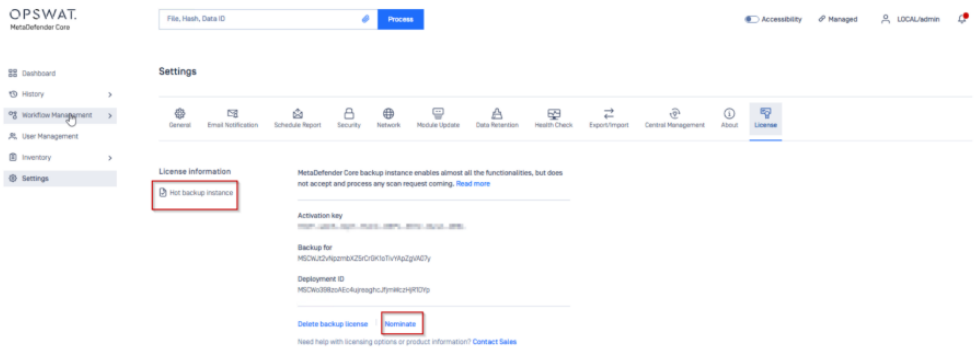
- API Key from the backup MetaDefender Core server
- Activation key
- Primary MetaDefender Core instance deployment ID
- .yml activation file from the Primary MetaDefender Core instance



- 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 is located in this Registry path:

html

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan
```

Linux

By default, the following are the directories used when MetaDefender Core is installed:

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:

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**, 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.

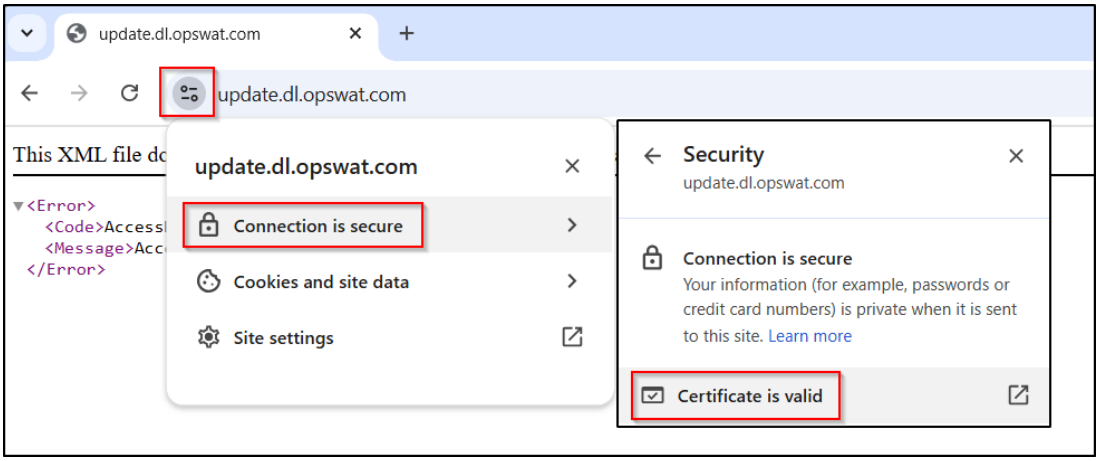
Steps for Chrome or Edge (similar steps for Firefox)

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.



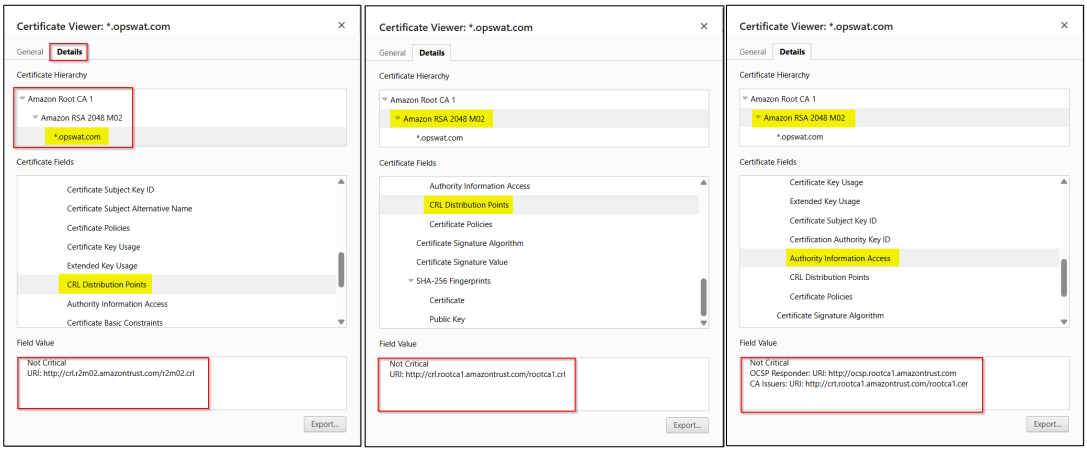
3. Examine the Certificate Chain

- 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?

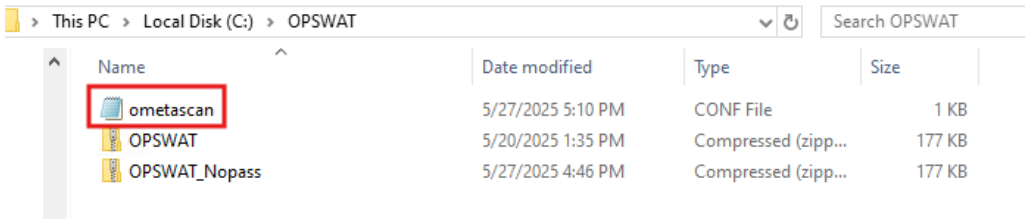
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows systems.

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.

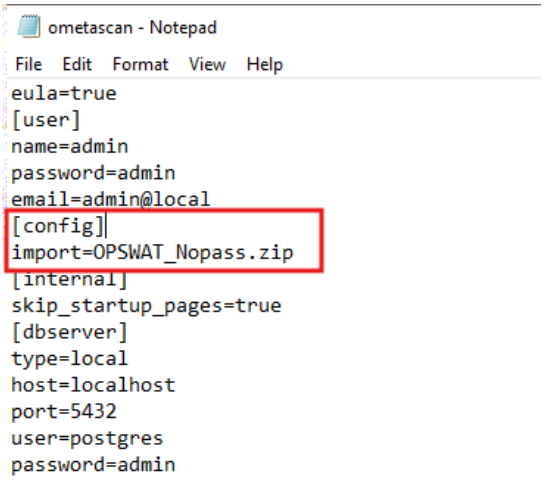
Example:



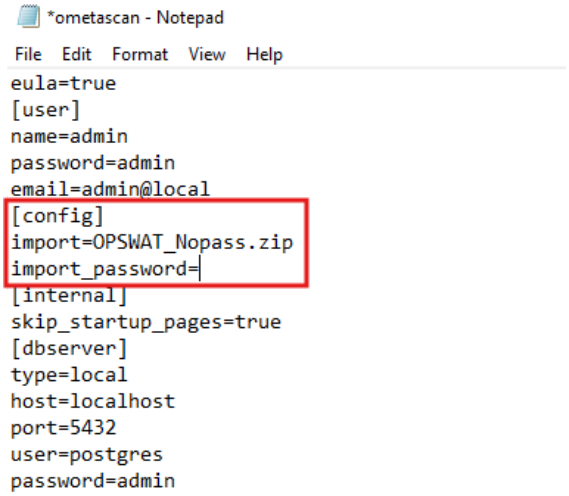
Here we have an example of two configuration files:

- `**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)



Example 1



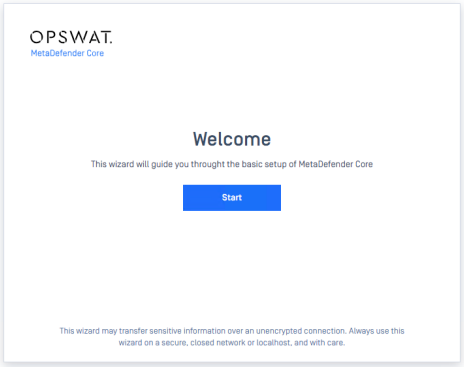
Example 2

When importing the configuration file above, MetaDefender Core reports it as an invalid format.

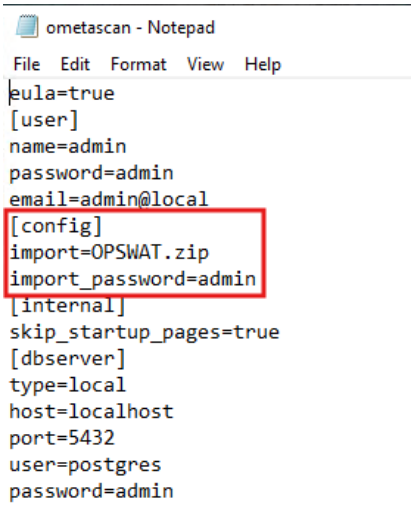
html

```
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)



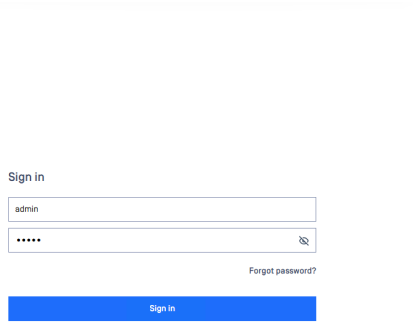
Once the ignition file is confirmed when installing MetaDefender Core, the configuration file is in the right format.

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.

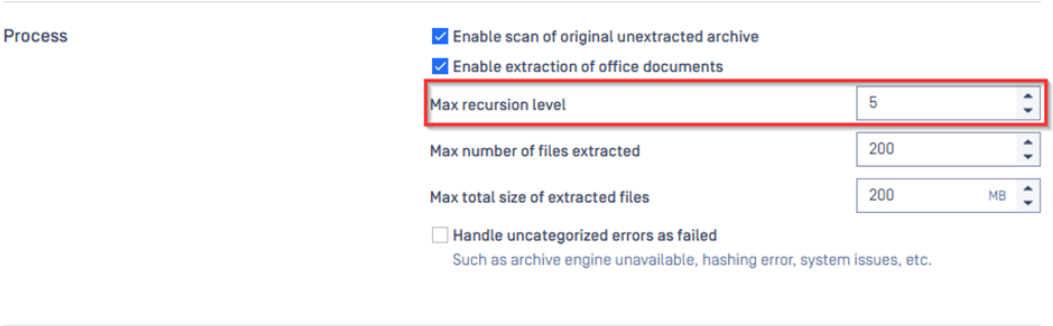
How does the recursion level work in MetaDefender Core?

> **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.

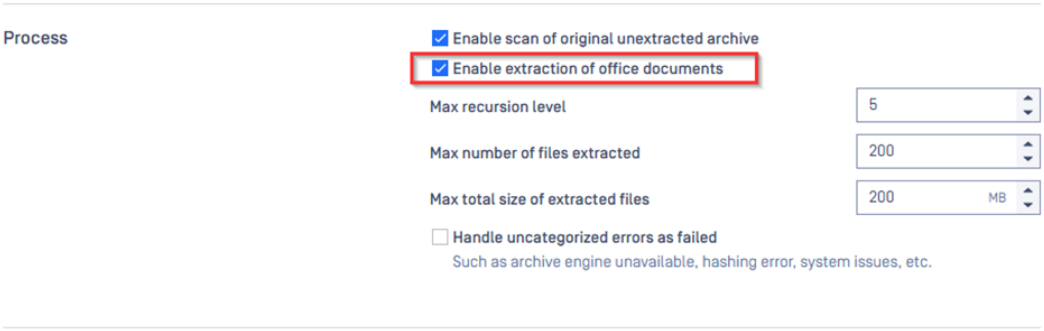
 What Is Archive Recursion Level?

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:

```
env:
- name: CORE_CONF_JSON
value: '{"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.

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: https://www.opswat.com/docs/mdcore/knowledge-base/how-do-i-manually-clean-up-the-md-core-engine-s-local-folder-

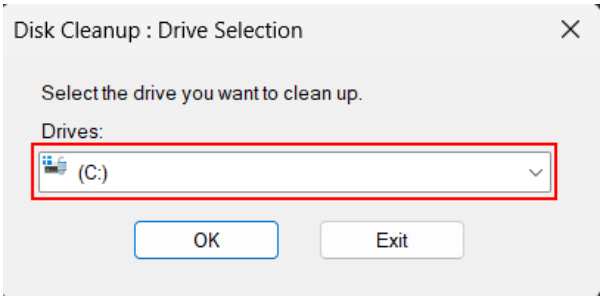
> **Warning:**
> Deleting MetaDefender Core Engines will require an engine update after that.

3. **Using the Windows Disk Cleanup tool:**

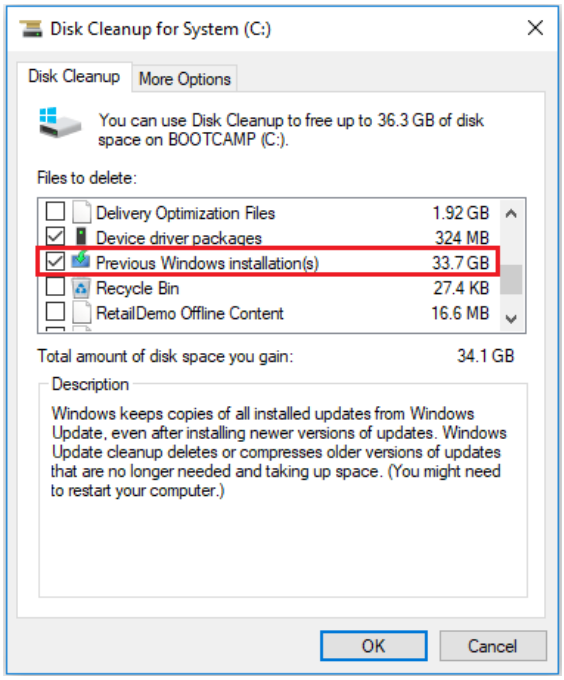
This method will not interfere with the MetaDefender Core installation files, but the Windows unnecessary files.

> **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`



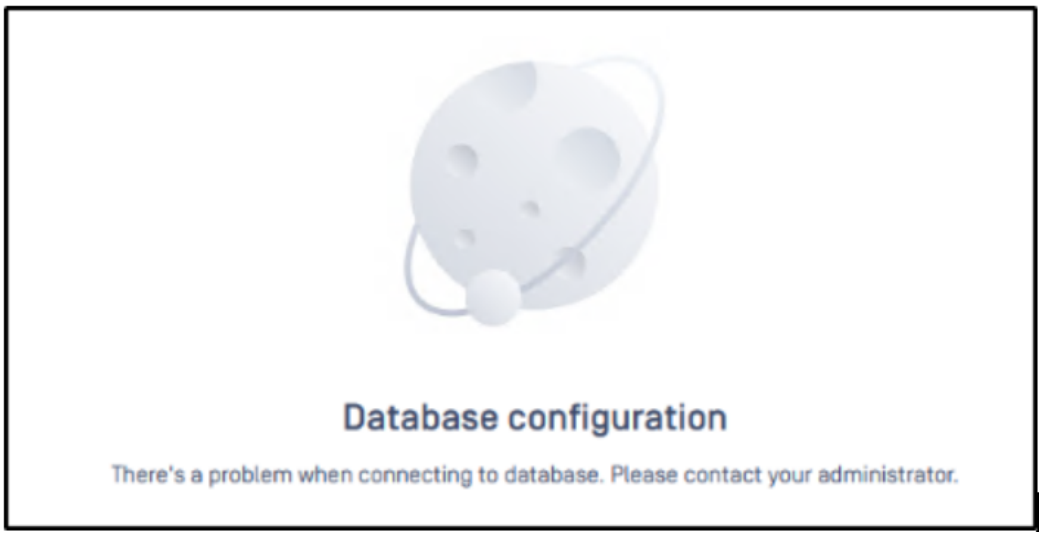
Choose OK in the end to start the Windows Disk Cleanup.

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 none

```
[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 none

```
[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:

bash

```
LD_LIBRARY_PATH=/usr/lib/ometascan:$LD_LIBRARY_PATH
```

Run the upgrade database tool using:

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:

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 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.

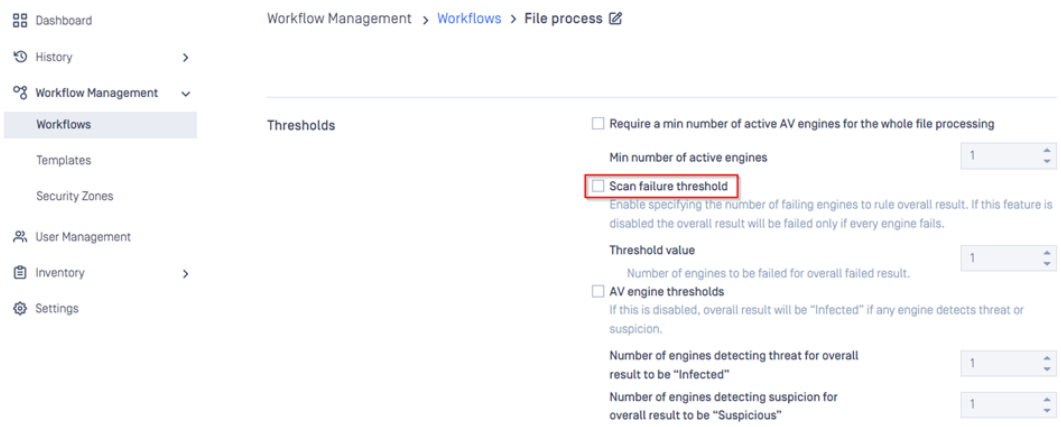
Failing to disable this option can give the illusion that MetaDefender Core is failing to scan large amounts of files.

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

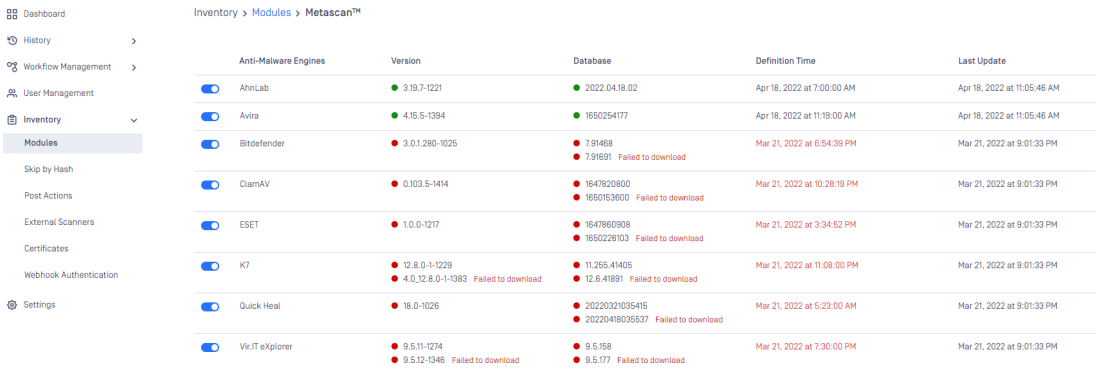
Uncheck the option **Scan Failure Threshold** shown below.



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 auto$, before creating a support case or chatting with our support engineer.

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.

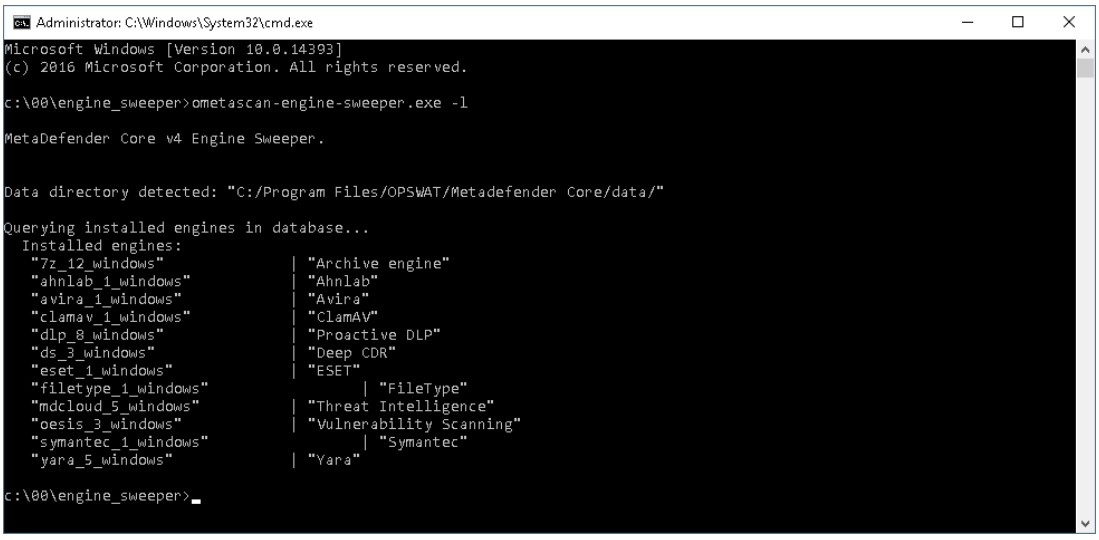
Support:
If you have followed all of these steps and your engines are still unusable, please proceed to create a support case or chat with our support engineer.

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: auto$
>
> 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 : auto$

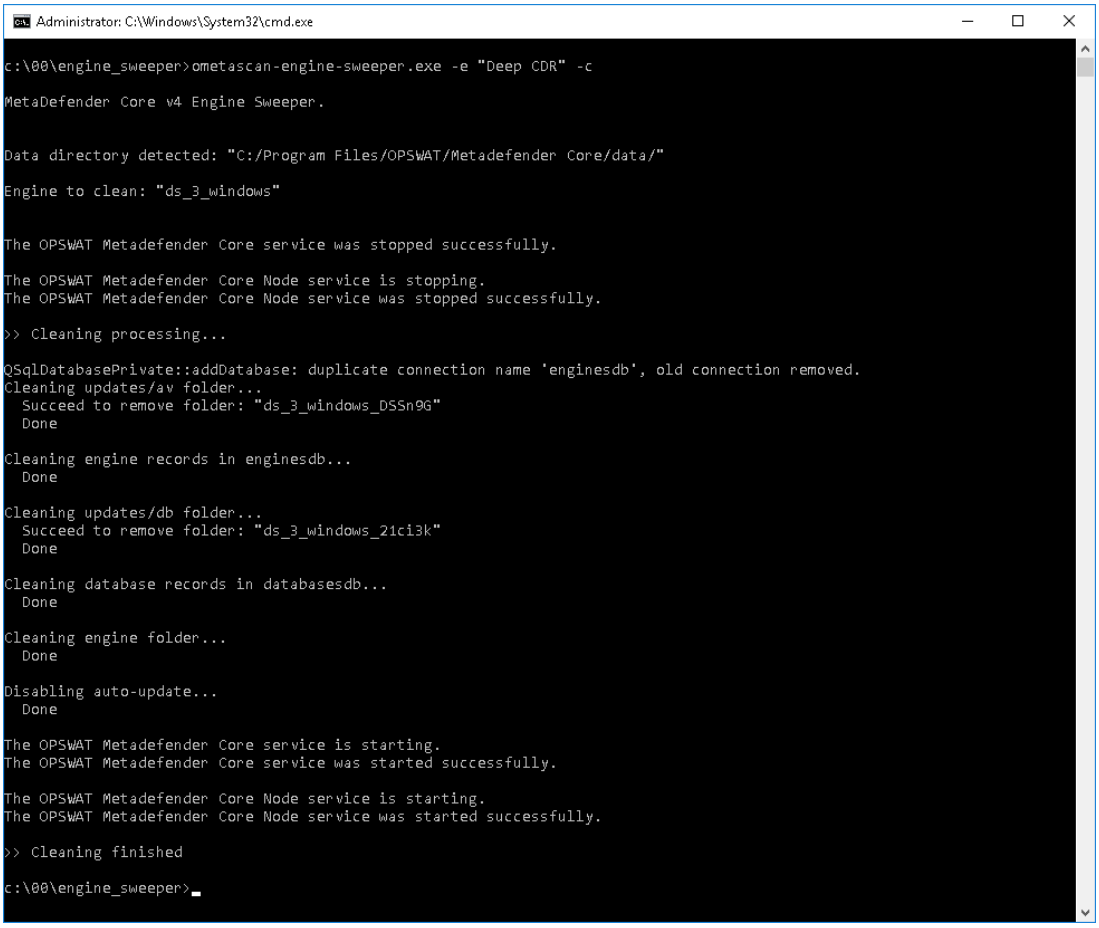
Sometimes, during the engines downloading/deployment process, some of them may remain in "**failed**" or "**permanently failed**" status. In this case, you can perform an engine clean-up by downloading the engine_sweeper tool and following the next steps:

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

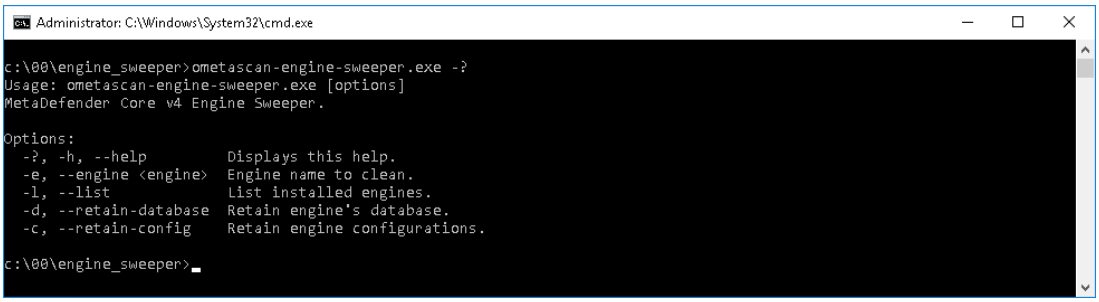
Note: During the cleanup process, the MetaDefender Core and Node services will be **restarted**, and the Updates Settings Source will be set to "**Manual**"



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 a support package, 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 auto$, before creating a support case or chatting with our support engineer.

Why have the ESET and Kaspersky scan engines 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 and Kaspersky engines will fail to initialize unless certain changes are made.

The ESET and Kaspersky scan engines require execution permission on a temp folder. However, while on the hardened system, such permissions are restricted on: **/var/tmp**, resulting in the engines' 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** and **Kaspersky** engines are still failing to initialize, please follow these instructions on auto$, 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.

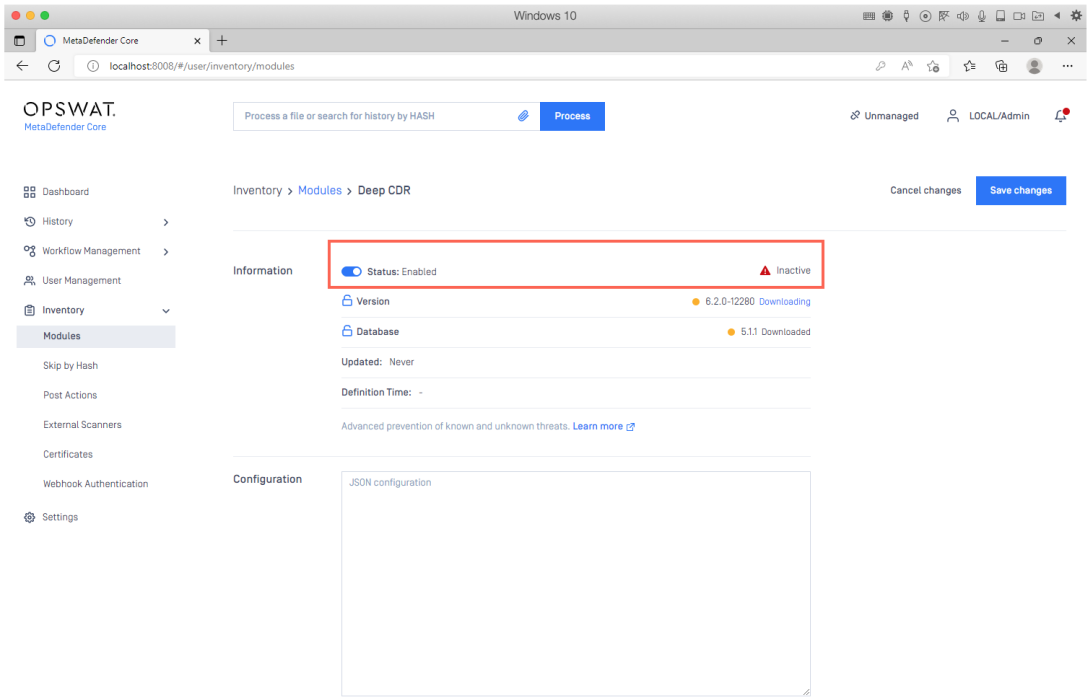
Support:
If you have followed the instructions above but are still experiencing a Symantec installation **timeout**, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

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:

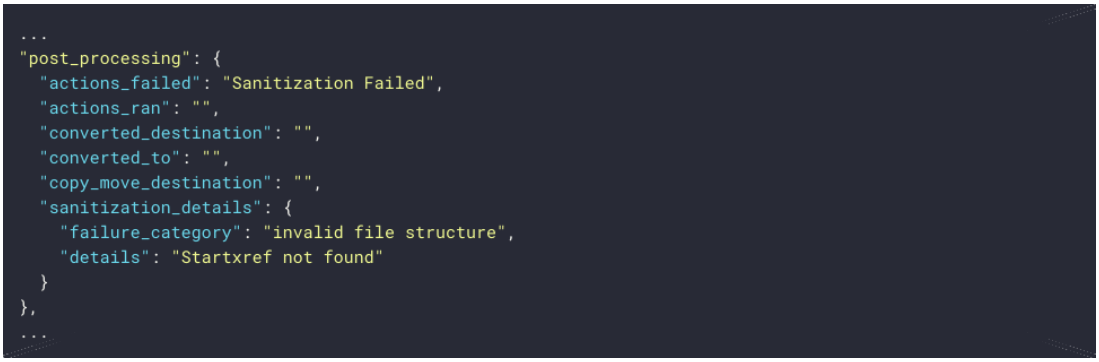
You are attempting to process a supported file type, but the version is outdated:

- 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.

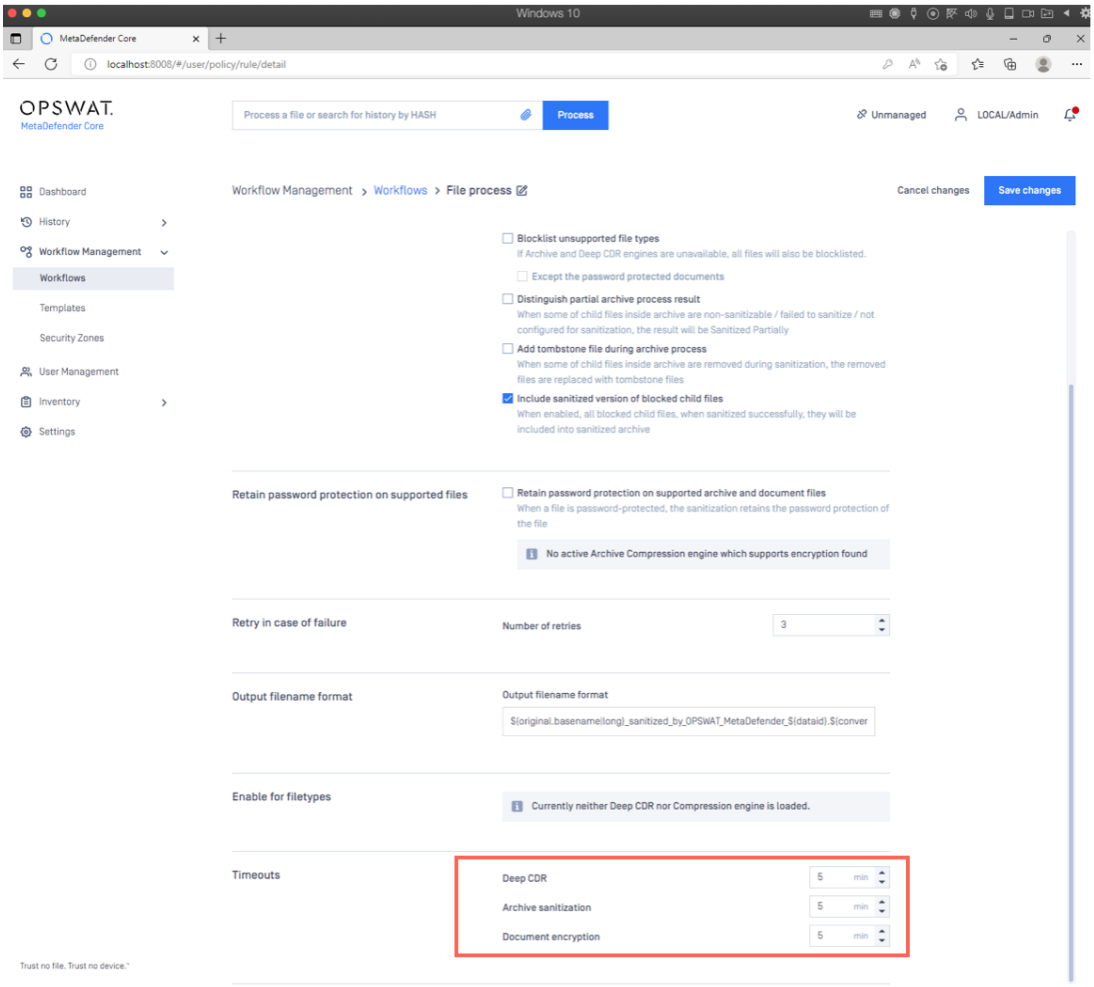
When retrieving your scan results via REST API, Deep CDR sanitization failures will be clearly classified under the **post_processing** section in your JSON output, as illustrated in the screenshot below.



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 auto$, 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
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
	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
GET/archive/{data_id}	200	Entire analysis report, containing all extracted files, generated by MetaDefender Core

API Request	API Response	Meaning
	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
	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

API Request	API Response	Meaning
	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 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
	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)

API Request	API Response	Meaning
	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 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

API Request	API Response	Meaning
POST/admin/user	200	Request processed successfully
	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
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/nomination	200	Request processed successfully
	400	Bad request (e.g. invalid header, API key missing or invalid)
	403	Invalid user information or not allowed

API Request	API Response	Meaning
	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

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
	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

API Request	API Response	Meaning
	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
	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

API Request	API Response	Meaning
PUT/admin/config/webhook	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/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
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

API Request	API Response	Meaning
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 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
	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
	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 auto$, 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 https://www.opswat.com/docs/mdcore/knowledge-base/how-to-retrieve-the-crl-and-ocsp-urls-for-the-server-and-ica-cer

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 auto$, 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 auto$.

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 auto$.

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 2010	grep
Microsoft Visual C++ Redistributable for Visual Studio 2013	lib32stdc++6 (>= 4.5)
Microsoft Visual C++ Redistributable for Visual Studio 2017	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.

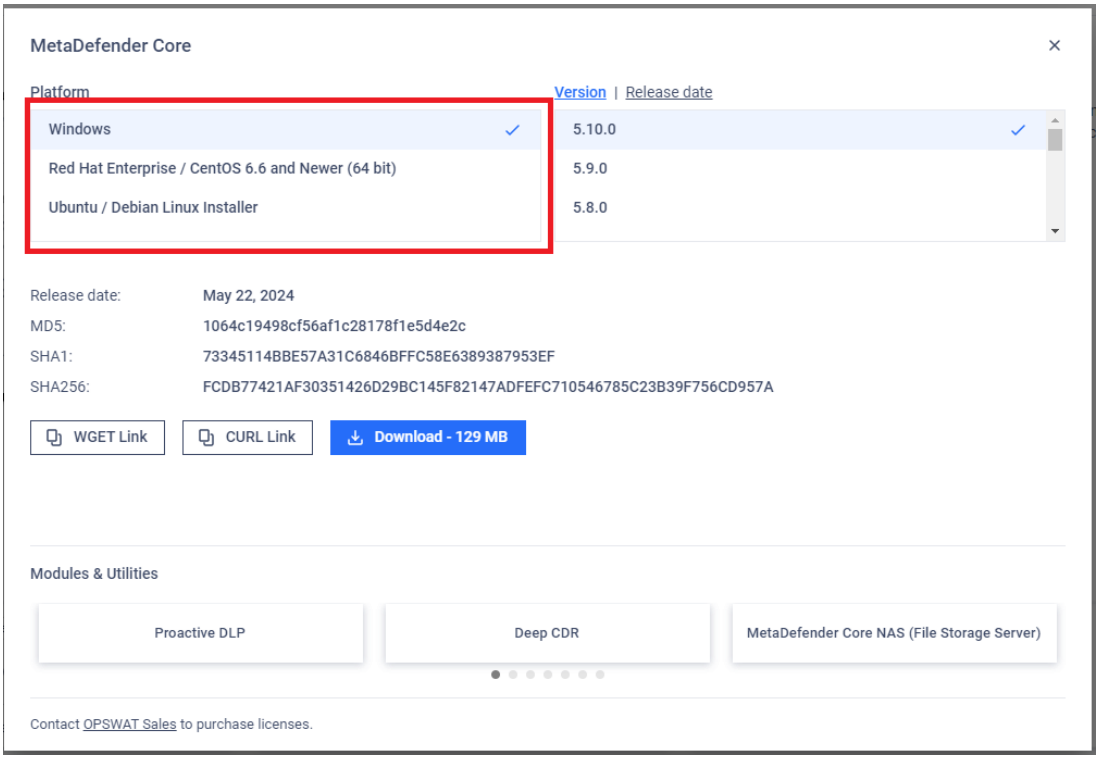
To learn more about engine-specific dependencies, please Read This.

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

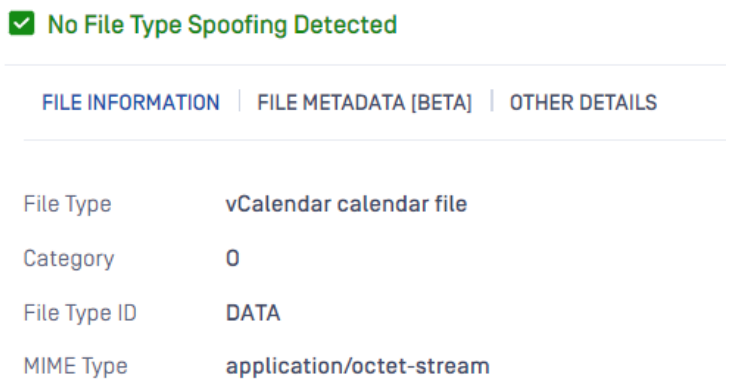
You will be unable to launch the **MetaDefender Core Management Console** and use the product unless these hosts are allowed.

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?

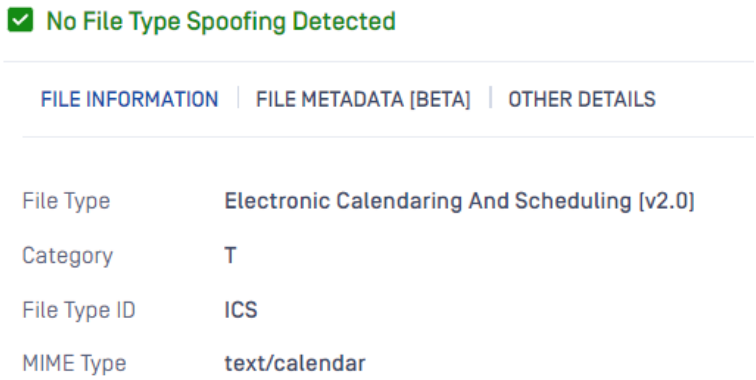
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

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.

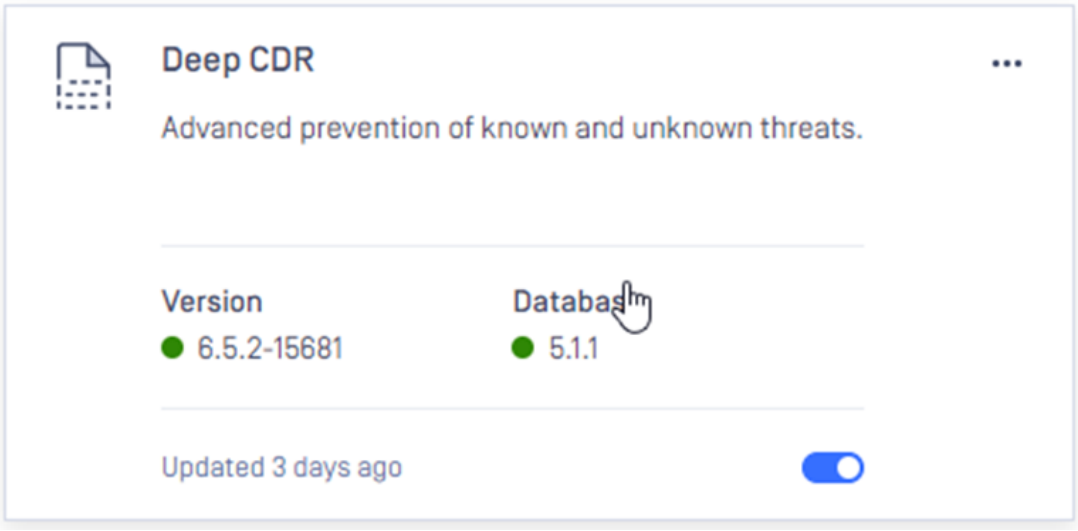
Support:
If Further Assistance is required, please follow the instructions on auto$, then proceed to create a support case or chat with our support engineer.

Why files are not sanitized?

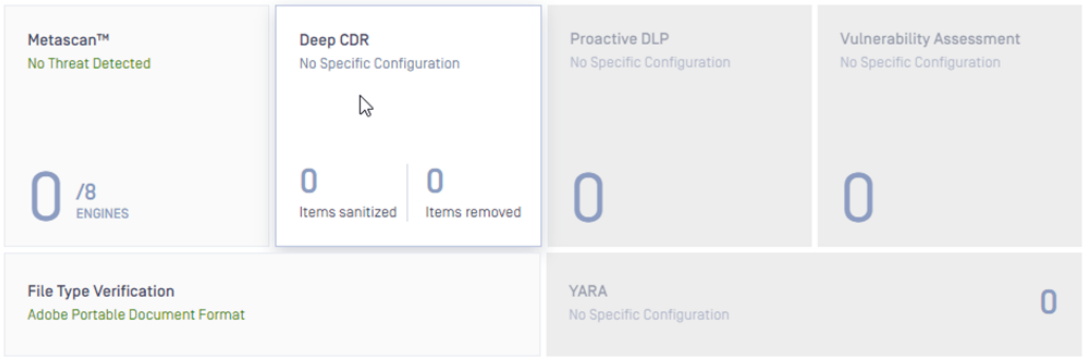
Check Your Version
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

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.

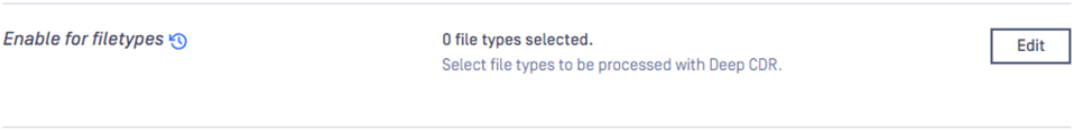


2. Make sure that the extension of the file you are scanning is enabled in the DEEP CDR module.

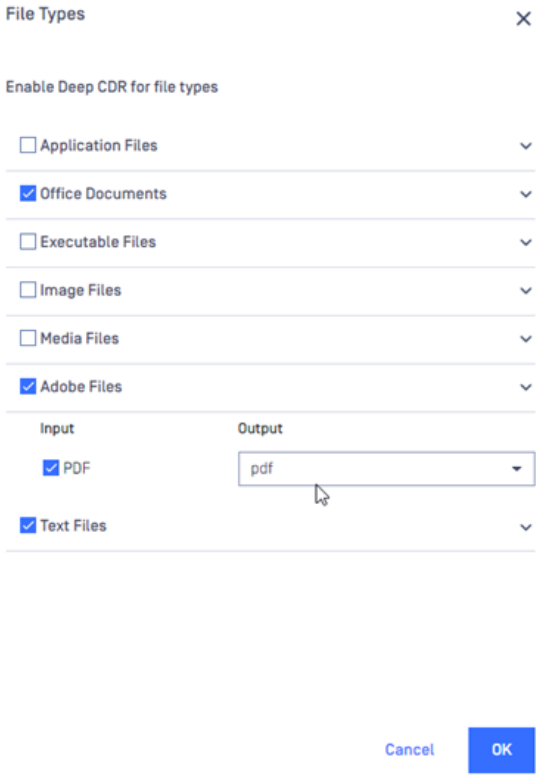


To do so, please access the MetaDefender Core WEB UI (by default http://localhost:8008) and access Workflow Management -> Workflows -> Select your workflow -> DEEP CDR tab

Here navigate to “Enable for filetypes”



Click on Edit and enable all extensions that you want to sanitize:

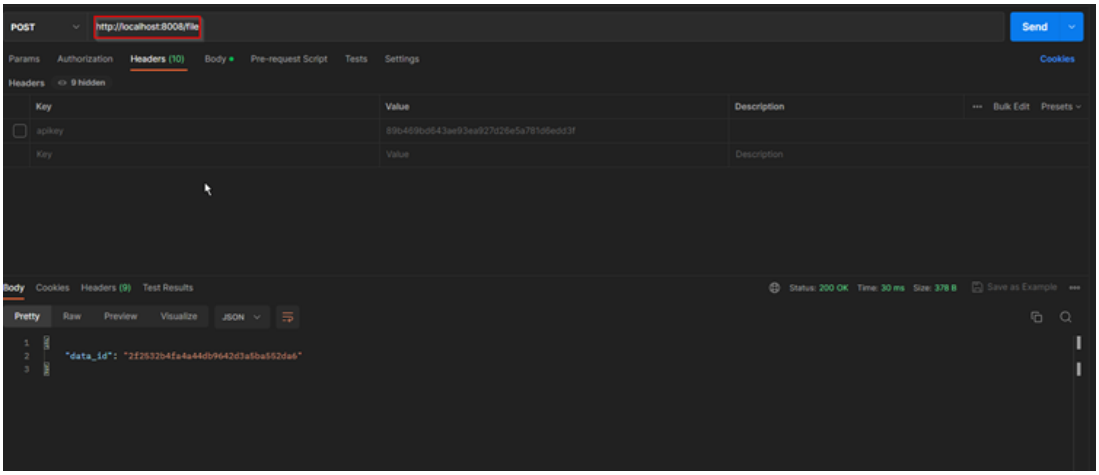


Once selected, click OK and then Save changes.

Support
If you have followed the instructions above but **Files are still not sanitized** please follow these instructions on auto$, 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.

https://docs.opswat.com/mdcore/metadefender-core/ref#fileanalysispost

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?

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 software operational on the MetaDefender Core Server.
2. Navigate to the settings of the identified real-time protection software.
3. Look for the 'exclusions' or 'exceptions' section in the settings.
4. Add full installation path of MetaDefender Core. Add the temporary upload path used by MetaDefender Core. Add **engineprocess.exe**, **engineprocess32.exe**, **ometascan.exe**, **postgres.exe** and **nginx.exe** processes, to the exclusions list. Note: If you are using Symantec Endpoint Protection as your local AV, please adjust the settings as instructed in This KB Article.
5. Save your changes and exit the settings.
6. Conduct a test scan to confirm the resolution of the problem.

> **Please Note**
> For more detailed instructions on adding exclusions or exceptions to your antivirus 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 auto$, 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
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 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:

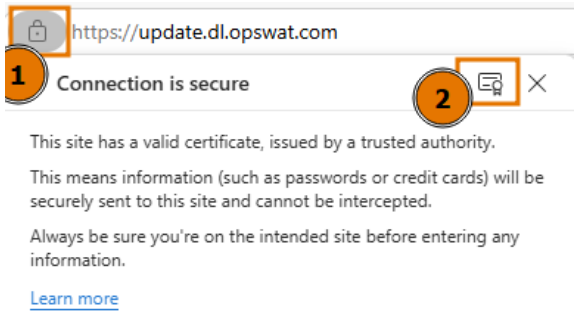
1. Make sure that update.dl.opswat.com is reachable from the MetaDefender Core server.
2. If a proxy is used, please make sure it’s configured correctly in Proxy configuration.

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.

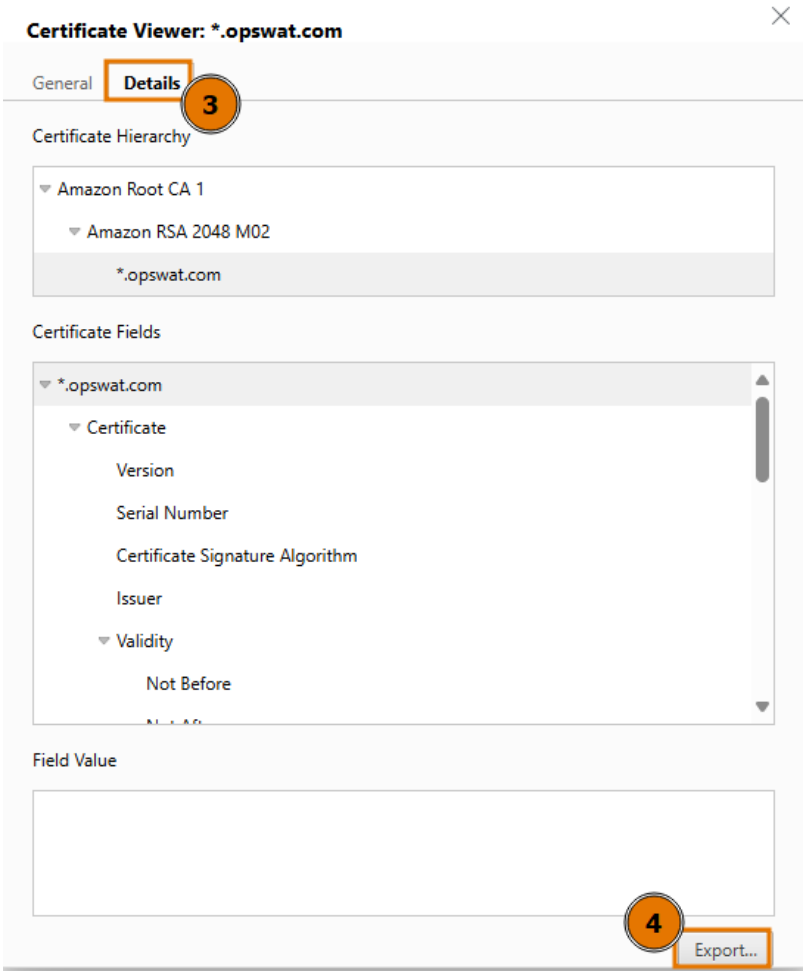
When the above certificate error occurs for update.dl.opswat.com, you might find the following in core.log:

```
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:

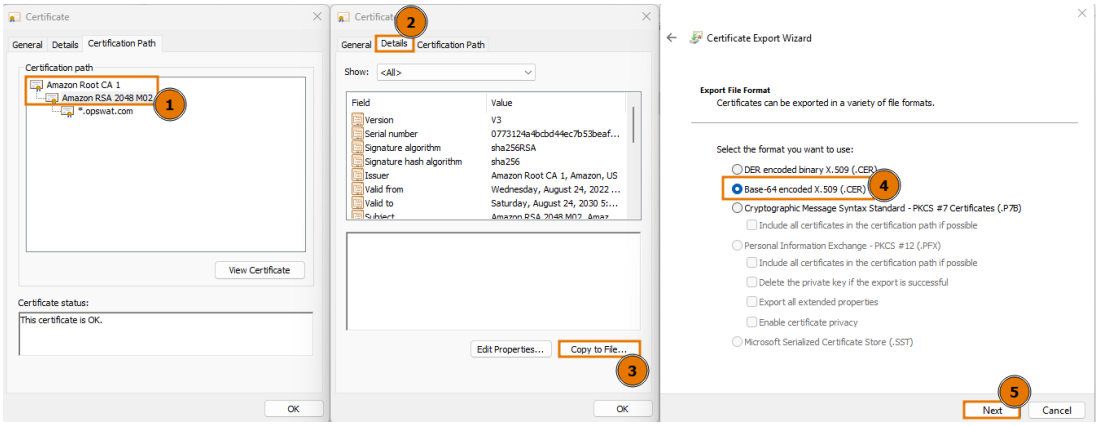


View “https://update.dl.opswat.com” certificate.



Export “https://update.dl.opswat.com” certificate.

Once the certificate chain is downloaded, export the root and intermediate certificates:



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

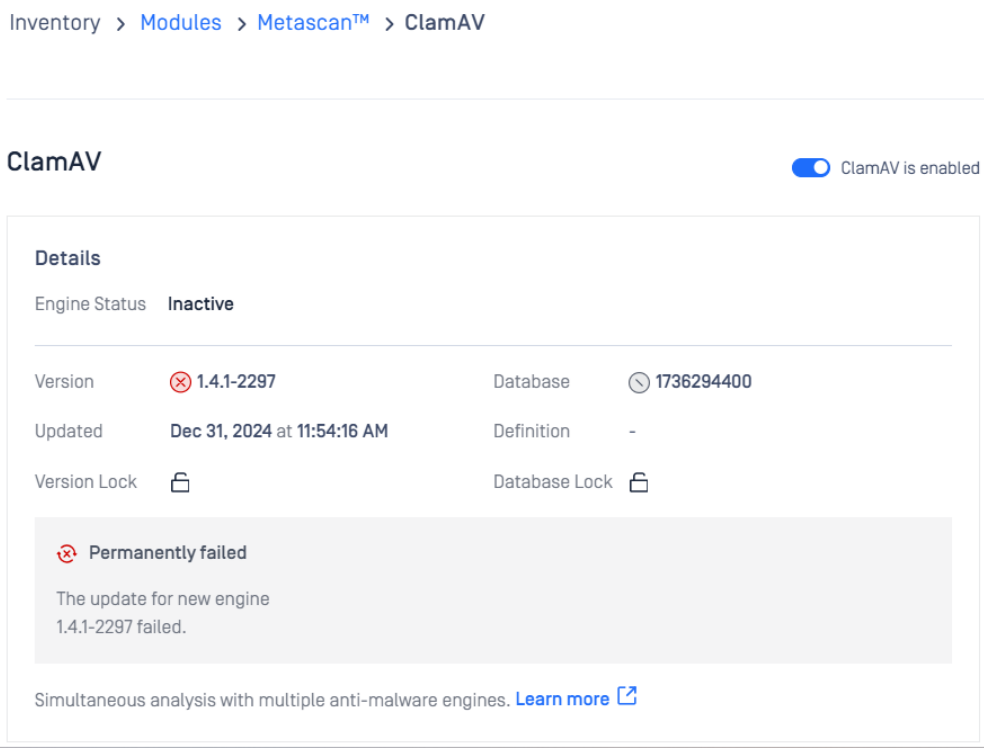
MetaDefender Core is pulling module updates from OCM, to avoid SSL issue:

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.

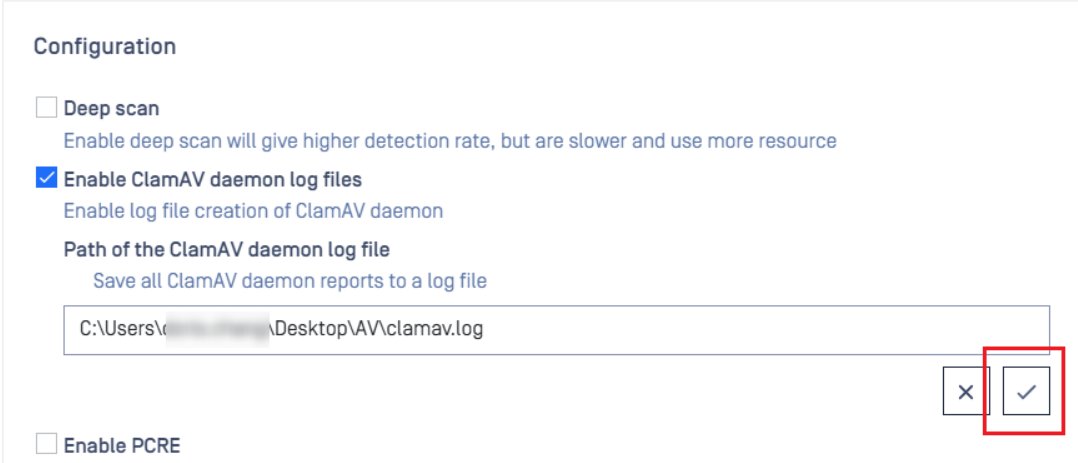
Cause:
This is intended behavior. The ClamAV engine requires a valid path for the daemon log file to function correctly. Without it, the daemon cannot initialize properly.

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:

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?**, 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.

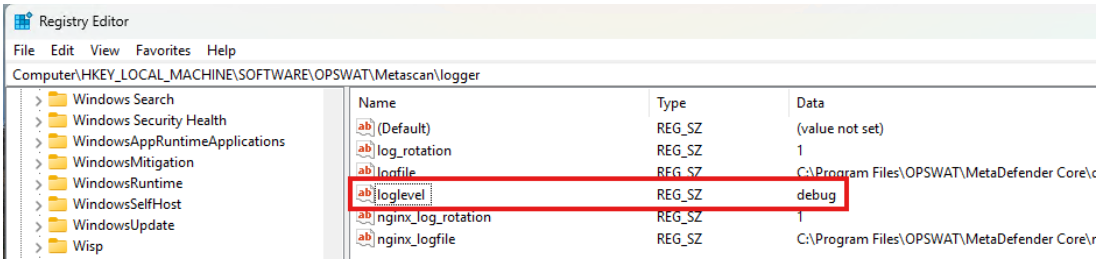
> **Note:**
> To assist the OPSWAT Support team in diagnosing issues more effectively, we may require debug logs from your MetaDefender Core for a detailed analysis.

Enabling debug logging level on MetaDefender Core should only be used when absolutely necessary, and advised by OPSWAT team.

To enable debug logging for MetaDefender Core, please follow the appropriate guide based on your operating system:

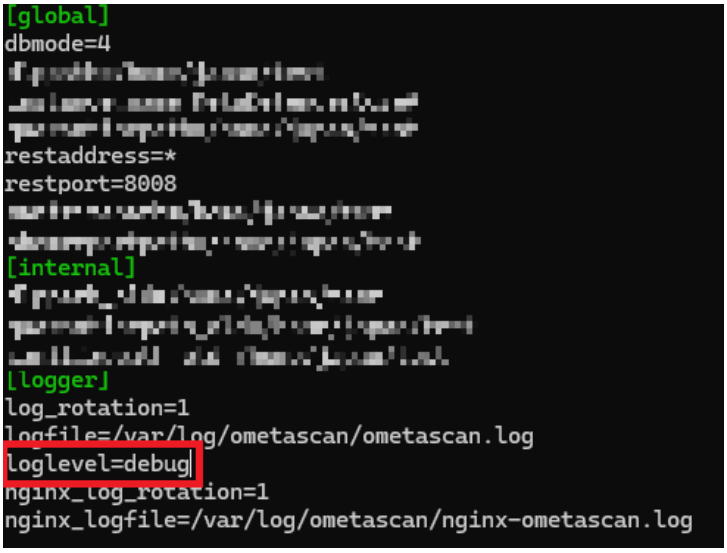
Windows

1. Open Registry Editor.
2. Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\logger`.
3. Change the value of `loglevel` to `debug`.
4. Restart your MetaDefender Core service.



Linux

1. Open the file at `/etc/ometascan/ometascan.conf` with your preferred text editor.
2. In the **[logger]** section, change the value of `loglevel` to `debug`.



3. Restart your 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

After you have enabled the debug logging mode, try to reproduce the issue that you are currently have. Next, please run generate the Support Package and send to the Support team so we can effectively investigate your issue.

For detailed guide on Generating Support Package, please follow this URL:

- How to Create Support Package With Web Console - MetaDefender Core
- 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.

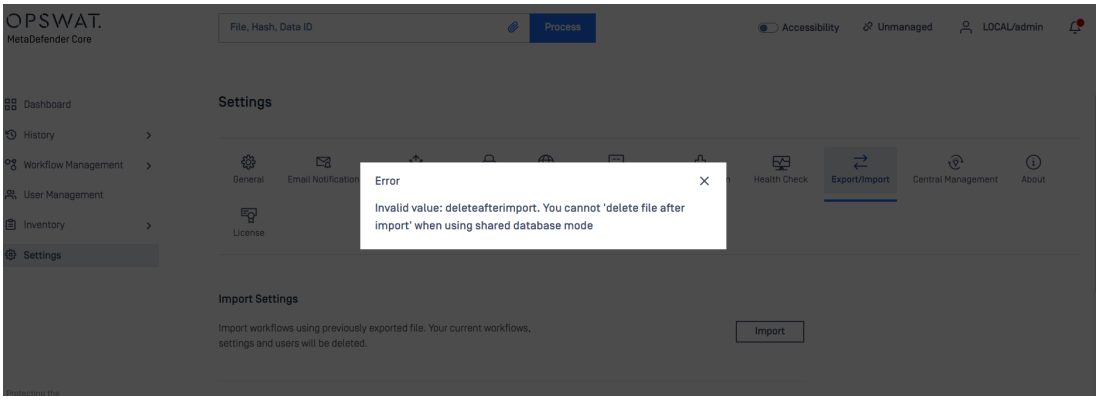
Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

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.

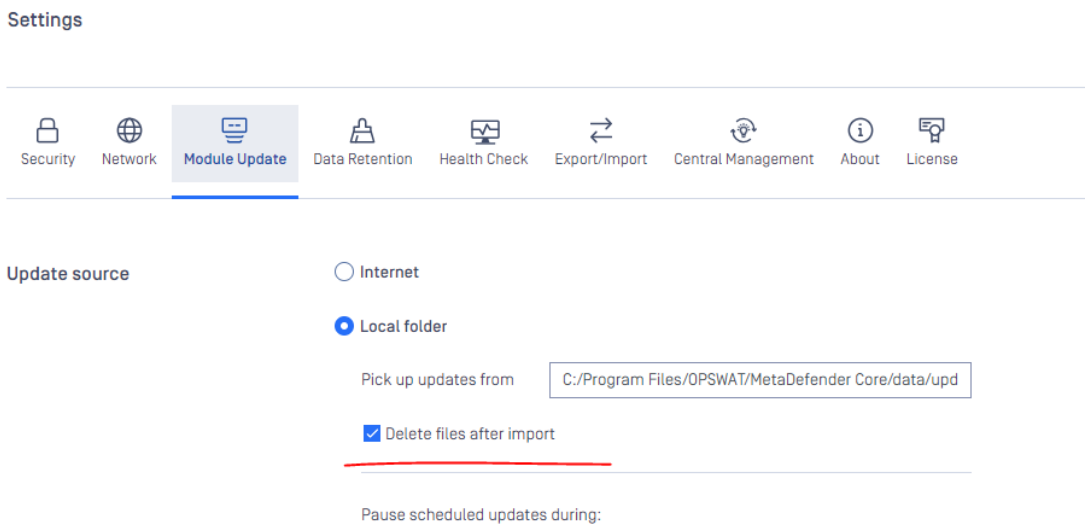
Issue: An error message, **“deleteafterimport”**, appears when a user imports an MD Core configuration from a standalone database mode instance to an MD Core instance using shared database mode.

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:

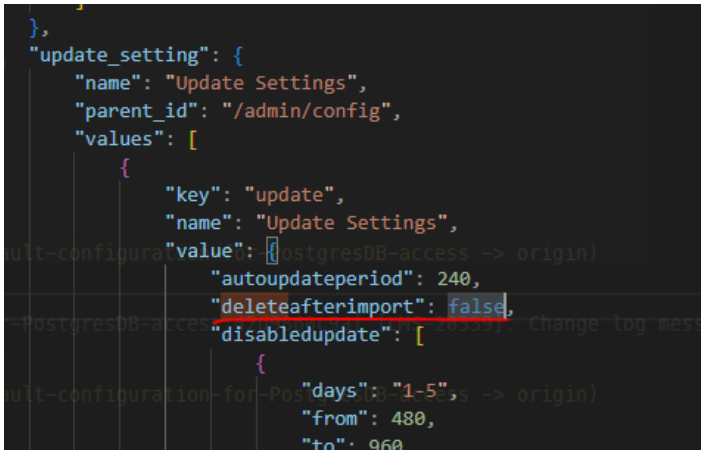
The `deleteafterimport` configuration is not valid for **Shared Database Mode**. This option specifies that once **MD Core** completes updating the package from the configured folder, it will automatically delete the files in that folder. However, this behavior is not supported in **Shared Database Mode** due to its centralized data management structure.



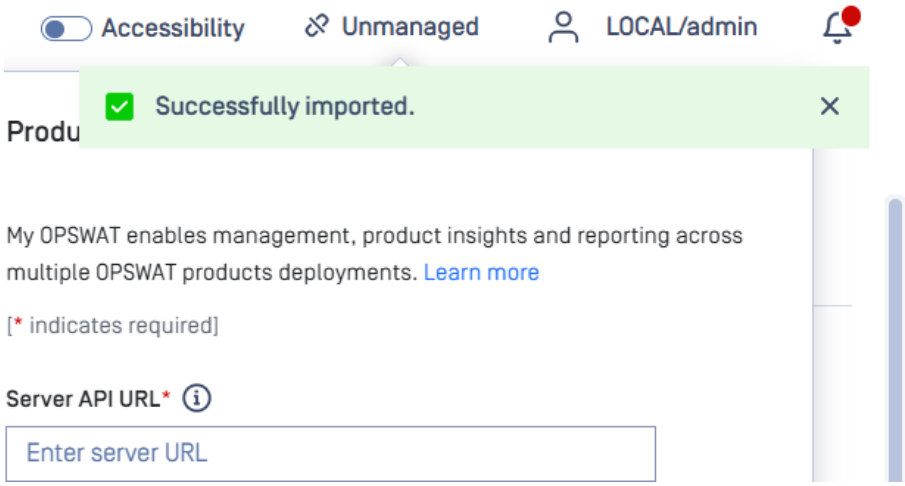
But in shared db mode → we can have multiple MD Core point to the same location (shared folder/network) → this option is always disabled in shareddb, because if not → it will delete all files → other MD Core cannot have the files → failed to updates.

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?

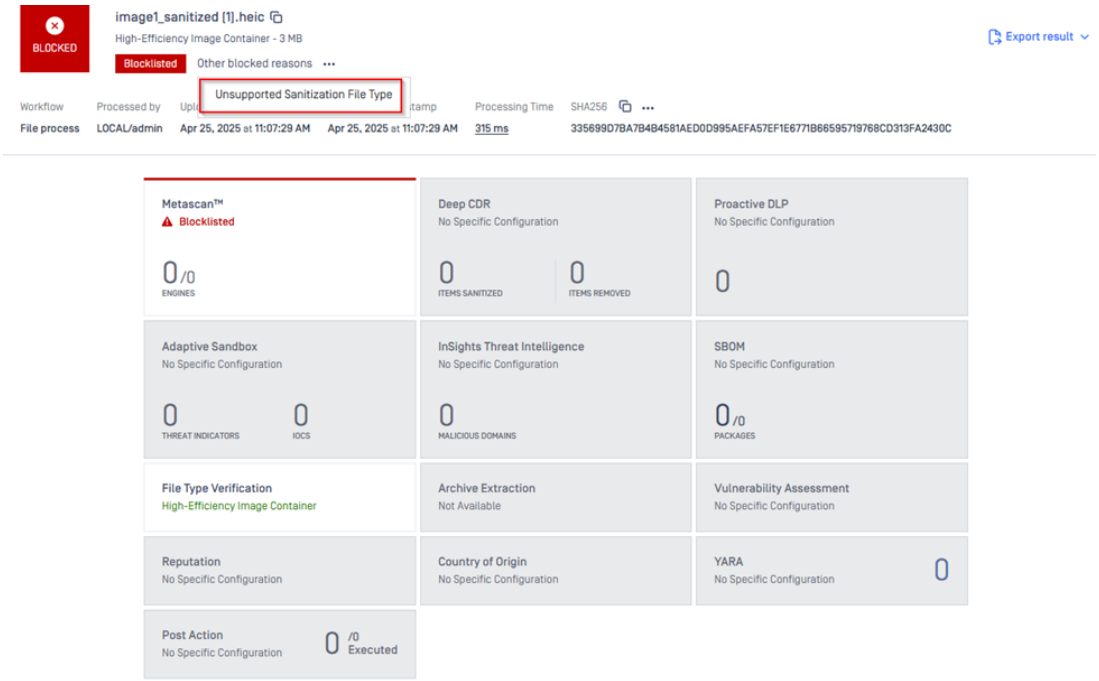
Issue:

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.

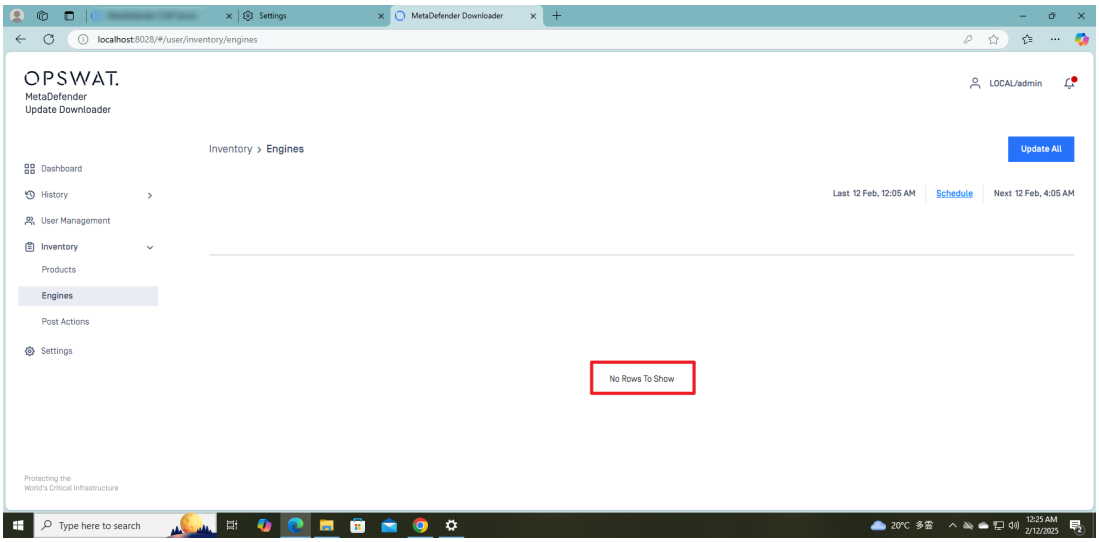
Our support team will be able to provide you with the necessary guidance and resolution for any license-related concerns.

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.

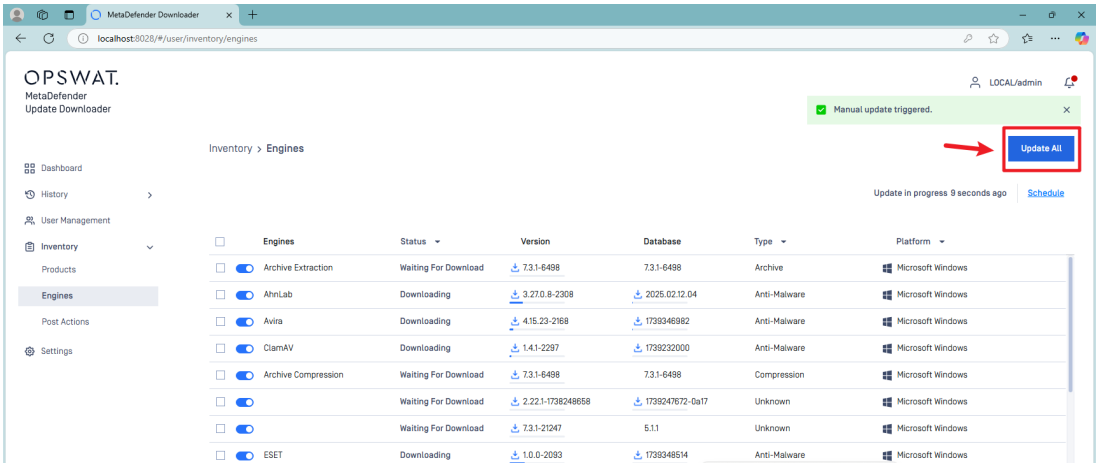
As a result, you may encounter the error: "No rows to show."



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.

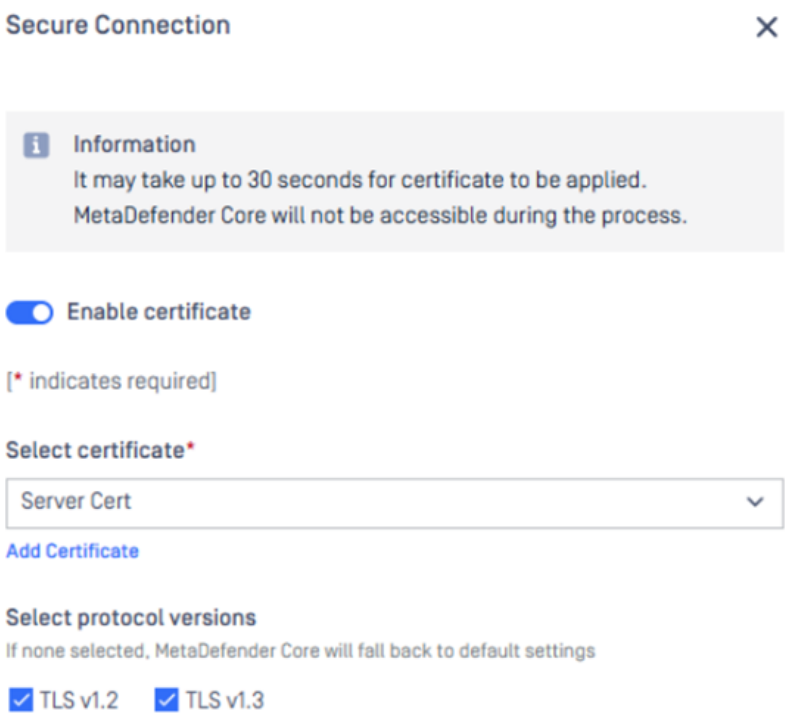
SSL customizations are easy to manage in MD Core v14+. We recommend using the new Secure Connection Settings shown below instead of the conf file. For best security please stick with a **minimum** of TLSv1.2.

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:
https://www.opswat.com/docs/mdcore/metascan-engines/anti-malware-vendors

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:
https://www.opswat.com/docs/mdcore/operating/inventory-management#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?, before creating a support case or chatting with our support engineer.

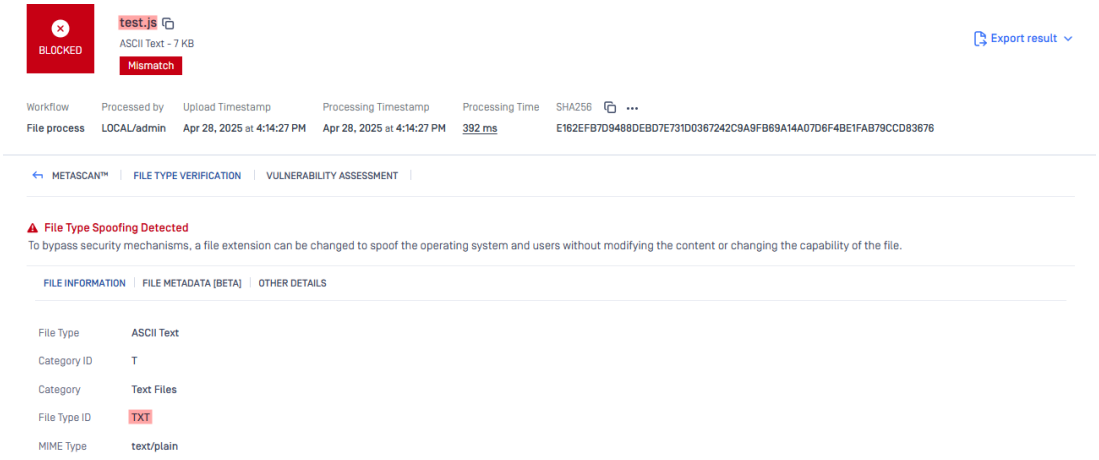
How do I fix files being blocked due to unexpected "File type mismatch" verdict?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

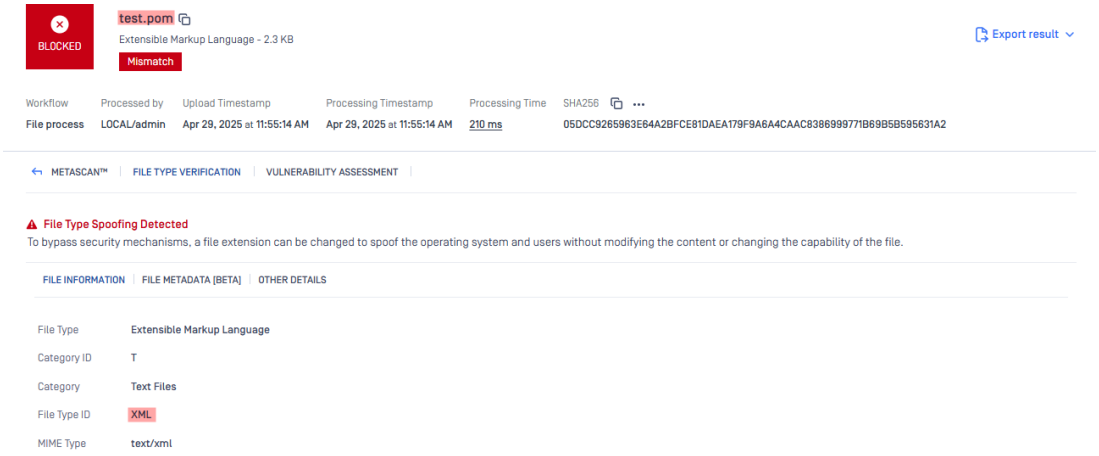
Behavior

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".

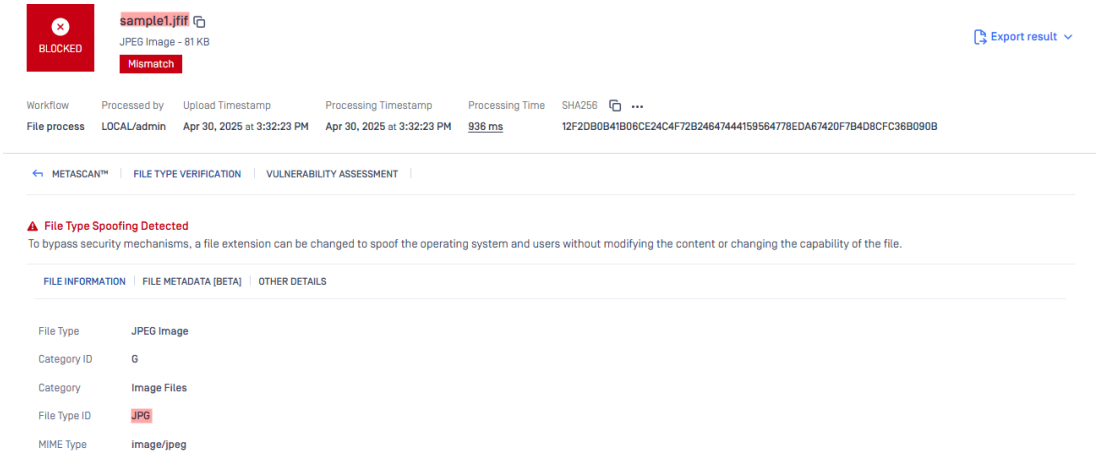
- Example 1: JavaScript source file being detected as TXT (plaintext) and blocked



- 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

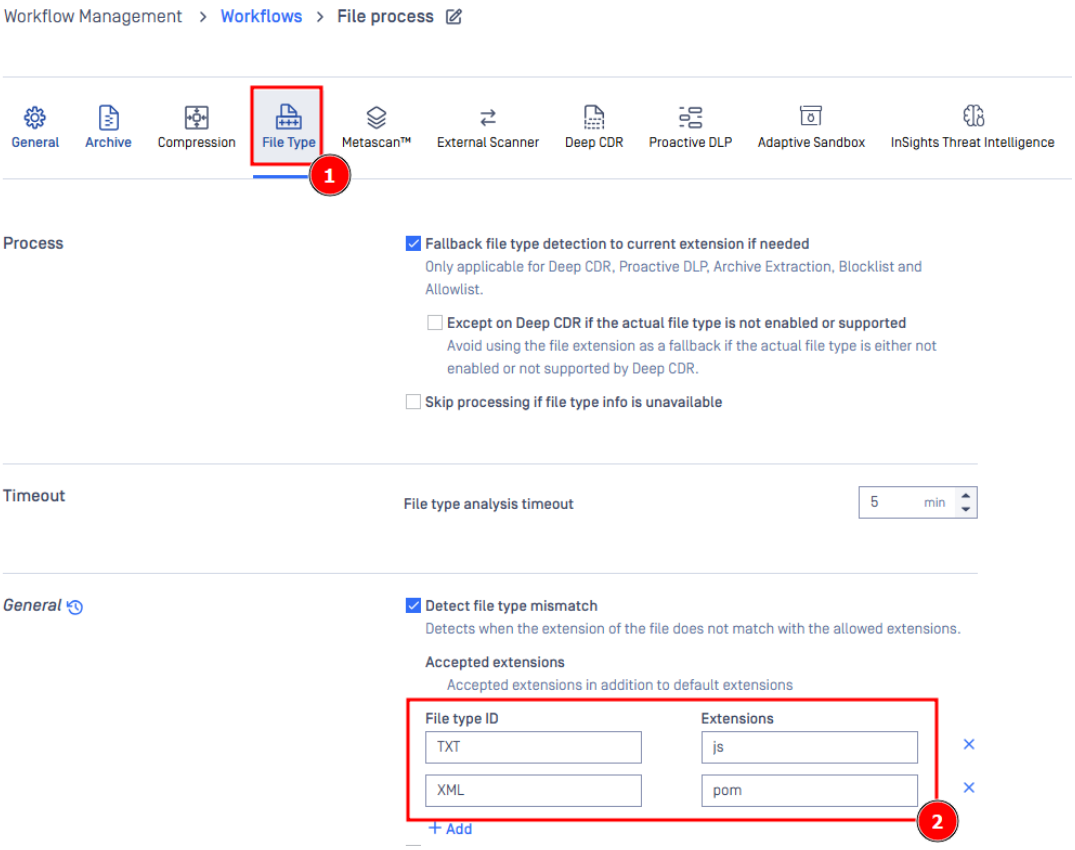


In these examples, the underlying file type used by JavaScript, POM, JFIF files are plain text, XML, and JPEG respectively, even though their names have the extensions .js, .pom, and .jfif. Thus, that's the type detected by File type engine.

Remediation

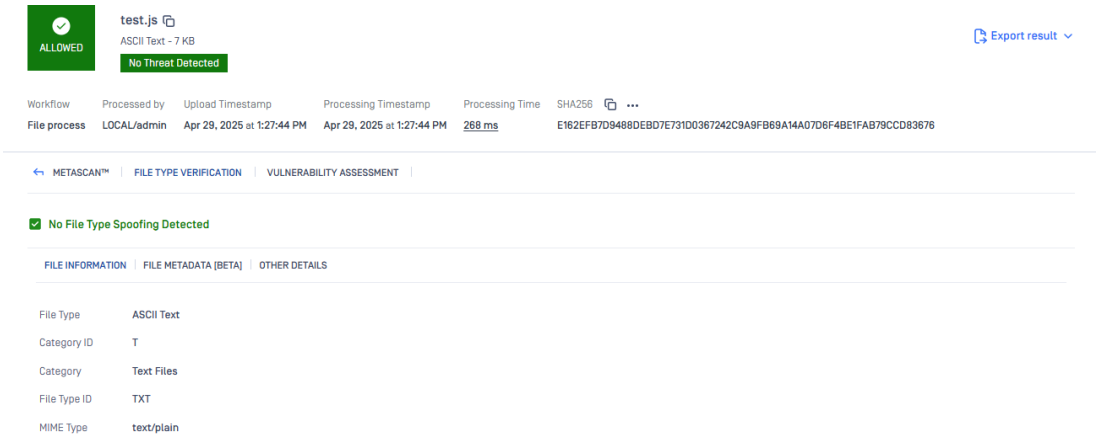
To avoid blocked verdict due to File type mismatch, there are 2 configurations you can use.

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 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):



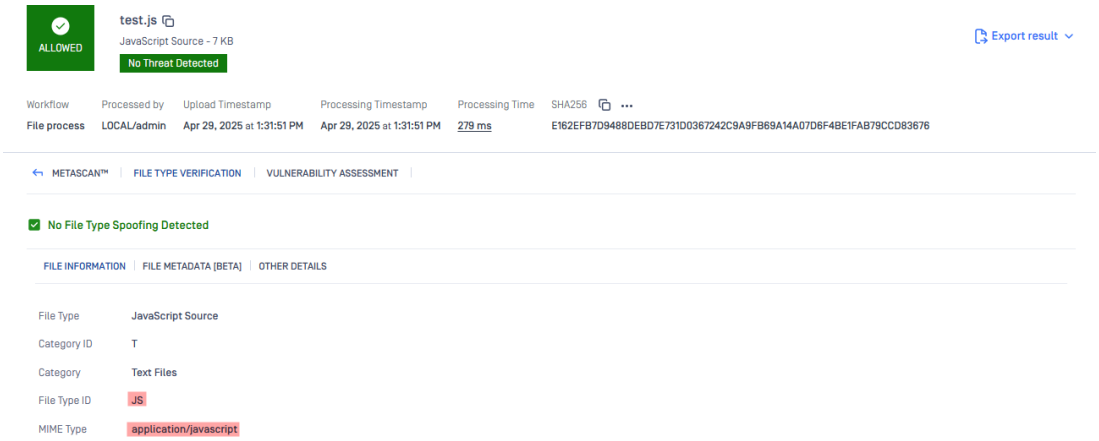
-

- 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

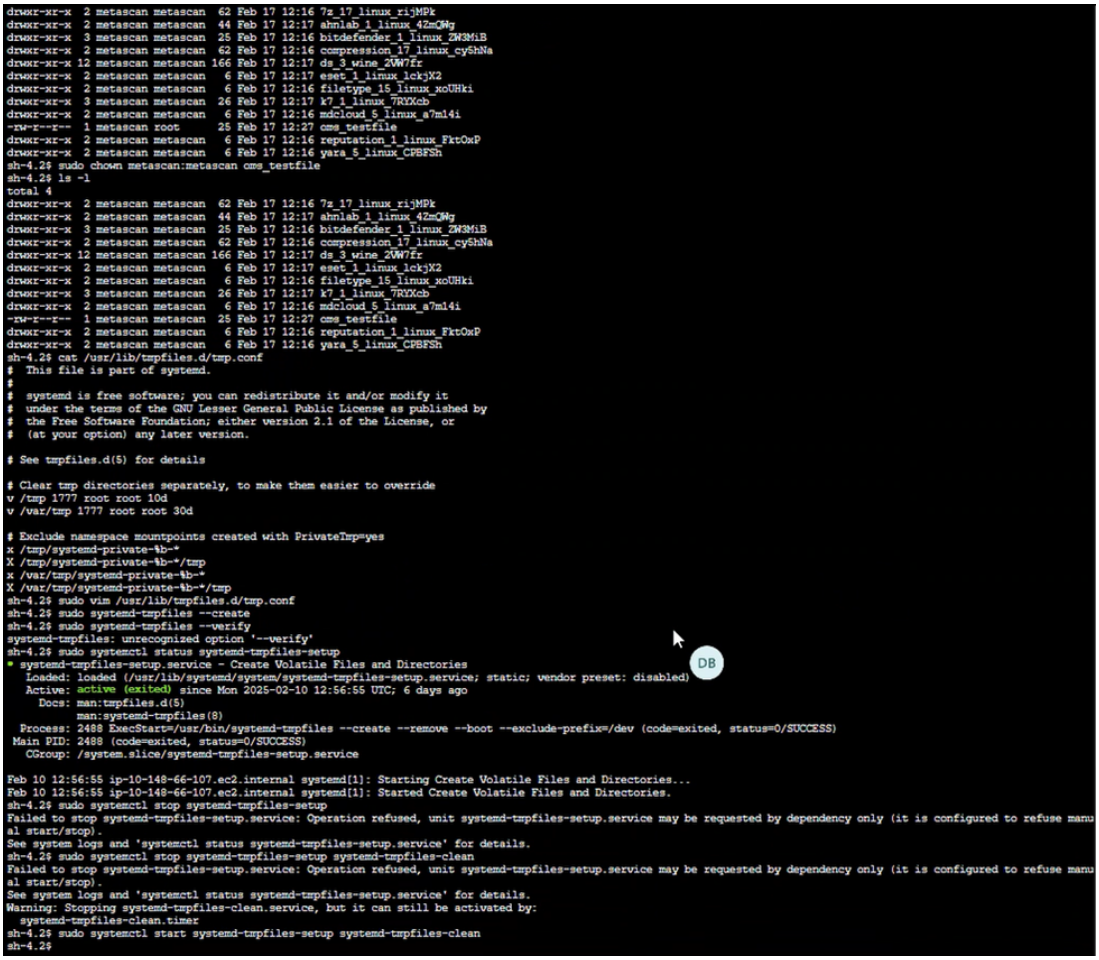
```
:wq
```

- 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.



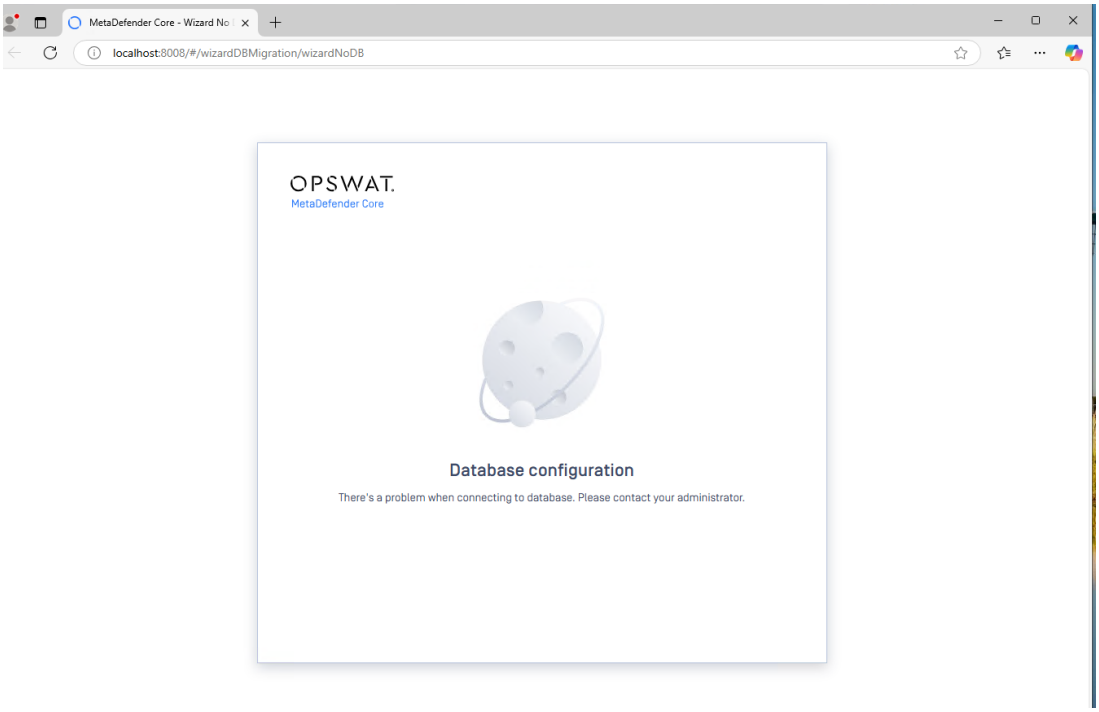
Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

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]
```

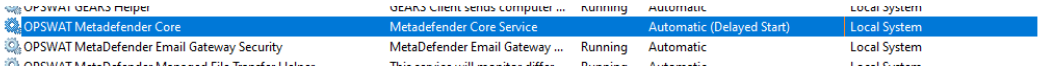
The root cause

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 **Engine 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:
    a. **Check Internet Connection**: Verify that the MetaDefender Core server has a stable internet connection. This is necessary for connecting to the update server.
    b. **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)
    c. **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.
    d. **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**
    e. **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.
    a. Have a backup version (snapshot, recovery point,...)
    b. Import/Export your configuration settings.
    c. 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 can I create a support package?
- 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:

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.

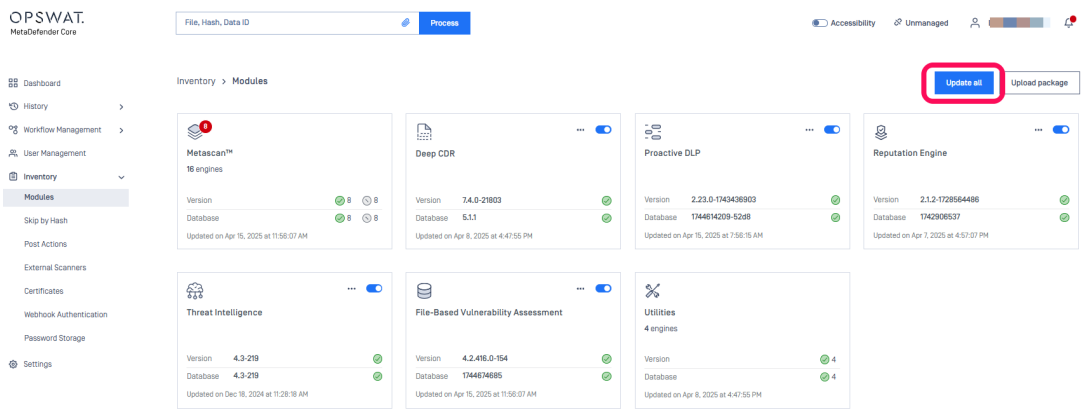
4. Check Internet Connection (if using online updates)

- 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:

docker

```
--shm-size=1gb
```

For **Kubernetes (K8s)** deployments, include this in your YAML:

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.

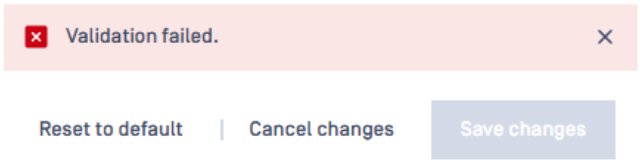
Best Practices

- 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

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:

https://www.opswat.com/docs/mdcore/configuration/metadefender-configuration#windows

- 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:

https://www.opswat.com/docs/mdcore/configuration/metadefender-configuration#linux

- 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 to disable the SSL configuration.

For Linux:

Please run the script /usr/sbin/ometascan-reset-ssl-conf

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.

Default paths:

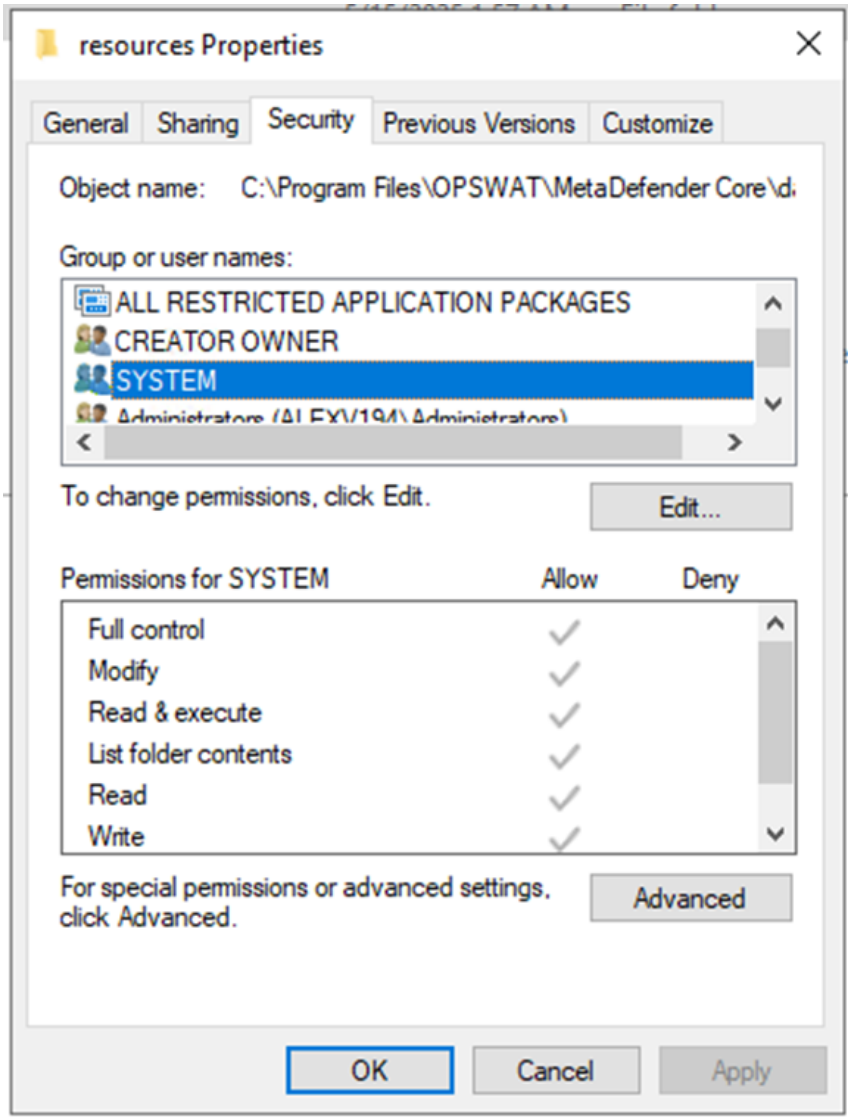
- **Windows**: C:\Program Files\OPSWAT\MetaDefender Core\data\resources\
- **Linux**: /var/tmp/ometascan/resources

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

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:

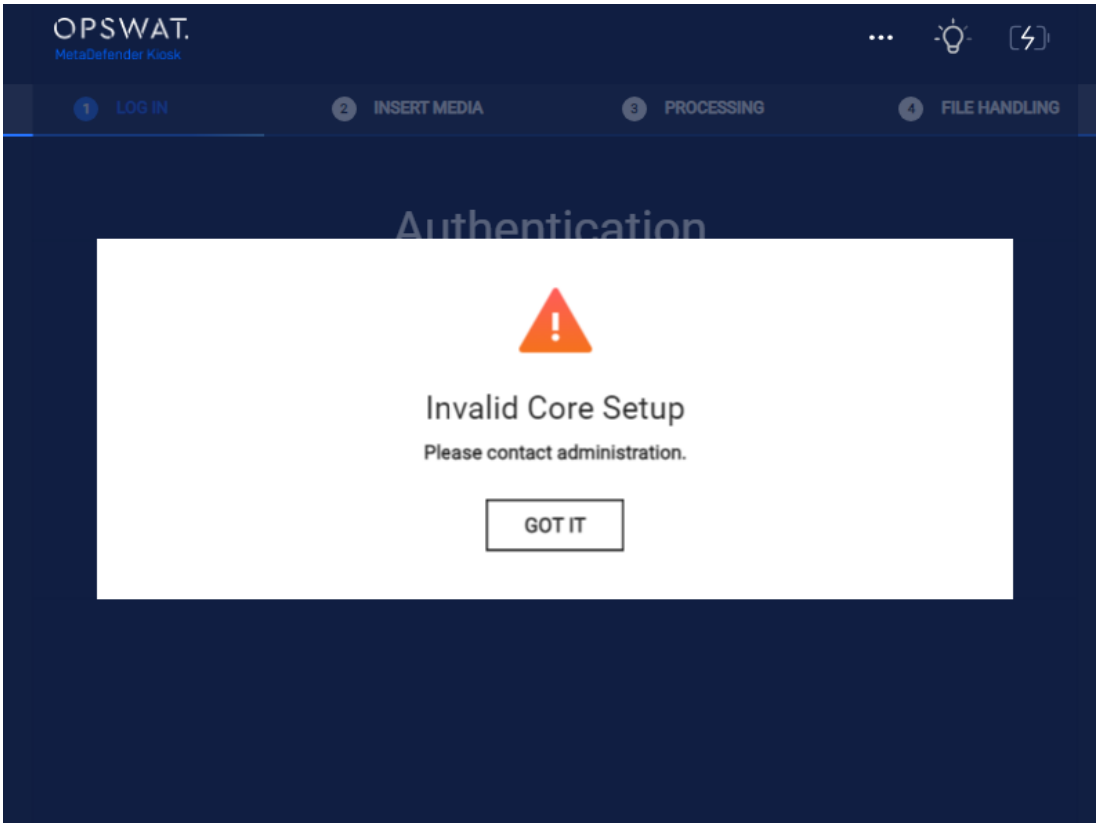
How to Create Support Package With Web Console - MetaDefender Core

How to Create Support Package With Bundle Tools? - MetaDefender Core

> **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.

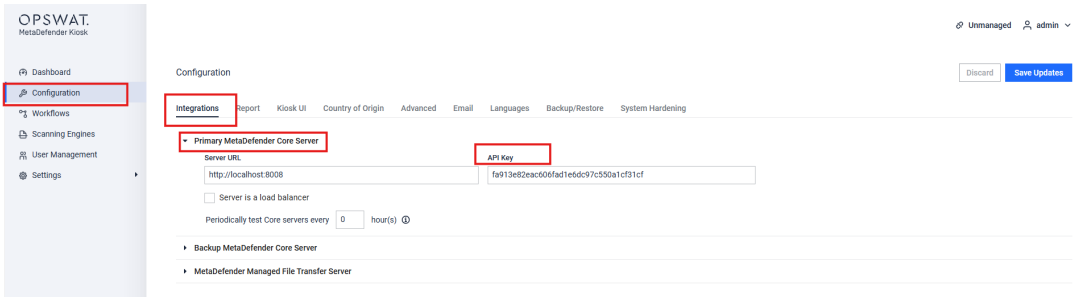


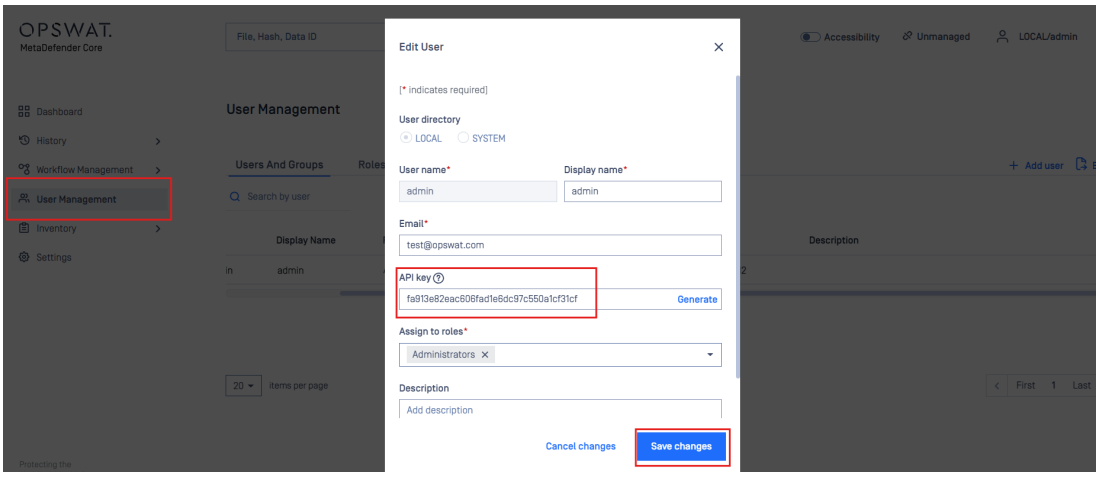
Introduction

There are many different reasons why you may view the “Invalid Core Setup”, please review the below:

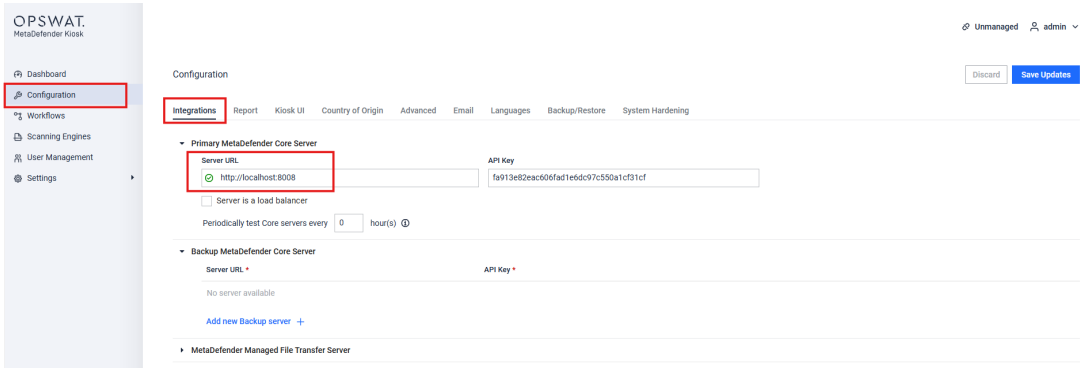
1. **Incorrect API key**

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

*

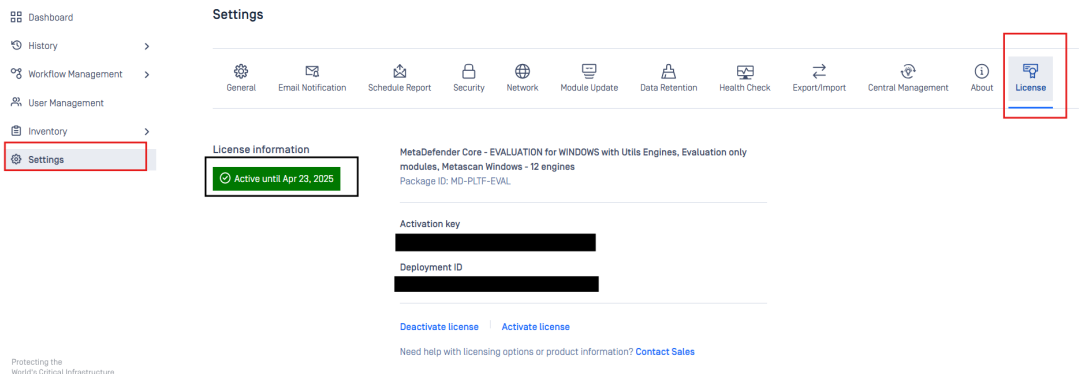
If MetaDefender Core has a certificate enabled for HTTPS, please use the https://<FQDN>:8008

Example: https://certificateSAN:8008

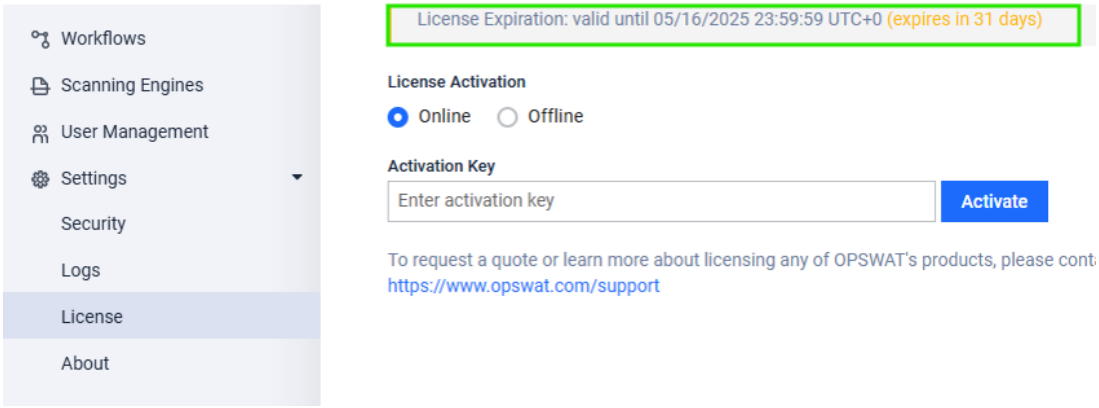
3. **Unlicensed MetaDefender Core or MetaDefender Kiosk**

Confirm that both the MetaDefender Core and MetaDefender Kiosk are properly licensed.

MetaDefender Core:



MetaDefender Kiosk:



How to activate Kiosk offline: https://www.opswat.com/docs/mdkiosk/installation/managing-license-information

4. **Network connectivity issue between MetaDefender Core and MetaDefender Kiosk**

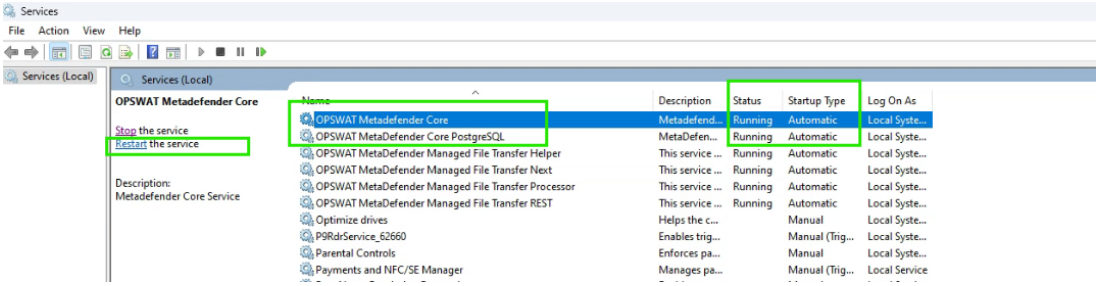
Check for network issues that might prevent communication between the Kiosk and Core. Consult with your networking team if necessary. Example: Firewall, proxy, low bandwidth

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**

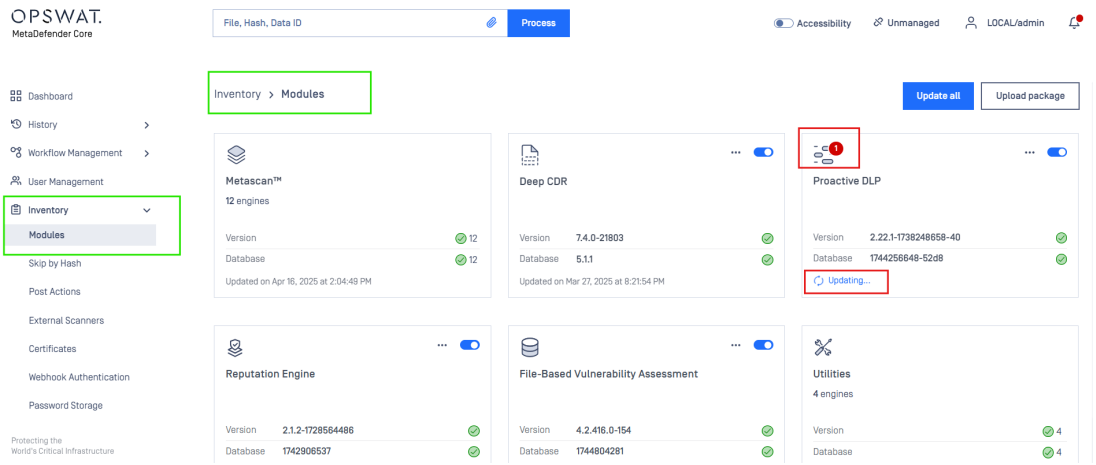
Verify that the MD Core services are running properly. Restart them if necessary.



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 “System Requirements” 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.

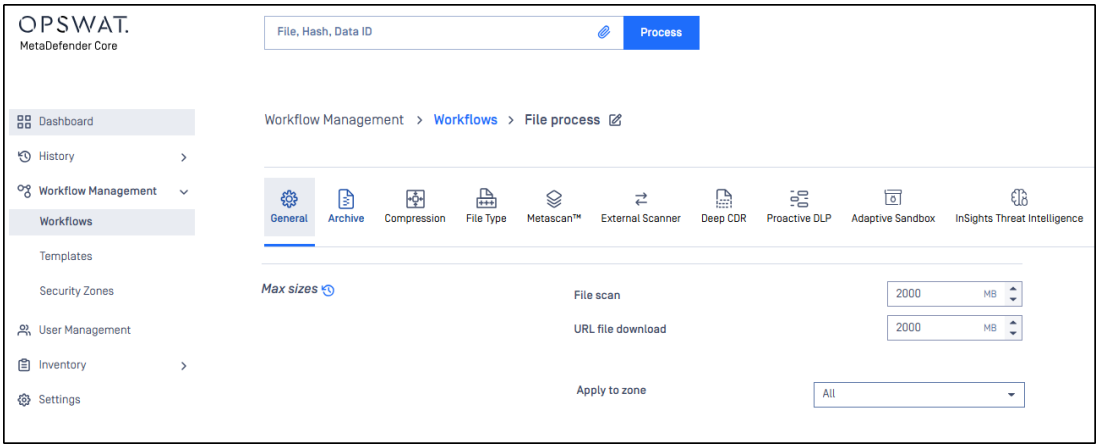
Comparison of Unit Systems

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

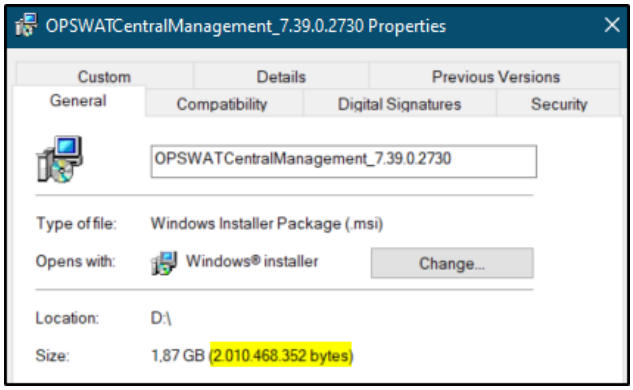
Suppose your MDCore upload limit is configured as **2000 MB (decimal)**:

- **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

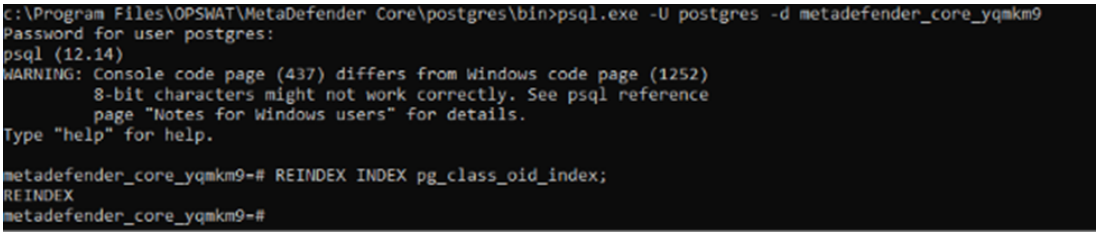
- Replace <your_postgres_username> with the correct PostgreSQL username used during installation.
- If prompted, enter the PostgreSQL password.

4. Run the Reindex Command

Once connected to the database prompt enter:

Reindex index scan.request1 pkey;

Please see the following example below:

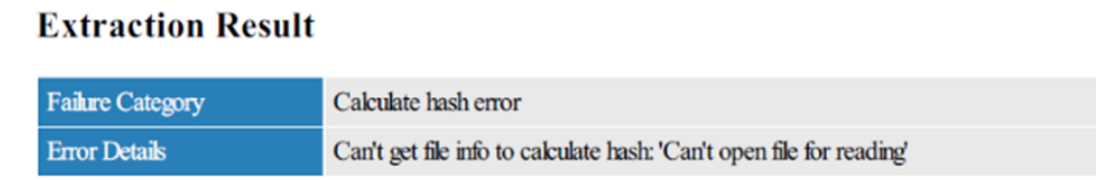


Support:
If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

Why Does MetaDefender Core Show a 'Calculate hash error' When Scanning Malicious Files?

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

To resolve this issue, you need to disable RTP or add exclusions for MetaDefender Core directories in your local antivirus software.

Recommended Steps:

1. Disable Real-Time Protection (RTP)

- This can usually be done through your antivirus software's settings panel.

OR

2. Add Exclusions in Antivirus Software:

- Exclude the following directories from real-time scanning:
  - C:\Program Files\OPSWAT
  - C:\Users<YourUsername>\AppData\Local\Temp

Replace <YourUsername> with the actual Windows username running the MetaDefender Core service.

3. 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.

Support:
If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

How to Use a RAMDISK for the tempdirectory?

In order to improve the file scan speed, a custom *tempdirectory* can be set for Metadefender Core.

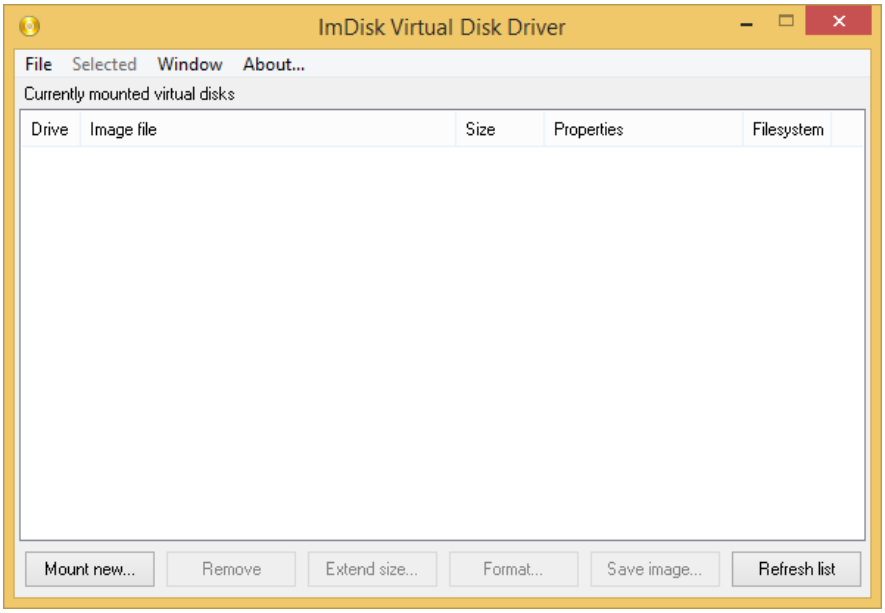
Instructions for Windows

Step 1:

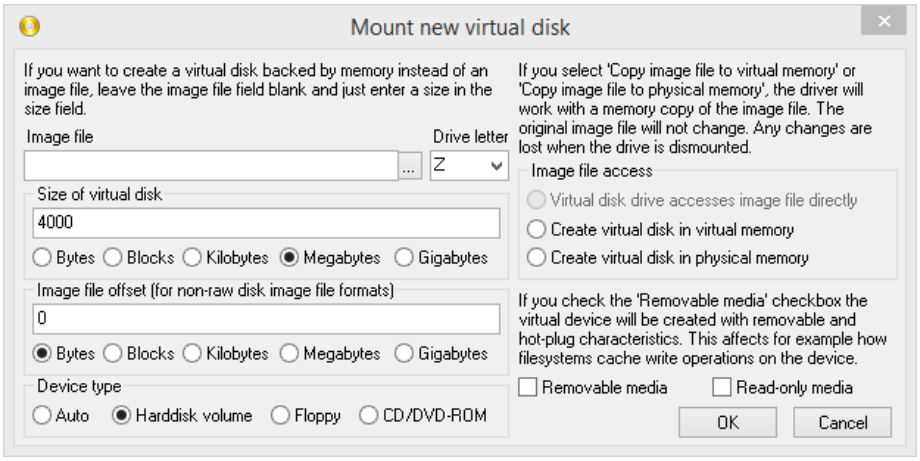
Create a RAMDISK on your system.

We recommend the following tool for this: http://www.ltr-data.se/opencode.html/#ImDisk

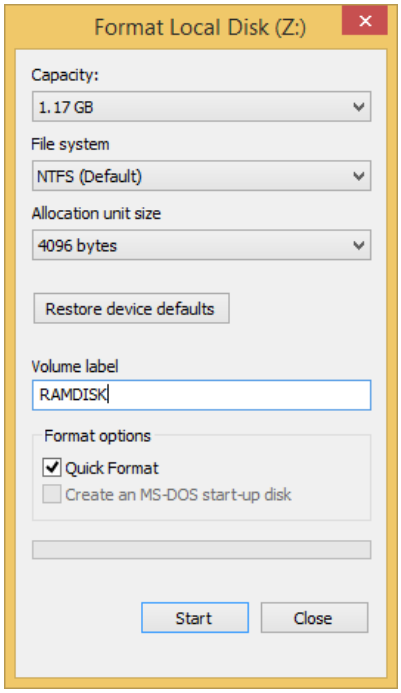
After installation, use the **ImDisk Virtual Disk Driver** application, to create a new RAMDISK.



Mount new virtual disk:



After disk creation, windows will ask you to format the new disk.

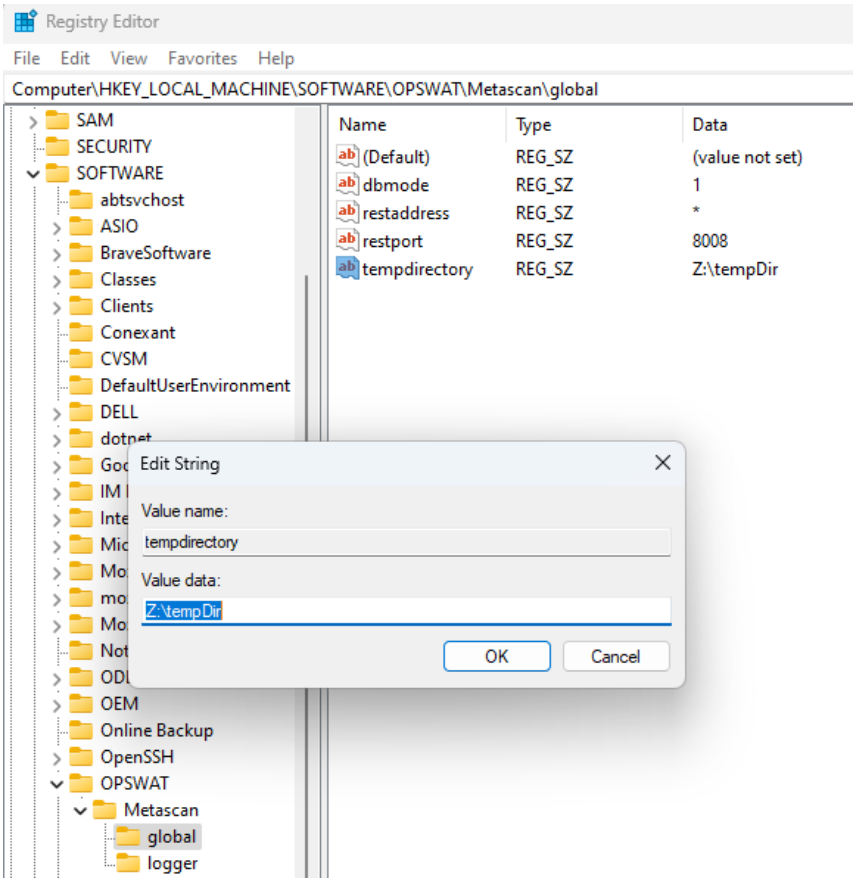


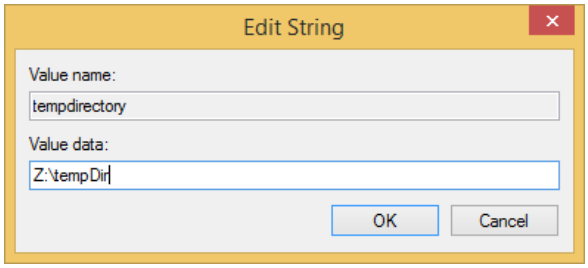
Step 2:

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.

Why does ClamAV seem to be slower than other engines?

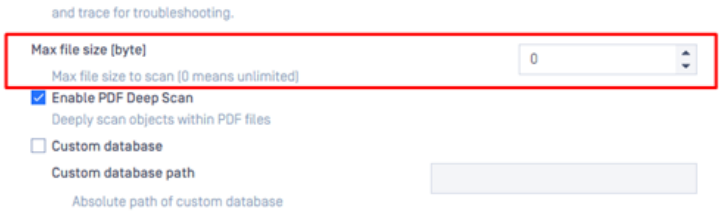
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

Due to certain engine-specific design factors, ClamAV is known to be one of the slower performing engines included in the MetaDefender Core system.

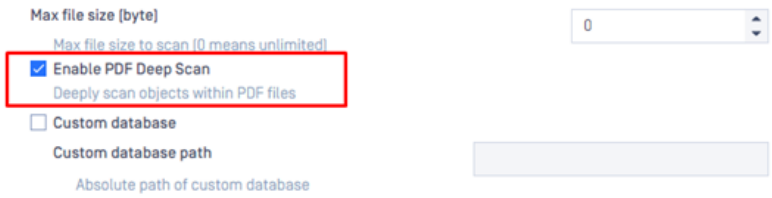
However, ClamAV's slow scan speed is easily counterbalanced by its high detection rate, quality, and stability.

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 auto$, before creating a support case or chatting with our support engineer.

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 auto$, 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 auto$, 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 auto$, 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 auto$, 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**.

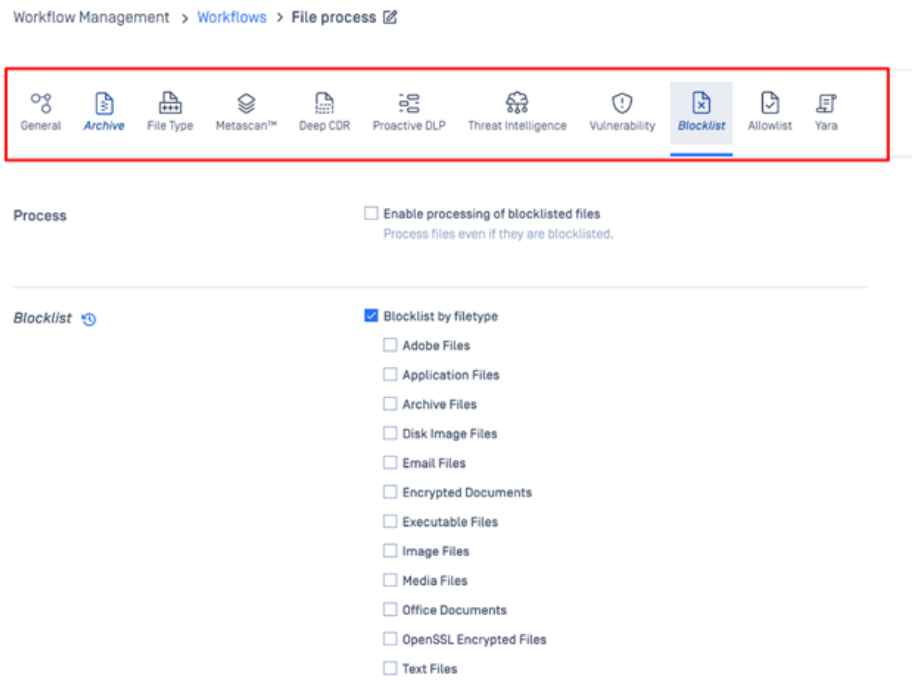
The workflow is identified by its name.

- 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 auto$, 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 auto$, 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 V4 and V5 come equipped with a queue mechanism designed to streamline and manage high-volume data processing (i.e. file scans).

The article below outlines the basic functioning of the MetaDefender Core queue mechanism.

The main topics include:

- **Queue mechanism**
- **Queue size**

Queue mechanism

MetaDefender Core manages its processing tasks in a queue.

Please Note:
Each item in the queue is handled by a **Workflow** (a thread) on the system. MetaDefender Core starts processing the queue on a 'first come first served' basis. However, this doesn't facilitate determination of the run-time for overall processing.

MetaDefender Core uses a maximum of a quarter (25%) of its maximum queue size for archive processing (this applies to all archives processed at a time, not to each one).

This means that, if you send only one archive file into the queue, the archive extraction will fill the queue up to only 25% of the queue size. It does this not only to leave room for further file submissions, but also to provide parallel processing capacity.

Queue size

The default queue size is set to 500.

The number of items in the queue can be extracted from the results of the REST API call **GET /stat/nodes**.

Pro Tip:
To increase or decrease this queue value, please read auto$ do I configure.

Support:
If you require further guidance around MetaDefender Core's **Queue Mechanism**, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

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.

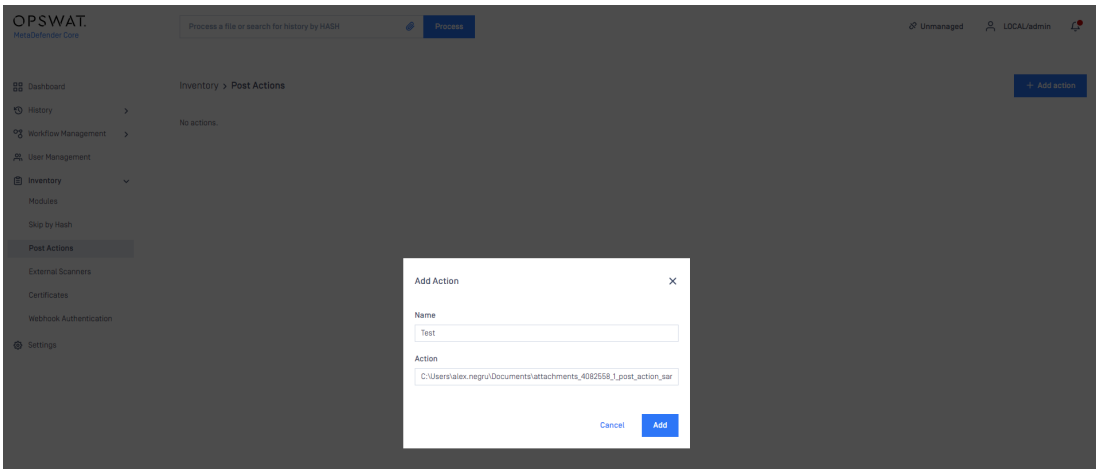
Disclaimer
The following sample script is provided for illustrative purposes only and is not guaranteed to be functional in a production environment. Please be aware that script only works Windows platforms.

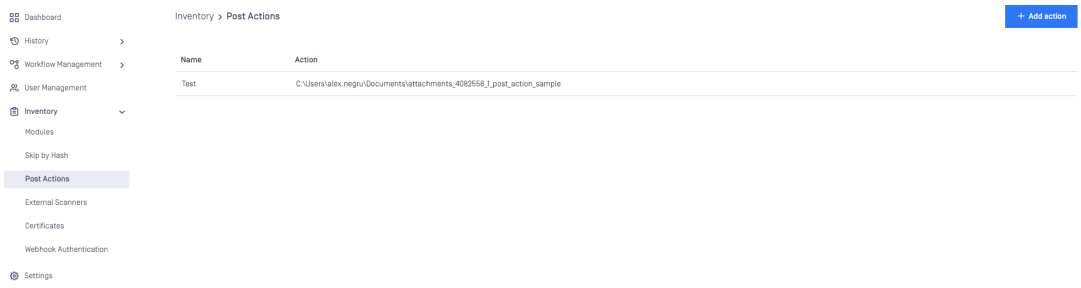
MetaDefender Core V4.8.0 and all subsequent releases include the **Post Actions** feature:

- 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 auto$, 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:

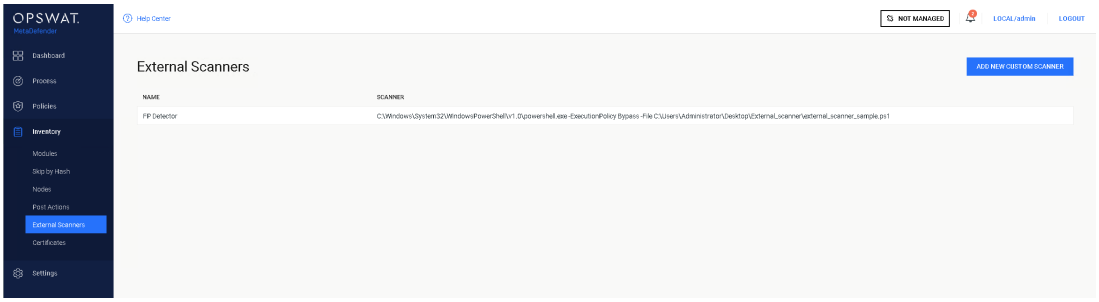
- This feature allows you to define an **External Scanner** which is then invoked through a command line executable or script.
- This executable/script will be called for each scanned file, after all the other engines have scanned - but before the final verdict is decided.
- Additional documentation 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 auto$, before creating a support case or chatting with our support engineer.

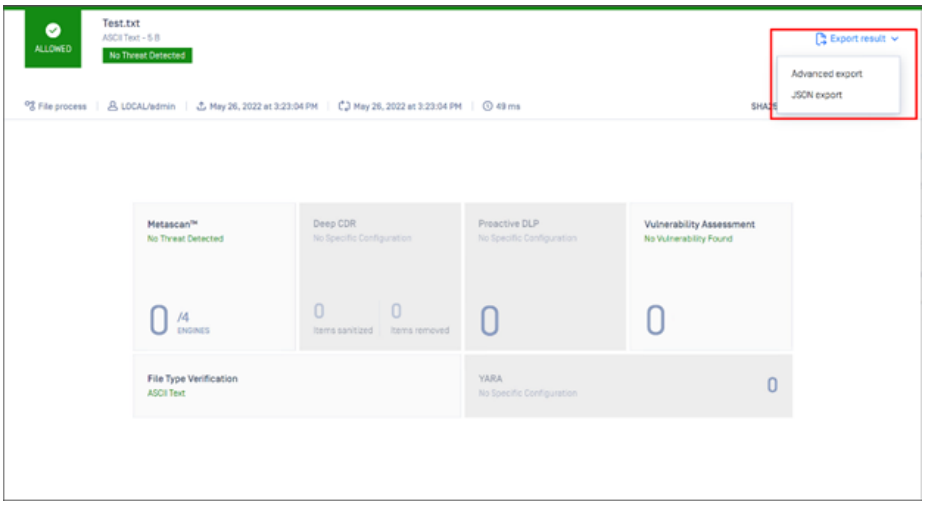
How to generate a Core's scan result summary in JSON format

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)

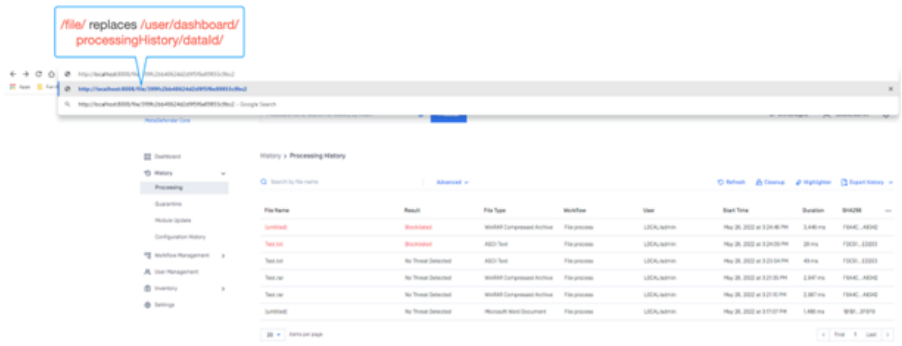
1. On the MetaDefender Core Scan Result Page, modify the URL as follows:

http://localhost:8008*/#/user/dashboard/processingHistory/dataId/*399fc2bb48624d2d9f5f6e89855c9bc2

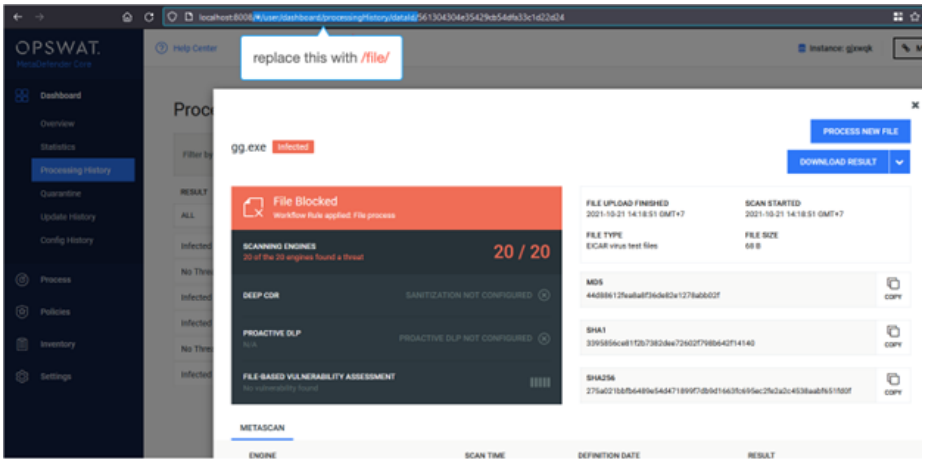
=> http://localhost:8008/file/399fc2bb48624d2d9f5f6e89855c9bc2

Or, as illustrated in the relevant screenshot below:

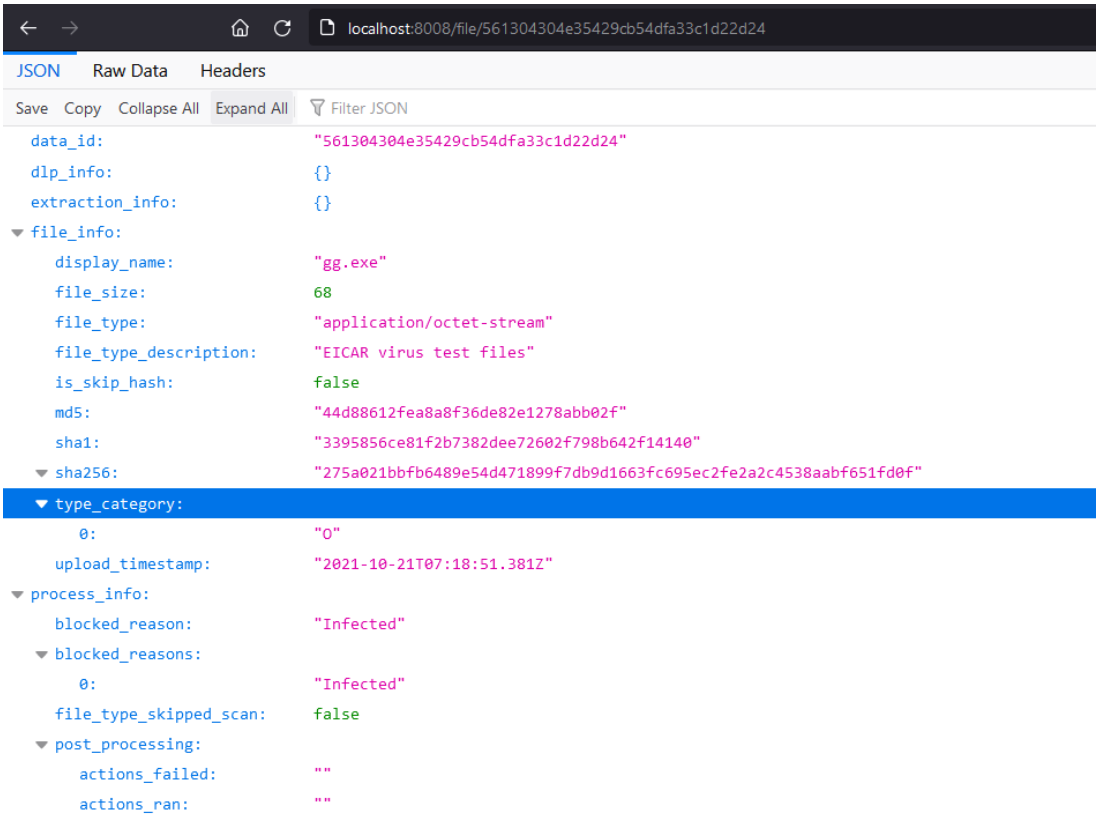
V5:



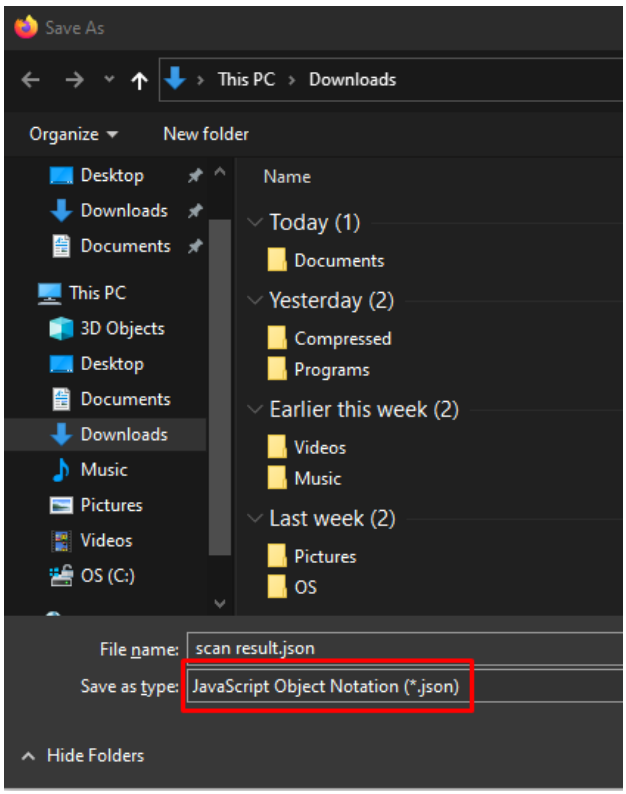
V4:



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 auto$, 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 to 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: https://www.opswat.com/products/metadefender/enterprise
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:

https://blog.avira.com/petya-strikes-back/

ESET:

https://www.eset.com/us/about/newsroom/corporate-blog/petya-ransomware-what-we-know-now-3/

Bitdefender:

https://labs.bitdefender.com/2017/06/massive-goldeneye-ransomware-campaign-slams-worldwide-users/

https://labs.bitdefender.com/2016/04/low-level-petya-ransomware-gets-bitdefender-vaccine/

Quick Heal:

http://blogs.quickheal.com/petya-ransomware-affecting-users-globally-things-can/

VirlTeXplorer:

http://www.tgsoft.it/italy/news_archivio.asp?id=843

MetaDefender Core 12

CYREN:

https://blog.cyren.com/articles/petya-ransomware-spreading-fast-using-same-wannacry-exploit

MetaDefender Core 16

Emsisoft:

http://blog.emsisoft.com/2017/06/27/petya-petna-ransomware/

Zillya!:

https://ru.tsn.ua/ukrayina/v-antivirusnoy-kompanii-rasskazali-kto-mozhet-stoyat-za-hakerskoy-atakoy-petya-a-i-chem-eto-grozit-885812.html

VirusBlokAda:

https://blog.fortinet.com/2017/06/27/new-ransomware-follows-wannacry-exploits

MetaDefender Core 20

McAfee:

https://kc.mcafee.com/corporate/index?page=content&id=KB89540

https://securingtomorrow.mcafee.com/mcafee-labs/new-variant-petya-ransomware-spreading-like-wildfire/

Sophos:

https://nakedsecurity.sophos.com/2017/06/27/breaking-news-what-we-know-about-the-global-ransomware-outbreak/

https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~Petya-AQ.aspx

Linux

MetaDefender Core 5

Bitdefender:

https://labs.bitdefender.com/2017/06/massive-goldeneye-ransomware-campaign-slams-worldwide-users/

https://labs.bitdefender.com/2016/04/low-level-petya-ransomware-gets-bitdefender-vaccine/

ESET:

https://www.eset.com/us/about/newsroom/corporate-blog/petya-ransomware-what-we-know-now-3/

MetaDefender Core 10

Avira:

https://blog.avira.com/petya-strikes-back/

CYREN:

https://www.cyren.com/blog/articles/petya-ransomware-spreading-fast-using-same-wannacry-exploit

Quick Heal:

https://blogs.quickheal.com/petya-ransomware-affecting-users-globally-things-can/

Support:
For more information on virus-specific protections, or to report a suspected vulnerability or threat, please follow these instructions on auto$, 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*
Antiy	900 MB	
Emsisoft	4 GB	
QuickHeal	2 GB	
RocketCyber Linux		Does not function on Redhat and CentOS versions earlier than 8.x.
Cylance		Does not function on Redhat and CentOS versions earlier than 8.x.
Nano AV		Does not function on Amazon Linux 2023
SentinelOne	4 GB	

Minimum System Resource Requirements

Minimum System Resource Requirements

Engine Name	—	*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

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 auto$, before creating a support case or chatting with our support engineer.

Can MetaDefender Core block renamed files (such as renamed .exe files)?

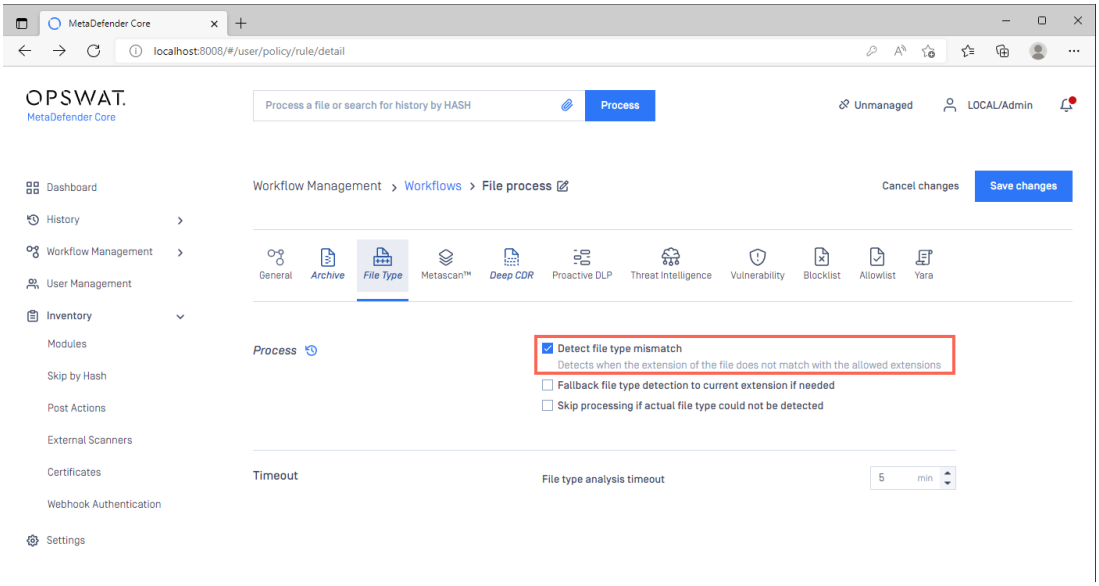
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

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:

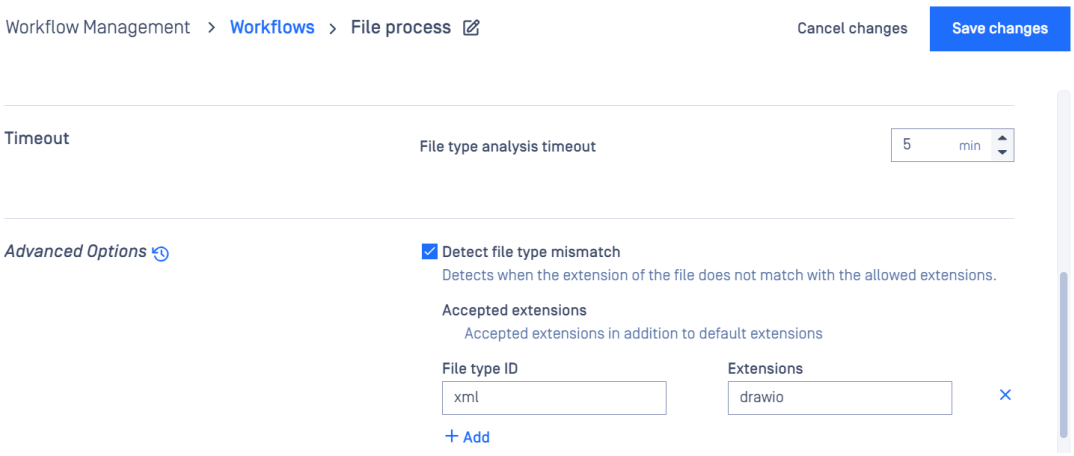
- When enabled, the File Type Verification Engine will work in conjunction with the Deep CDR module to detect and sanitize files according to their true file type.
- This means Deep CDR will only attempt to sanitize according to the given extension in the event that 'true type' sanitization fails.

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.

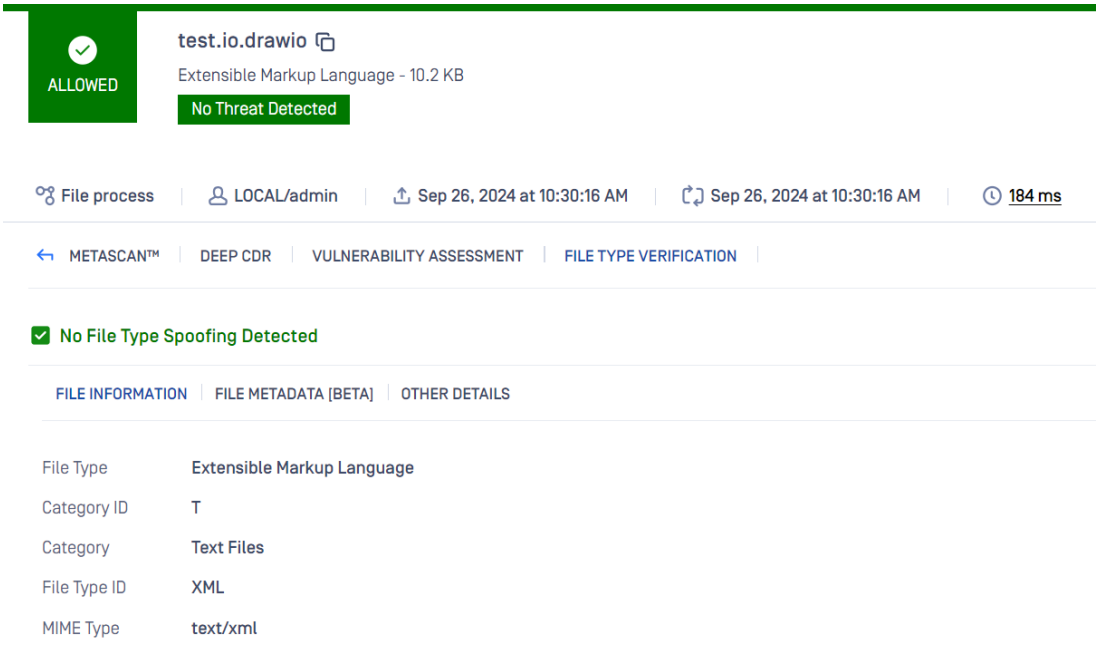
Skip processing if actual file type could not be detected:

- 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.

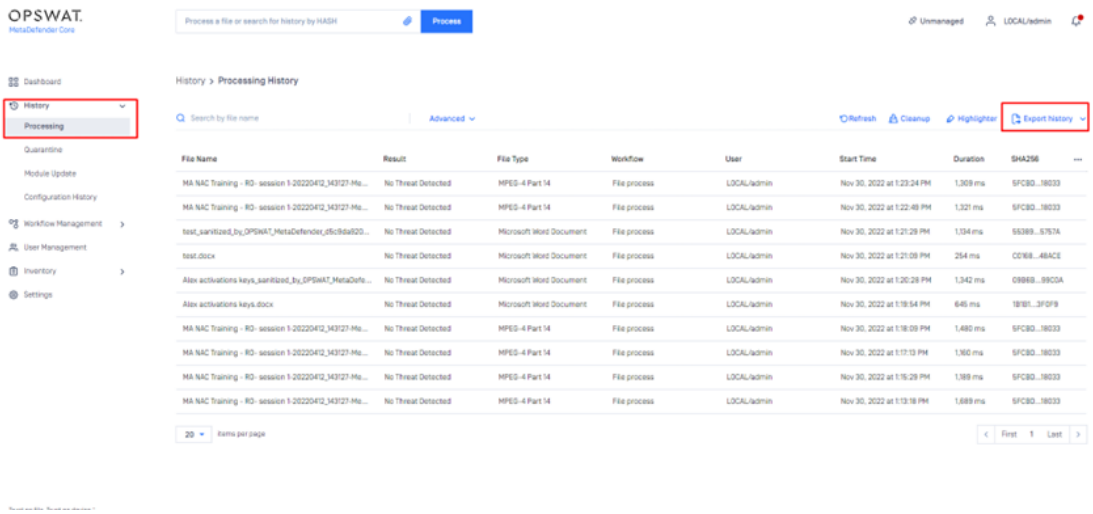


Support:
If you have followed the instructions above but are **Having Difficulty Configuring the Filetype Mismatch Feature**, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

How can I export the processing history from the Core console?

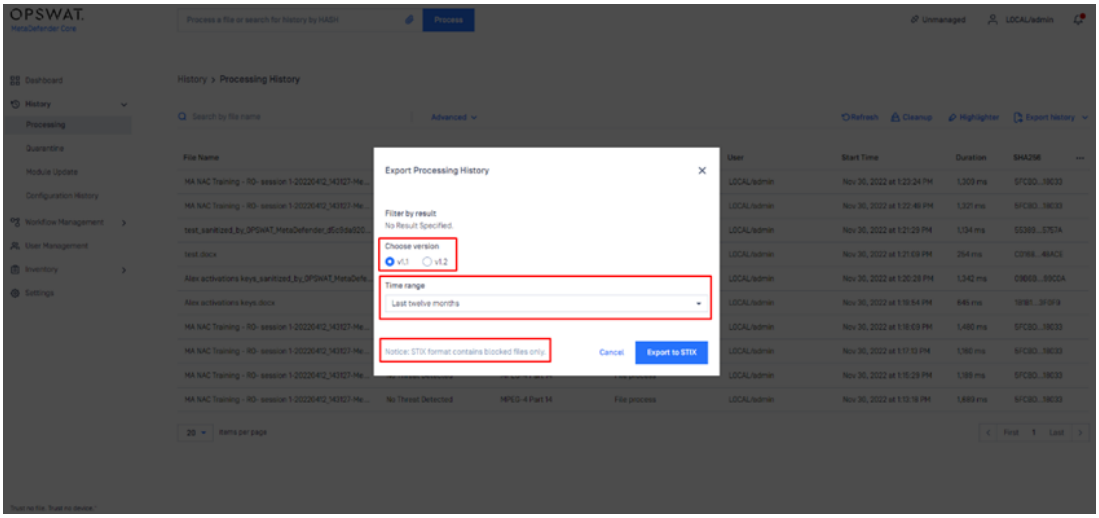
Check Your Version:
This article applies to all MetaDefender V5 releases deployed on Windows and Linux systems.

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



Support:
If **Further Assistance** is required, please follow the instructions on auto$, then proceed to log a support case or chat with our support engineer.

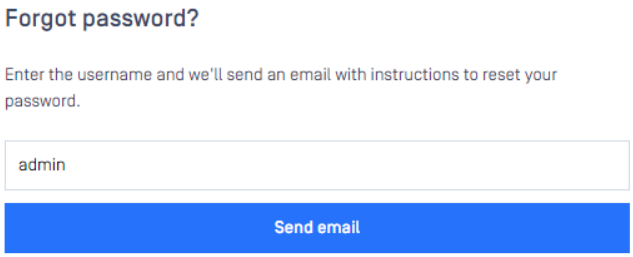
I forgot my login password, what can I do?

Check Your Version:
This article applies to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

The 'Forgot Password' tab will be visible within the login screen. Once the username has been provided, instructions for password reset will be sent via email to the admin user created for password recovery.

If a mail server has been integrated with the MetaDefender Core product, users will be able to use the feature to recover their credentials.





Please Note:
SMTP server must be configured to use this feature. You can follow this article to configure.

Support:
If **Further Assistance** is required, please follow the instructions on auto$, then proceed to create a support case or chat with our support engineer.

How to Utilize YARA Rules with MetaDefender Core?

This guide is intended for users who wish to leverage YARA rules within MetaDefender Core for enhanced malware detection and specific file pattern recognition during the scanning process.

1. **Access MetaDefender Core User Interface**: Open your preferred web browser and navigate to the MetaDefender Core's IP address or hostname on the nginx port (default 8008) to access its user interface.
2. **Access the YARA Rules Configuration**: Once in the MetaDefender Core UI, navigate to the YARA Rules configuration. This can be found under Inventory - Modules - Utilities - YARA.
3. **Create a YARA Rule**: In the provided text field, create a new YARA rule that describes the characteristics of the malware or file patterns you wish to detect. Alternatively, if you have a URL or a path to a file containing YARA rules, you can provide that instead.

Here are a few example rules:

Simple String Match Rule:

YARA sql

```
yara

rule HelloWorld

{

    strings:

        $my_text_string = "Hello, World"

    condition:

        $my_text_string

}
```

Case-Insensitive String Match Rule:

YARA sql

```
yara

rule CaseInsensitiveStringDetect

{

    strings:

        $a = /malware/i

    condition:

        $a

}
```

Binary Pattern Match Rule:

YARA sql

```
yara

rule BinaryPatternDetect

{

    strings:

        $a = { DE AD BE EF }

    condition:

        $a

}
```

Multiple Strings Match Rule:

YARA sql

```
yara

rule MultipleStringsDetect

{

    strings:

        $a = "string 1"

        $b = "string 2"

    condition:

        $a and $b

}
```

Regular Expression Match Rule:

YARA sql

```
yara

rule RegexpDetect

{

    strings:

        $a = /[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+/

    condition:

        $a

}
```

4. **Save the YARA Rule**: After creating your rule or providing the URL or file path, save it in the MetaDefender Core UI.
5. **Enable YARA in Workflow:** Go to the Workflow section in the MetaDefender Core UI. This is accessible by following this path: Workflow - Select desired workflow - YARA. Ensure that YARA is enabled for each workflow where you wish to apply the rule.
6. **Review the Results:** Examine the scan results to see if the YARA rule identified any matches in the uploaded files. If a match is found, it indicates that the file contains the pattern described in the rule.

Conclusion:

By integrating YARA rules with MetaDefender Core, users can significantly enhance their malware detection capabilities. These rules provide a powerful way to identify specific file patterns during the scanning process, thereby leading to a more robust and effective threat detection. This guide should enable users to create, implement, and utilize YARA rules effectively within MetaDefender Core.

For more information on YARA, please refer to the official YARA Documentation: https://virustotal.github.io/yara/

Support:
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.

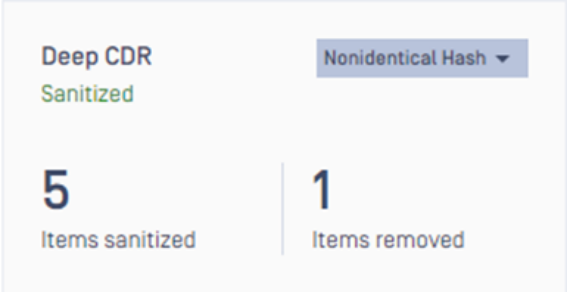
This mechanism ensures the highest level of safety and reliability, as no files can be triggered or activated during their time on your system.

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.



METASCAN™ | DEEP CDR | VULNERABILITY ASSESSMENT | FILE TYPE VERIFICATION | ARCHIVE EXTRACTION

Object	File Name	Action
PDF file	/word/embeddings/oleobje...	✓ Sanitized
XLSX file	/word/embeddings/microso...	✓ Sanitized
XLSX file	/word/embeddings/microso...	✓ Sanitized
XLSX file	/word/embeddings/microso...	✓ Sanitized
[3] hyperlink	-	✓ Removed
[4] image	-	✓ Sanitized

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 auto$, 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.

https://docs.opswat.com/mdcore/deep-cdr/supported-file-types#supported-file-types-both-windows-and-linux-https://www.fujifilm.com/fb/download/software/docuworkshttps://www.fujifilm.com/fb/support/software/docuworks_91/faq/contents/faq_00655.html

- 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.

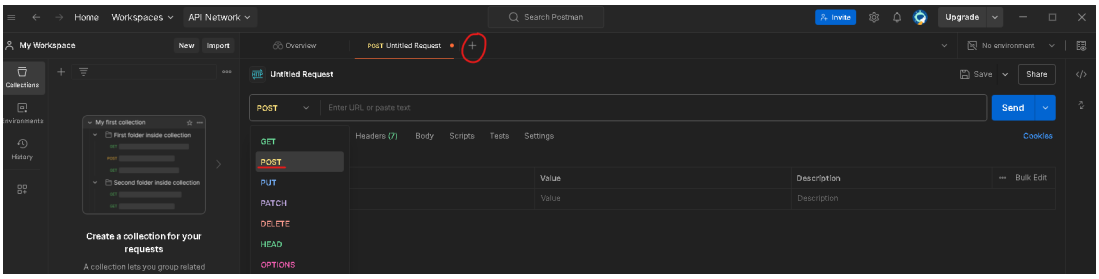
How do we utilize API uploads so the correct file type and sanitization occurs every time?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

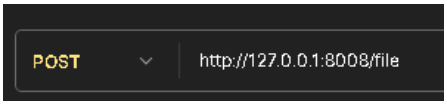
Use the 'binary' upload method when sending files to Core for scanning via API. If you add files to the body of a 'Post' API call using another method such as 'form-data', the content of the file that our Core server receives gets modified. This can result in our system incorrectly identifying the file 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:

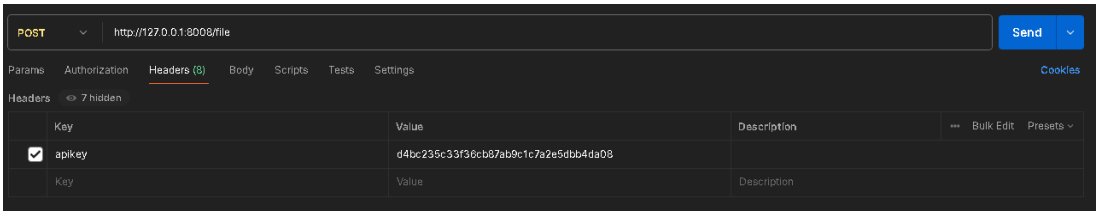
1. Press the + Button at the top of the Postman tool to begin a new API request.
2. Select 'Post' as the call type from the drop down menu



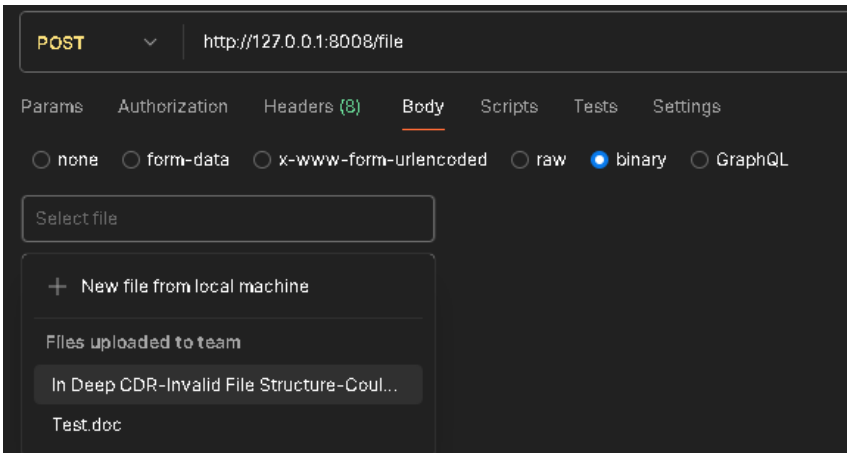
3. Enter the full URL for your MetaDefender Core server's API interface.



4. Add any optional headers outlined in https://docs.opswat.com/mdcore/metadefender-core#fileanalysispost. 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

- **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?

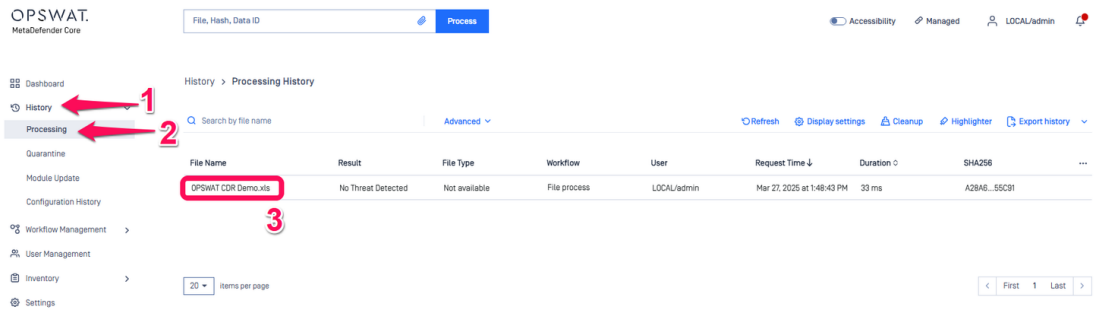
> **Check Your Version:**
> This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

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:

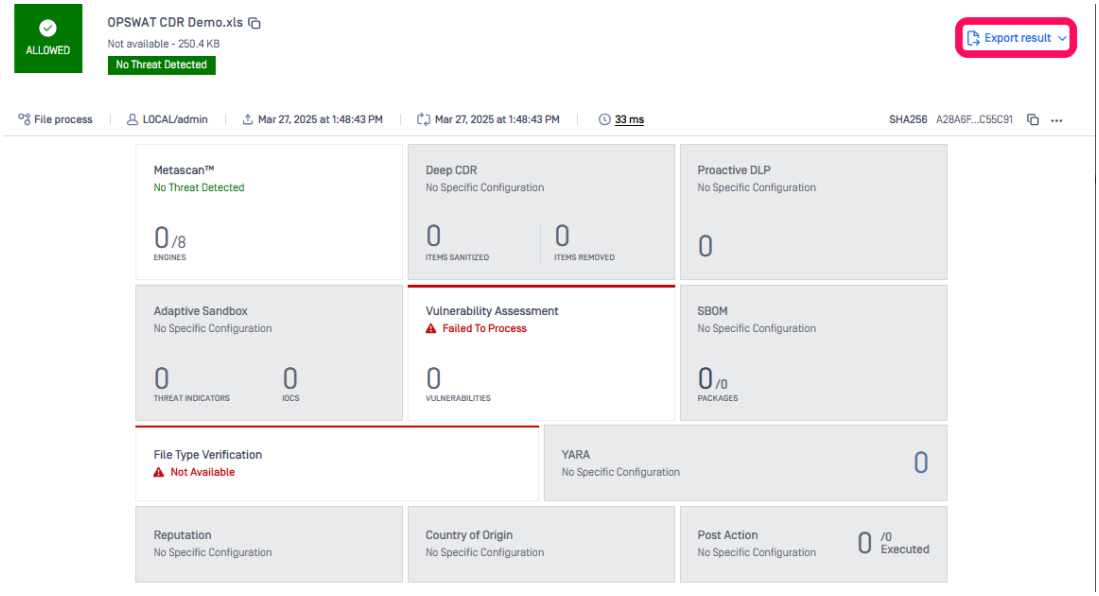
- **Scanning Engine**: The antivirus or security engine that flagged the file.
- **Threat Name**: The identifier assigned to the detected threat.
- **Metadata**: Additional details such as file hash, scan timestamp.

Using the Management Console UI:

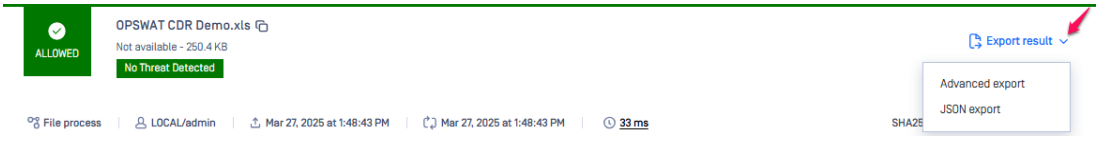
- Click the **History** tab on the left, select **Processing** from the drop-down list, and then click the file to view the scan result.



- 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:

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.

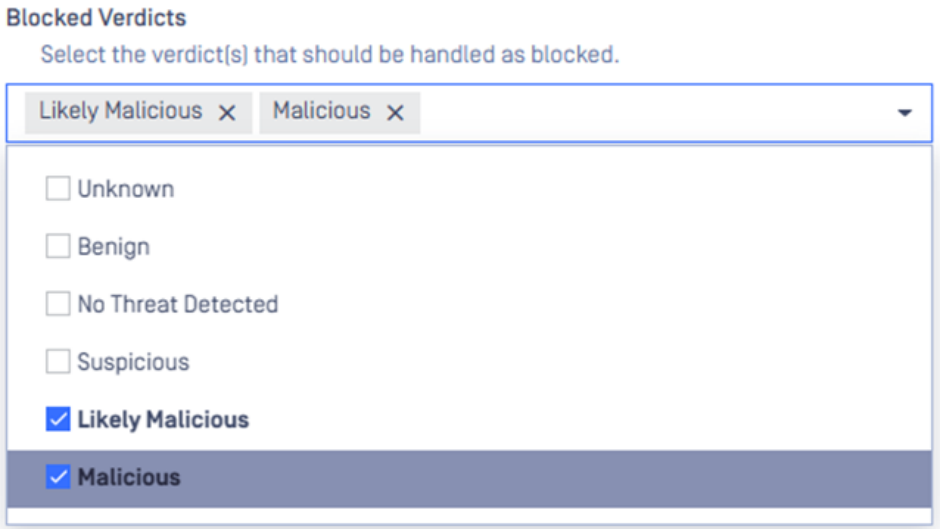
By leveraging both MetaDefender's scan results and vendor-specific threat intelligence, you can gain a deeper understanding of the nature of the threat and take appropriate remediation actions.

> **Support:**
> If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

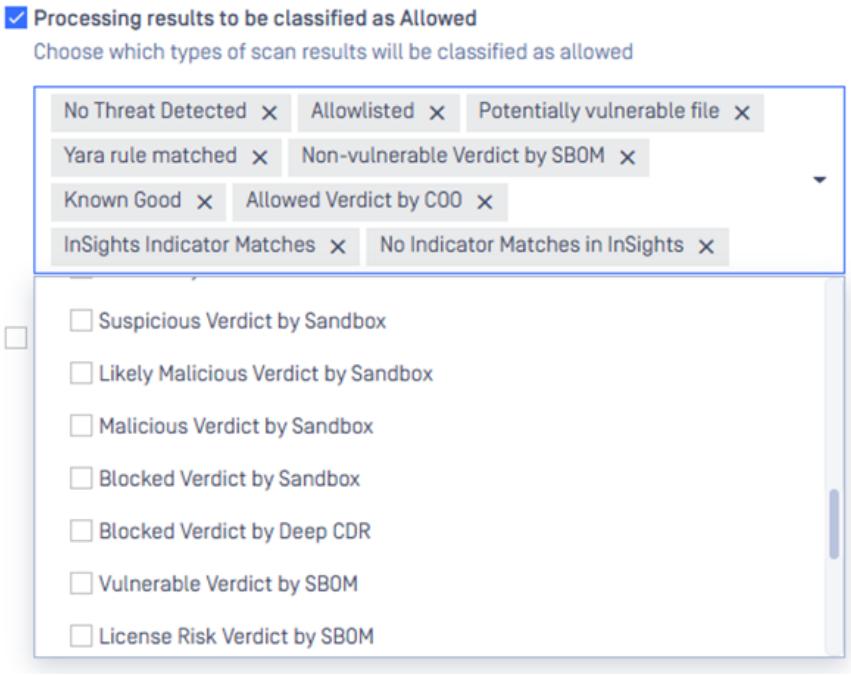
Why MetaDefender Core blocks Sandbox Verdicts that are not marked to be blocked?

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

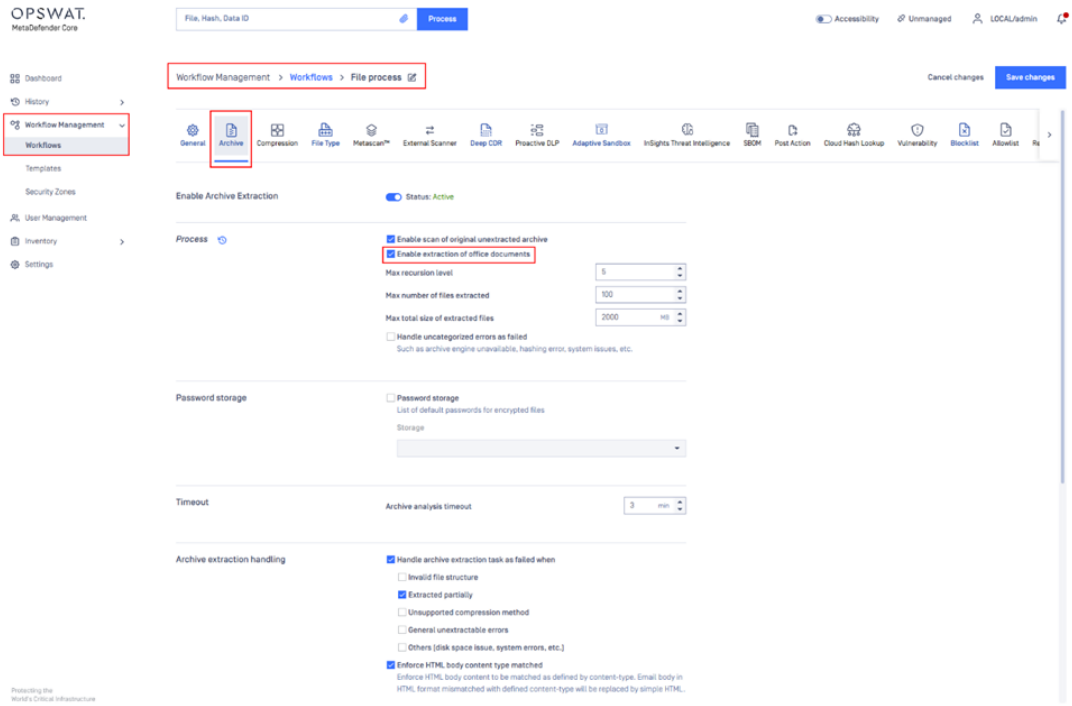
Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

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, pptm** 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?

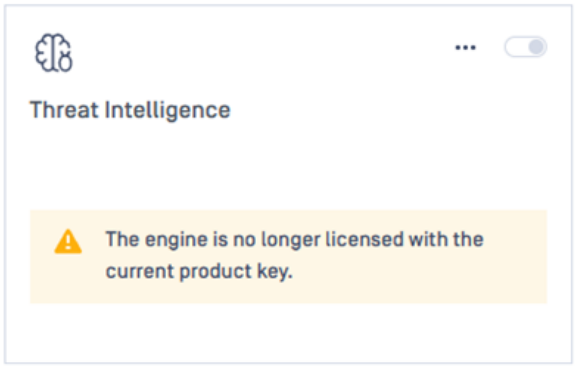
Starting on March 12th, 2025, OPSWAT will release InSights Threat Intelligence and discontinue Threat Intelligence in the module to avoid confusion.

How will it affect the current customers, and what actions should they take?

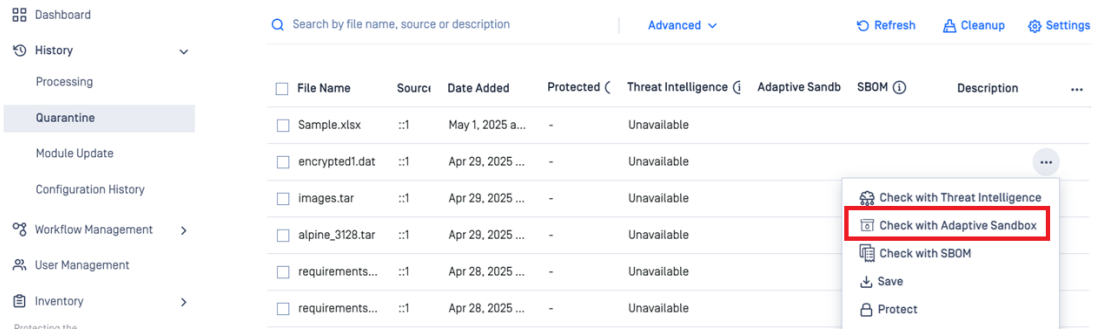
The Threat Intelligence module was **included** with all MetaDefender Core licenses. However, it requires customers to purchase a MetaDefender Cloud API key in order to use it.

So there's no impact for customers that are not including the API key as part of their Threat Intelligence module usage.

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:
https://www.opswat.com/docs/mdcore/sandbox

Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I Validate FileType Digital Signature on MetaDefender Core?

Check Your Version:
This article applies to all MetaDefender Core versions, deployed on Windows and Linux systems.

License Requirement:
Please ensure you have Deep CDR licensed and enabled in the Modules

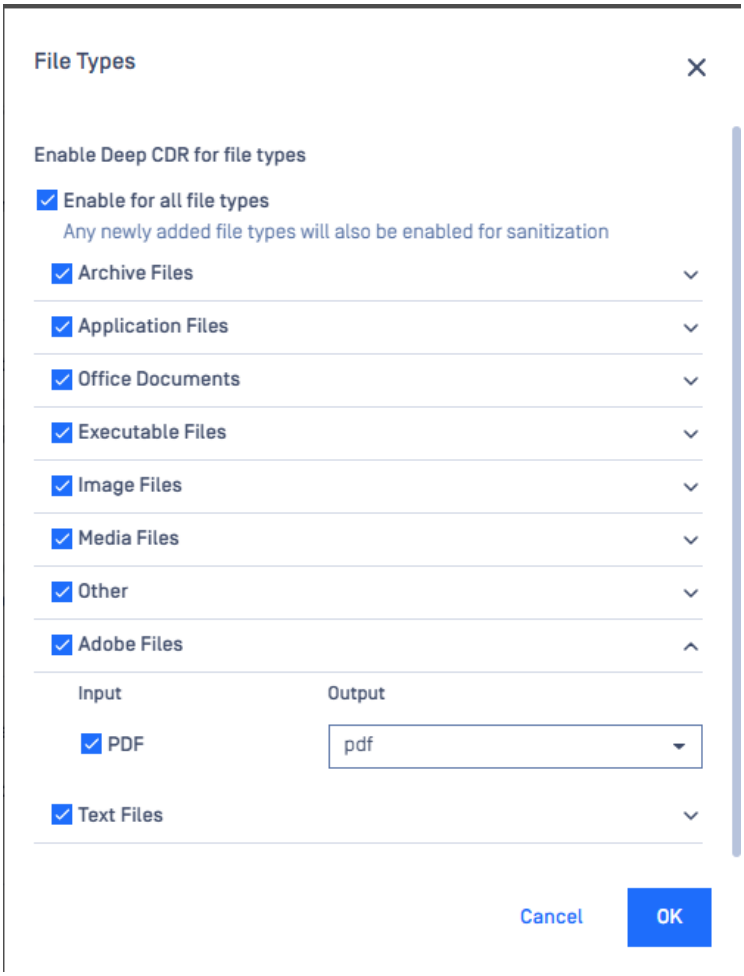
Introduction

MetaDefender Core's Deep CDR (Content Disarm and Reconstruction) technology enables robust digital signature validation by verifying integrity of the signed content.

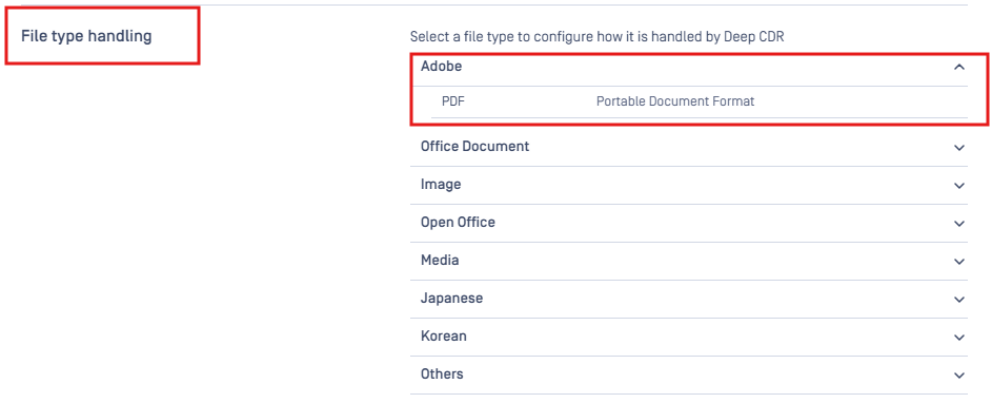
PDFs are one example, other filetypes with a digital signature configuration include: Microsoft Word, Microsoft Excel, and Microsoft PowerPoint.

Configuration

1. Enabled PDF FileType under MetaDefender Core > Workflow Management > desired workflow > Deep CDR tab >



2. Navigate to Deep CDR > Advanced configuration > desired FileType



Example: PDF

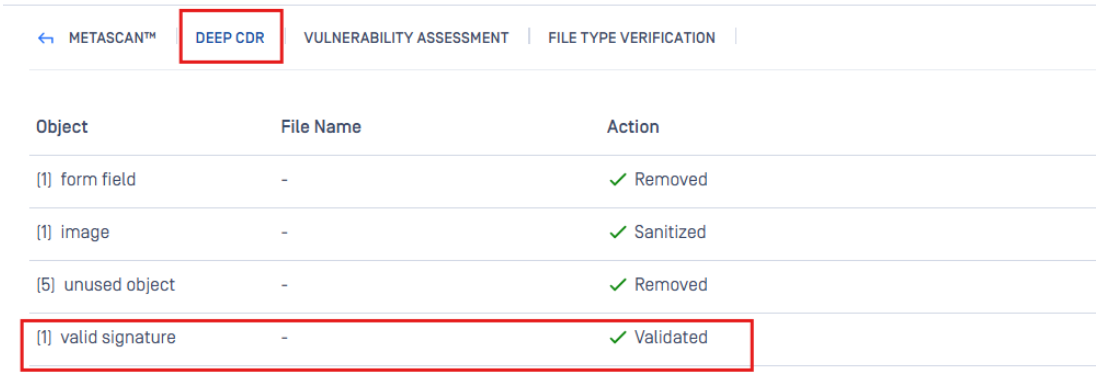
Check the box: “Validate digital signature” > Ok > Save Changes



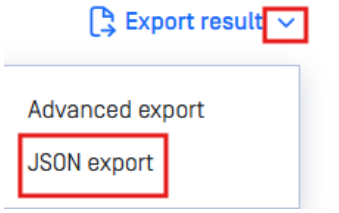
Results

We can see under the Deep CDR tab a confirmation action: Validated. Deep CDR was able to successfully validate the integrity of the digital signature

MetaDefender Core UI:



Exported JSON results:



Please find the “object_name”: “valid signature”

```
 "sanitization_details": {
        "description": "File is sanitized successfully and its structure is validated.",
        "details": [

          {
            "action": "validated",
            "count": 1,
            "object_name": "valid signature"
          }
```

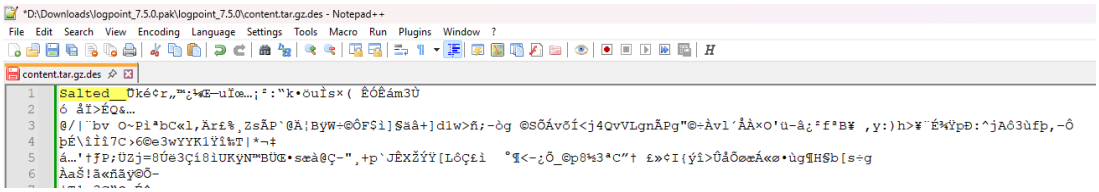
Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How to add custom detection for a file type?

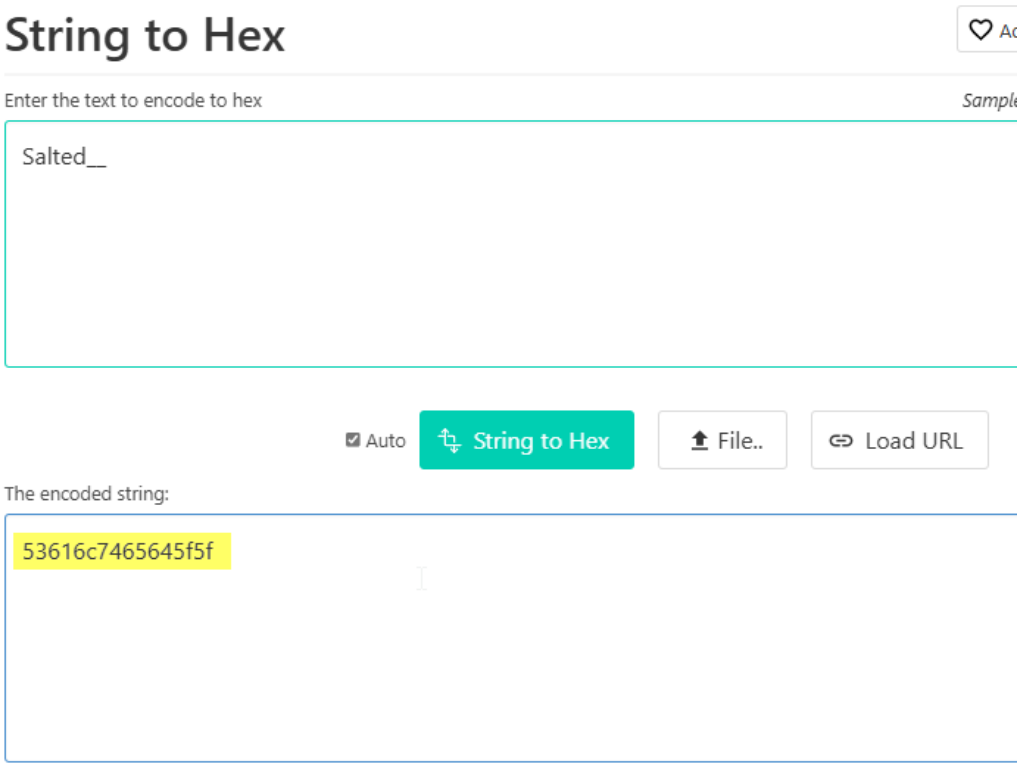
> **Info:**
> The custom detection is only supported from File Type version 7.4 and later

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>`

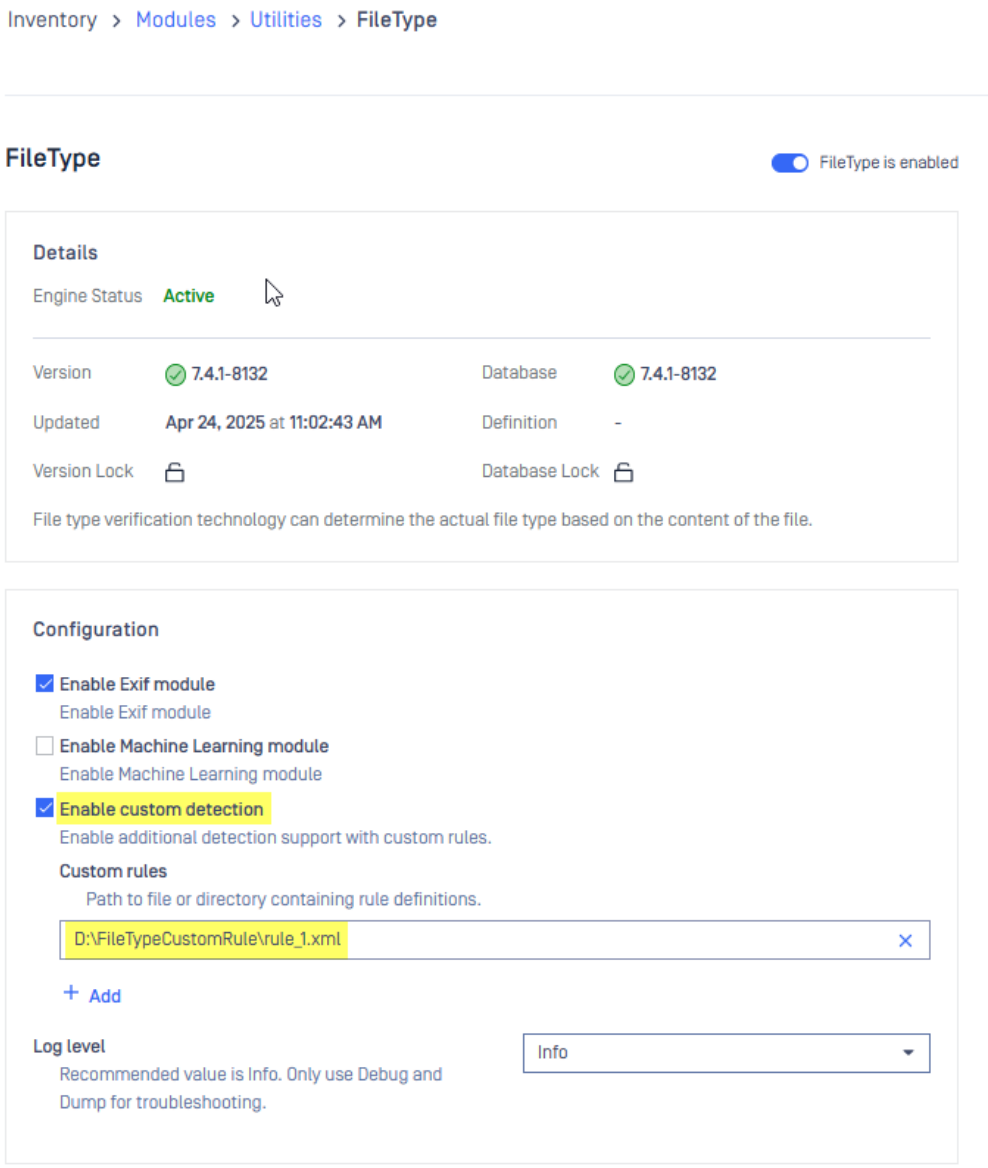
- This is a **hexadecimal byte sequence** in the step #2

`<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).

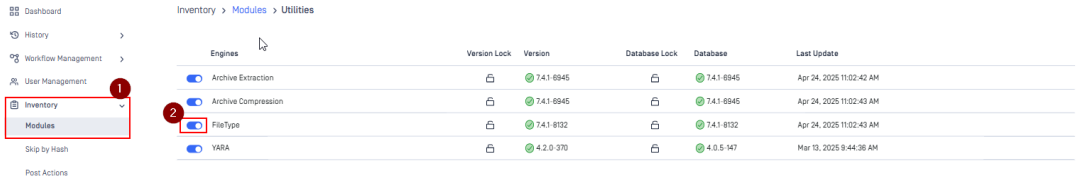
For details about the XML format with fields description, check Custom detection - MetaDefender Core

- 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

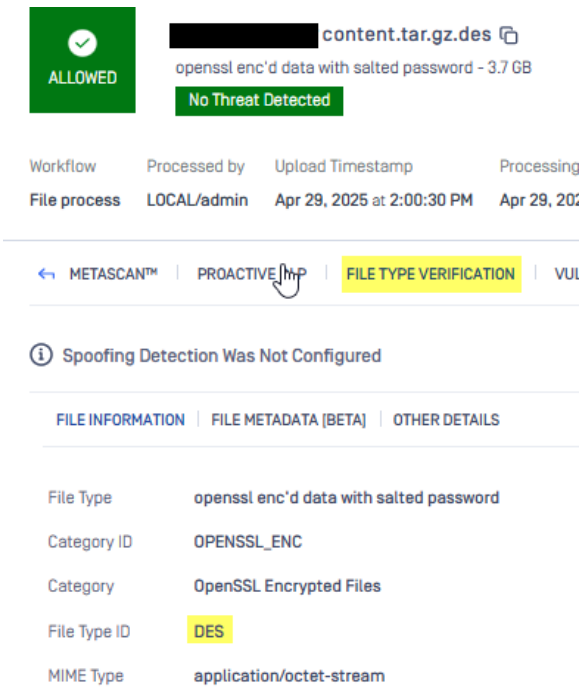


Check Custom detection - MetaDefender Core to see how to enable custom detection

- 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.

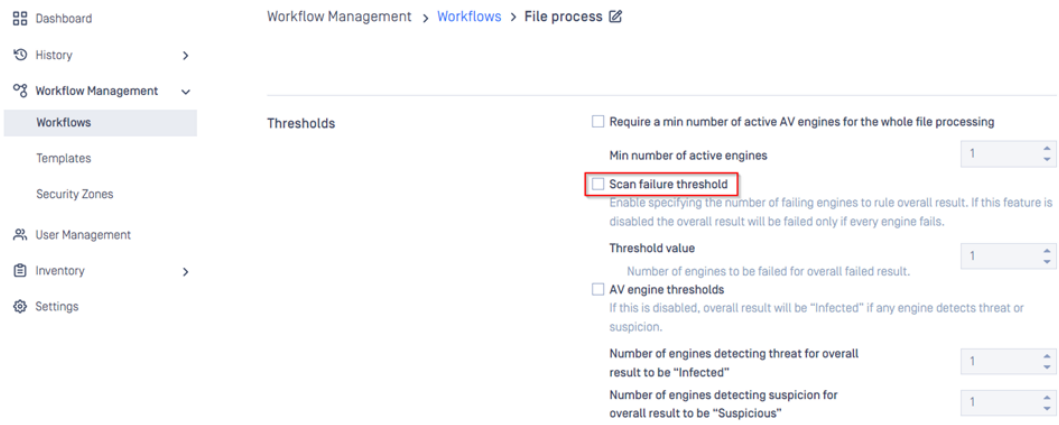
MetaDefender Core V5 Releases

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

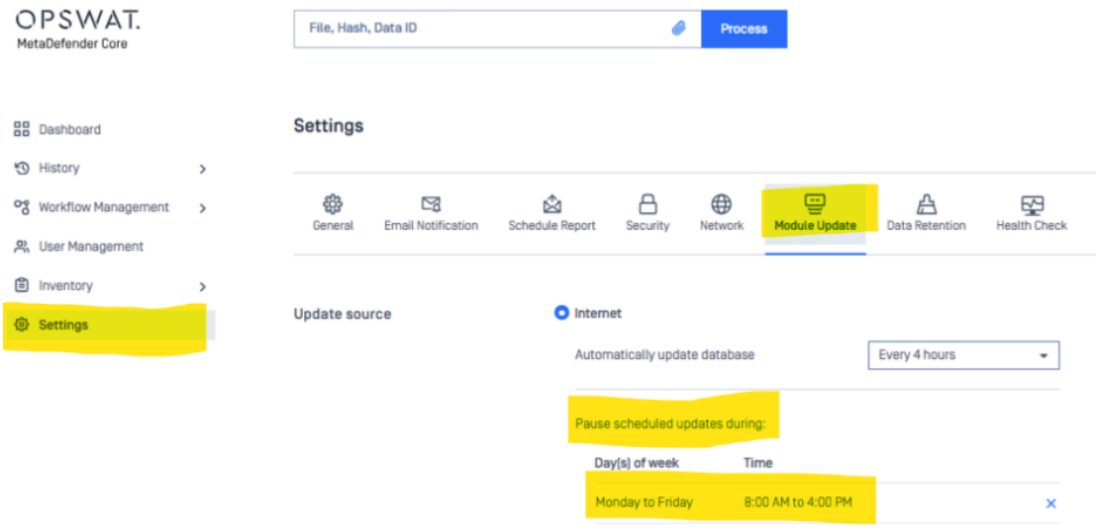
Uncheck the option **Scan Failure Threshold** shown below.



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?

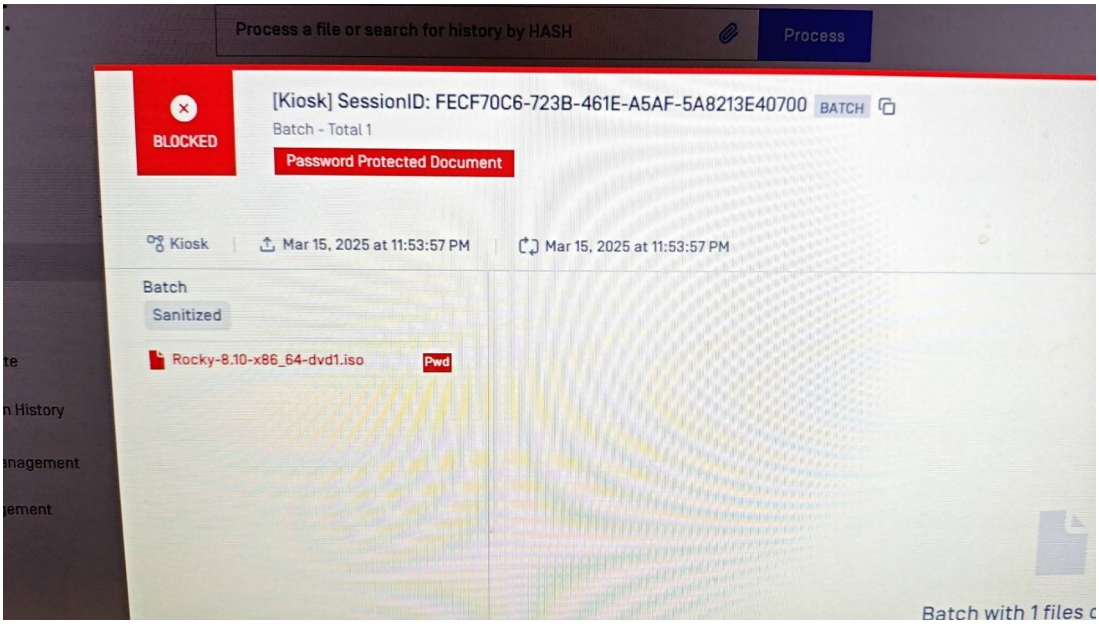
This article explains why MetaDefender Core or MetaDefender Kiosk may unexpectedly prompt for a password when scanning files that are not encrypted.

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)



MD Core Results

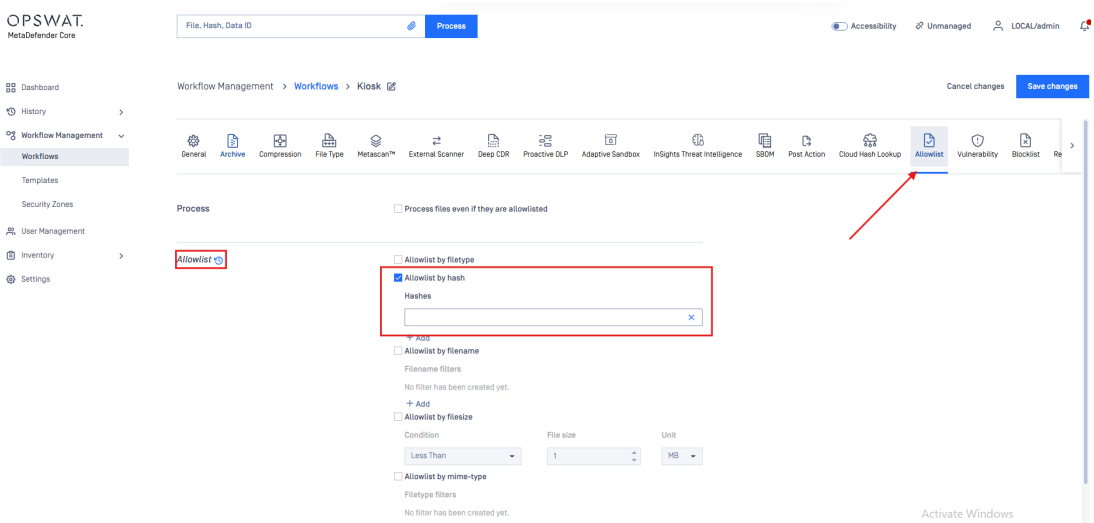


Kiosk UI

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.**



Allow list by hash

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

```
$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

```
$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

```
1 echo "<34>Test Syslog message" | nc -u -w1 <syslog_server_ip> 514
```

TCP Syslog

```
1 echo "<34>Test Syslog message" | nc <syslog_server_ip> 601
```

Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

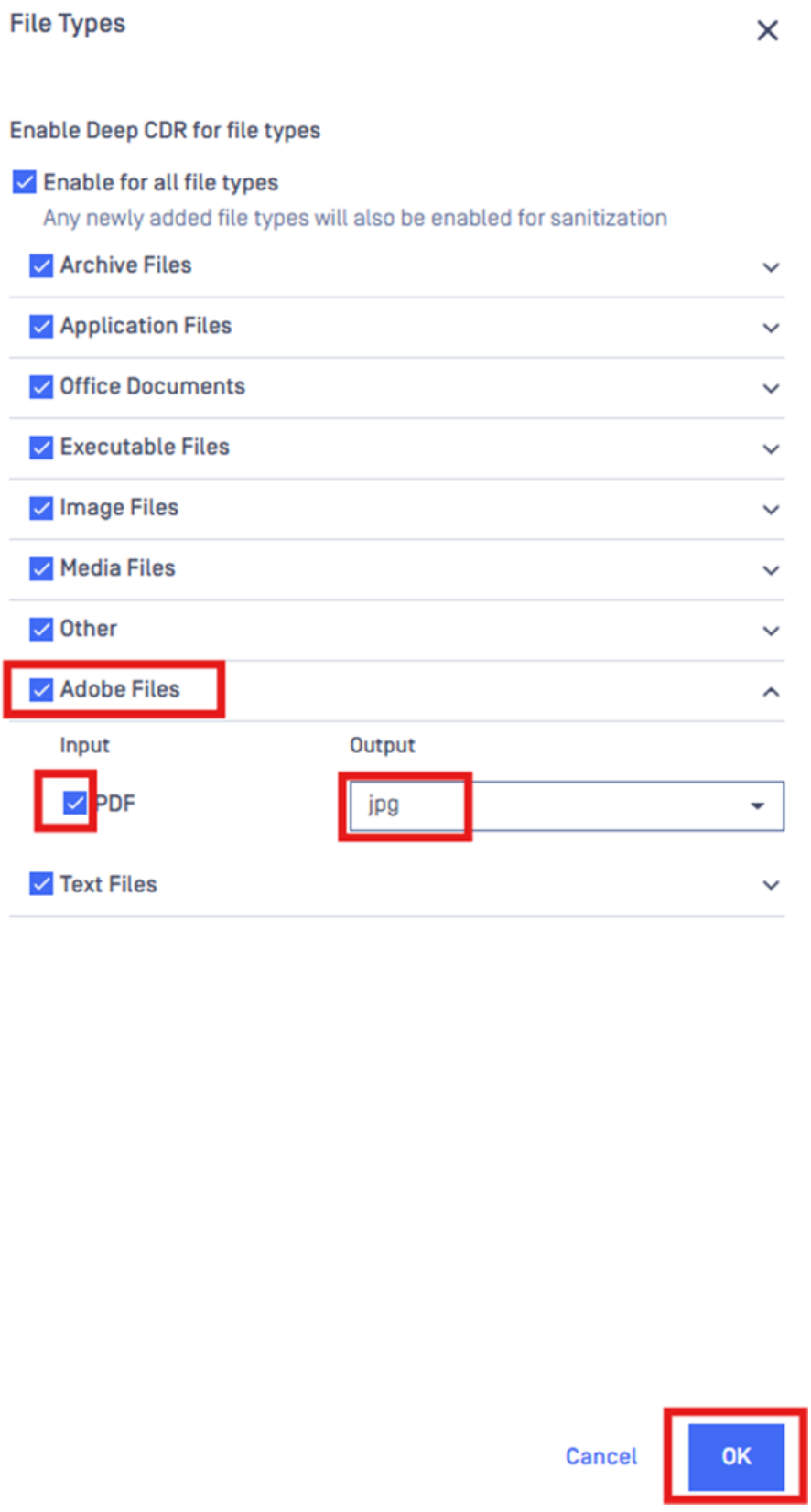
When and how to use “PDF to Image Configuration” Deep CDR feature?

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.

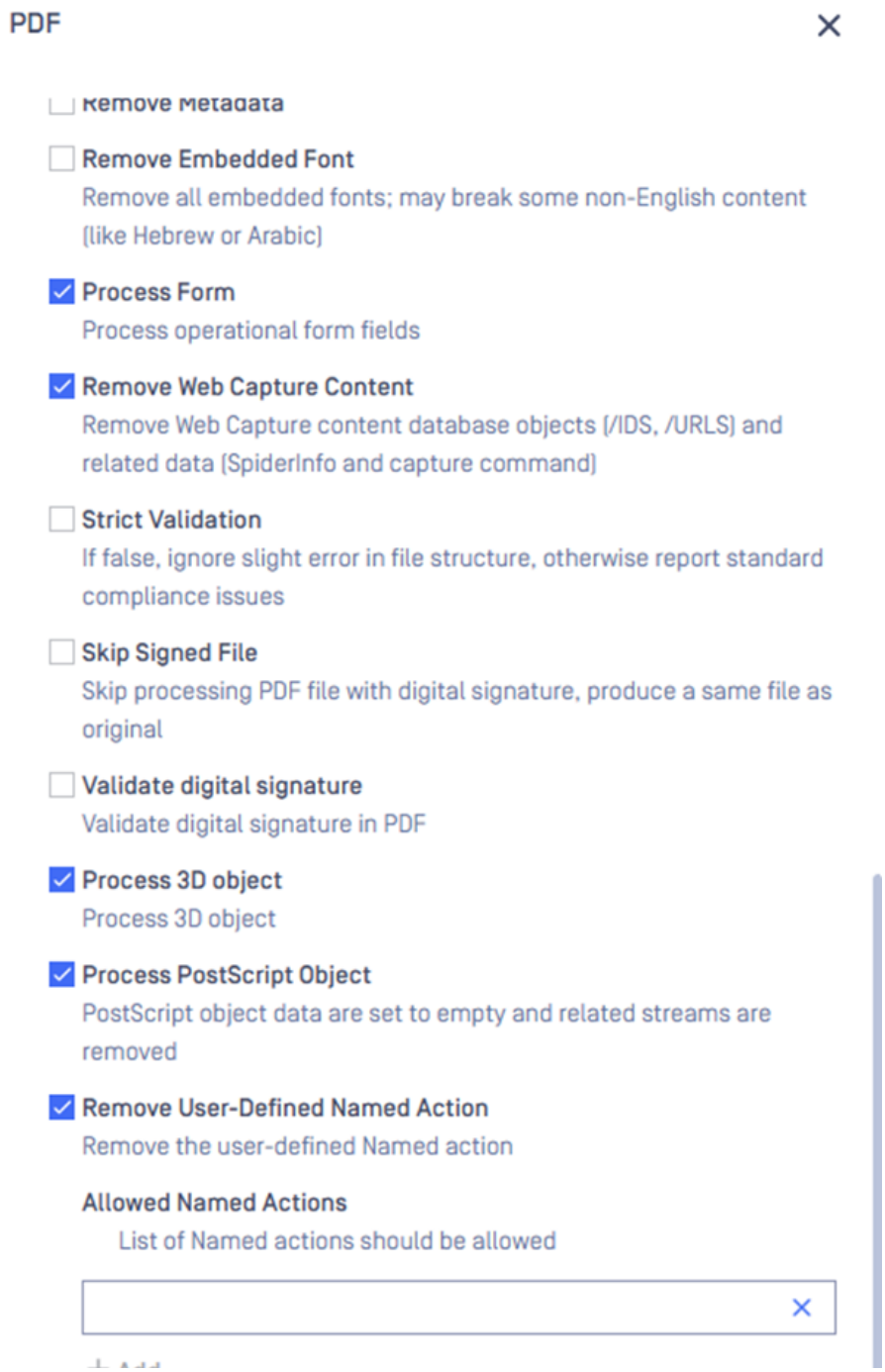
Details:

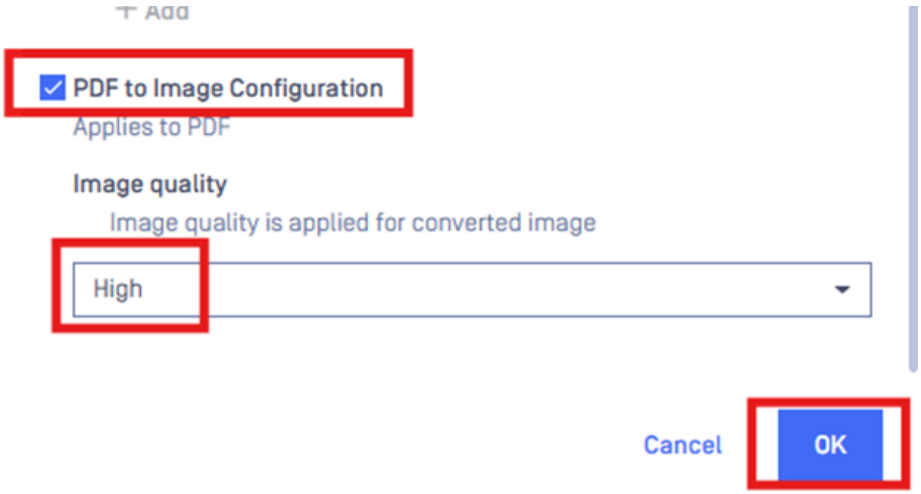
“PDF to Image Configuration” feature is taking effect only when you use PDF to image(ex.: jpg etc.) conversion for PDF files.


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”





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]

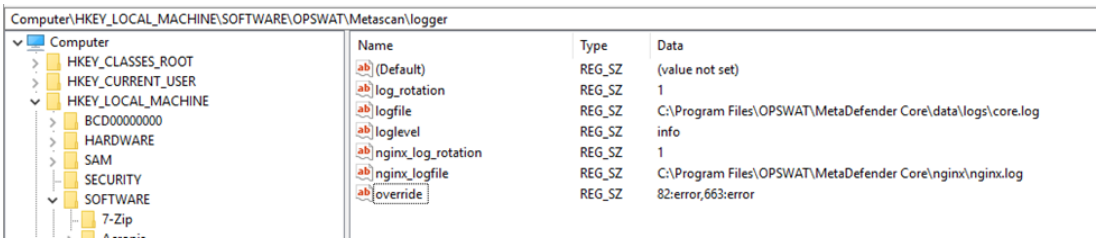
To make the above change please follow the instructions below for each OS:

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:

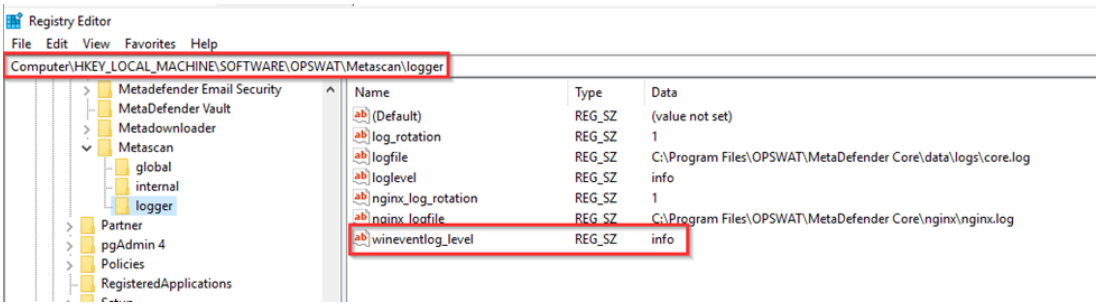
There are situations where the syslog integration is already configured with the Windows event log and it's easier to get the MetaDefender Core logs from the Windows event log than to create a new integration with the MetaDefender Core.

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--
```

CSS style

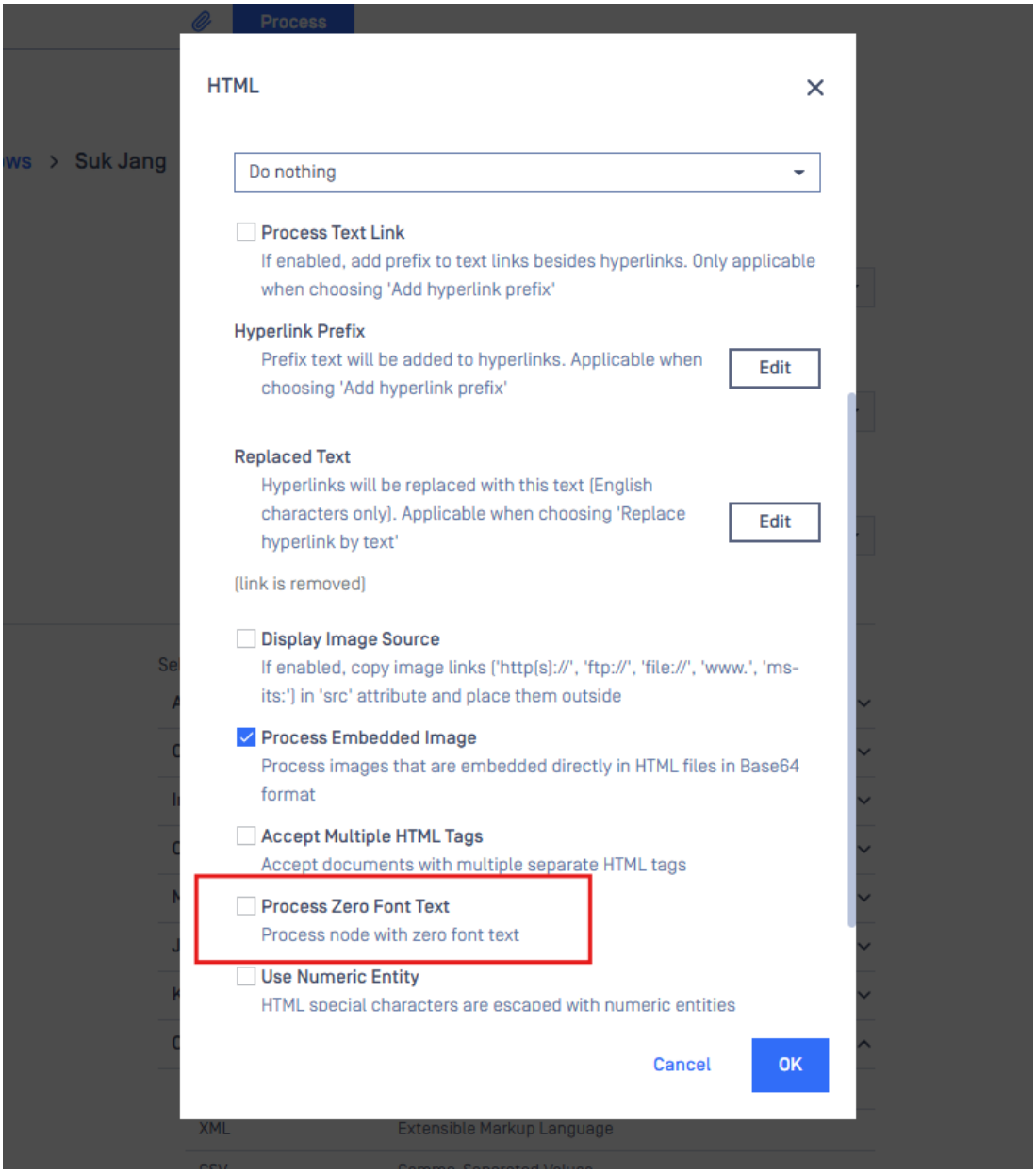
```
17  ------=_Part_316646736_828665848.1743118257422
18  Content-Type: text/html; charset=UTF-8
19  Content-Transfer-Encoding: quoted-printable
20
21  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
22      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
23  <html xmlns=3D"http://www.w3.org/1999/xhtml"><head><title></title></head><b=
24  ody><div id=3D"emailPreHeader" style=3D"mso-hide:all; visibility:hidden; op=
25  acity:0; color:transparent; mso-line-height-rule:exactly; line-height:0; fo=
26  nt-size:0px; overflow:hidden; border-width:0; display:none !important;"></d=
27  iv></body></html>=
28
29  ------=_Part_316646736_828665848.1743118257422--
30
```

The content removed

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.

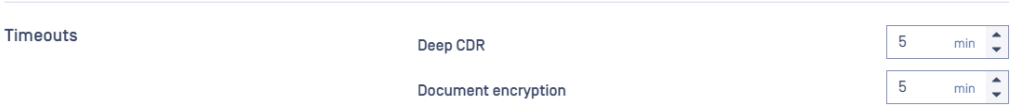


Support:
If you have followed all of the steps above but are still unable to resolve the issue. Please follow these instructions on How to Create Support Package With Bundle Tools?, before creating a support case or chatting with our support engineer.

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.

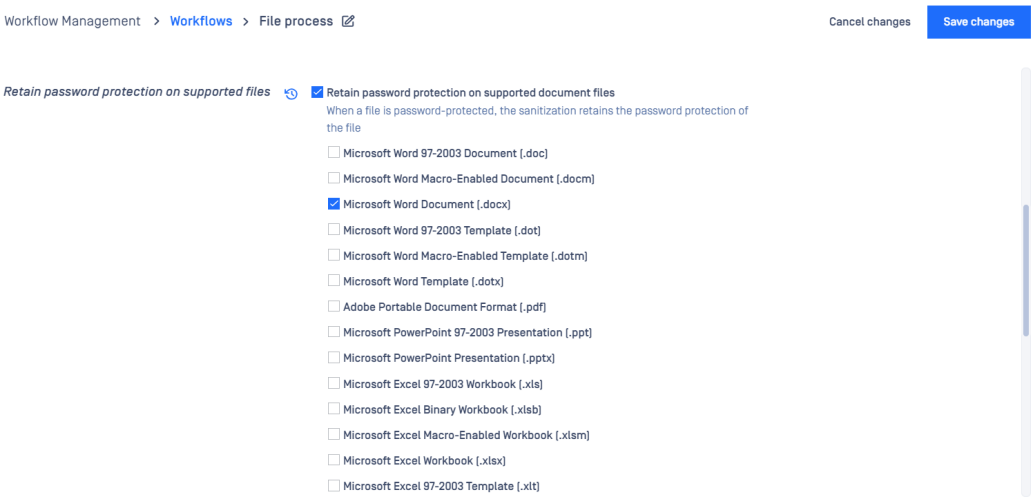
The **Document Encryption Timeout** is a configurable setting located under: **Workflows > DeepCDR > Timeouts > Document encryption**



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**
    a. Use system monitoring tools to assess CPU, memory, and disk I/O usage.
    b. Identify if CPU usage spikes coincide with specific scanning activities.
3. **Review MetaDefender Core Configuration**
    a. Verify that the engine parallel count and maximum queue size settings match your workload. Adjust these settings if necessary.
    b. Check for any misconfigurations in the MetaDefender Core settings that might be triggering unnecessary load.
4. **Update Scanning Engines**
    a. 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**
    a. 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).
    b. Consider increasing CPU cores if the current hardware is consistently overburdened.
6. **Review Logs and System Health**
    a. Examine MetaDefender Core logs to detect any error messages or alerts related to CPU load.
    b. 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**
    a. 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.

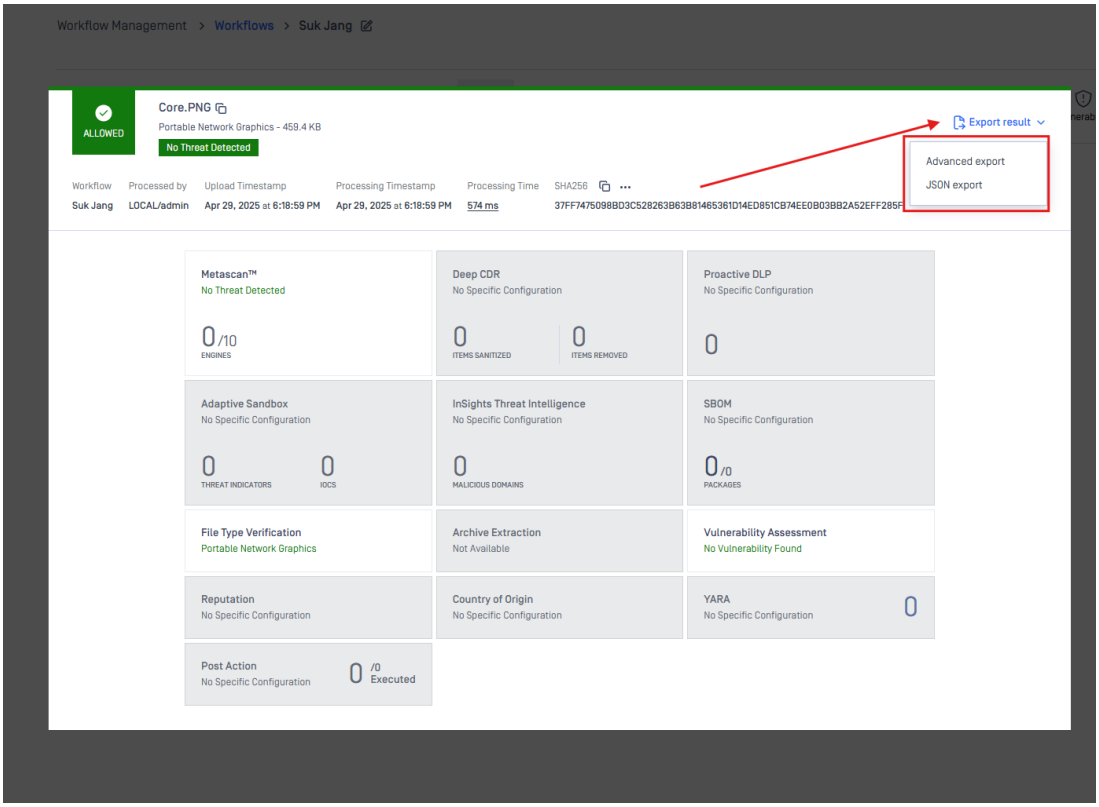
Workflow without DeepCDR, Proactive DLP

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.

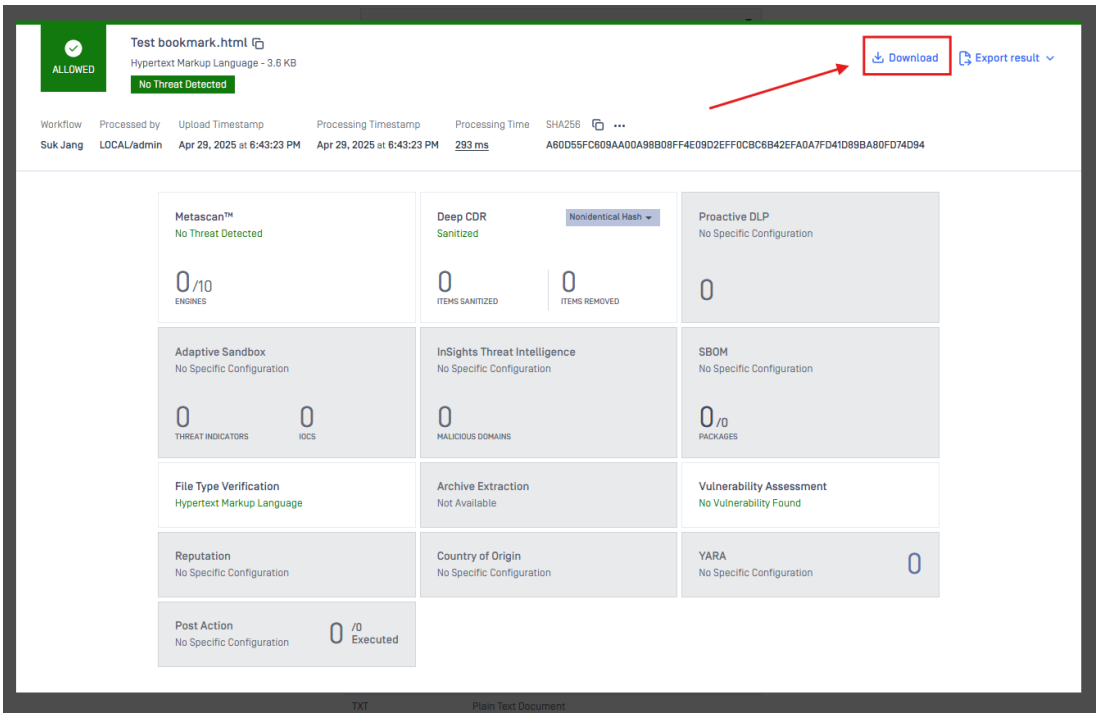


Workflow with DeepCDR Or DLP

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.

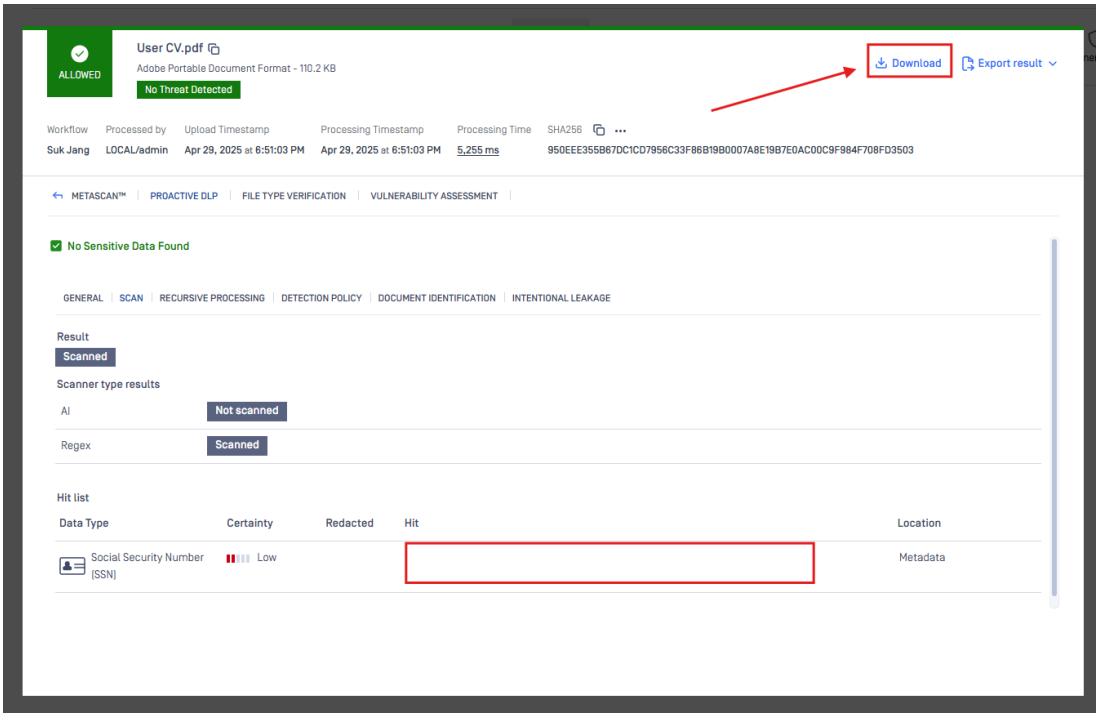


Scan with Deep CDR

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.



Scan with DLP

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.

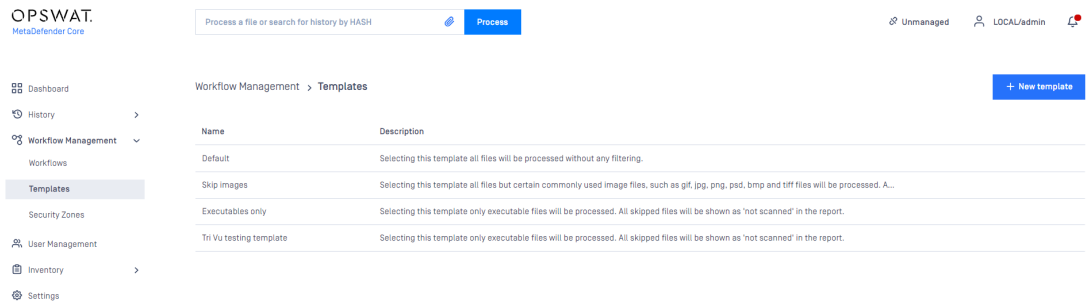
Can Workflow Settings Be Updated Globally Across Multiple Workflows?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows and Linux systems.

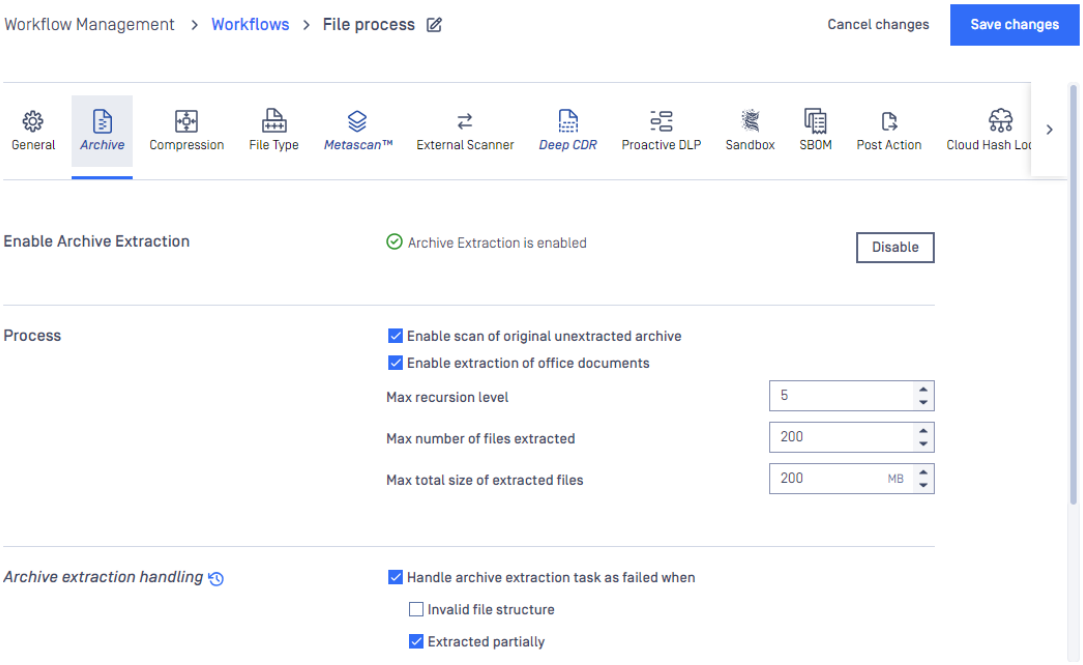
Description

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

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:

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:

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:

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:

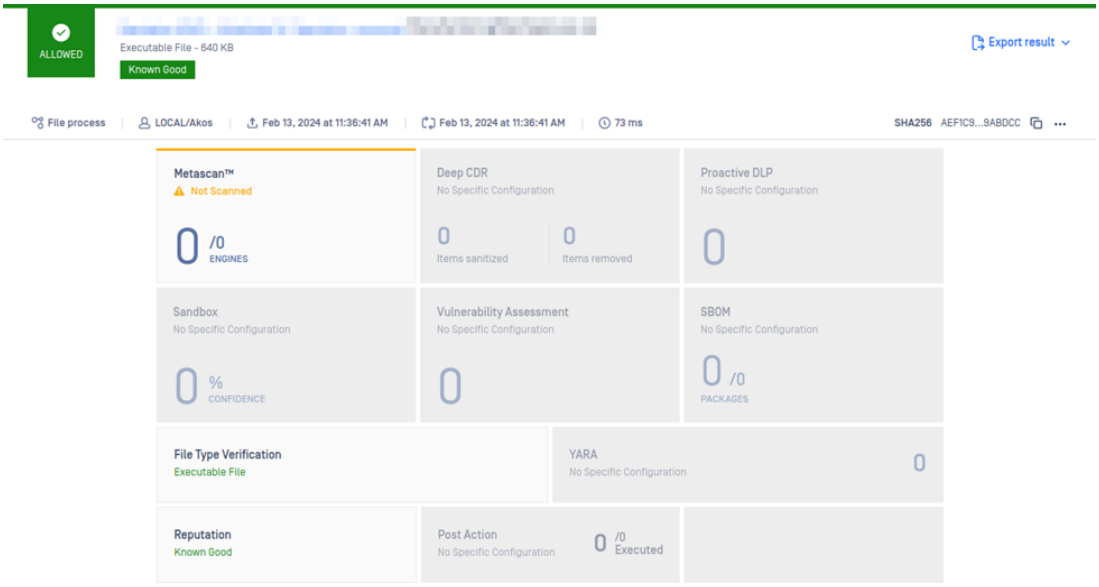
1. Verify That the File Is a False Positive, Not an Actual Threat

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)?

- 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.

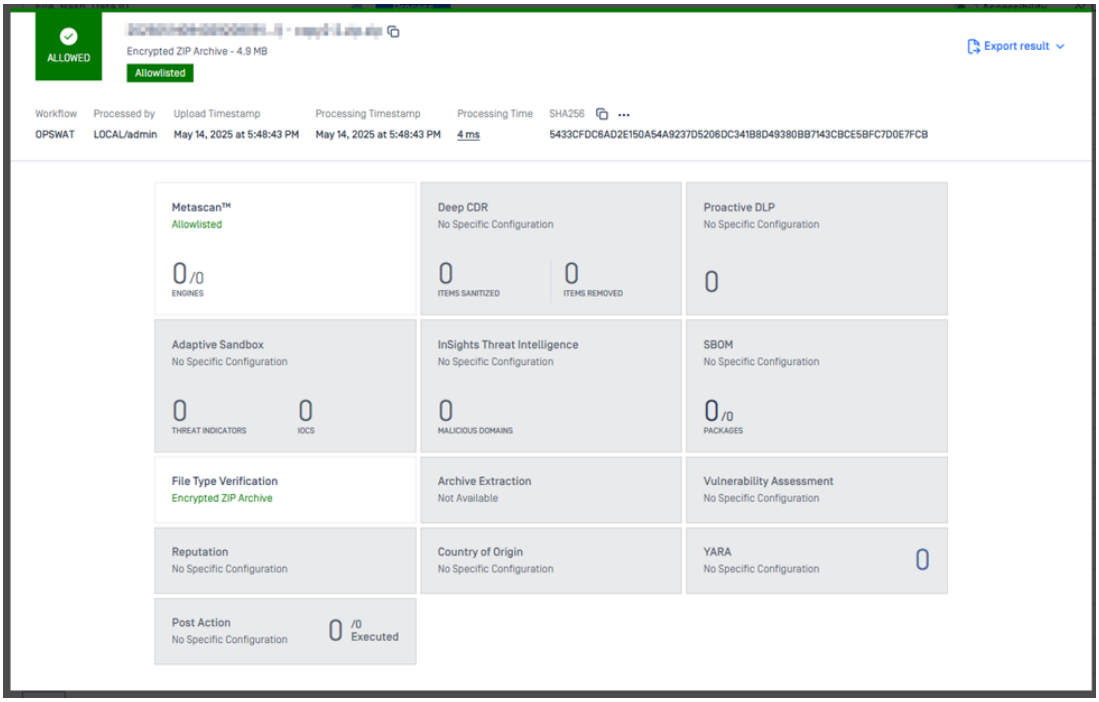
- The sandbox analyzes files in a controlled virtual environment to detect malicious behavior beyond traditional AV signatures.
- This additional layer helps validate whether a detection is accurate or a false positive.

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

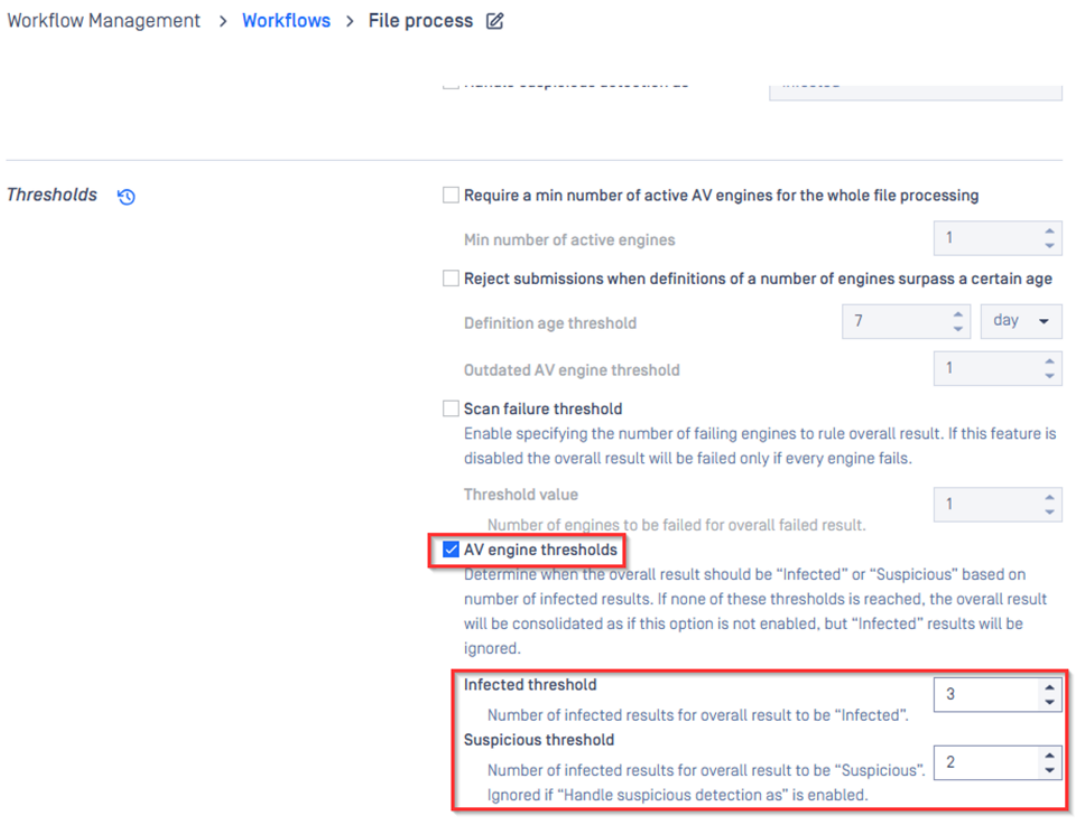


⚠ **Warning**: Adding a file to the Allow List means it will bypass all scanning. This creates a potential security risk. Always ensure that internal security policies are followed, and use this option **only when absolutely necessary**.

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?

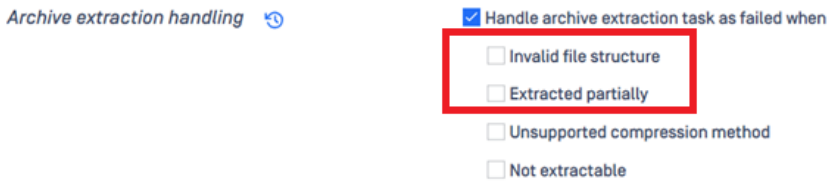
A **split archive file** is a compressed file that has been divided into multiple smaller parts, typically to simplify storage, sharing, or transfer of large data sets. These files usually follow a naming convention such as `.001`, `.002`, and so on. Each part contains a portion of the original data.

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.

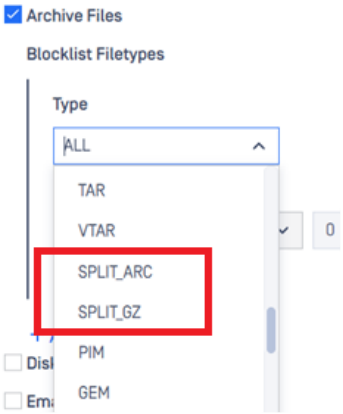
The **Archive Extraction** module attempts to extract as much data as possible and passes the extracted content through a scanning process to ensure maximum security. However, due to the nature of split archives, the extraction process often results in outcomes like **"Partially Extracted"** or **"Corrupted Archive."**

What can customers do with split archive files?

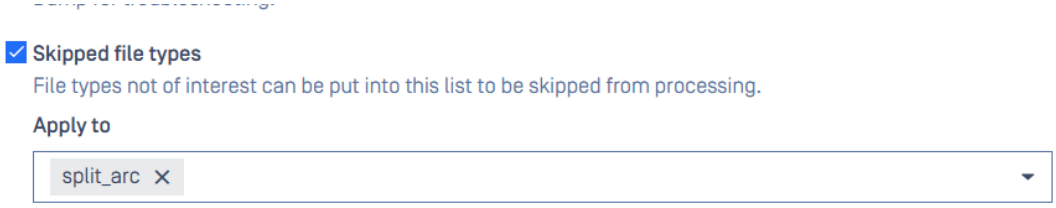
- If customers prefer not to have these files blocked under such conditions, they should avoid enabling specific options in the **Archive Extraction** tab within the workflow configuration.



- 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.

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:

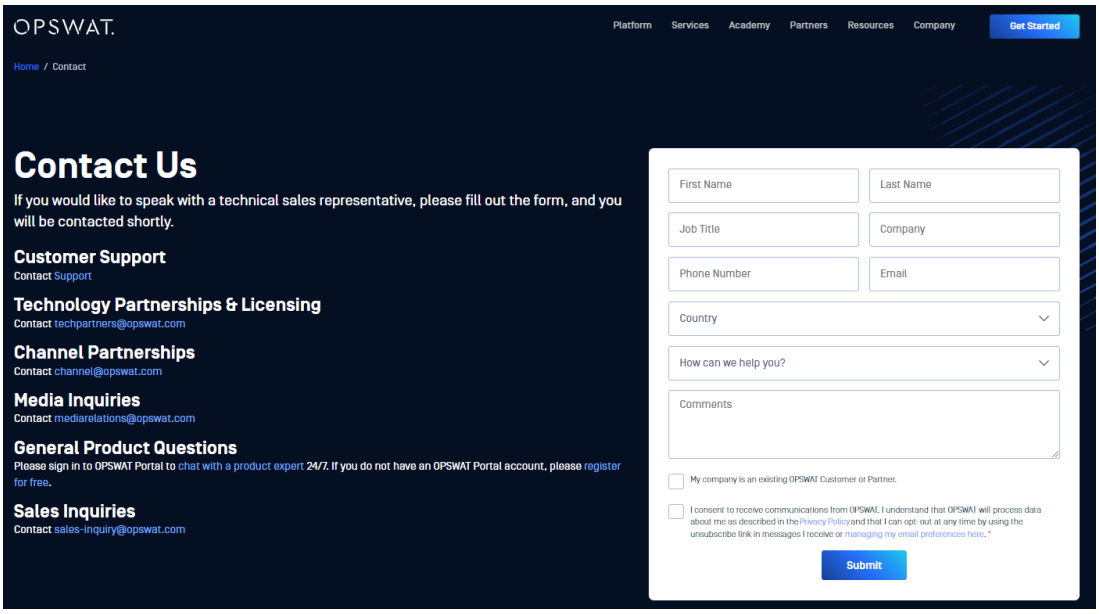
- Enable the setting: `Preserve entry name in sanitized files`. This feature has been available starting from DeepCDR version 7.4 and later, and it needs MetaDefender Core version 5.14.0 or above to work correctly.

Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

How do I 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.

Looking for help now?

Phone Support
For Premium Customers

US:	+1 855 677 9281
UK:	+44 870 820 0400
Japan:	+81 345 209 785
France:	+33 975 181 808
Australia:	+61 87 100 1170
Israel:	+972 722 577 008

Chat with us
Open a chat session through OPSWAT portal.

You will need to have an OPSWAT Portal account to log a case. Don't have one? It's easy to create one now.

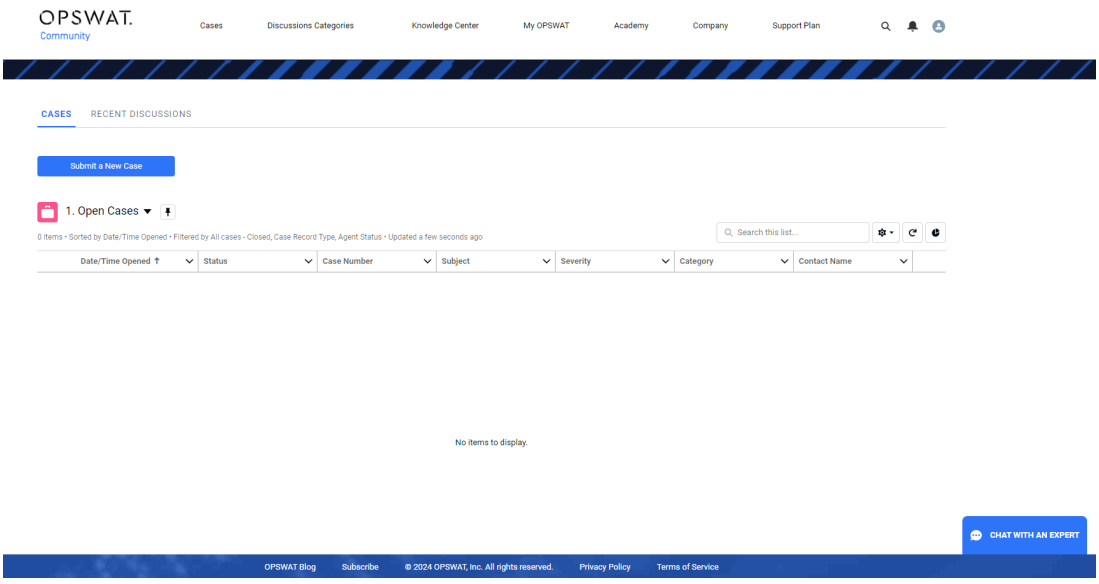
Online Chat

Submit a Case
Use the form to submit your information and describe the situation you are experiencing, including the issue/s. Our team of experts will review, research, and respond directly to you.

Open a Case

Join our Forum
Visit our forum to find an extensive collection of Questions and Answers related to OPSWAT products. The answers are contributed by OPSWAT Experts, our Partners, and other Customers.

Ask a Question



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.

Please Note:
The active support period for the previous MetaDefender Core release, Version 5.14.2, will expire on **0 3 Jan 2027.**

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 the engines failure at anytime.

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 the 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.

MetaDefender Core V5 Support Lifecycles:

Release number	Release date	End-of-life date
5.15.0	03 July 2025	
5.14.2	28 May 2025	03 Jan 2027
5.14.1	28 Apr 2025	28 Nov 2026
5.14.0	31 Mar 2025	28 Oct 2026
5.13.3	27 Feb 2025	01 Oct 2026
5.13.2	21 Jan 2025	27 Aug 2026
5.13.1	16 Jan 2025	21 Jul 2026
5.13.0	18 Dec 2024	16 Jul 2026
5.12.1	30 Oct 2024	18 June 2026
5.12.0	30 Sep 2024	30 Apr 2026
5.11.1	28 Aug 2024	30 Mar 2026
5.11.0	31 Jul 2024	28 Feb 2026
5.10.1	27 Jun 2024	31 Jan 2026
5.10.0	21 May 2024	27 Dec 2025
5.9.0	21 Mar 2024	21 Nov 2025
5.8.0	20 Dec 2023	21 Sep 2025
5.7.1	09 Nov 2023	20 Jun 2025
5.7.0	09 Oct 2023	09 May 2025
5.6.1	28 Aug 2023	09 Apr 2025

Attention:
The versions not mentioned on this page are no longer supported.

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.

Report False Detection

Submit | Submission History

Submission file*

Select a file

Maximum file size: 2 GB

If the submitted file is an archive file, please specify the issues for each child file in the **'Why do you think this is a false detection?'** field.

Submission type*

Clean file (false positive)
Falsely detected as infected

Infected File (false negative)
Falsely detected as clean

In which product did the false detection occur?*

Select products

Which antivirus engine triggered the false detection?*

Select antivirus engines or Input

AV detection/Threat name*

E.g. Trojan/Win32.Downloader

Image of Detection/Report Alert

Attach an image

What is the origin of the file?*

Select source

What is the purpose of the submitted file?*

E.g. Design program, image viewer, informational document, document editor installer ...

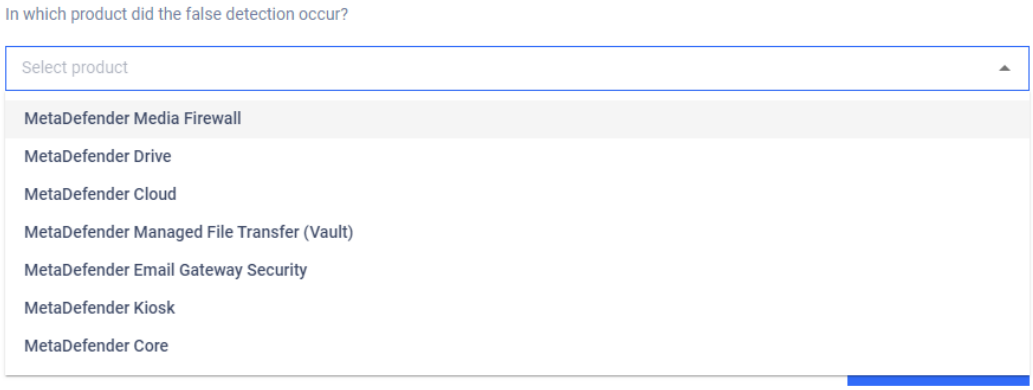
Why do you think this is a false detection?*

Please let us know why the file was falsely detected as infected or clean.

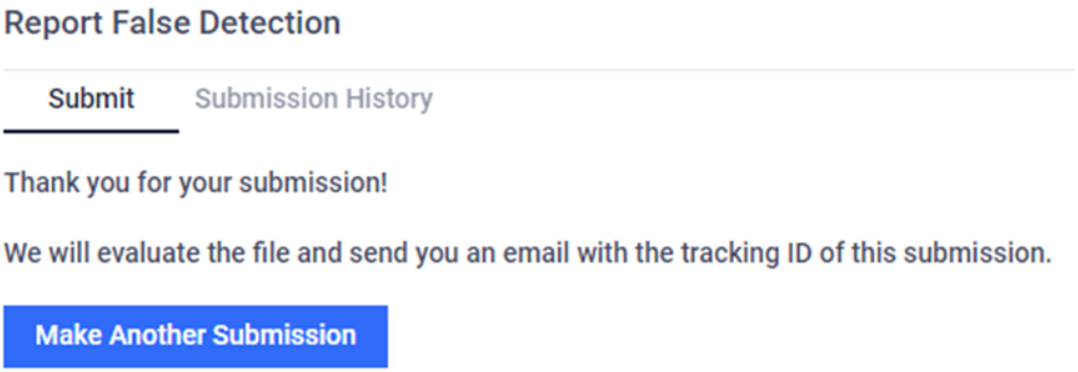
I agree to share the files with AV vendors for further analysis.*

Clear | Submit for Analysis

- 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 dependencies does MetaDefender's Deep CDR for Linux have?

Check Your Version:
This article applies to all MetaDefender Core releases deployed on Linux systems.

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.

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
```

Ubuntu 22.04

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
```

Red Hat 8

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

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

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

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 auto$, 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 https://www.opswat.com/docs/mdcore/knowledge-base/how-to-retrieve-the-crl-and-ocsp-urls-for-the-server-and-ica-cer

Support:
If you have any difficulty **Installing** and **Setting Up** MetaDefender Core in a restricted environment, please follow these instructions on auto$, 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 auto$, 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.

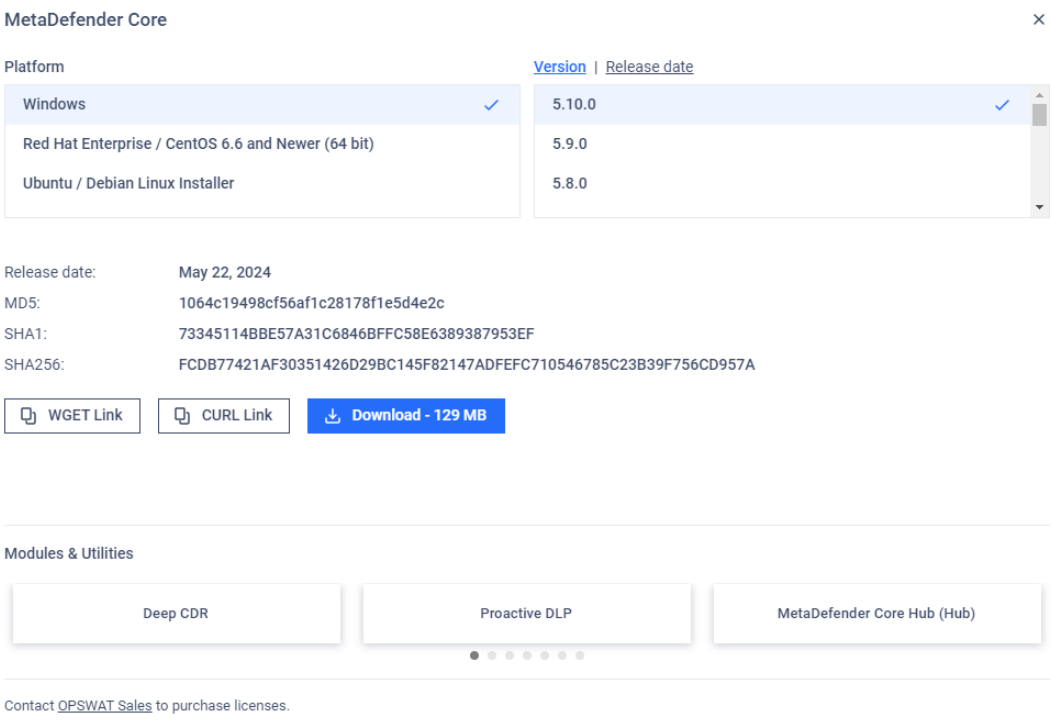
There are four steps to getting the product up and running in an offline environment:

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.

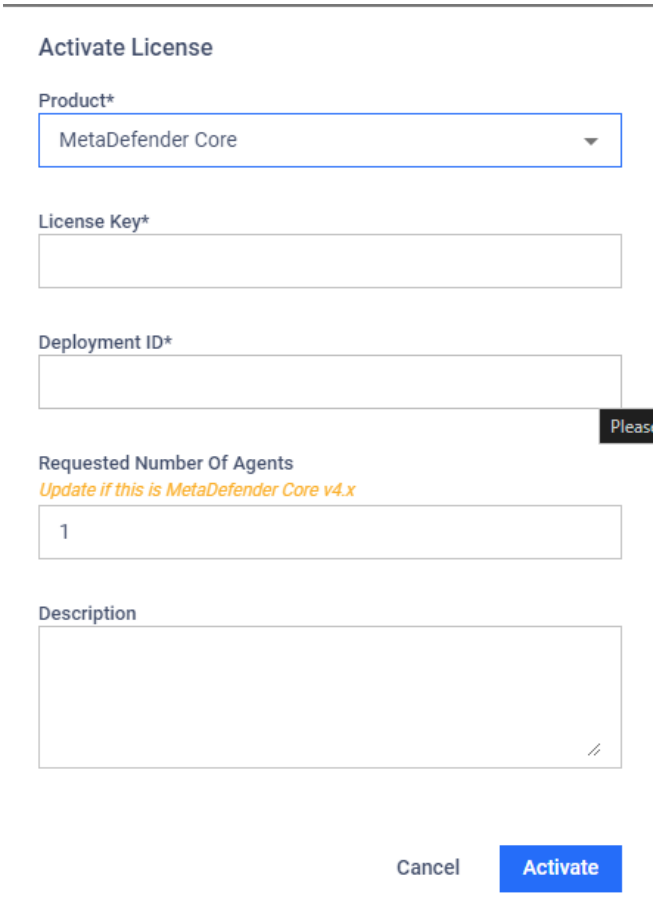
1. After logging into the portal, go to the Downloads page to select the MetaDefender Core package you wish to download.
2. The supported platforms include Windows 7 and later releases.
3. Download the appropriate installer and run it on the Windows system:



B. Activating your license

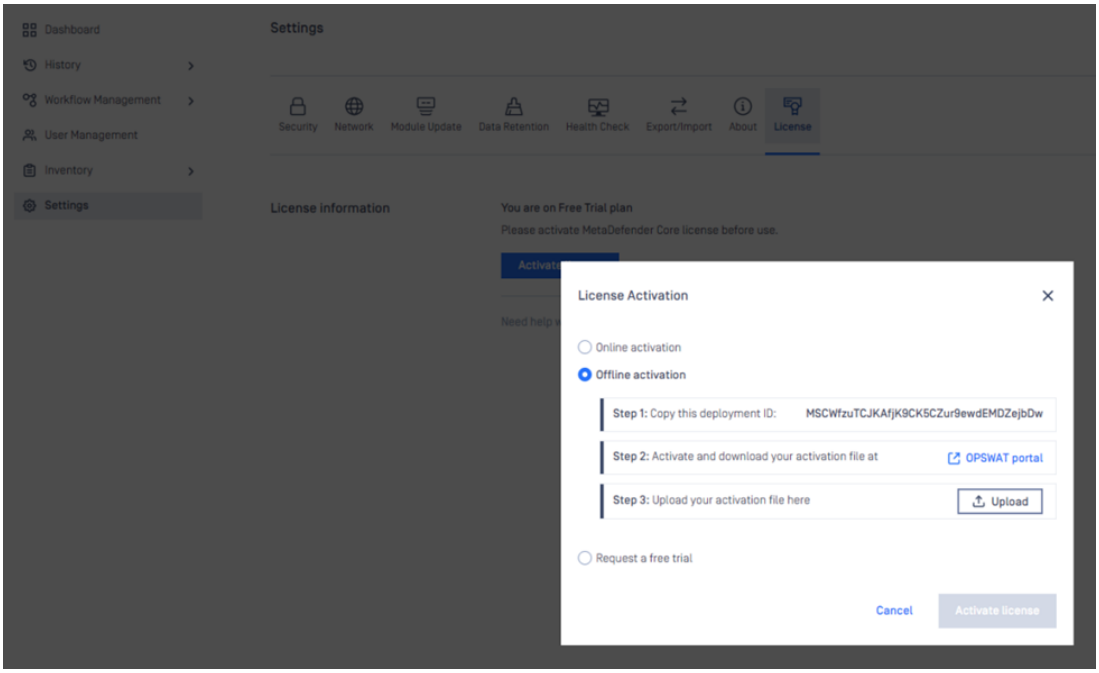
After MetaDefender Core is installed, you will need to activate your MetaDefender Core installation:

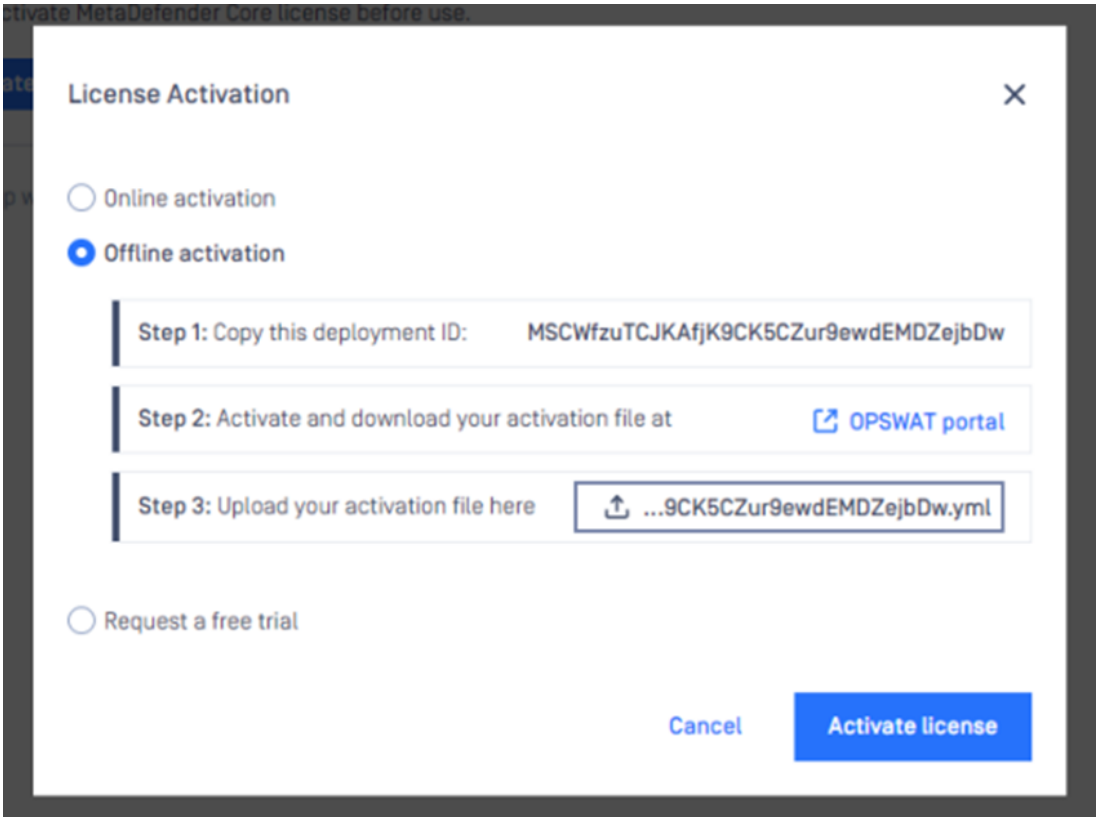
1. Log in to your MetaDefender Core **Management Console** at http://localhost:8008/. Complete the required steps of the Basic Configuration Wizard.
2. Go to **Settings > License > Activate > Offline**.
3. Here, you can follow the steps to:
    a. Copy your **Deployment ID** to the clipboard, then open a new browser tab.
    b. Log in to the My OPSWAT and navigate to the **License Management** page and click on **Activate License** button on the top right.
    c. 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.

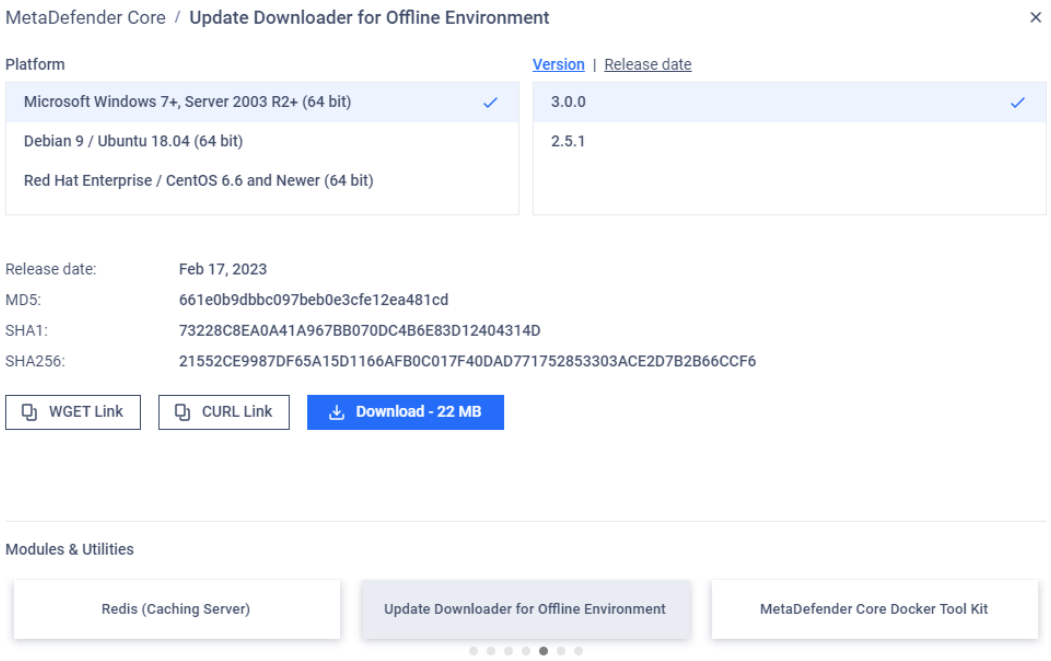




C. Installing the MetaDefender Update Downloader utility

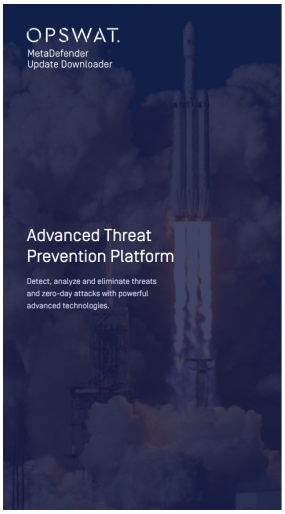
Please Note:
If your MetaDefender Core server is deployed offline, you will need to install the **Update Downloader** utility to download the anti-malware definition updates to be applied to the server.

You can download the **Downloader** utility from My OPSWAT under **Modules & Utilities**.



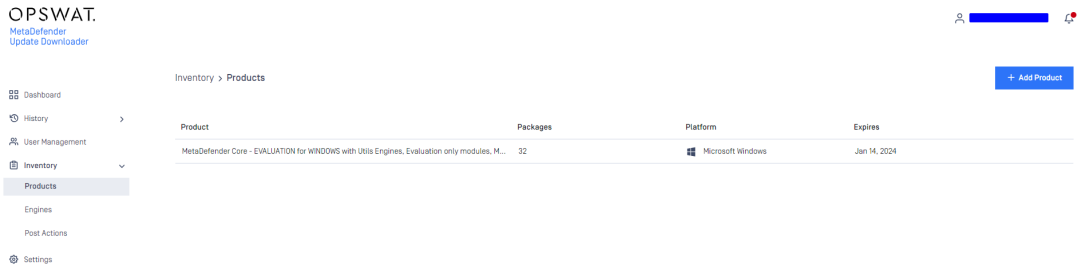
Once you have installed the Update Downloader, apply your license key to activate the product:

1. Log into the **Update Downloader Management Console at http://<server>:8028/**.

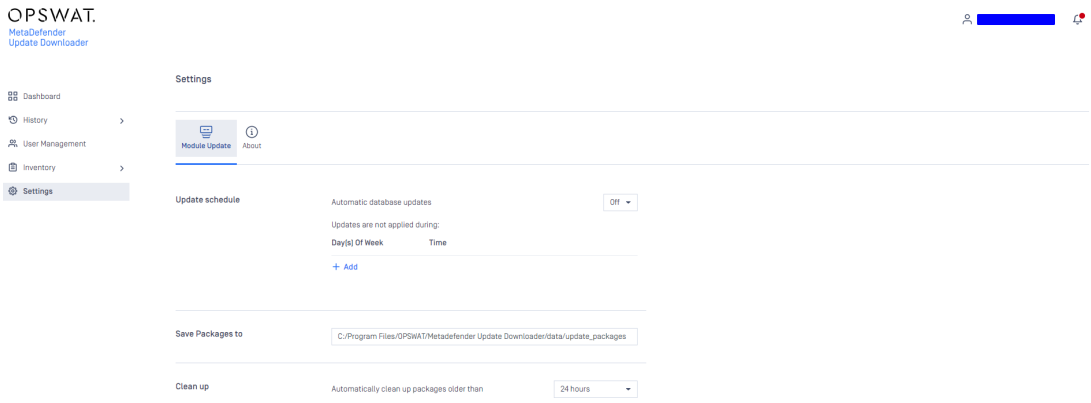




2. Add your product on the **Inventory** page, **Product** tab.



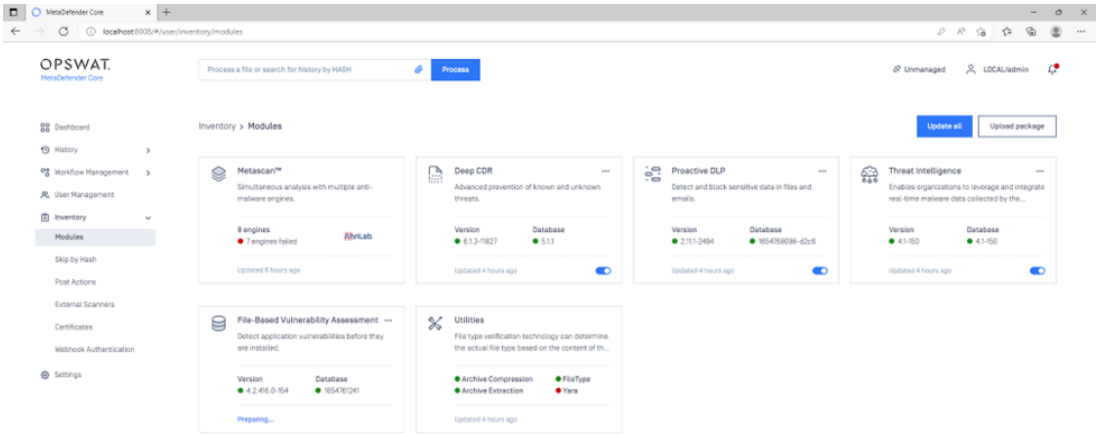
3. Update the configuration for update download and package generation on the **Settings** page of the **Management Console.**



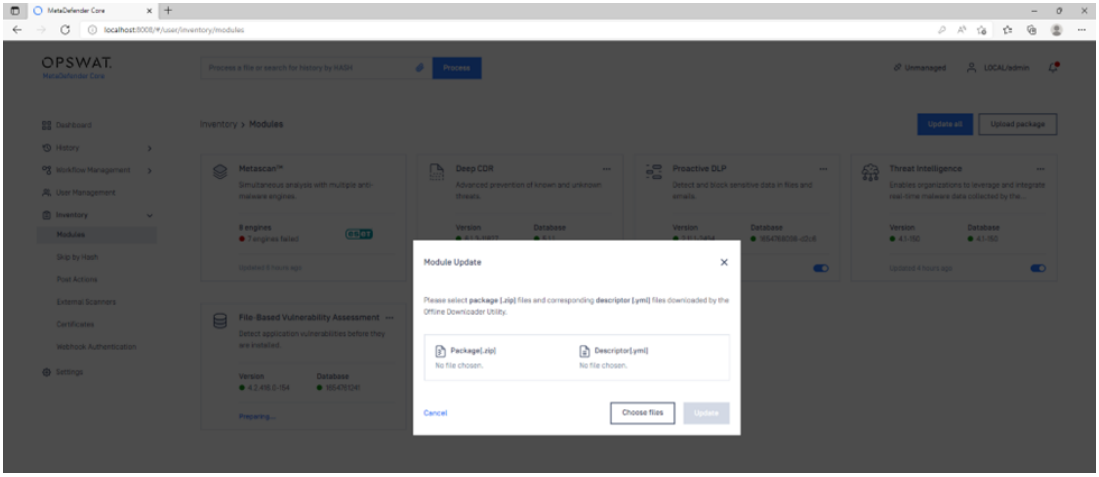
D. Applying offline updates

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 auto$, 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.

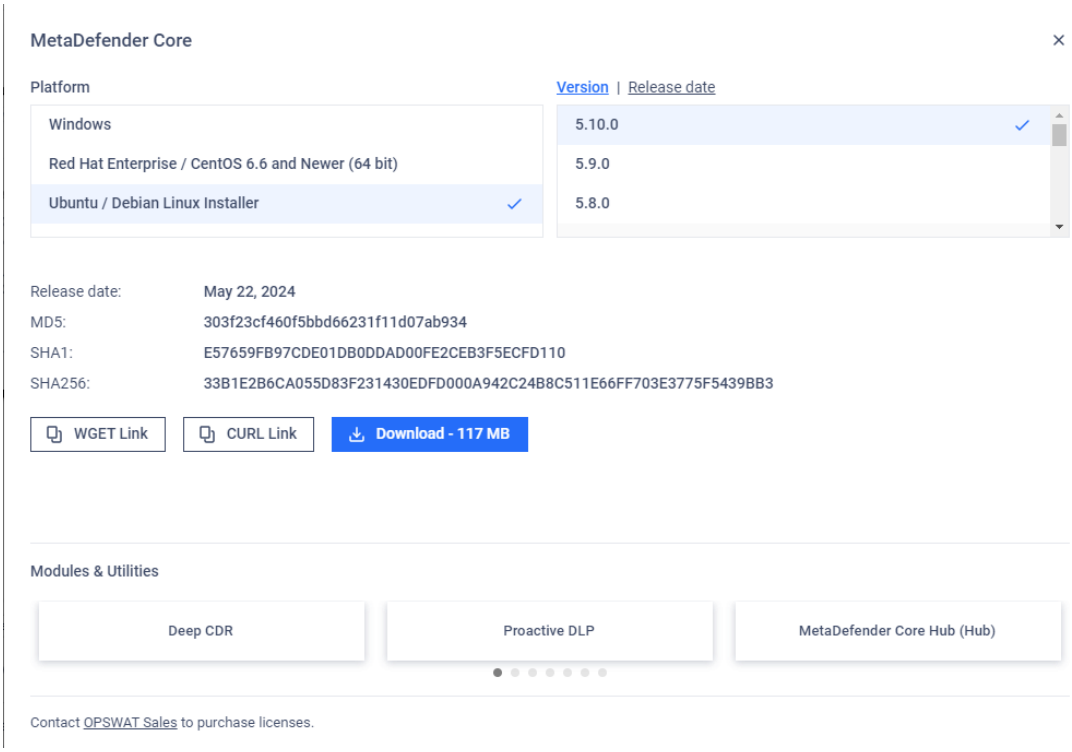
There are four steps to getting the product up and running in an offline environment:

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.

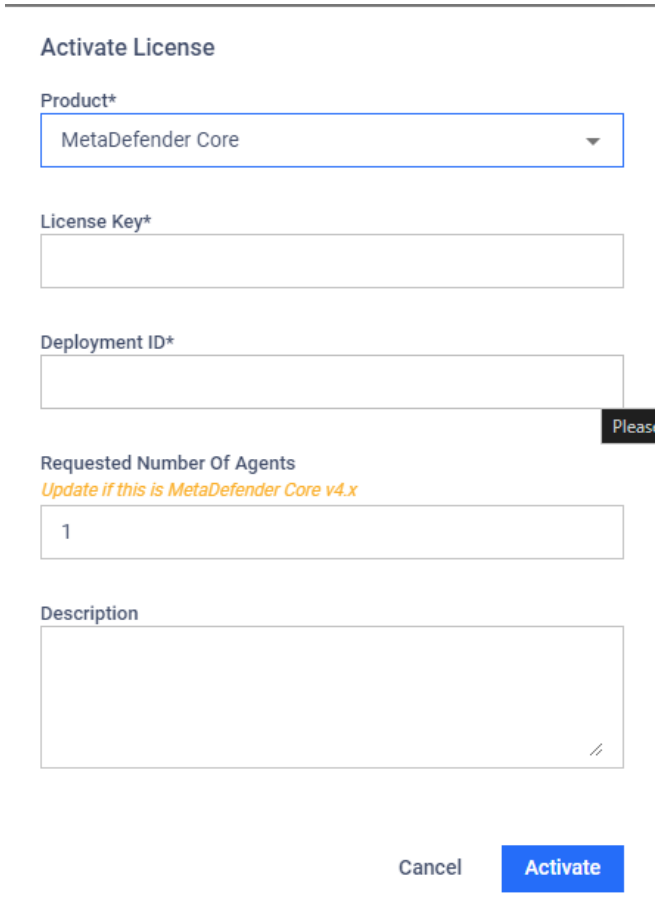
1. After logging into the portal, go to the Downloads page to select the MetaDefender Core package you wish to download.
2. The supported platforms include Red Hat Enterprise, CentOS, Debian, and Ubuntu.
3. Download the appropriate installer for your distribution:



B. Activating your license

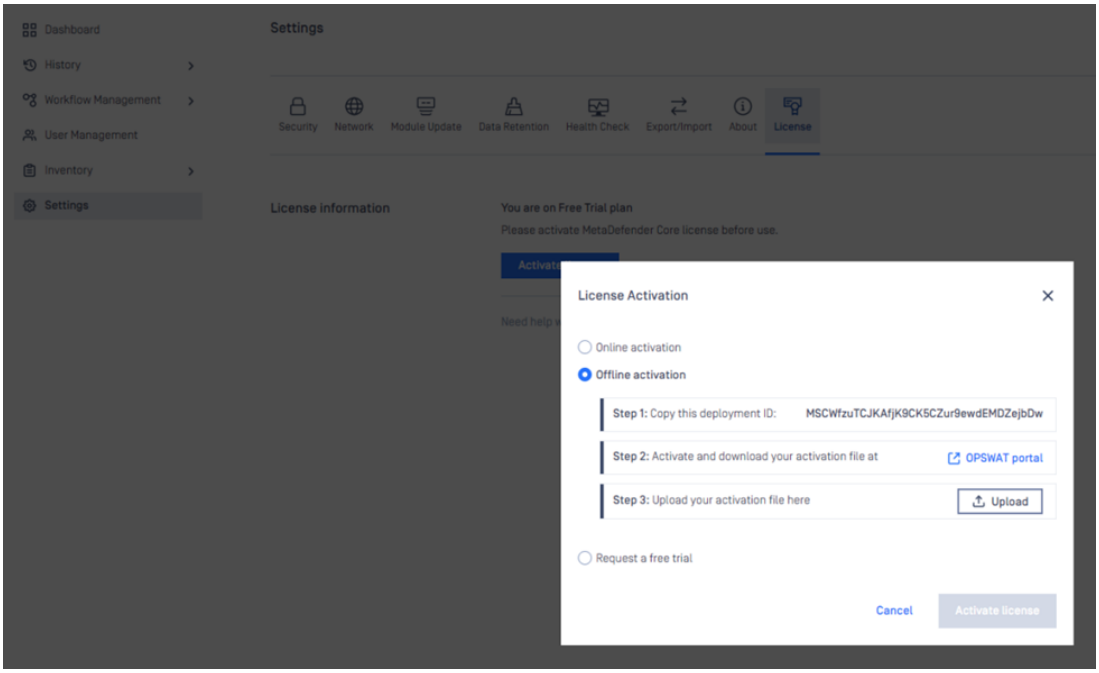
After MetaDefender Core is installed, you will need to activate your MetaDefender Core installation:

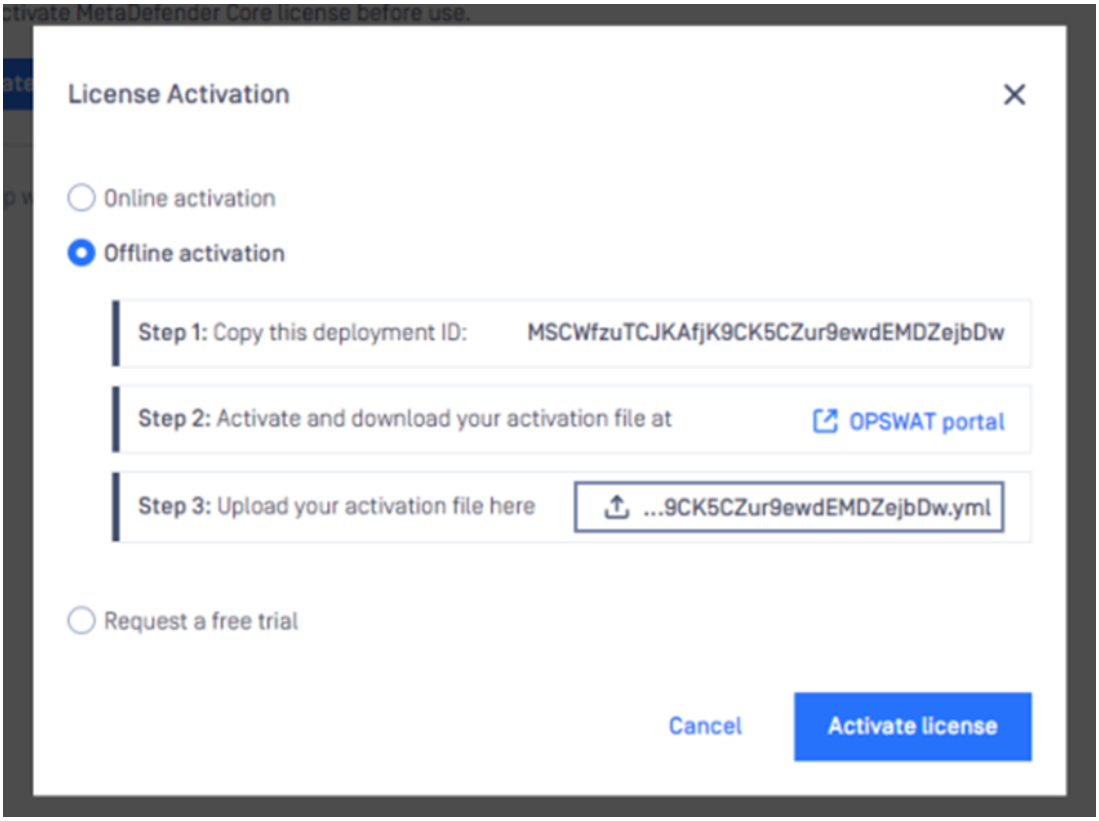
1. Log in to your MetaDefender Core **Management Console** at http://localhost:8008/. Complete the required steps of the Basic Configuration Wizard.
2. Go to **Settings > License > Activate > Offline**.
3. Here, you can follow the steps to:
    a. Copy your **Deployment ID** to the clipboard, then open a new browser tab.
    b. Log in to the My OPSWAT and navigate to the **License Management** page and click on **Activate License** button on the top right.
    c. 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.

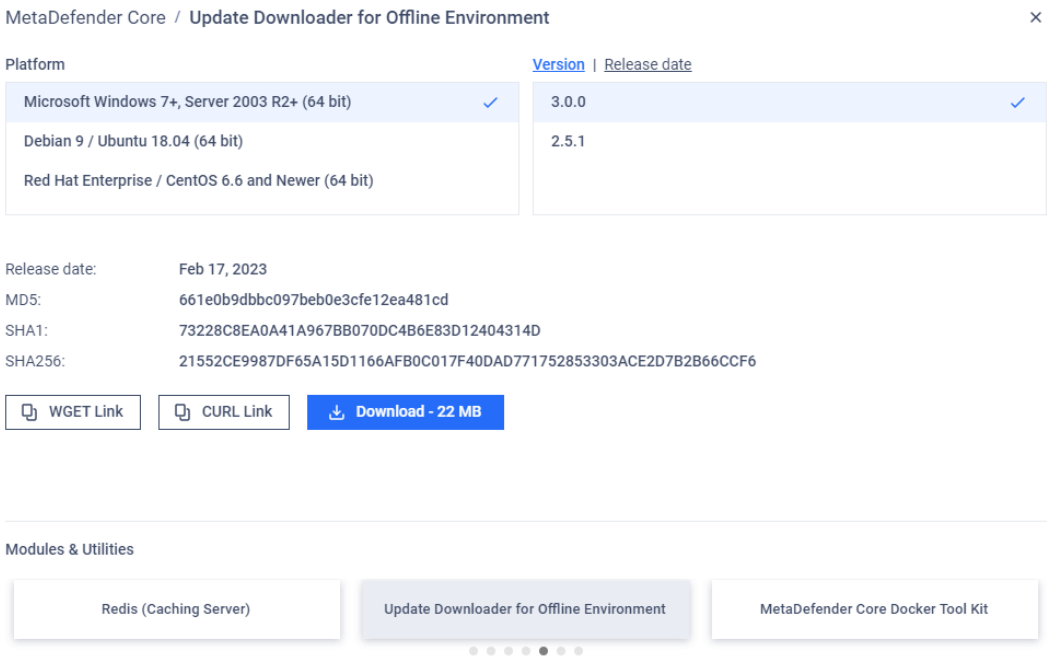




C. Installing the MetaDefender Update Downloader utility

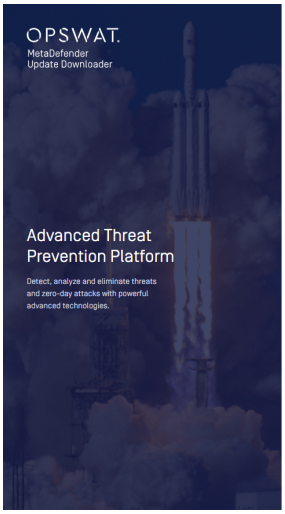
Please Note:
If your MetaDefender Core server is deployed offline, you will need to install the **Update Downloader** utility to download the anti-malware definition updates to be applied to the server.

You can download the **Downloader** utility from My OPSWAT under **Modules & Utilities**.



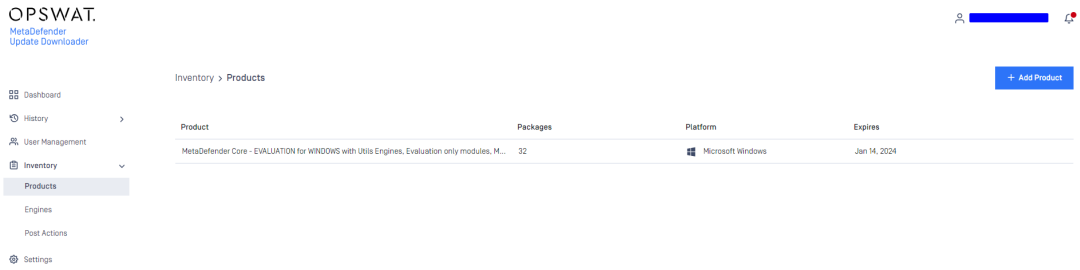
Once you have installed the Update Downloader, apply your license key to activate the product:

1. Log into the **Update Downloader Management Console at http://<server>:8028/**.

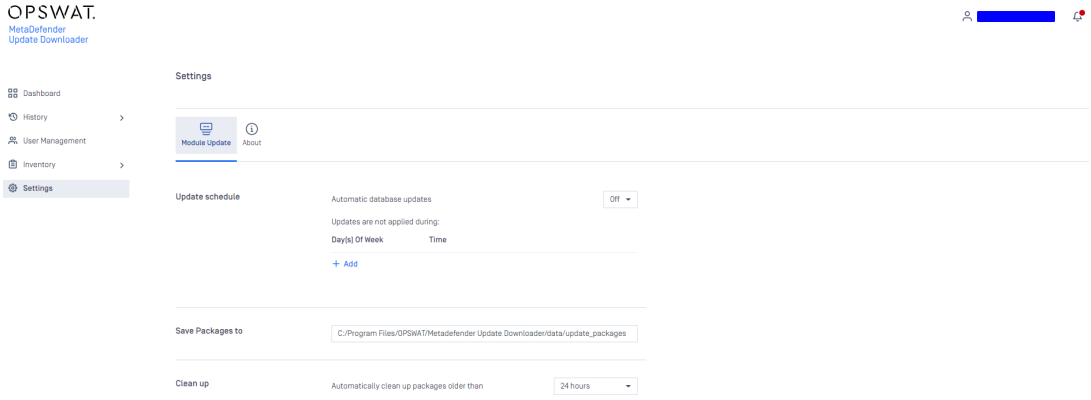




2. Add your product on the **Inventory** page, **Product** tab.



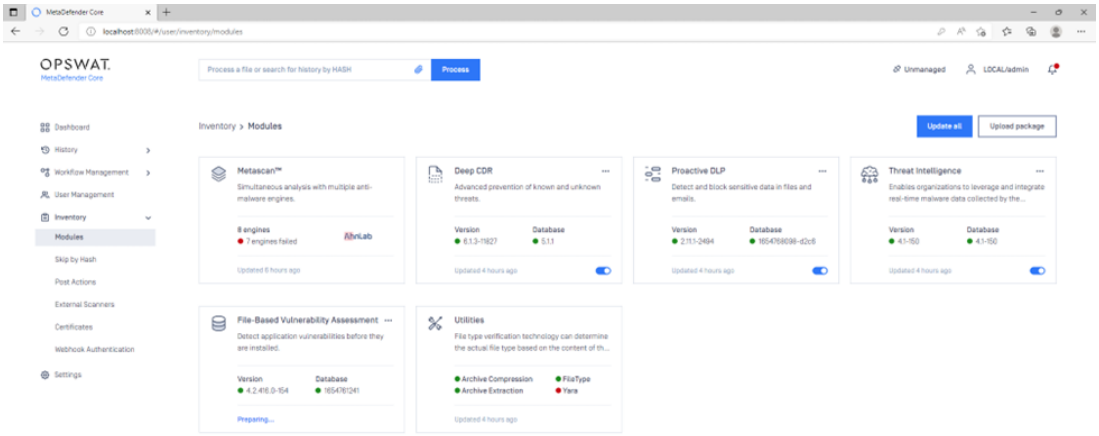
3. Update the configuration for update download and package generation on the **Settings** page of the **Management Console.**



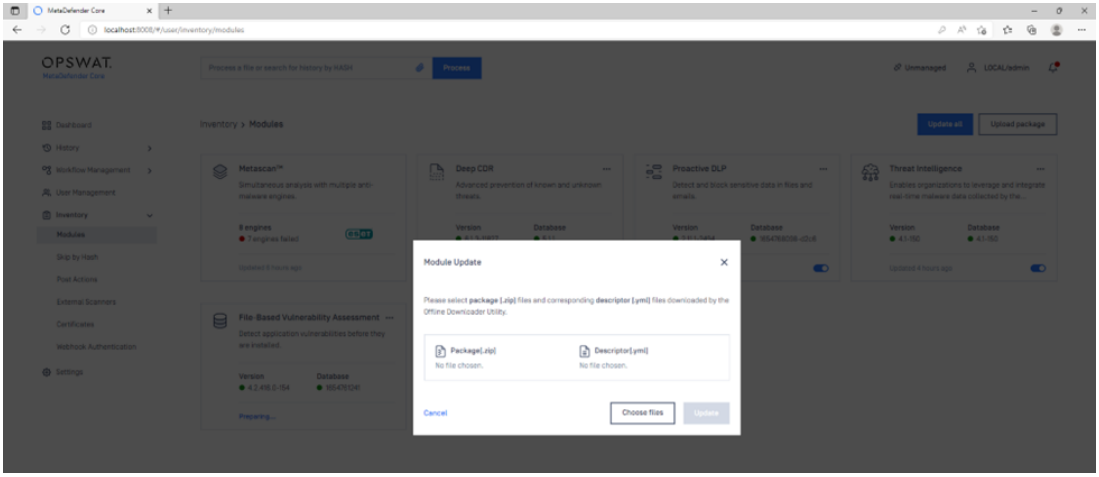
D. Applying offline updates

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.



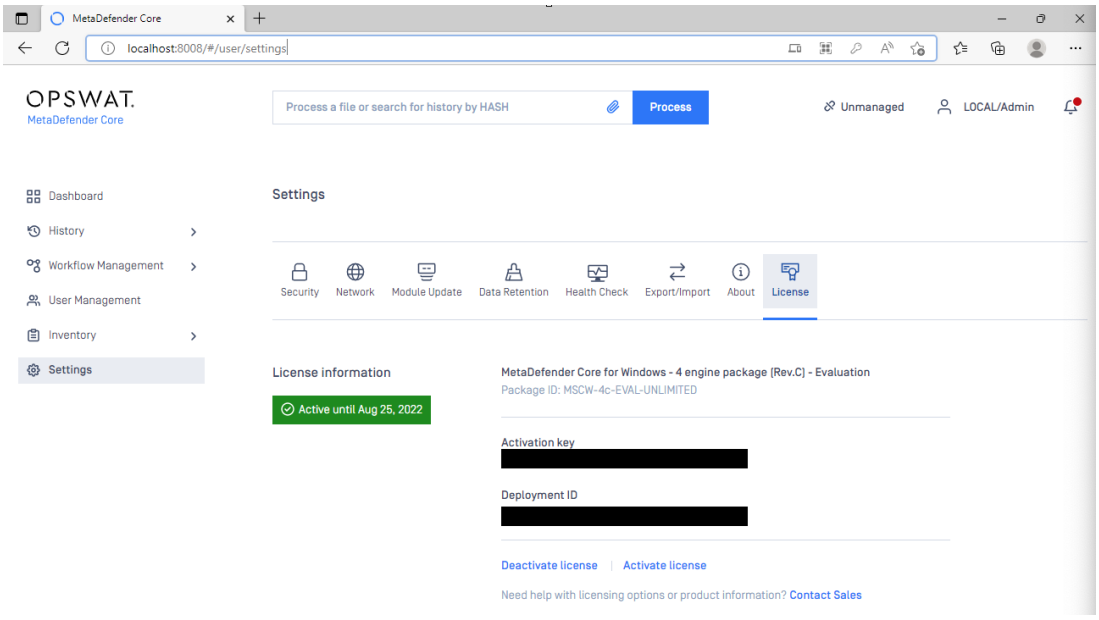
Support:
If you have any difficulty setting up MetaDefender Core in an offline Linux environment, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

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.

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:

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.

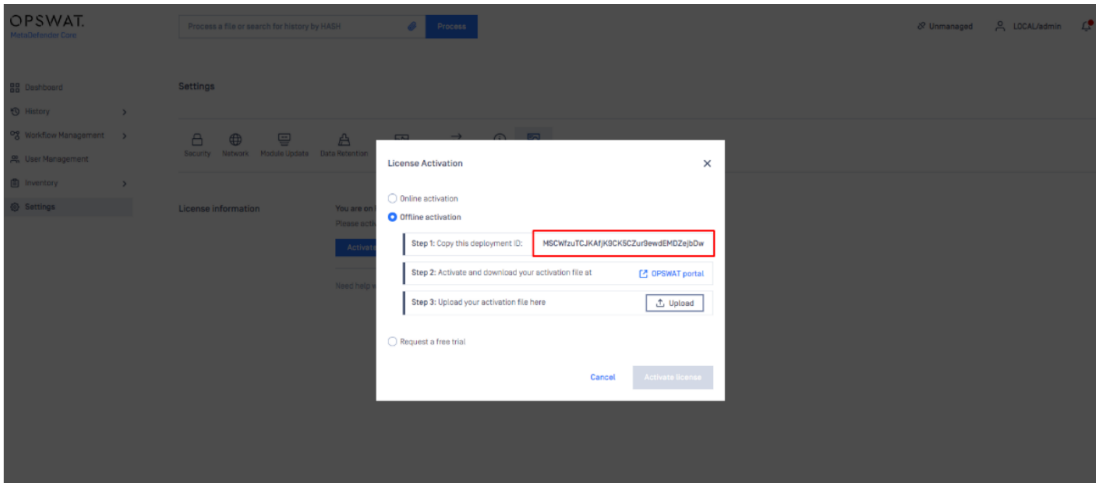
Support:
If you require **Further Assistance With Your Licensing or Deployment ID**, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

How do I activate my offline deployment of MetaDefender Core?

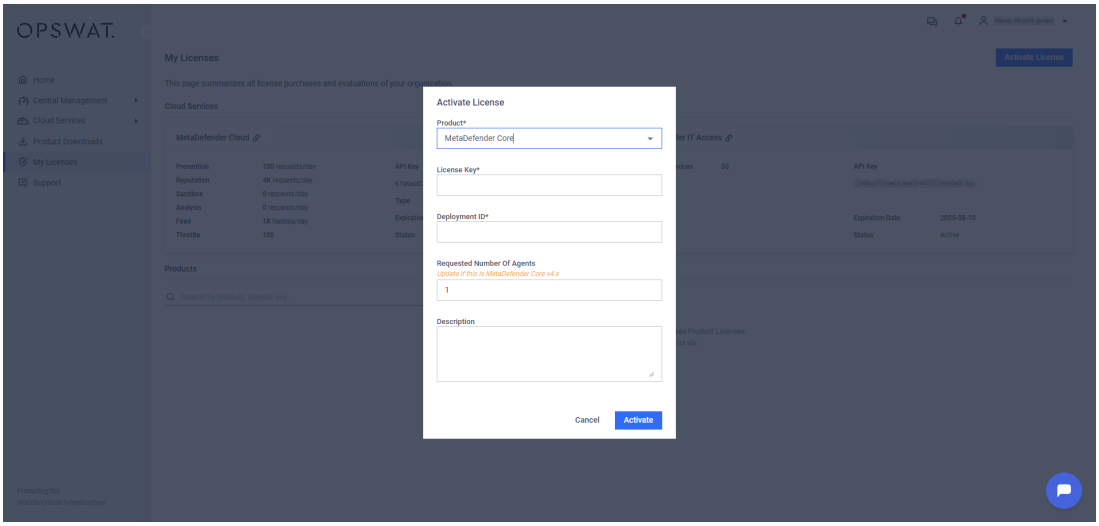
> **Check Your Version:**
> This article applies to all MetaDefender Core V5 releases deployed on offline Windows or Linux systems.

The steps for activating MetaDefender Core in an offline environment are as follows:

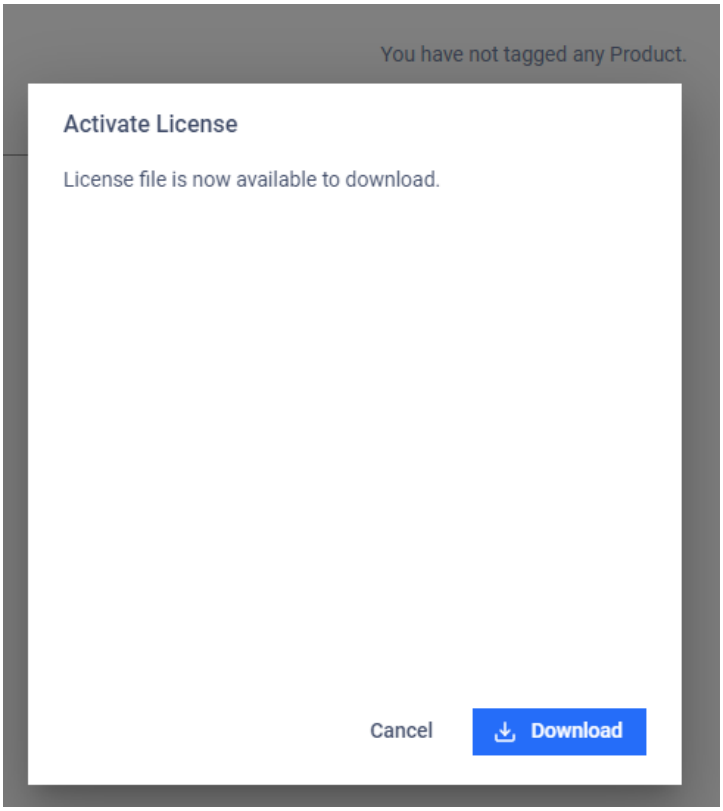
1. Go to your **MetaDefender Core Management Console>Settings>License** tab.
2. Select the **Offline Activation** option.
3. Copy your unique **Deployment ID**, as highlighted in the screenshot below.



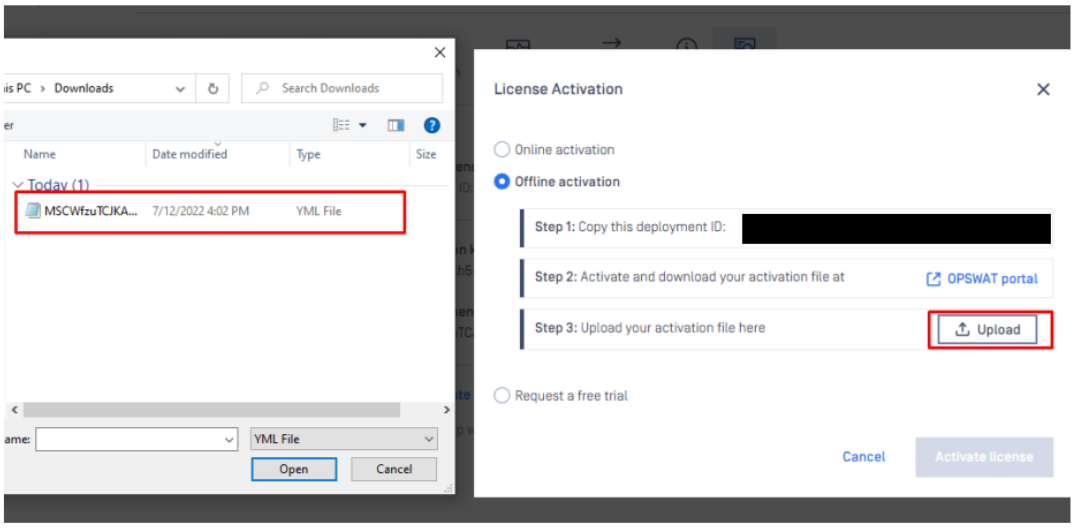
4. Navigate to MyOPSWAT here and click Activate License on the left side panel.
5. Fill out all necessary information, including your **Activation Key**, **Deployment ID** and selection of the **Package** you require.



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 auto$, 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:

bash

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



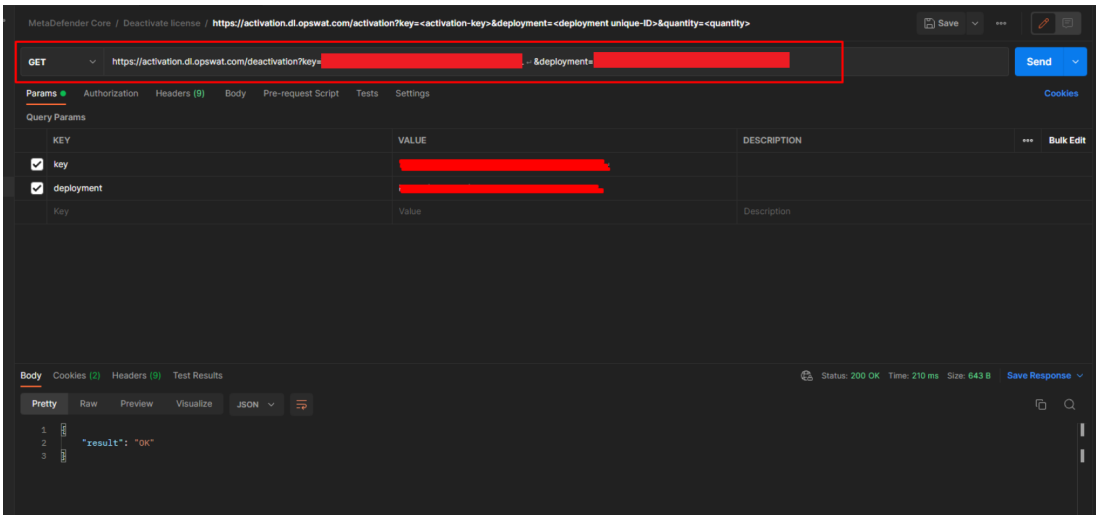
After running the API call described above users will need to use the “save response file” option and use the .yml file to activate the Core deployment.

Deactivate your deployment via API

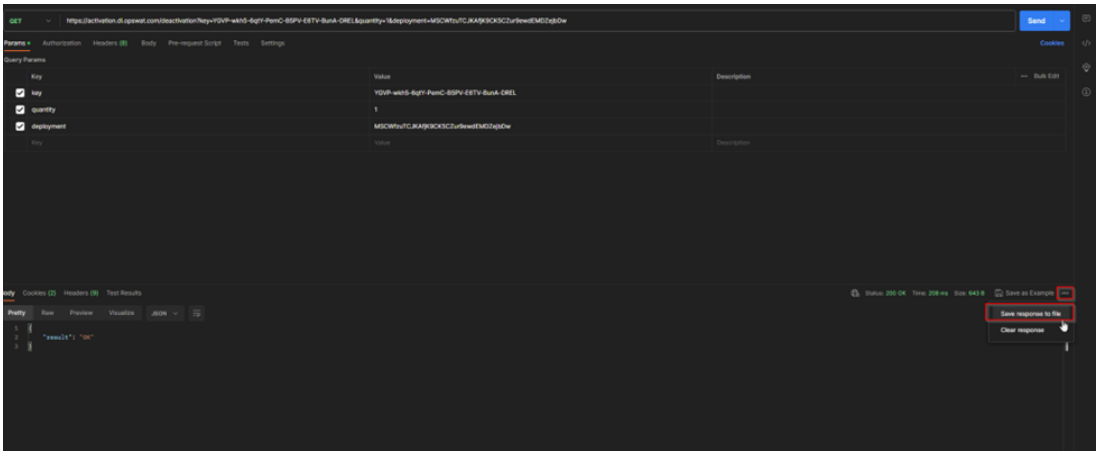
Apply the following API call, as illustrated in the screenshot below:

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 auto$, before creating a support case or chatting with our support engineer.

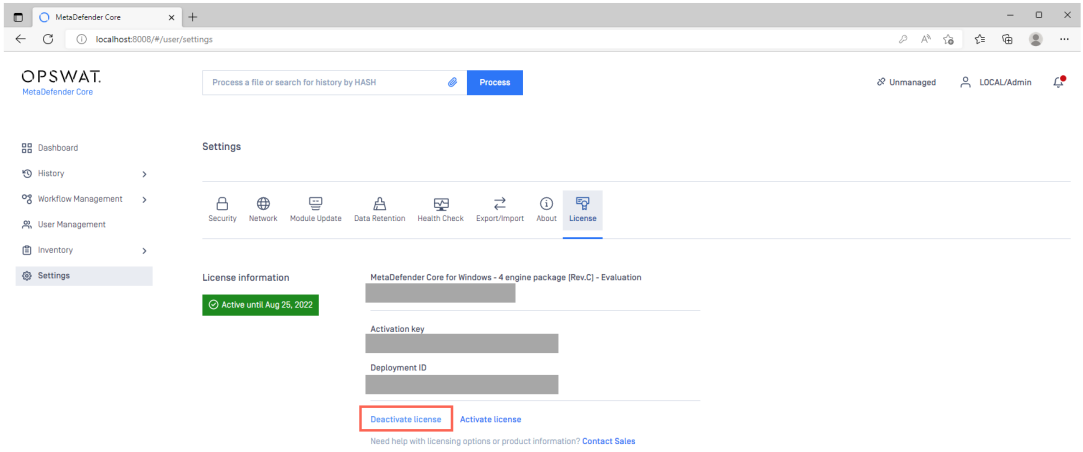
How do I update to a new license key on a Metadefender Core server?

Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

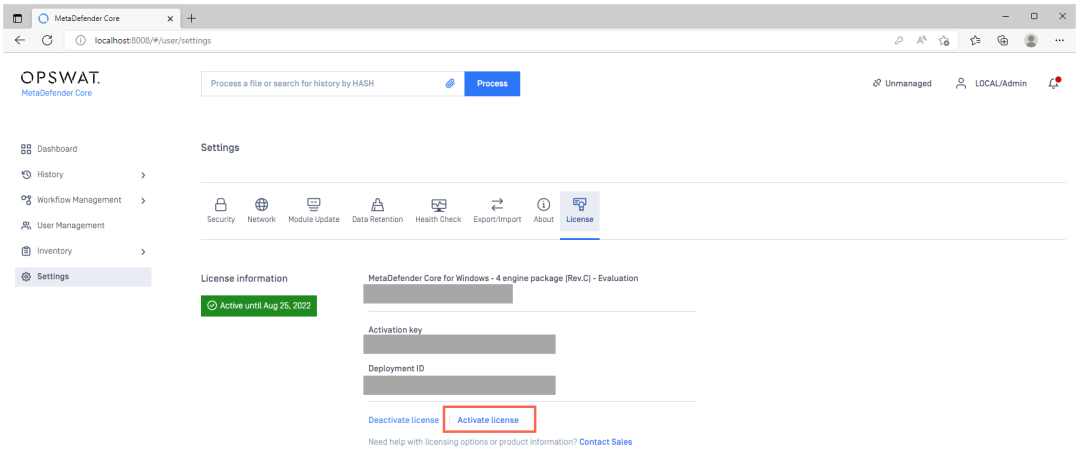
To update to a new license key, you will first need to deactivate your existing deployment (if it hasn't expired or been deactivated already), before activating again under a new license.

Online Activation

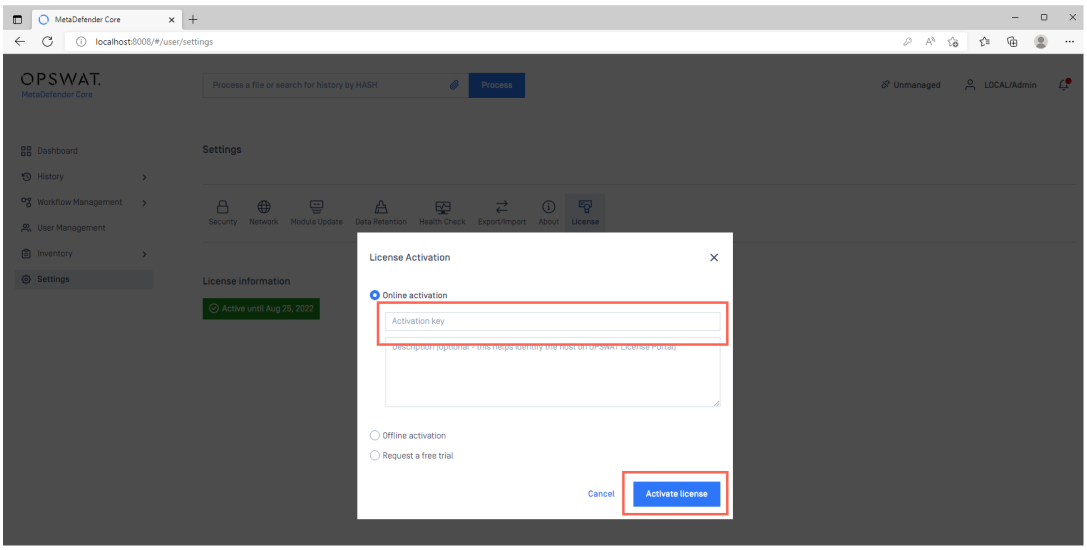
1. To do this, go to your **MetaDefender Core Management Console>Settings>License** tab, then click **Deactivate License**, as highlighted in the screenshot below.



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 auto$, 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.

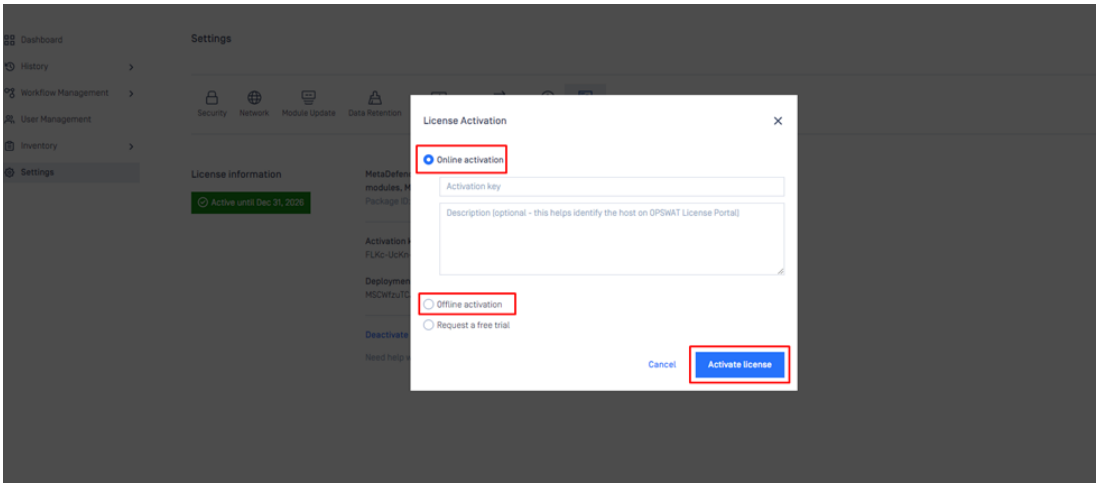
Support:
If you need further guidance in **Choosing The Appropriate License Option For Your Enterprise**, feel free to contact OPSWAT support by creating a support case or chatting with our support engineer.

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 auto$, 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

bash

```
sudo yum install libnsl
sudo dnf install krb5-libs libicu openssl-libs zlib
```

Red Hat Enterprise Linux 9.x / Rocky Linux 9

bash

```
sudo yum install libnsl
sudo dnf install krb5-libs libicu openssl-libs zlib
sudo dnf install compat-openssl11
```

Debian 11.x

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

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

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

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, or feel free to ask the community on our OPSWAT Expert Forum.

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

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 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
    a. Deb/Ubuntu:

bash

```
cp -f ldap.crt /usr/local/share/ca-certificates
update-ca-certificates
```

b. Centos/RHEL:

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

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 need any third party application.
- Windows OS
- PostgreSQL build-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 command below, and you need to run them on CMD (Command Prompt) with Administrator privilege.
- To backup database you need to run the command below.

none

```
<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:

none

```
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:

```
PS D:\prod_2\mdcore\postgres\bin> .\psql.exe --host=localhost --port=5432 --username=postgres --password
Password:

psql (14.11)
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.
Type "help" for help.

postgres=#
```

- type `\l` to show all database instances in PostgreSQL.

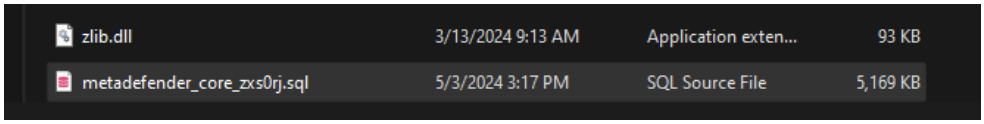
```
postgres=# \l
                                                    List of databases
          Name           |  Owner   | Encoding |          Collate           |           Ctype            |                  Access privileges
-------------------------+----------+----------+----------------------------+----------------------------+-----------------------------------------------------
 metadefender_core_zxs0rj | postgres | UTF8     | English_United States.1252 | English_United States.1252 | postgres=CTc/postgres                              +
                          |          |          |                            |                            | usr_a686fade0737fb766781618526d826b0046b3245=CTc/postgres
 postgres                 | postgres | UTF8     | English_United States.1252 | English_United States.1252 |
 template0                | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres                                        +
                          |          |          |                            |                            | postgres=CTc/postgres
 template1                | postgres | UTF8     | English_United States.1252 | English_United States.1252 | =c/postgres                                        +
                          |          |          |                            |                            | postgres=CTc/postgres
(4 rows)
```

- 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.

none

```
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.

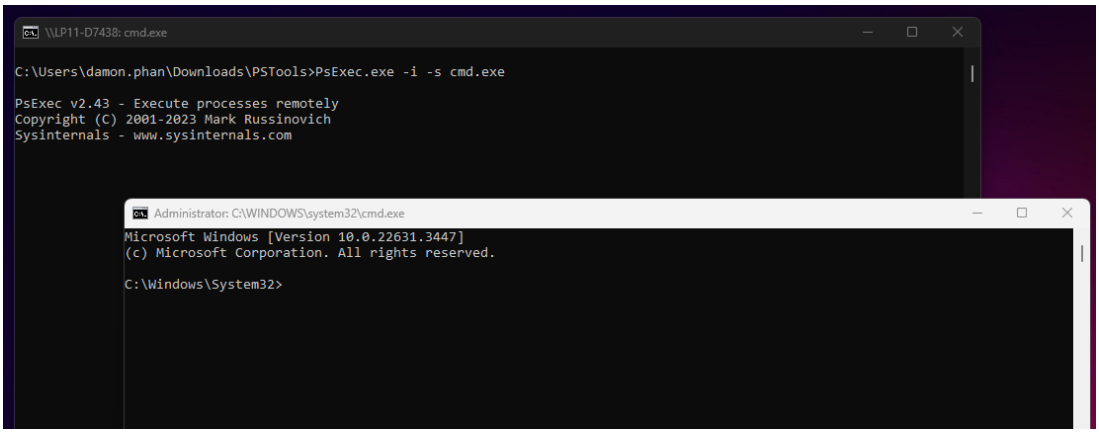
none

```
<installation_folder_MDCore>\postgres\bin\pg_ctl.exe start -D "<PG_DATA_DIR>" -w
-o -p5432
```

- For example:

```
D:\prod_2\mdcore\postgres\bin>pg_ctl.exe start -D "D:\prod_2\mdcore\data\pg_data" -w -o "-p5432"
waiting for server to start....2024-05-03 16:24:20.784 +07 [20312] LOG:  redirecting log output to logging collector process
2024-05-03 16:24:20.784 +07 [20312] HINT:  Future log output will appear in directory "D:/prod_2/mdcore/data/pg_data/log".
 done
server started
```

- **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.

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>whoami
nt authority\system

C:\Windows\System32>
```

- 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.

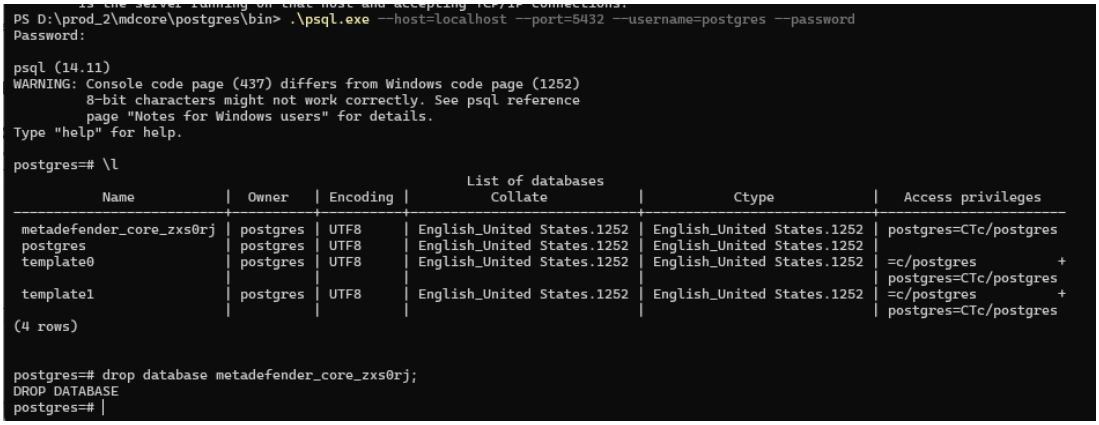
```
C:\Windows\System32>cd "C:\Program Files\OPSWAT\MetaDefender Core\postgres\bin>"

C:\Program Files\OPSWAT\MetaDefender Core\postgres\bin>pg_ctl.exe start -D "C:\Program Files\OPSWAT\MetaDefender Core\da
ta\pg_data" -w -o -p5432
waiting for server to start.... done
server started
```

- 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:

none

```
drop database metadefender_core_zxs0rj;
```

- Here is example:

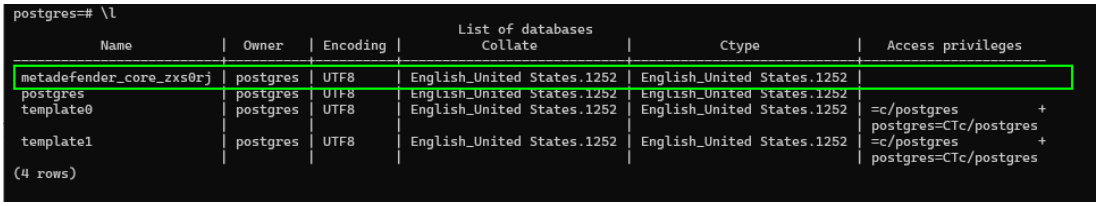


- When you dropped the current database of MD Core, you create the new database again with the same instance name.
- To create database, you can use the command below. Please note that the name of database should be the same.

none

```
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.

none

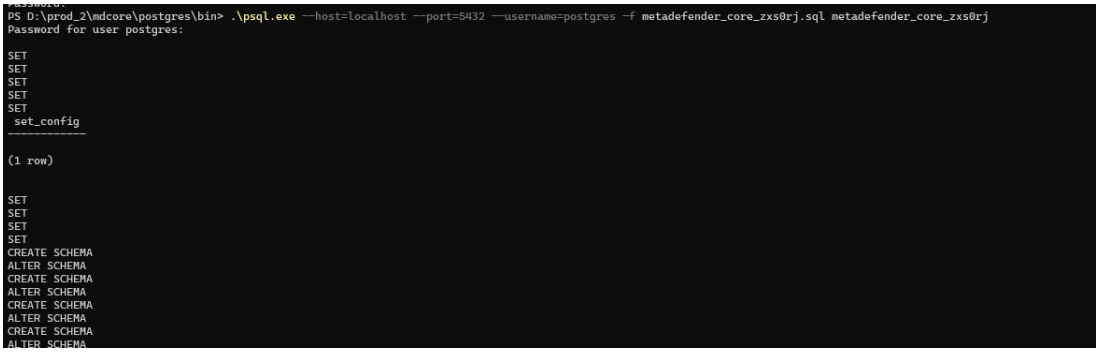
```
<installation_folder_MDCore>\postgres\bin\psql.exe --host=localhost --port=5432 --username=postgres -f <YOU_DUMP_FILE> <database_name>
```

- For example:

none

```
.\psql.exe --host=localhost --port=5432 --username=postgres -f metadefender_core_zxs0rj.sql metadefender_core_zxs0rj
```

- You will the output like this

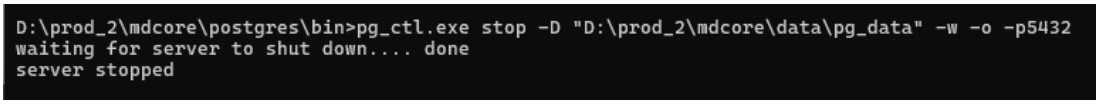


- After you finish import database, you need to stop PostgreSQL. You can use command below to stop

none

```
<installation_folder_MDCore>\postgres\bin\pg_ctl.exe stop -D "<PG_DATA_DIR>" -w -o -p5432
```

- For example:

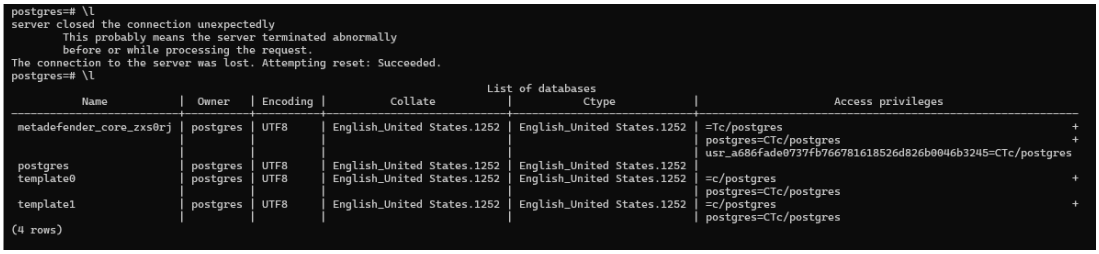


- Finally, you start Metadefender Core service. While starting, MD Core will create new internal user and grant access priviledge for the new database.
- You can recheck by access to PostgreSQL and show status all database, after MD Core service successfully start and running.

none

```
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_a686fade0737fb766781618526d826b0046b3245`

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.

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`.

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:

bash

```
sudo yum install curl cabextract xorg-x11-font-utils fontconfig
```

Once the dependencies have been installed, you can install the .rpm package itself:

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:

bash

```
dnf install curl fontconfig cabextract
```

If you get the following error: `No match for argument: cabextract` you need to enable the EPEL repository:

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:

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:

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`:

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: https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/downloads-list.html
2. Follow the installation instructions provided on the download page.
3. For detailed guidance, refer to the MetaDefender Core Sandbox engine system requirements documentation: https://www.opswat.com/docs/mdcore/sandbox/system-requirements#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.

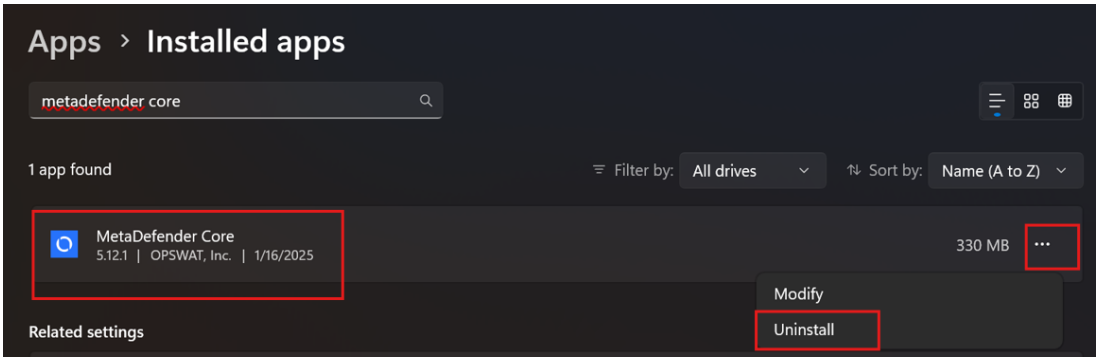
How to completely uninstall MetaDefender Core?

> **Check Your Version:**
> This article applies to all MetaDefender Core V5 releases deployed on Windows and Linux systems.

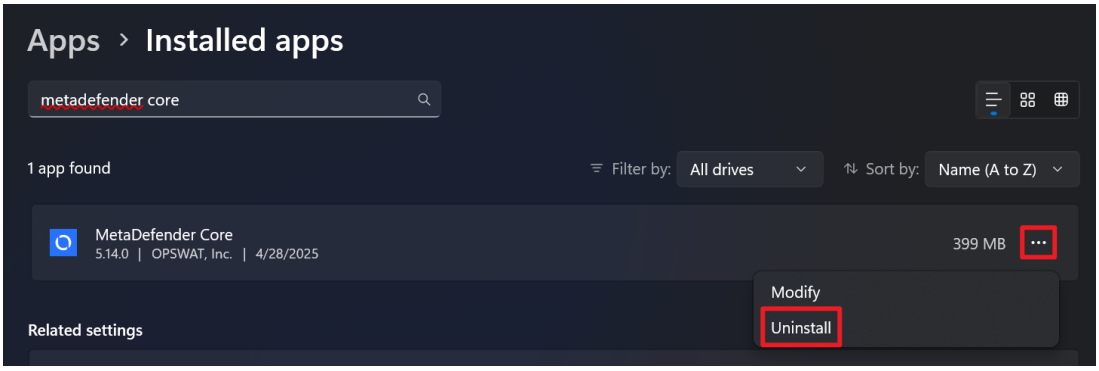
This article provides comprehensive steps to completely uninstall MetaDefender Core, including removal of the application, services, configuration files, logs, and database files on both Windows and Linux platforms.

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: https://www.opswat.com/docs/mdcore/knowledge-base/what-are-the-list-of-folders-that-should-be-removed-after-metade

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.

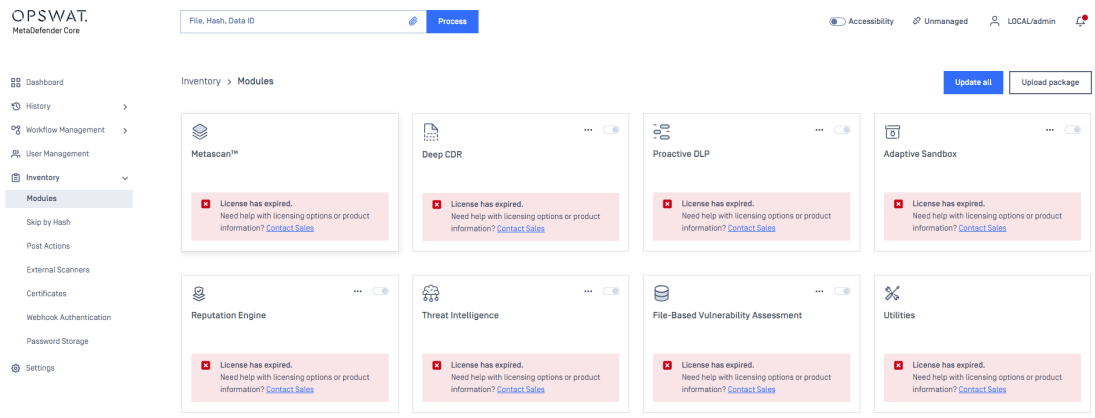
Hardware limitation:
If all software tuning options have been exhausted and CPU utilization remains high, you might need to consider increasing the hardware resources in order to meet the demanding processing load.

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**, 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**.

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**'.

none

```
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 auto$, 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.

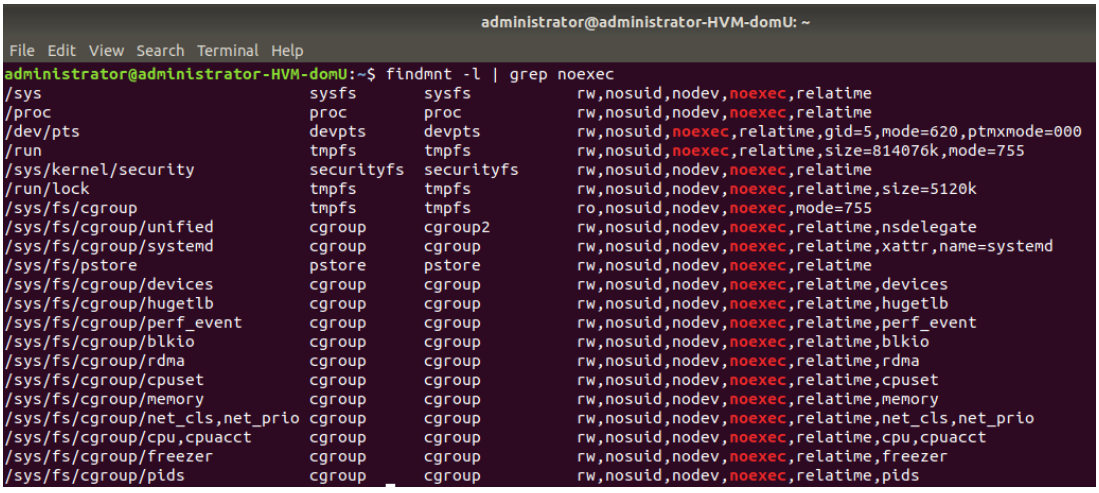
Pro Tip:
On Linux, MetaDefender Core deploys its engines and files in the **/var/lib** and **/usr/lib** folders. Depending on your file mount security policy, you may or may not have mounted **/var** or **/usr** with the flag **noexec**.

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.

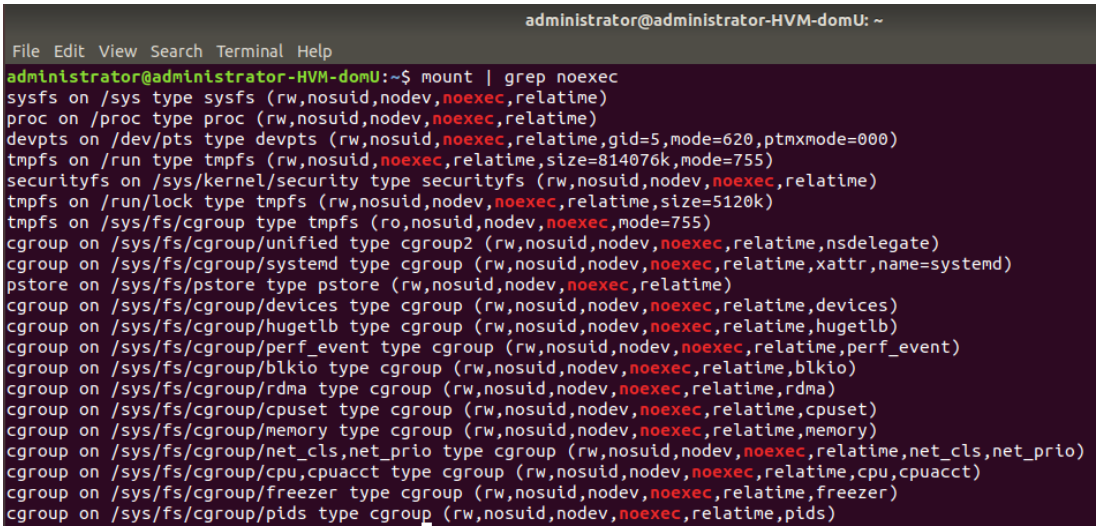
To c heck if the 'noexec' flag exists on /var or /usr simply do the following:

- 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 auto$, 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 auto$, 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.

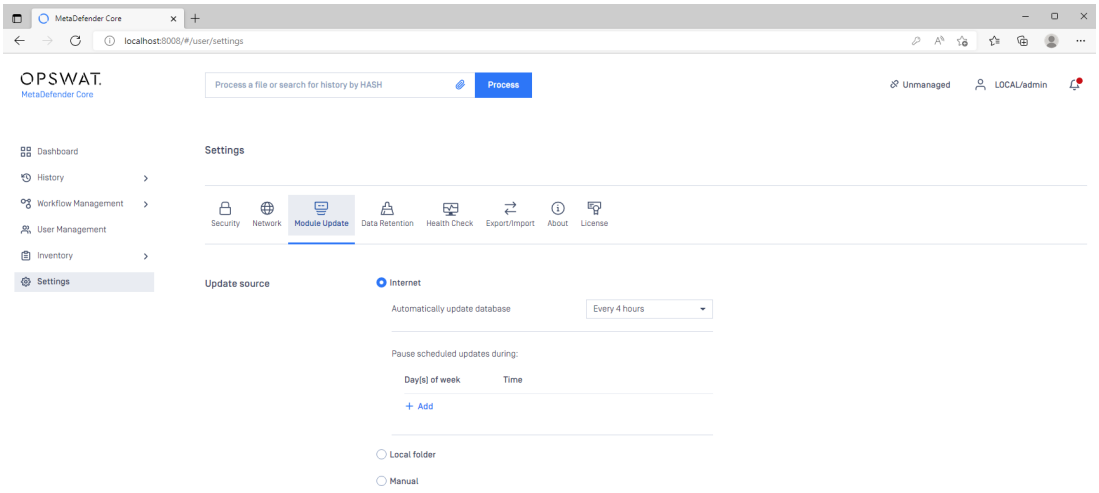
By default, engines and their dependent databases are downloaded and updated automatically and on schedule thanks to MetaDefender Core update settings.

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 auto$, 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.

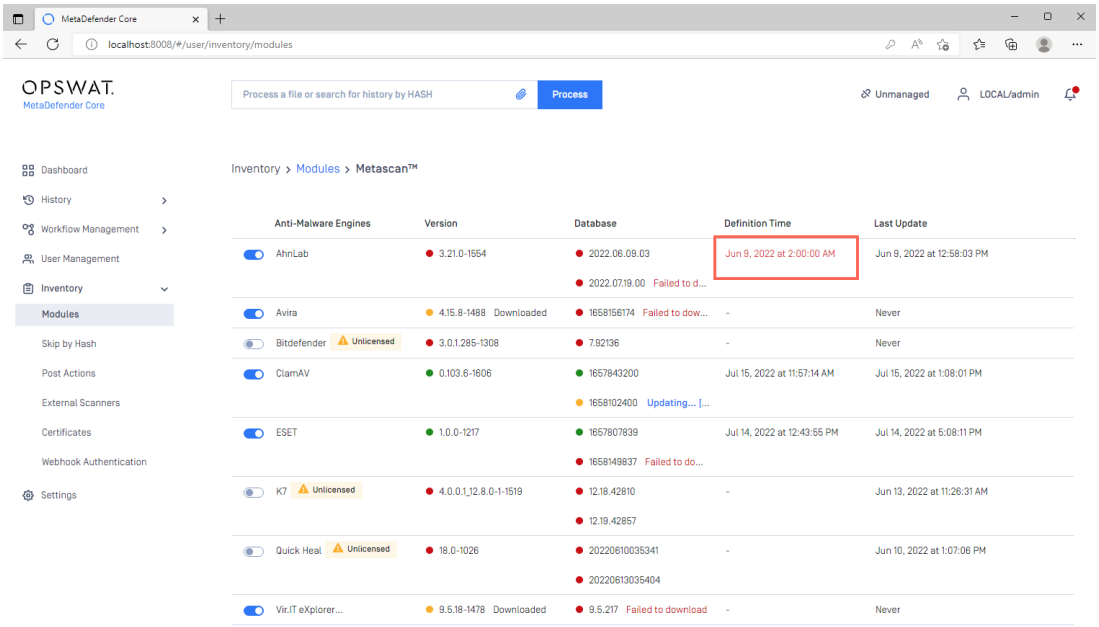
OPSWAT highly recommends that you keep the operating system hosting MetaDefender Core completely updated with the latest OS version updates, patches and bug fixes.

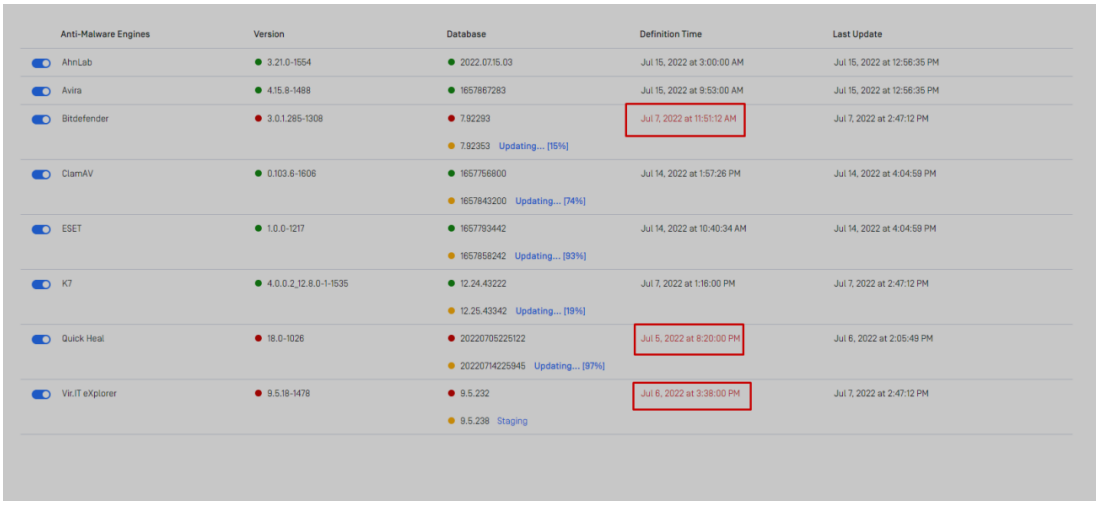
Support:
If your **Operating System** is fully updated but MetaDefender Core is not run optimally, please follow these instructions on auto$, before logging a Support Case with the OPSWAT team.

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.

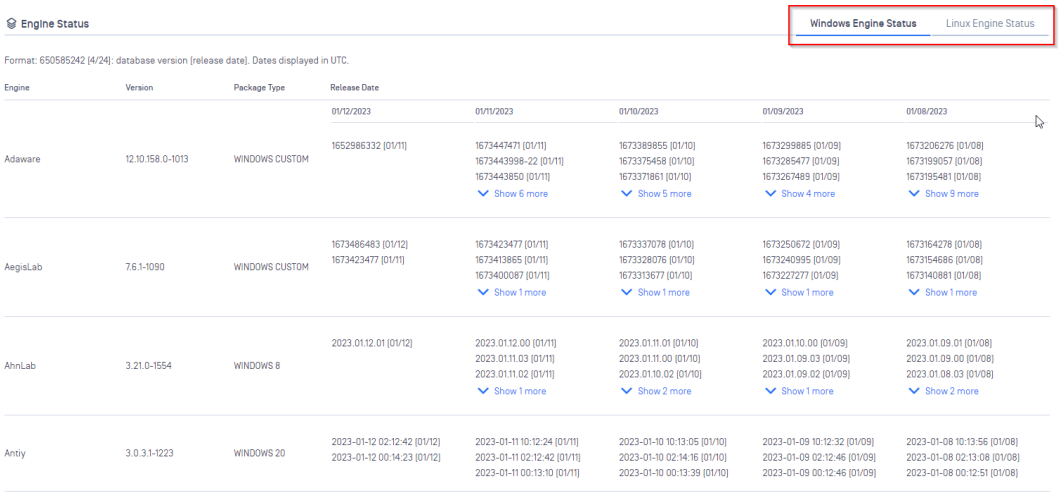




Anti-Malware Engines	Version	Database	Definition Time	Last Update
AhnLab	3.21.0-1554	2022.07.15.03	Jul 15, 2022 at 3:00:00 AM	Jul 15, 2022 at 12:56:35 PM
Avira	4.15.8-1488	1657867283	Jul 15, 2022 at 9:53:00 AM	Jul 15, 2022 at 12:56:35 PM
Bitdefender	3.0.1.285-1308	7.92293	Jul 7, 2022 at 11:51:12 AM	Jul 7, 2022 at 2:47:12 PM
		7.92353 Updating... [15%]		
ClamAV	0.103.6-1606	1657756800	Jul 14, 2022 at 1:57:26 PM	Jul 14, 2022 at 4:04:59 PM
		1657843200 Updating... [74%]		
ESET	1.0.0-1217	1657793442	Jul 14, 2022 at 10:40:34 AM	Jul 14, 2022 at 4:04:59 PM
		1657858242 Updating... [93%]		
K7	4.0.0.2_12.8.0-1-1535	12.24.43222	Jul 7, 2022 at 1:16:00 PM	Jul 7, 2022 at 2:47:12 PM
		12.25.43342 Updating... [19%]		
Quick Heal	18.0-1026	20220705225122	Jul 5, 2022 at 8:20:00 PM	Jul 6, 2022 at 2:05:49 PM
		20220714225945 Updating... [97%]		
Vir.IT eXplorer	9.5.18-1478	9.5.232	Jul 6, 2022 at 3:38:00 PM	Jul 7, 2022 at 2:47:12 PM
		9.5.238 Staging		

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.



Engine Status

Windows Engine Status | Linux Engine Status

Format: 650585242 [4/24]: database version [release date]. Dates displayed in UTC.

Engine	Version	Package Type	Release Date	01/12/2023	01/11/2023	01/10/2023	01/09/2023	01/08/2023
Adaware	12.10.158.0-1013	WINDOWS CUSTOM	1652986332 [01/11]	1652986332 [01/11]	1673447471 [01/11] 1673443998-22 [01/11] 1673443850 [01/11] Show 6 more	1673389855 [01/10] 1673375458 [01/10] 1673371861 [01/10] Show 5 more	1673299885 [01/09] 1673285477 [01/09] 1673267489 [01/09] Show 4 more	1673206276 [01/08] 1673199057 [01/08] 1673195481 [01/08] Show 9 more
AegisLab	7.6.1-1090	WINDOWS CUSTOM	1673486483 [01/12] 1673423477 [01/11]	1673486483 [01/12] 1673423477 [01/11]	1673423477 [01/11] 1673413865 [01/11] 1673400087 [01/11] Show 1 more	1673337078 [01/10] 1673328076 [01/10] 1673313677 [01/10] Show 1 more	1673250672 [01/09] 1673240995 [01/09] 1673227277 [01/09] Show 1 more	1673164278 [01/08] 1673154686 [01/08] 1673140881 [01/08] Show 1 more
AhnLab	3.21.0-1554	WINDOWS 8	2023.01.12.01 [01/12]	2023.01.12.01 [01/12]	2023.01.12.00 [01/11] 2023.01.11.03 [01/11] 2023.01.11.02 [01/11] Show 1 more	2023.01.11.01 [01/10] 2023.01.11.00 [01/10] 2023.01.10.02 [01/10] Show 2 more	2023.01.10.00 [01/09] 2023.01.09.03 [01/09] 2023.01.09.02 [01/09] Show 1 more	2023.01.09.01 [01/08] 2023.01.09.00 [01/08] 2023.01.08.03 [01/08] Show 2 more
Antiy	3.0.3.1-1223	WINDOWS 20	2023-01-12 02:12:42 [01/12] 2023-01-12 00:14:23 [01/12]	2023-01-12 02:12:42 [01/12] 2023-01-12 00:14:23 [01/12]	2023-01-11 10:12:24 [01/11] 2023-01-11 02:12:42 [01/11] 2023-01-11 00:13:10 [01/11]	2023-01-10 10:13:05 [01/10] 2023-01-10 02:14:16 [01/10] 2023-01-10 00:13:39 [01/10]	2023-01-09 10:12:32 [01/09] 2023-01-09 02:12:46 [01/09] 2023-01-09 00:12:46 [01/09]	2023-01-08 10:13:56 [01/08] 2023-01-08 02:13:08 [01/08] 2023-01-08 00:12:51 [01/08]

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 auto$, before creating a support case or chatting with our support engineer.

What is the frequency of signature/definition updates on MetaDefender Core?

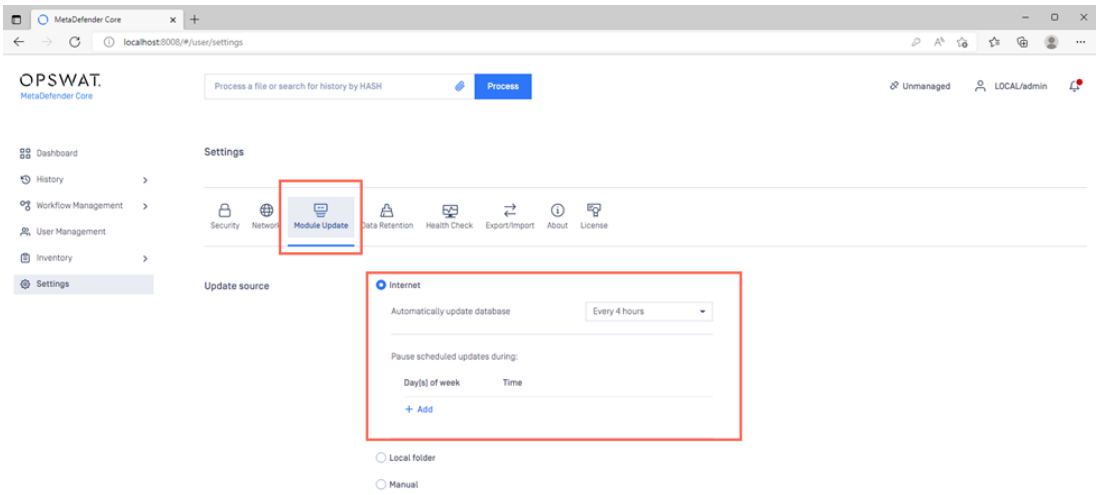
Check Your Version:
This article applies to all MetaDefender Core V4 and V5 releases deployed on Windows or Linux systems.

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.

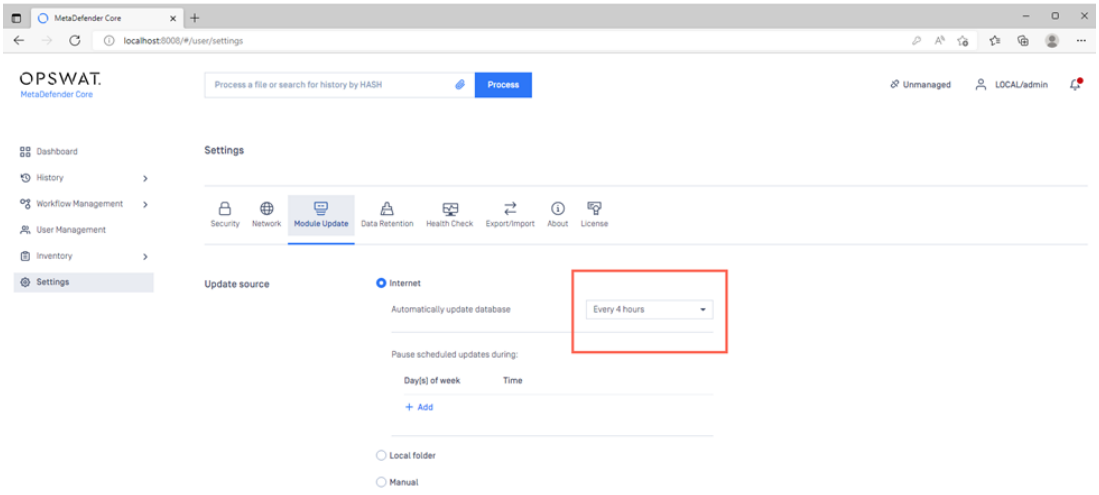
Please Note:
Most antivirus vendors release definitions at least once per day. Many have multiple daily releases. Some vendors release updates on weekends while others do not.

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 auto$.

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 auto$, 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 auto$, 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.

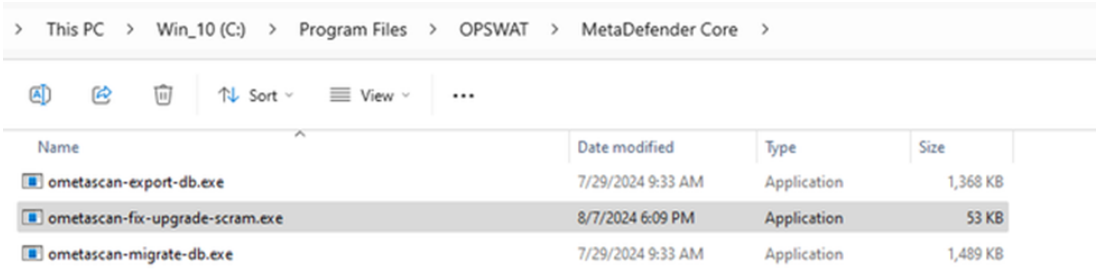
In the long term, a fix will be included in the v5.11.1 release.

Steps to Run the Tool (Password to unzip: **opswat123**)

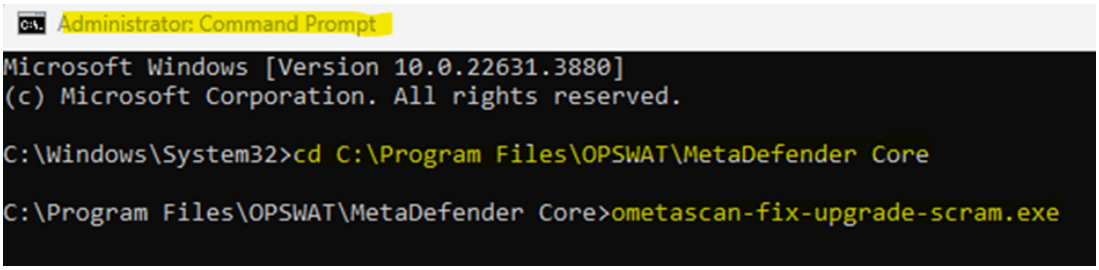
Windows:

Link to download: ometascan-fix-upgrade-scram-Windows.zip (SHA1: F6F0C8444CB99C903B52E0435A088EC398A5483E)

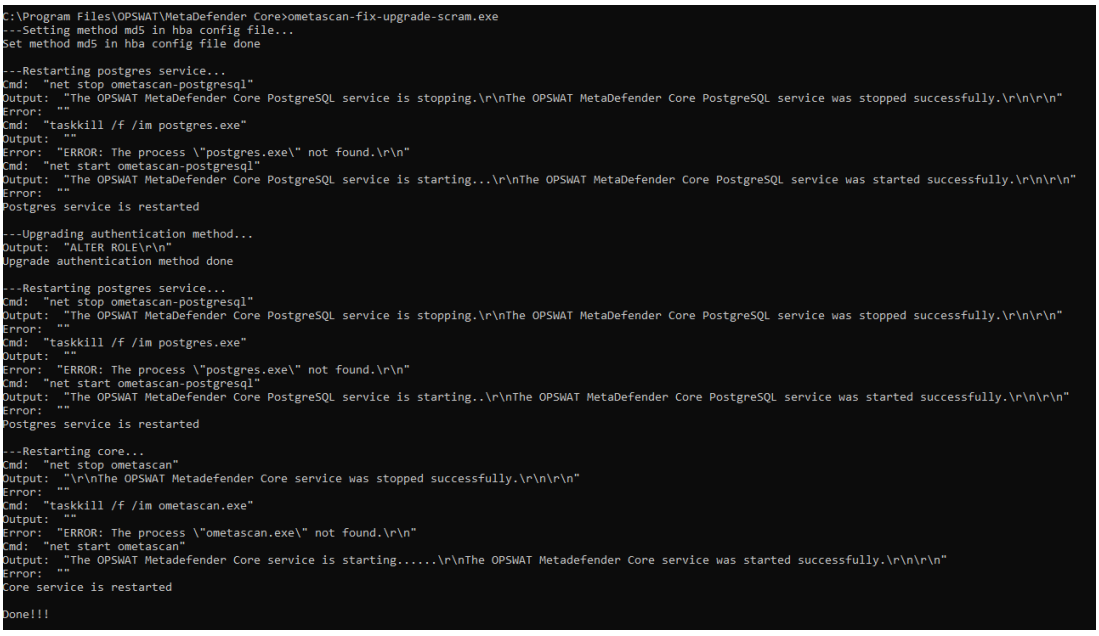
1. Extract and copy the tool to Core’s installation folder on Windows.



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.

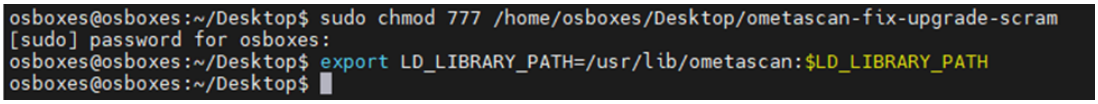
Linux:

Link to download: ometascan-fix-upgrade-scram-Linux.zip (SHA1: F8B0AA96A844EC9C455125007D673CD5F5B7C410)

1. Run the below commands:

bash

```
chmod 777 /path_to_the_tool/ometascan-fix-upgrade-scram
export LD_LIBRARY_PATH=/usr/lib/ometascan:$LD_LIBRARY_PATH
```

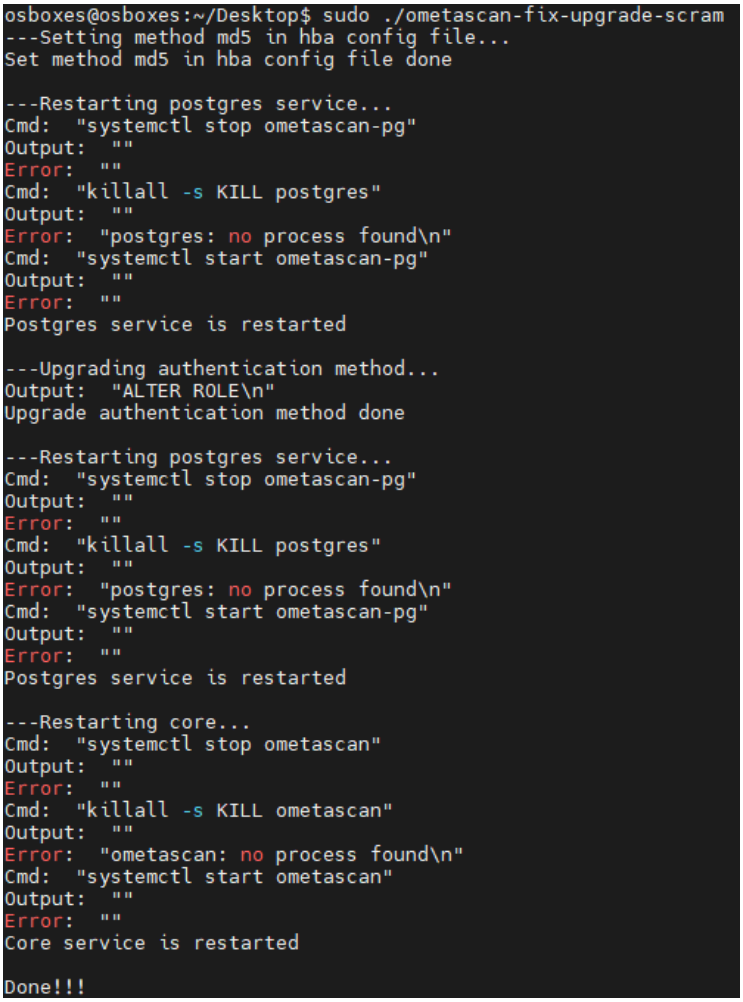


2. Run the tool with *sudo*

bash

```
sudo /path_to_the_tool/ometascan-fix-upgrade-scram
```

3. Check the output, it should looks as below



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 auto$, 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 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	Summary
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 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 : auto$
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)

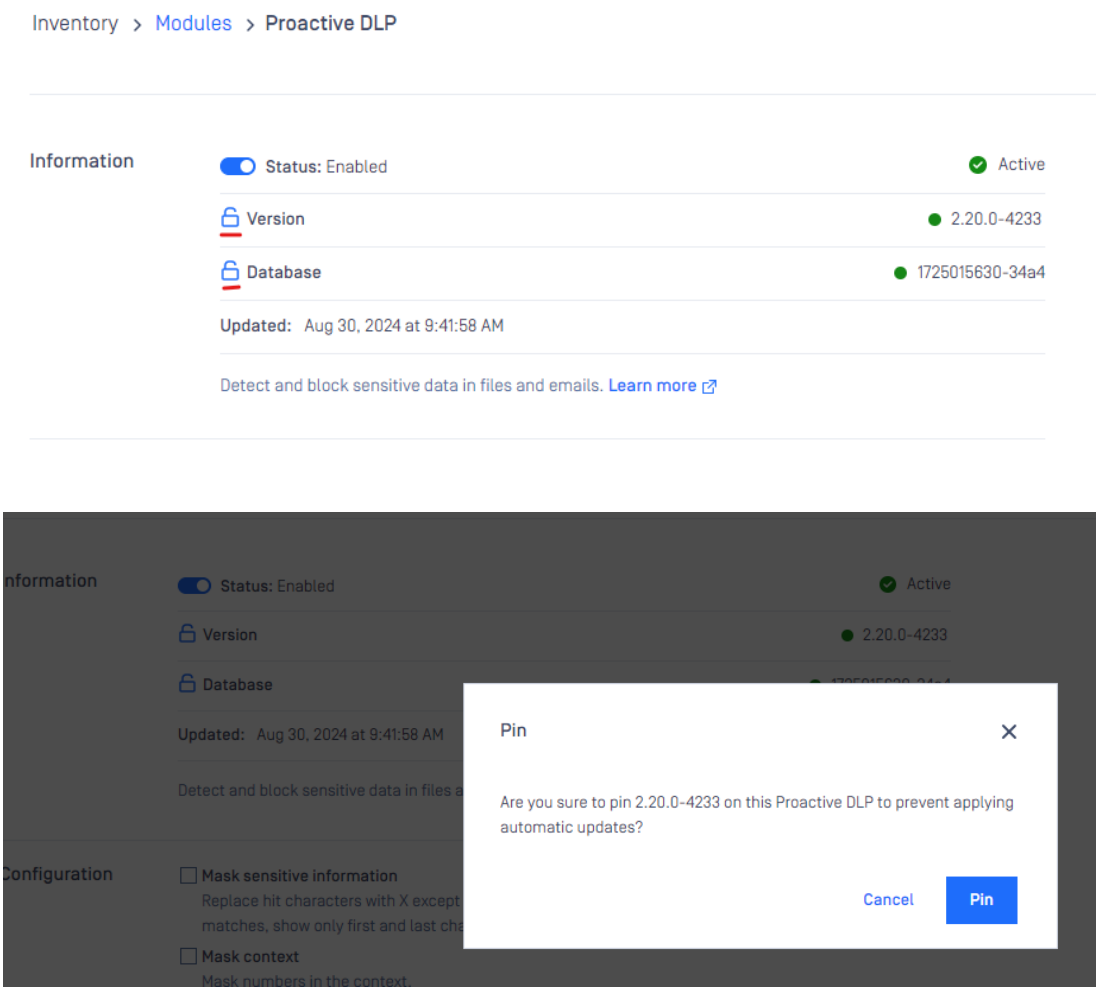
> **Important Notice**
> If you do not transition to a supported operating system, you will no longer receive the latest updates and features from OPSWAT MetaDefender Core and its engines. This could significantly impact the functionality and security of your system.

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**.

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.

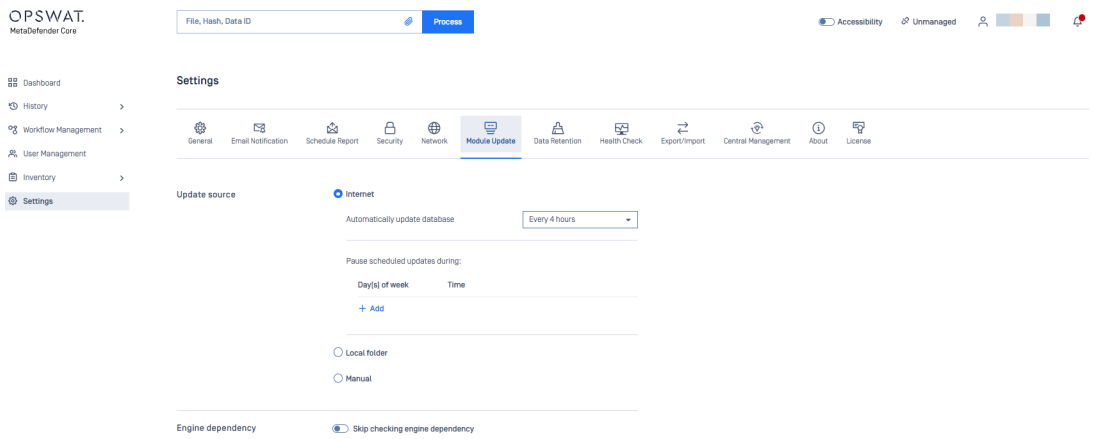
MetaDefender Core is built to ensure high availability and minimize system downtime, even during antivirus engine updates.

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.

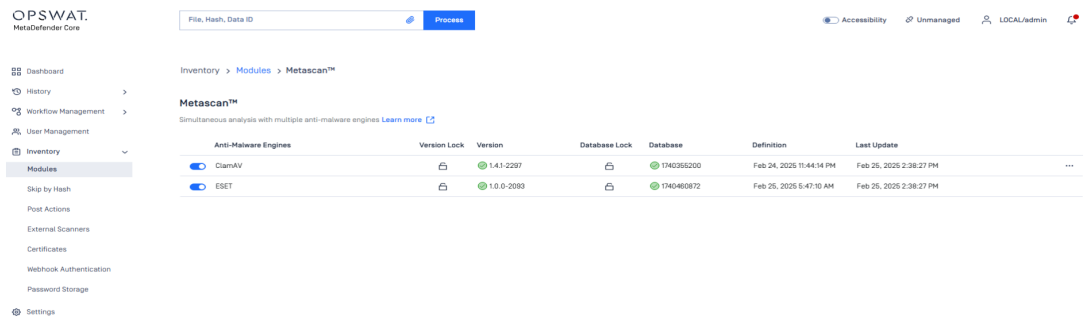
Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

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.

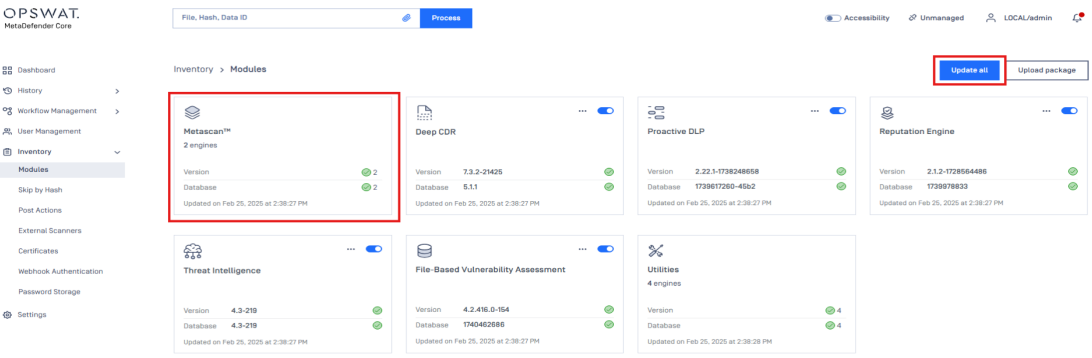
Online download

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**.



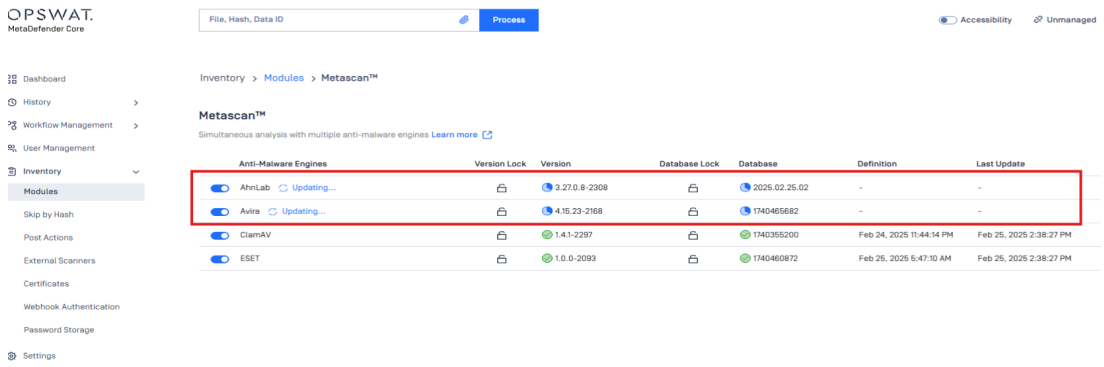
Ahn Lab and Avira missing

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.



Step 1

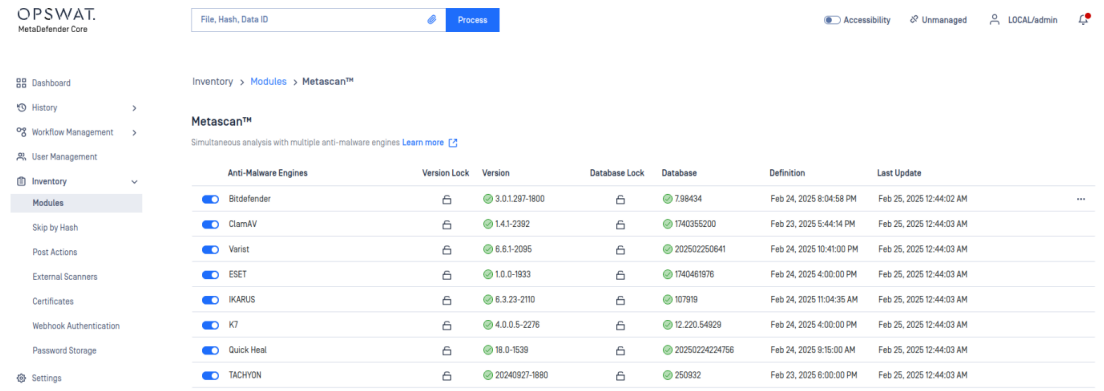
3. After hitting the button, MetaDefender Core will automatically find a suitable engine and download it to the Metascan tab.



Step 2

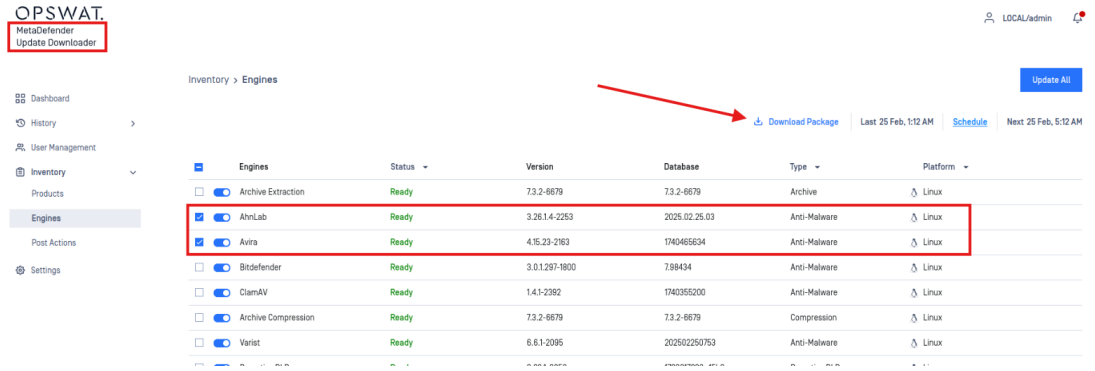
Offline download

1. For the offline download, we will choose the Antivirus engine that is the same as the online downloader, which is **AhnLab** and **Avira**.

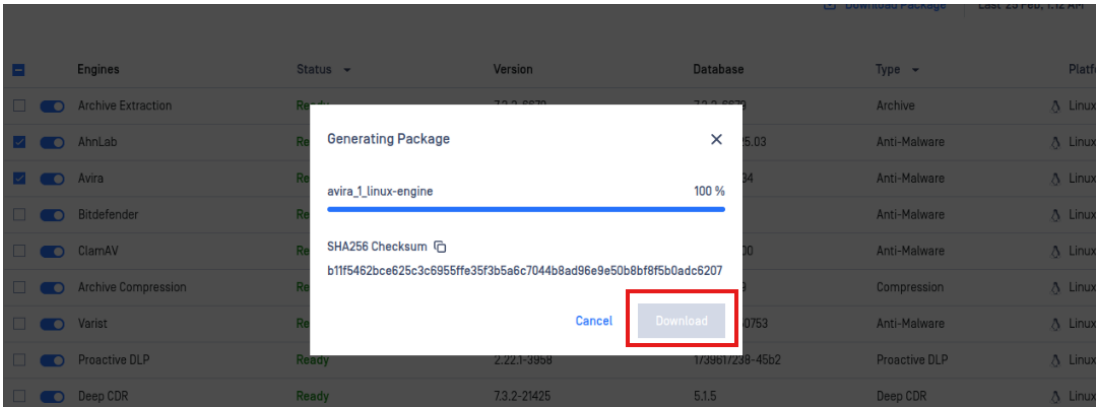


Ahn Lab and Avira missing

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? - MetaDefender Core

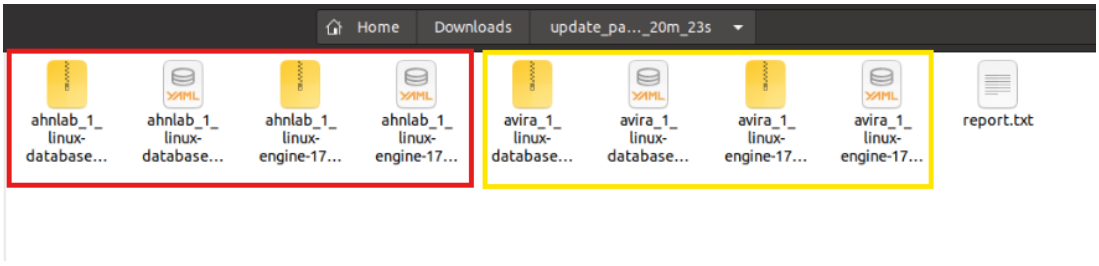


3. From this step, we choose the option that are needed from the tab and click on “Download Package” to download the zip that contains all the package we need for the Antivirus engines.



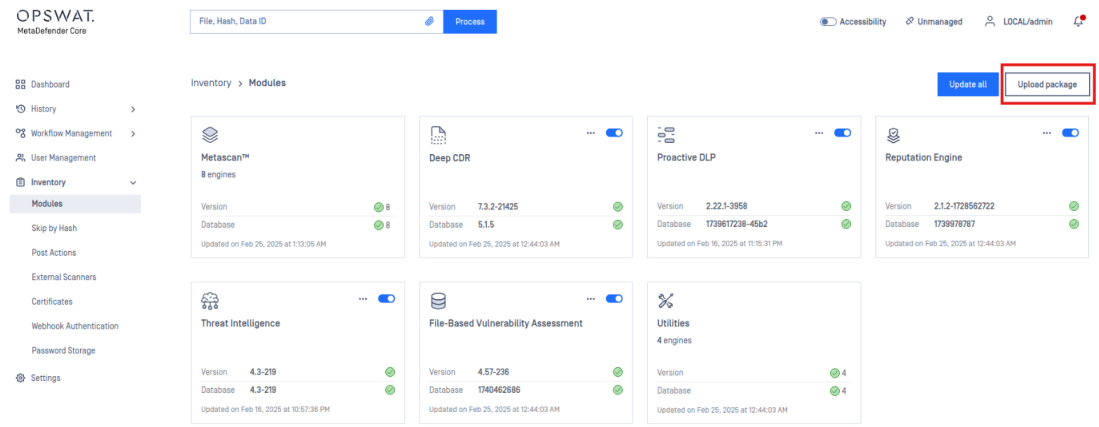
Download the package zip

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.

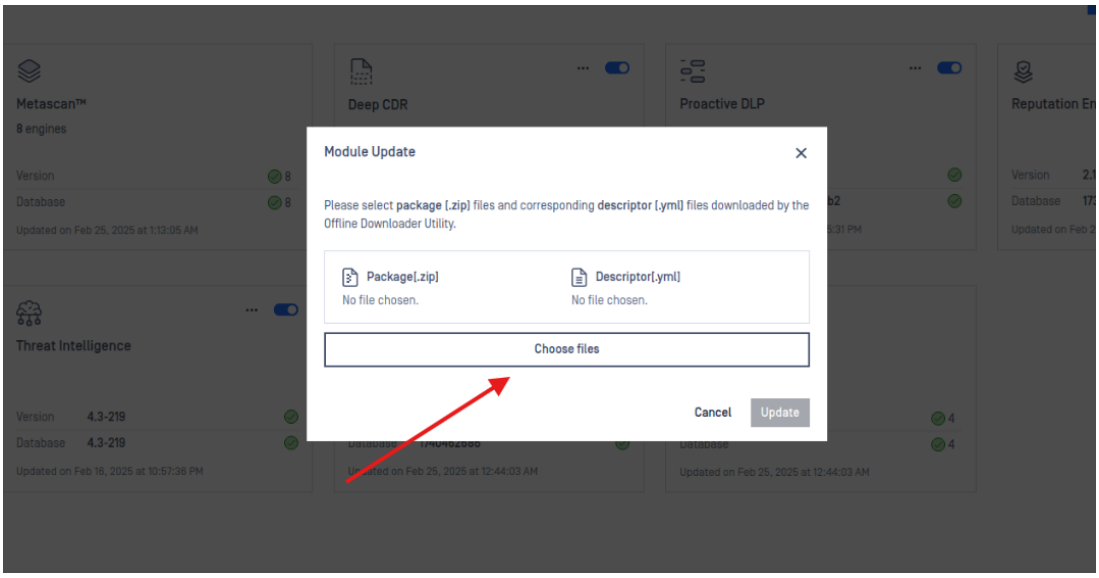


AhnLab and Avira packages

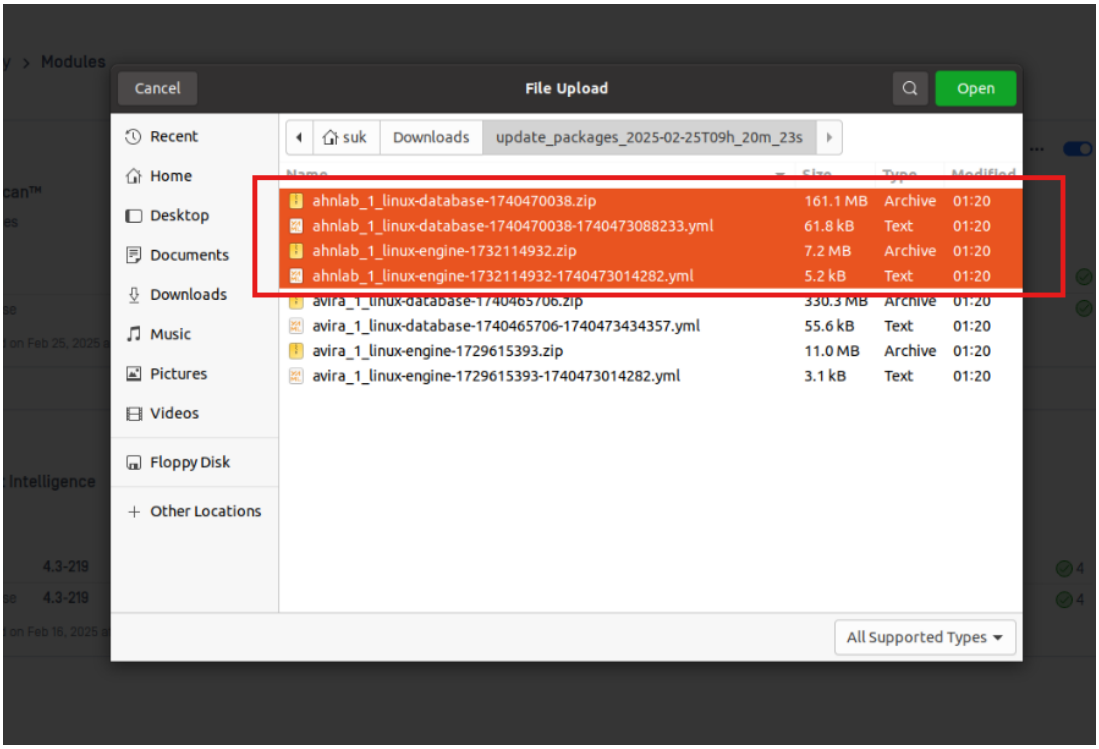
5. Navigate back to the MetaDefender Core Module tab, click on the “Upload Packages”. And manually upload 1 engine at a time as follows from the image below.


OPSWAT.
MetaDefender Core
File, Hash, Data ID
Process
Accessibility
Unmanaged
LOCAL/admin
Dashboard
History
Workflow Management
User Management
Inventory
Modules
Skip by Hash
Post Actions
External Scanners
Certificates
Webhook Authentication
Password Storage
Settings
Inventory › Modules
Update all
Upload package
Metascan™
8 engines
Deep CDR
Proactive DLP
Reputation Engine
Threat Intelligence
File-Based Vulnerability Assessment
Utilities
4 engines

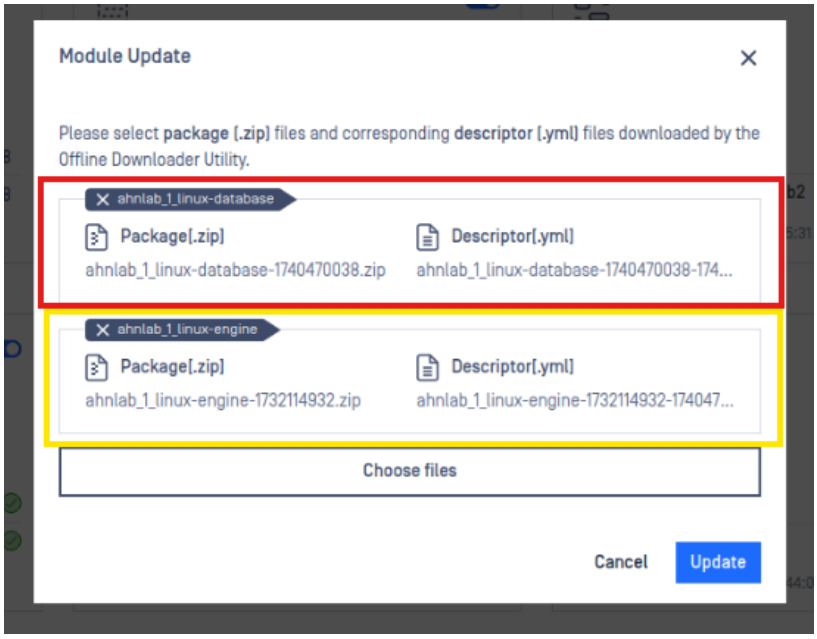
Step 1


Module Update
Please select package [.zip] files and corresponding descriptor [.yml] files downloaded by the Offline Downloader Utility.
Package[.zip]
No file chosen.
Descriptor[.yml]
No file chosen.
Choose files
Cancel
Update

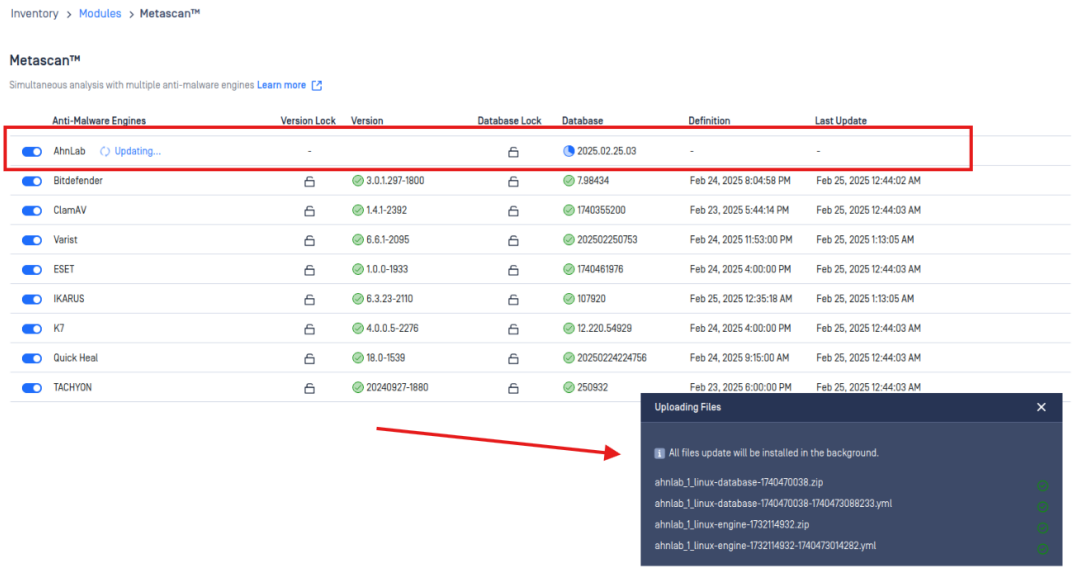
Step 2


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

Step 3


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

Step 4



Step 5

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**.

Are MetaDefender Core v5 upgrades free?

Check Your Version:
This article applies to all active-license MetaDefender Core releases deployed on Windows or Linux systems.

Yes.

The latest version of MetaDefender Core is free for all license holders to download and run.

In fact, OPSWAT recommends that you upgrade to the latest release as soon as possible, so that you can benefit from, state-of-the-art features, necessary bug fixes and ongoing support (Read This).

If you are interested in upgrading, please:

1. Browse our Release Notes to view current and prior versions, comparisons and notes.
2. Visit the My OPSWAT to download the latest version of MetaDefender Core.
3. Finally, check our Upgrade Guide and follow the links to install.

Support:
If you are a **MetaDefender Core Custom** customer, OPSWAT highly recommends that you contact OPSWAT Support for expert guidance through the upgrade process. Creating a support case or chatting with support engineer today to get in touch with the OPSWAT team.

Are MetaDefender Core version upgrades automatic?

Check Your Version:
This article applies to all MetaDefender Core V5 releases deployed on Windows or Linux systems.

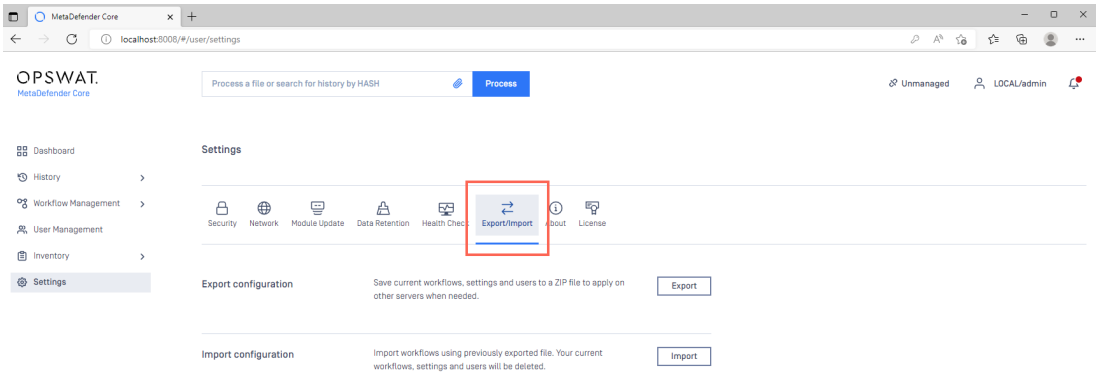
No.

New versions of the MetaDefender Core system must be manually installed. Manual installation allows users to follow prompts and engage the necessary customizations along the way.

V5 releases allow for quick manual installation of new versions directly over existing installations.

As a best practice policy, we urge clients to perform a disk file snapshot of their machine prior to upgrade (for systems deployed on virtual machines), or make use of our built-in **Config Export/Import** feature (for all systems, virtual or physical).

The **Config Export/Import** feature, as illustrated in the screenshot below, allows users to preserve their configurations through the upgrade process . Read This learn more.



Please Note:
If you are upgrading from a V4 or V5 release, visit Product Upgrading and follow the links to install your new version.

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 available for all license holders to download.

Updates of specific features within the MetaDefender Core system (such as scan engines) can be configured to either **Internet** (**Automatic**), **Local** (**Automatic**) or **Manual**.

Visit our Release Note section to learn more about feature updates and bug fixes.

Support:
For further guidance through the upgrade process, please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

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.

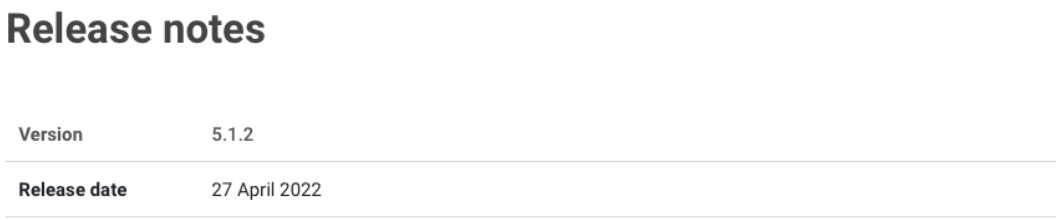
Please scroll down to find the instructions applicable to your OS and mode.

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.

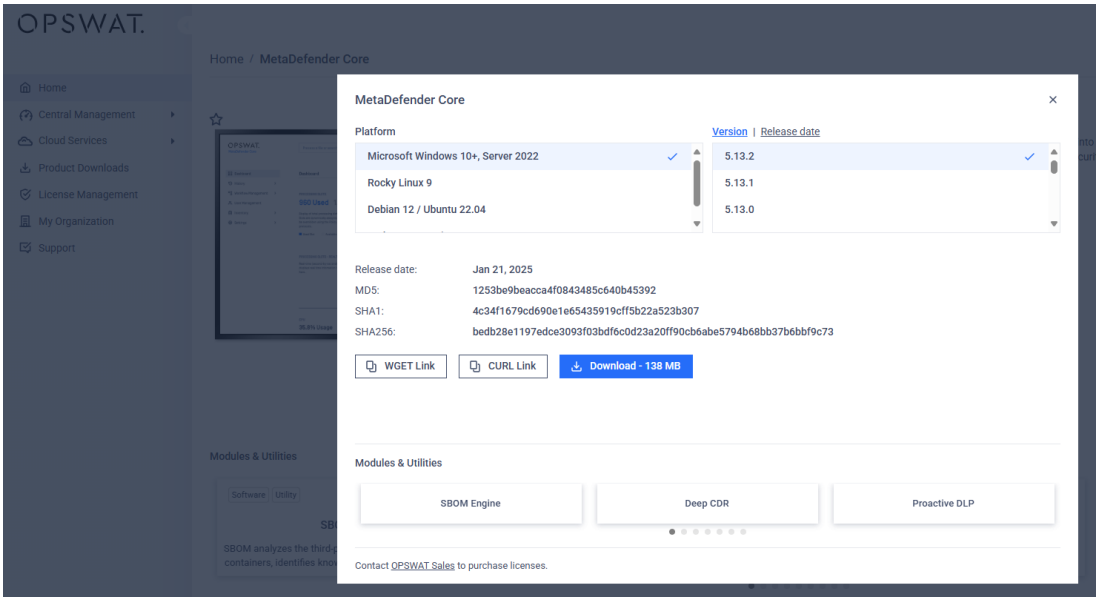


Release notes

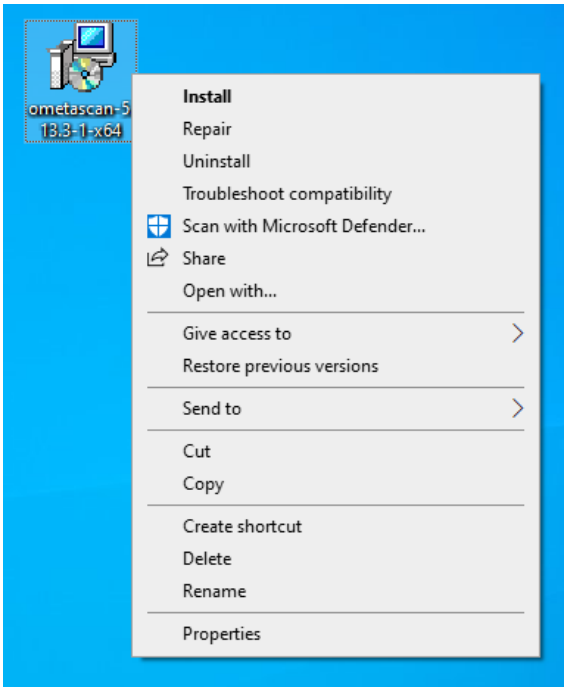
Version	5.1.2
Release date	27 April 2022

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.

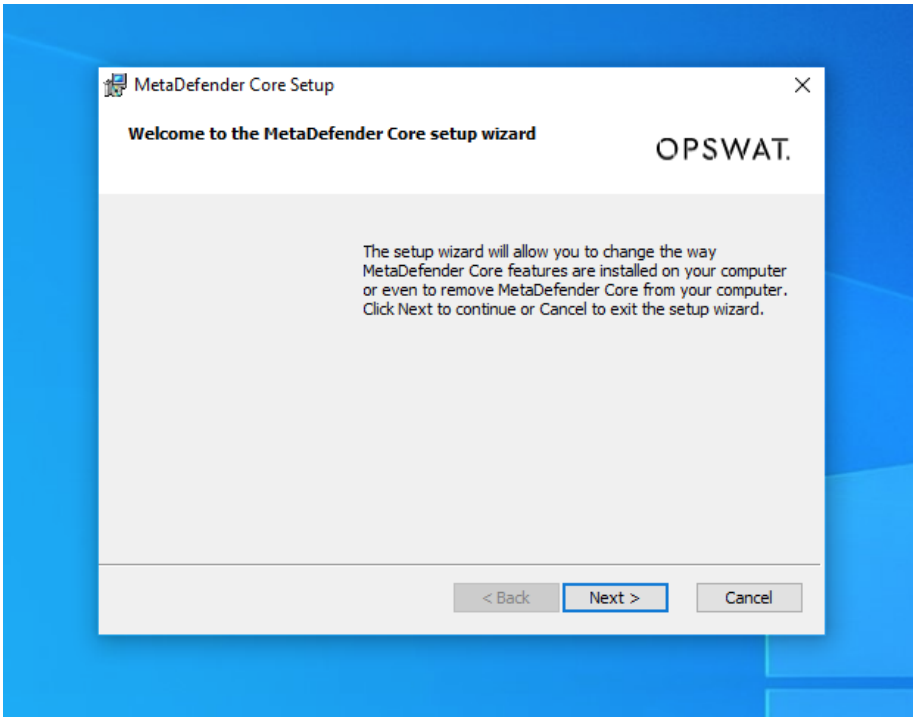
2. Visit My OPSWAT to download the latest MetaDefender Core version for Windows.



3. Log in as Admin user on the MetaDefender Core server, then **Run** the installer file.

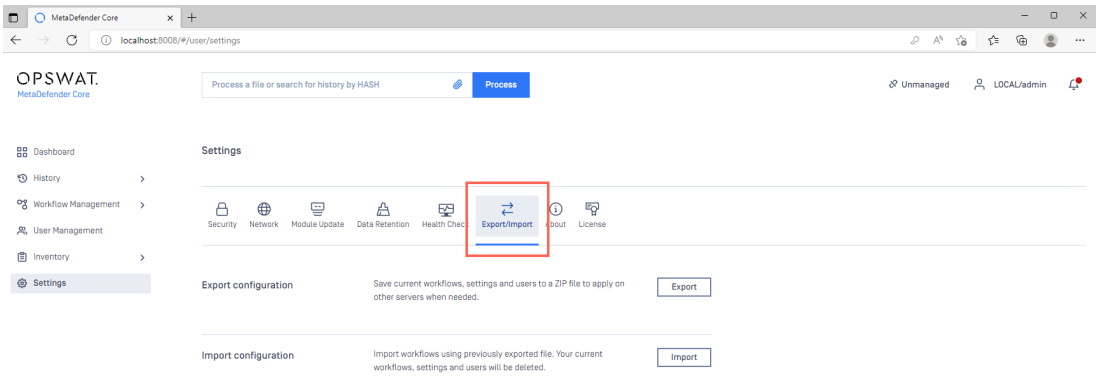


4. Let the **MetaDefender Core Setup Wizard** guide you through the process.



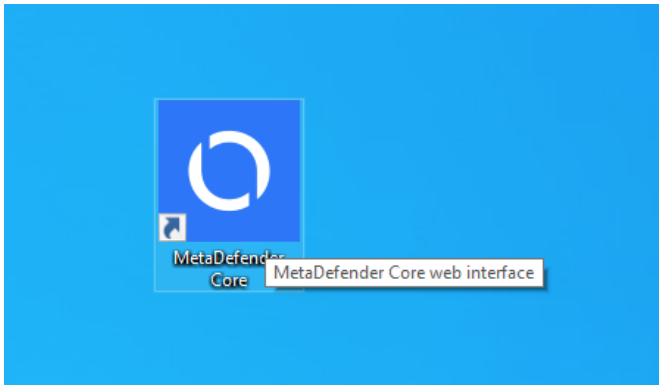
5. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more.

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.

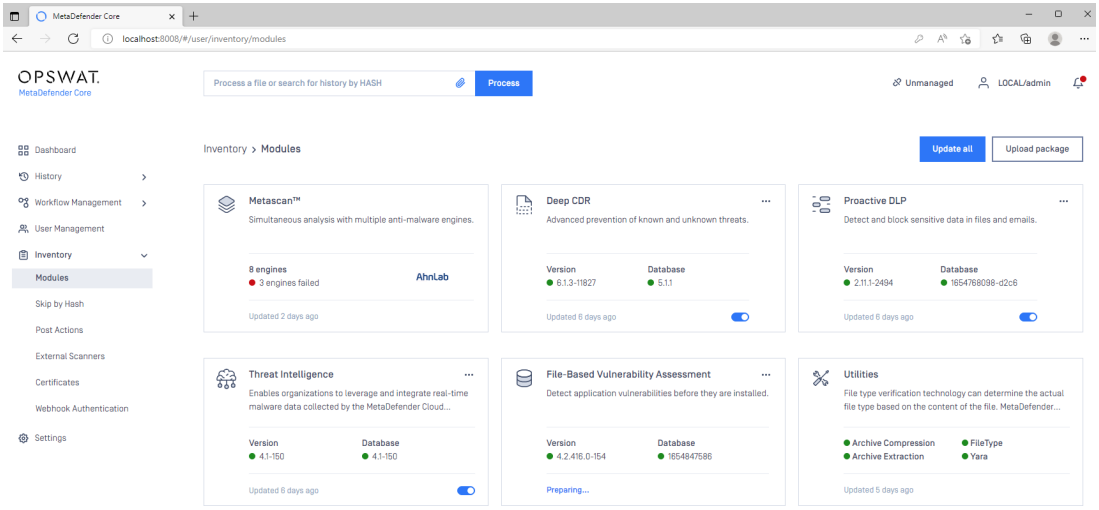


Please Note:
Once the installer has finished, your system will be upgraded to the latest version.

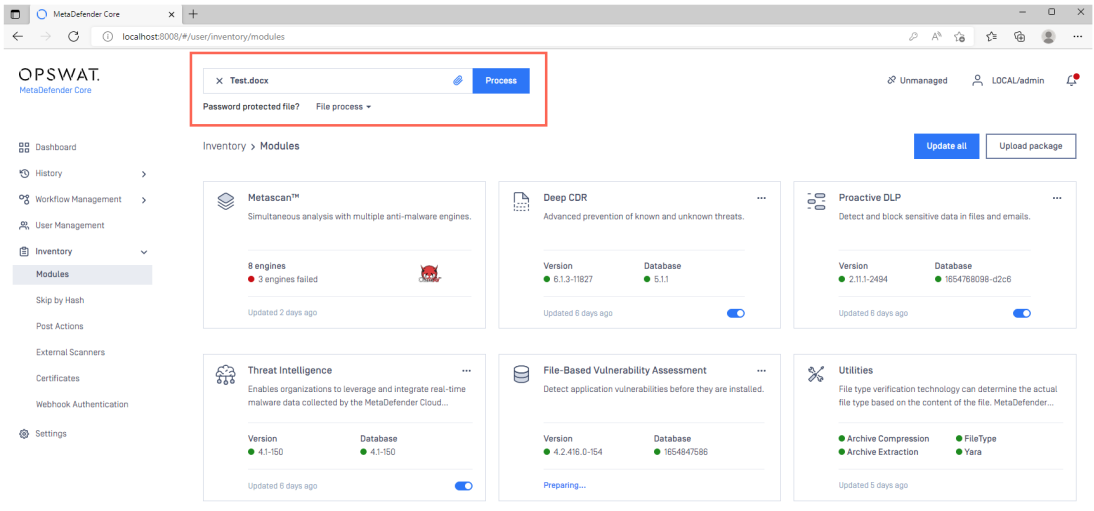
6. Once installed, click the MetaDefender Core icon to log into the **Management Console** and verify that the system is functioning as expected.



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 none

```
[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.

none

```
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.
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 none

```
[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.

none

```
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.
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:

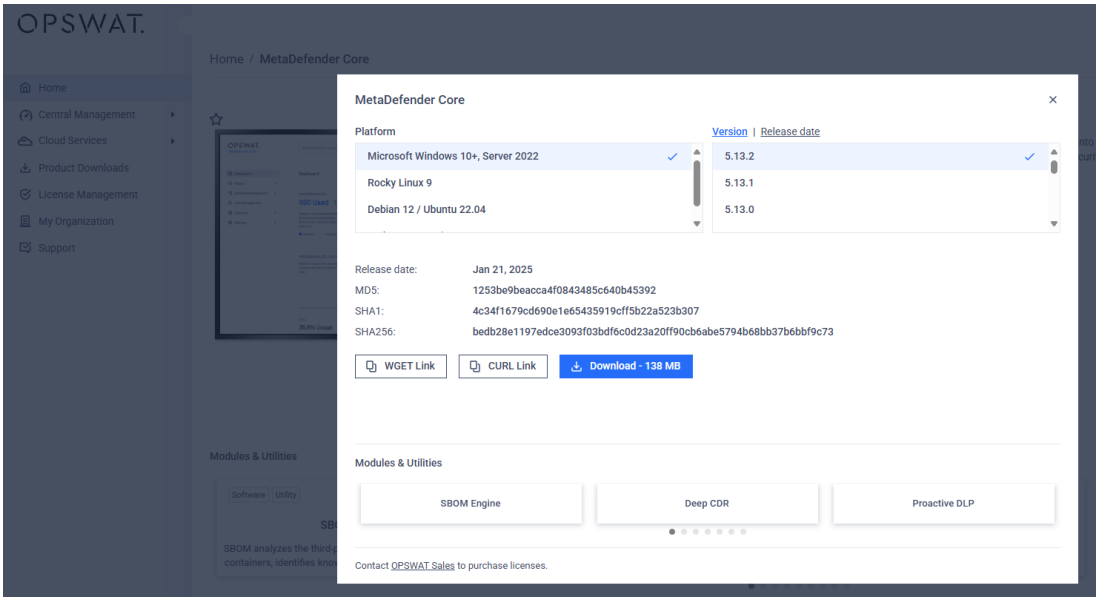
1. Browse Release Notes for details on the latest MetaDefender Core version release.

Release notes

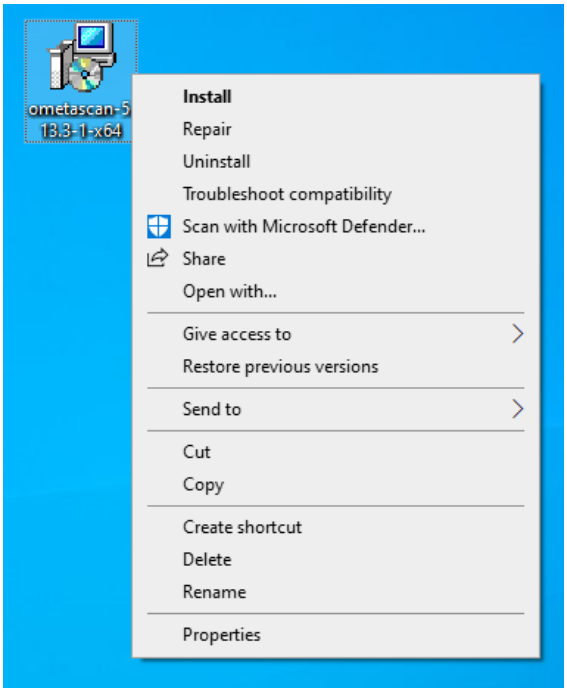
Version	5.1.2
Release date	27 April 2022

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.

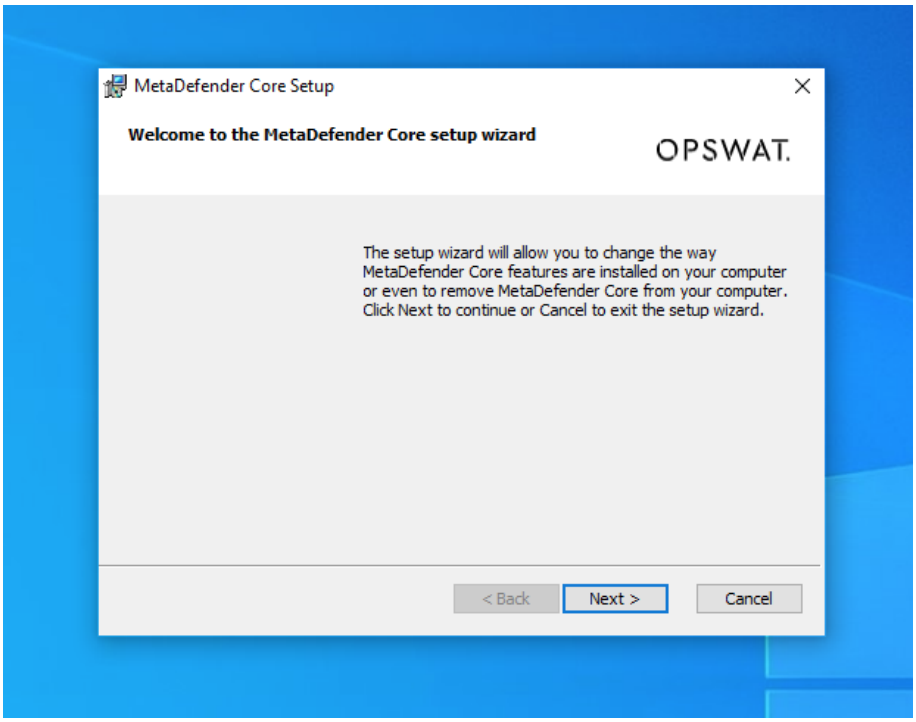
2. Visit My OPSWAT to download the latest MetaDefender Core version for Windows.



3. Log in as Admin user on the MetaDefender Core server, then **Run** the installer file.

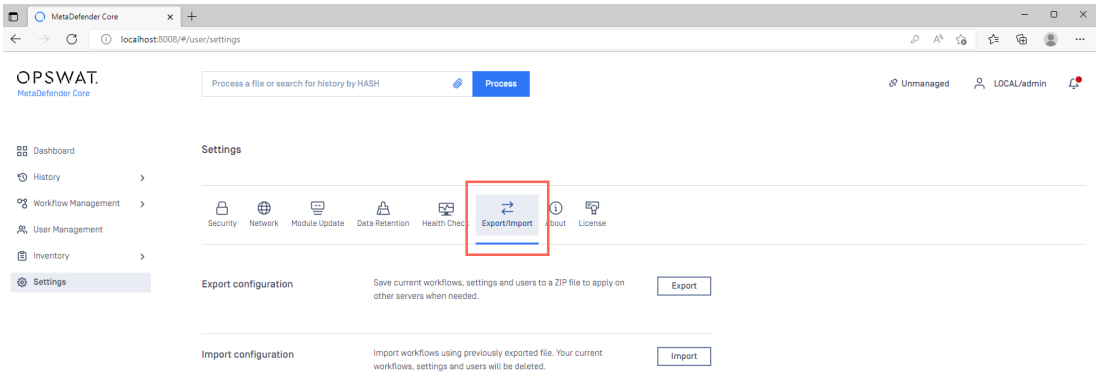


4. Let the **MetaDefender Core Setup Wizard** guide you through the process.

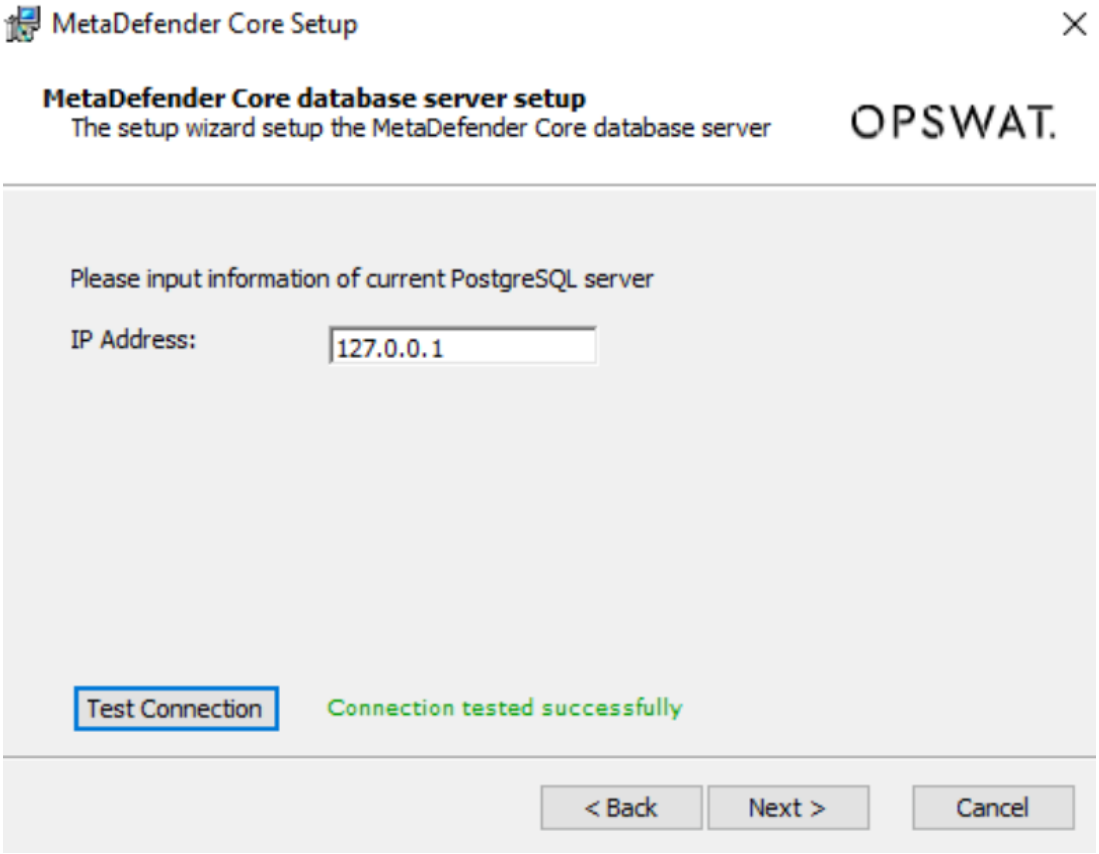


5. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more.

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.

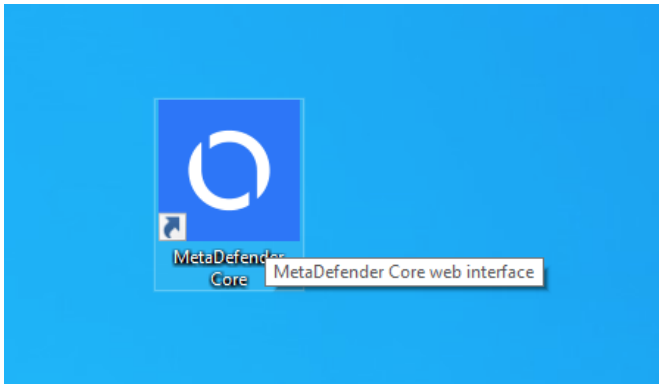


6. For **Shared Database** mode, the same upgrade instructions apply as described above for **Standalone** mode, with one exception: at the Setup stage illustrated in the screenshot below, you will need to specify your current PostgreSQL server IP address.

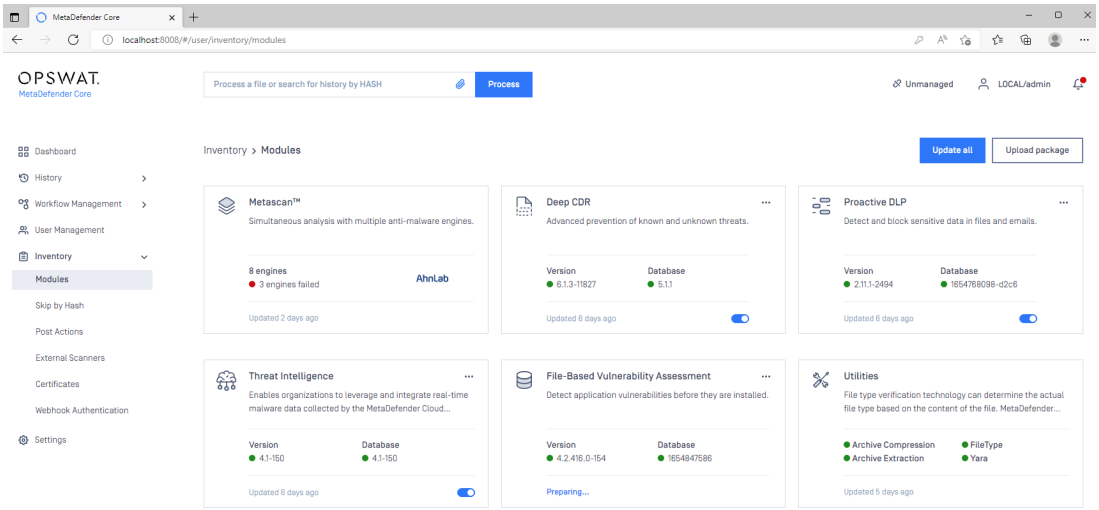


Please Note:
Once the installer has finished, your system will be upgraded to the latest version.

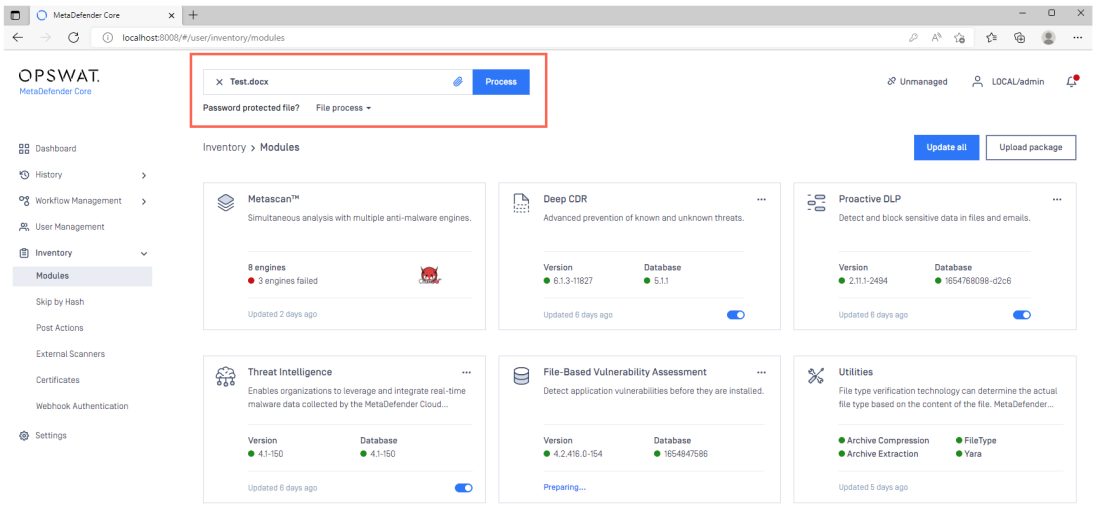
7. Once installed, click the MetaDefender Core icon to log into the **Management Console** and verify that the system is functioning as expected.



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 none

```
[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.

none

```
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.
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 none

```
[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.

none

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

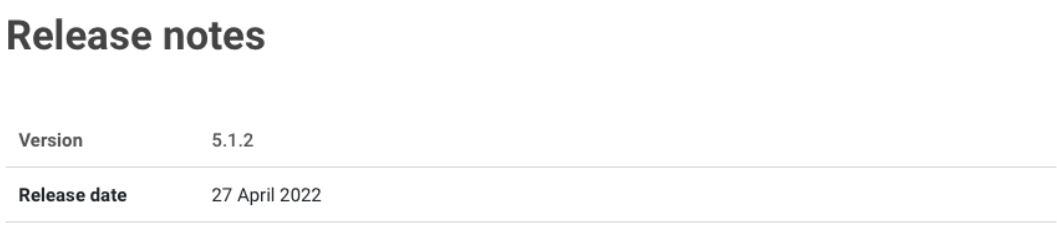
Where the <option keys>, their default values and their descriptions are as follows:

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.

Linux (Standalone mode)

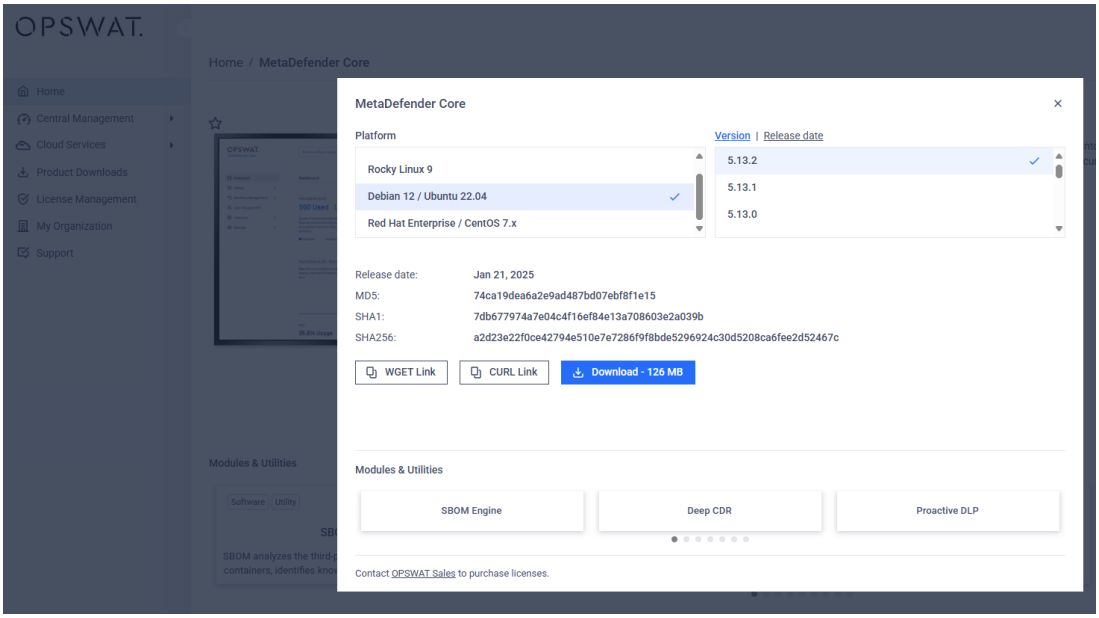
To perform a **MetaDefender Core Deployment Upgrade**, please follow the steps below:

1. Browse Release Notes for details on the latest MetaDefender Core version release.

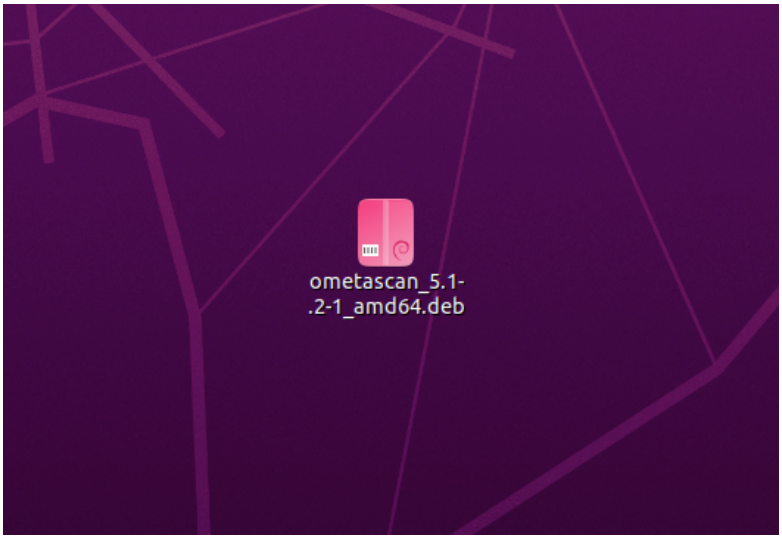


Pro Tip:
OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

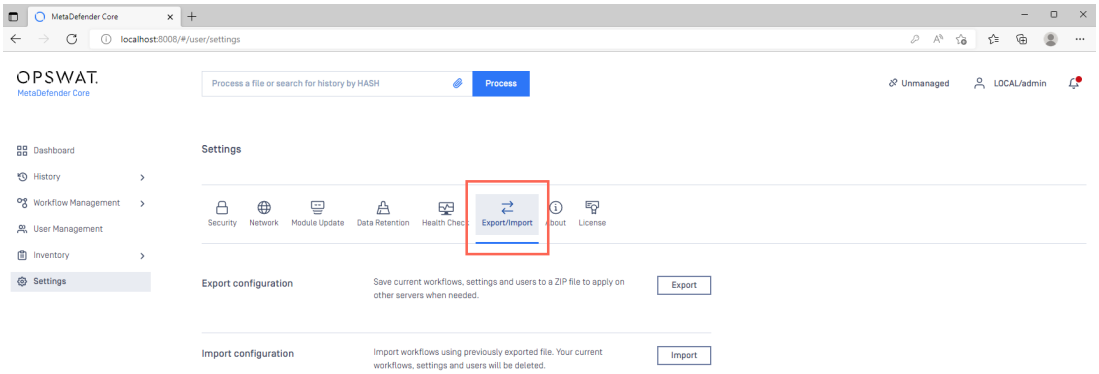
2. Visit My OPSWAT to download the latest MetaDefender Core version for your Linux distro.



3. Log in as **Root** user on the MetaDefender Core server, then **Run** the installer file.

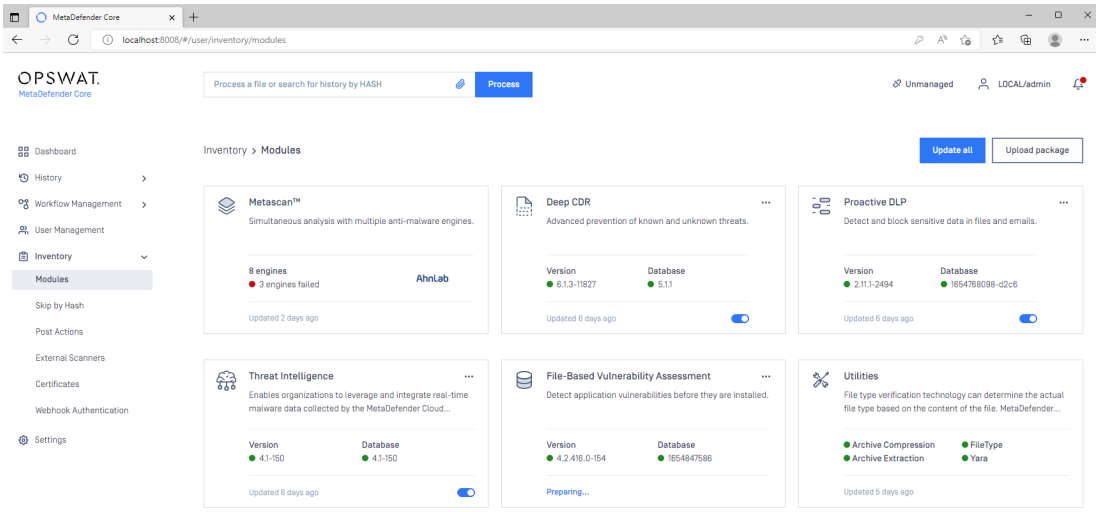


4. Depending on your distro, **Software Install** or a similar utility will guide you through the process.
5. If you are upgrading from a prior release, ensure you use the **Config Export/Import** feature, as illustrated below Read This to learn more. With the help of this feature, and provided no errors occur during the upgrade process, your configurations and licensing information will be fully preserved through the upgrade process.

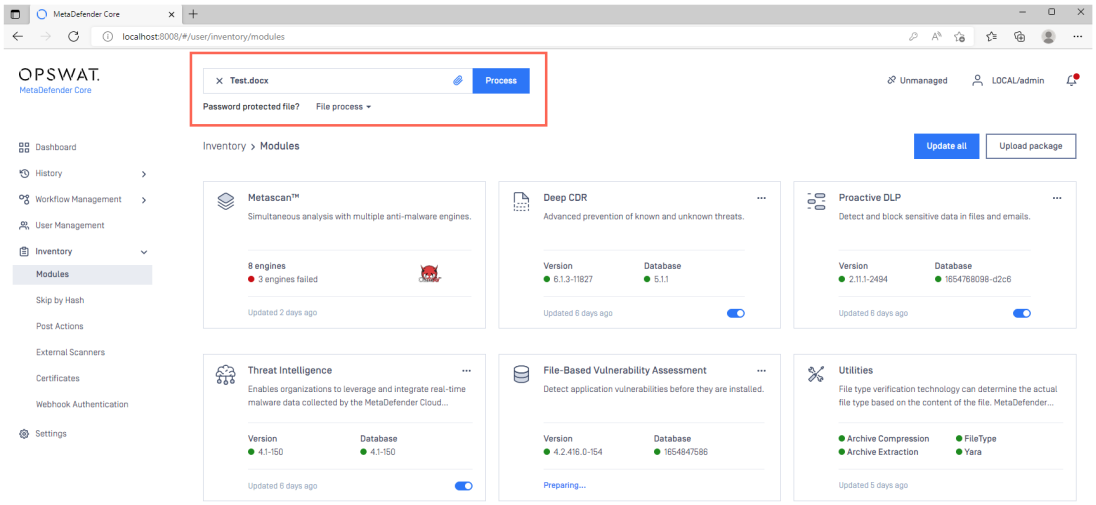


Please Note:
Once the installer has finished, your system will be upgraded to the latest version.

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 none

```
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 none

```
[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 none

```
[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.

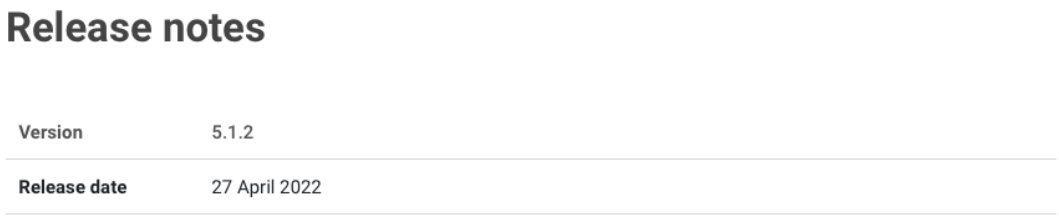
Debian package (.deb)	*sudo dpkg -i <filename> || sudo apt-get install -f*
On Red Hat Enterprise 7.x Linux / CentOS package (.rpm)	*sudo yum install <filename>*
For systems which enabled GPD check flag	*sudo yum install –nogpgcheck <filename>*
Red Hat Enterprise Linux 8.x package (.rpm)	*sudo dnf install -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)

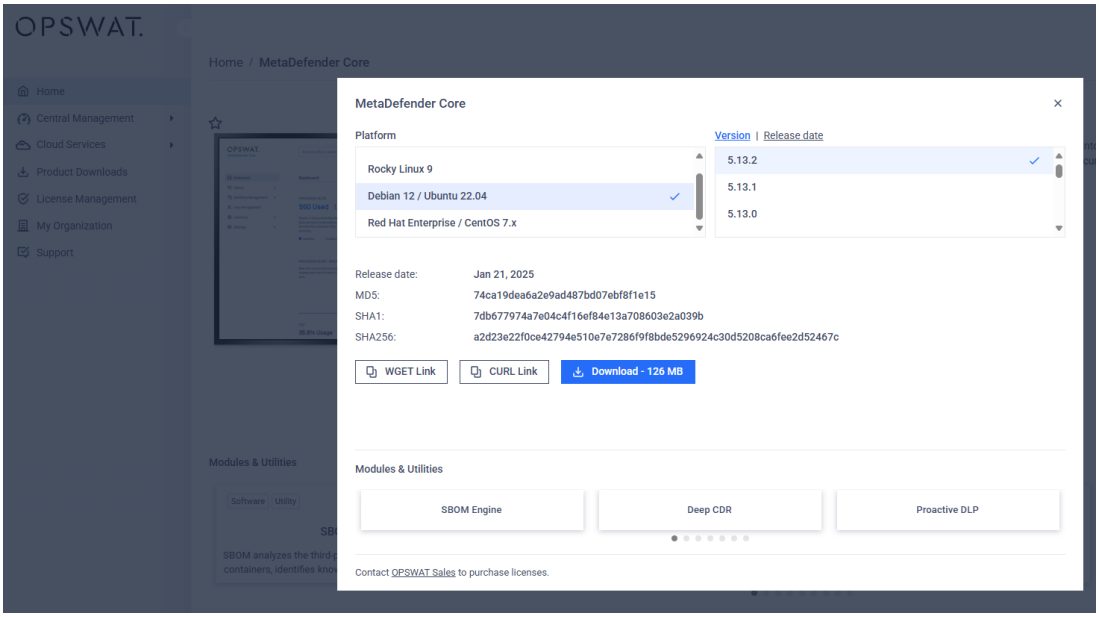
To perform a **MetaDefender Core Deployment Upgrade**, please follow the steps below:

1. Browse Release Notes for details on the latest MetaDefender Core version release.

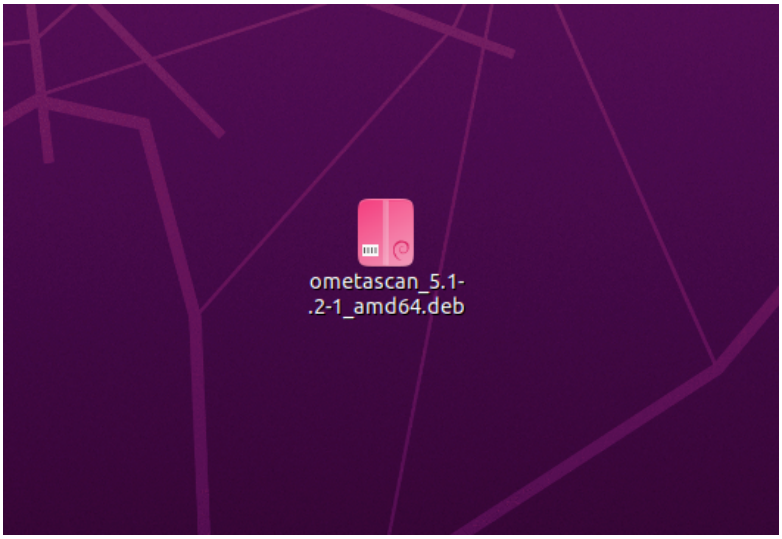


Pro Tip:
OPSWAT highly recommends that users upgrade to the latest version of MetaDefender Core. Current versions incorporate state-of-the-art features and necessary bug fixes, and are freely available for all license holders to download.

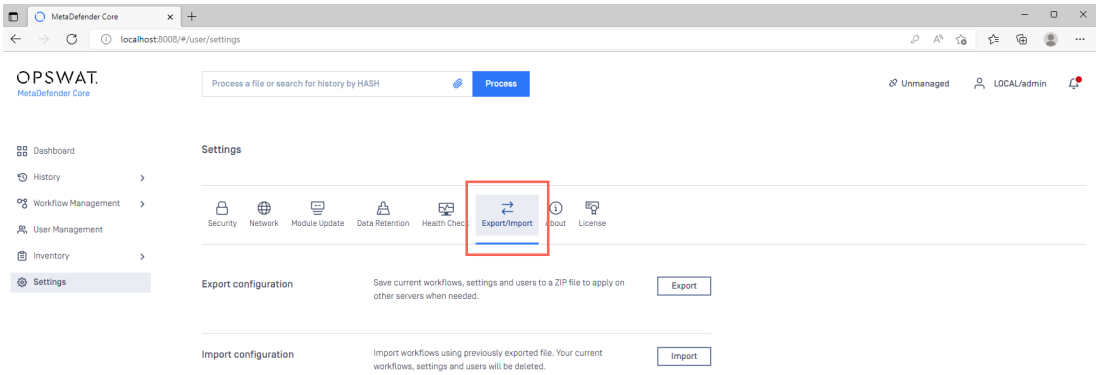
2. Visit My OPSWAT to download the latest MetaDefender Core version for your Linux distro.



3. Log in as **Root** user on the MetaDefender Core server, then **Run** the installer file.

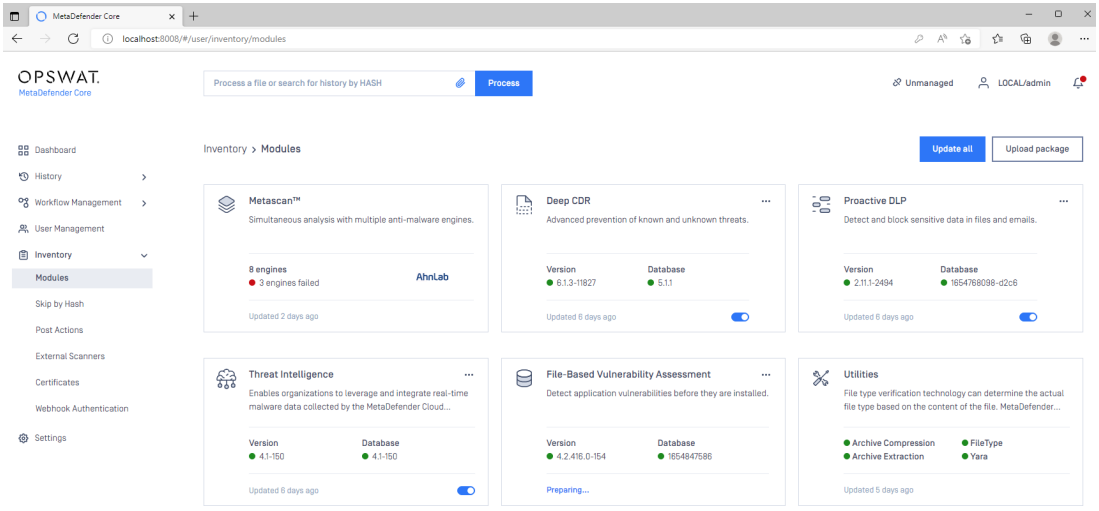


4. Depending on your distro, **Software Install** or a similar utility will guide you through the process.
5. For Shared Database mode, the same upgrade instructions apply as described above for Standalone mode, with one exception: during setup, you will need to specify your current PostgreSQL server 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.

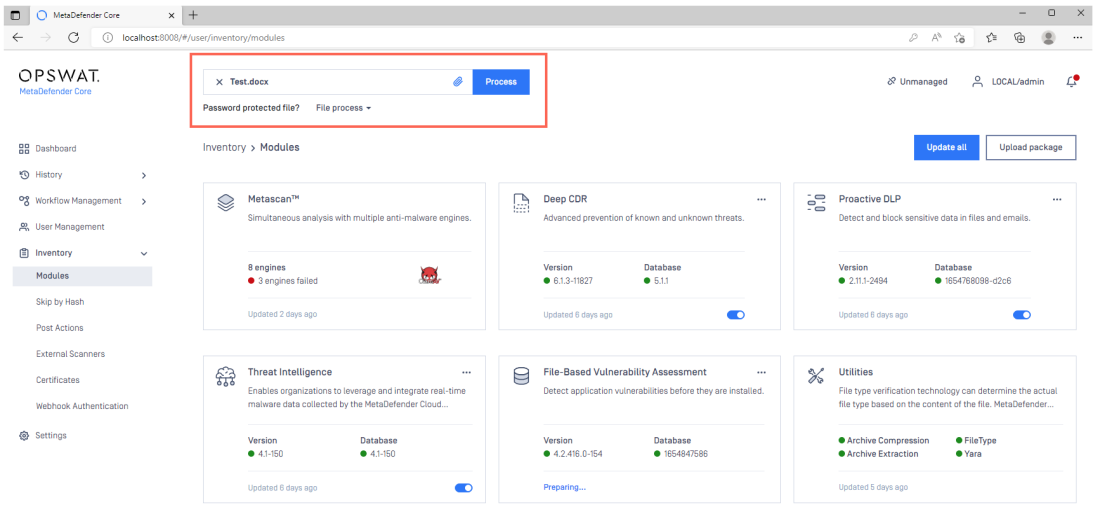


Please Note:
Once the installer has finished, your system will be upgraded to the latest version.

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 none

```
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 none

```
[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 none

```
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 auto$, before creating a support case or chatting with our support engineer.

What is 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.

Browse MetaDefender Core Release notes for details on the latest version, version history, version comparisons and release notes.

Alternatively, go directly through the My OPSWAT for a list of available downloads.

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.

Support:
Feel free to contact OPSWAT support for further guidance through your **Upgrade** process. Please follow these instructions on auto$, before creating a support case or chatting with our support engineer.

How to upgrade an End-of-Life [EOL] MetaDefender Core?

Check Your Version:
This article applies to all MetaDefender Core releases deployed on Windows and Linux systems.

Please review our EOL policy:

Please find our End-of-Life policy here: https://www.opswat.com/docs/mdcore/knowledge-base/how-long-is-the-support-life-cycle-for-a-specific-version-releas

System Requirements:
Please review the system requirements and ensure to meet them before upgrading

Good Practices:

- Export configurations under MetaDefender Core > Settings
- Take a snapshot of the machine
- Follow installation guide for your deployment model. Ex. Standalone Mode, Shared Database Mode

Version 4.18 and older:

These versions of MetaDefender Core utilize the SQLite Database, whereas the newer version of MetaDefender Core utilizes the Postgres Database. We currently do not have a migration plan, therefore we require a complete uninstall and fresh install of MetaDefender Core.

*
Before upgrading, please take screenshots and/or notes of the current settings, user management, and workflows as they will be lost in the uninstall

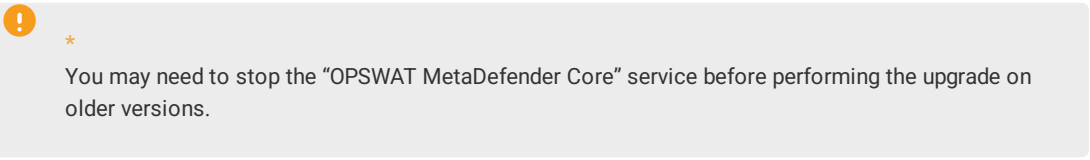
Version 4.19 and newer:

Upgrade Options

If you are jumping from an older version to a much newer version, we suggest the following two options:

Option 1: Upgrade Gradually

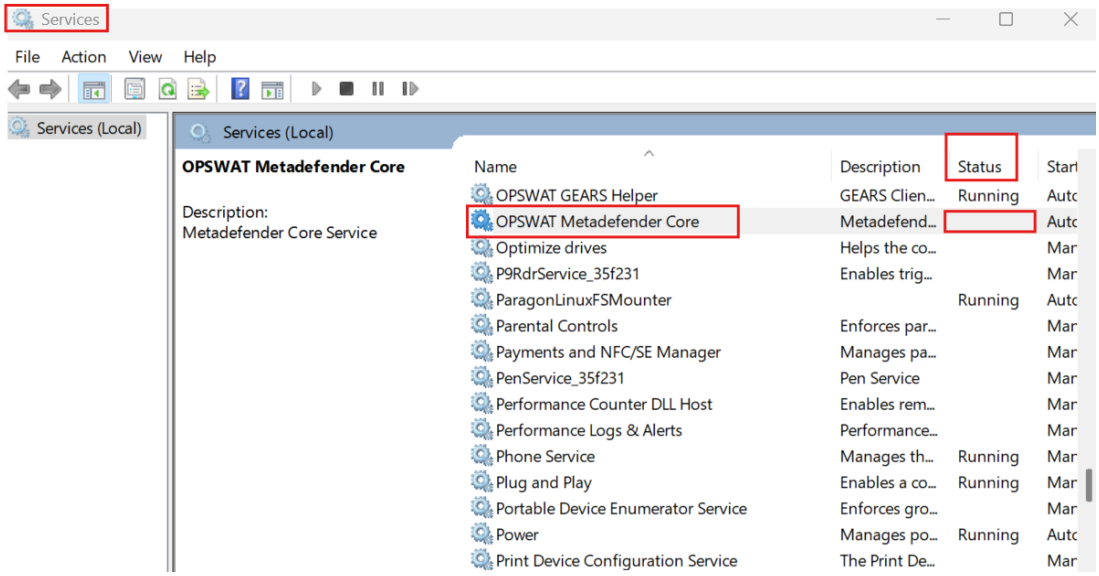
Upgrade Gradually, every other main version until you reach a version that is no longer EOL. For example: 4.19 > 5.0 > 5.2 > 5.4 > 5.6 > 5.8



*

You may need to stop the “OPSWAT MetaDefender Core” service before performing the upgrade on older versions.

Windows:



Linux:

Run command in terminal

```
sudo systemctl stop ometascan
```

After upgrading, please reboot the machine for the necessary changes to take effect.

Once on a supported version, you can upgrade to the latest version available.



*

If on version 5.11.1, upgrade directly to 5.14 or newer.

Disclaimer:
Some versions are only compatible with certain Postgres versions. Please check the release notes and system requirements of the desired version.

Option 2: **Fresh Install**

Export configuration settings in settings or take screenshots and/or notes of the configuration in workflows, settings, user management. Export processing data for your records [History > Processing > Export History]. Uninstall the current version and fresh install the latest version. Import configuration settings or manually configure the configuration based on notes/screenshots.

Support:
If **Further Assistance** is required, please proceed to log a support case or chatting with our support engineer.

Can I upgrade MetaDefender Core from an very old version such as version 4.x.x to the latest version 5.x.x?

TLDR: Yes, we can basically run the latest installer to upgrade the MetaDefender Core from any older version to the latest. **However**, if the gap between the 2 versions is too far, it's not recommended, and you should perform a clean/fresh installation.

Recommended Path:

If your Core release is more than *[time period]* old, In the vast majority of cases it will be a cleaner, easier, more reliable path to simply install the latest version of Core, including a clean database install.

The downside to this is that you will lose historical processing data. And you will need to rebuild your settings and workflows from scratch.

Specific recommendations documented below.

If you absolutely MUST keep your historical data

If you have absolutely no leeway about keeping your historical data, you can follow our incremental upgrade path below.

Caveats:

- The incremental upgrade process is very time intensive, and prone to unforeseen interactions between different Core versions, e.g. database structure mismatches, incompatible settings export files, etc.
- The process is not a science, but an art, that comes down to reading Release Notes at each stage to determine which Core version is the next safe landing point.
- Especially if the existing Core instance is very old, there is no guarantee that this path will be free of database errors. Meaning, you might end up with a corrupted database after several hours of effort, and need to drop back to a clean install anyway.

With all the above in mind, again this approach may turn out to require far more effort and time than the clean install approach, and with no guarantee of a clean result in the end.

Incremental upgrade path:

- Have a backup version (snapshot, recovery point,...)
- Divide the version gap into multiple stages to minimize the gap.
- Choosing the most stable versions to upgrade to. (Please review the Release note).
- Confirm that you have the proper Recommended System Configuration.

You can **attempt** to upgrade to MD Core version 5.6.1. However, the 5.X.X versions of Core come with a new Database. Since the current version is very old it may not be a smooth transition. After upgrading to 5.6.1 you can do incremental upgrades until you reach the latest version 5.13.2

About the Installer:

The latest installer is designed to work with both the old and new versions of the software. If you're currently using an older version, such as 4.16.3, it can detect the old database format (e.g., SQLite) and automatically convert it to the new format (e.g., PostgreSQL). It will also upgrade the database step-by-step through each version update. For example, if your current database is version v53 (from version 5.13.1) and you're upgrading from v1 (from version 4.19.0), the installer will migrate the database to each version from 1-53. This process can take a long time and might lead to issues during migration.

That is why it's not recommended to do the direct upgrade if the version gap is too far.

***Please note the following:**

- We do not have a formal recommended incremental upgrade path. So if you do decide to do incremental upgrade there is a good chance that you will run into a database error. In this situation you will most likely have to perform a clean/fresh installation.
- It is suggested that you take a Snapshot of your server before performing these upgrades.
- If you are on an older version of MD Core, such as v4.16.3 and upgrading to the latest version, you can attempt to export/import the configuration. However, you may not be able to successfully import the configuration. This is due to all the fixes and feature updates that have been made on latest version of MD Core.

Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

Can I upgrade MetaDefender Core from a very old version directly to the latest available?

Before upgrading to the latest version of MetaDefender Core, please make sure that all the System Requirements and the prerequisites of the newer version are met.

It is very important to consider that very old versions of MetaDefender Core - especially those that are no longer supported (see MetaDefender Core v5 Support Lifecycles) - may not be able to upgrade directly to the latest available version. This limitation can arise from a variety of factors, such as changes in database schemas, altered data relationships, third-party dependency updates, and other architectural differences.

In this case, when speaking of a large gap of versions between the current one and the latest, our recommendation is to incrementally upgrade the application, from close to close (for example, 3 or 2 versions at a time), to successfully guarantee a smooth upgrade path.

If running an out-of-support version (MetaDefender Core Support Lifecycle) or for any additional support on this, please do not hesitate to reach out to our Support Team which can assist you with the upgrade process.

Info:
We do always recommend having a back-up solution in place before proceeding with any upgrade. Taking a snapshot of the machine(s) holding the component(s) [application/database] is recommended.

Support:
If Further Assistance is required, please proceed to log a support case or chatting with our support engineer.

What is the impact of upgrading MetaDefender Core from version 5.12.0 to 5.13.3?

Overview This article provides insights into the key changes introduced in MetaDefender Core version 5.13.3, including updates to OpenSSL, NGINX, PostgreSQL, and the Syslog Server Configuration API. It also explains the potential impact on current operations and offers recommendations to ensure a smooth upgrade process.

Affected Versions

- **From**: MetaDefender Core 5.12.0
- **To**: MetaDefender Core 5.13.3

Key Changes and Their Impact

1. OpenSSL and NGINX Security Enhancements Upgrade Details:

- MetaDefender Core's NGINX web server will no longer start if weak cipher suites are configured for HTTPS.
- Starting with MetaDefender Core 5.2.0, OpenSSL 1.x has been replaced with OpenSSL 3.x.
- This upgrade affects key dependencies like PostgreSQL and NGINX, enhancing security and resolving known vulnerabilities.

Impact on Operations:

- If weak ciphers are present in your SSL/TLS configuration, the NGINX service will fail to start post-upgrade.
- To avoid downtime, review and update SSL/TLS configurations to remove any unsupported cipher suites prior to upgrading.

2. Syslog Server Configuration API Enhancements Upgrade Details:

- Starting in MetaDefender Core 5.13.0, API support has been added for configuring Syslog settings.
- Previously, these settings could only be managed via the GUI.
- The new API provides greater control and flexibility for managing log forwarding and system monitoring.

Impact on Operations:

- Existing Syslog configurations remain unaffected.
- However, if changes are made using the new API, the MetaDefender Core service must be restarted for changes to take effect.

Recommendations Before Upgrading

- **Review SSL/TLS Cipher Configuration** Ensure no weak cipher suites are used in your HTTPS settings to prevent NGINX startup failures. Refer to the official release notes for supported ciphers.
- **Backup Current Configurations** Back up all configuration files, SSL certificates, and log settings before performing the upgrade.
- **Test in a Staging Environment** Validate compatibility and stability by deploying the new version in a test environment prior to production rollout.
- **Restart MetaDefender Core Service if API Changes Are Made** Any Syslog configuration updates made through the API require a service restart to take effect.

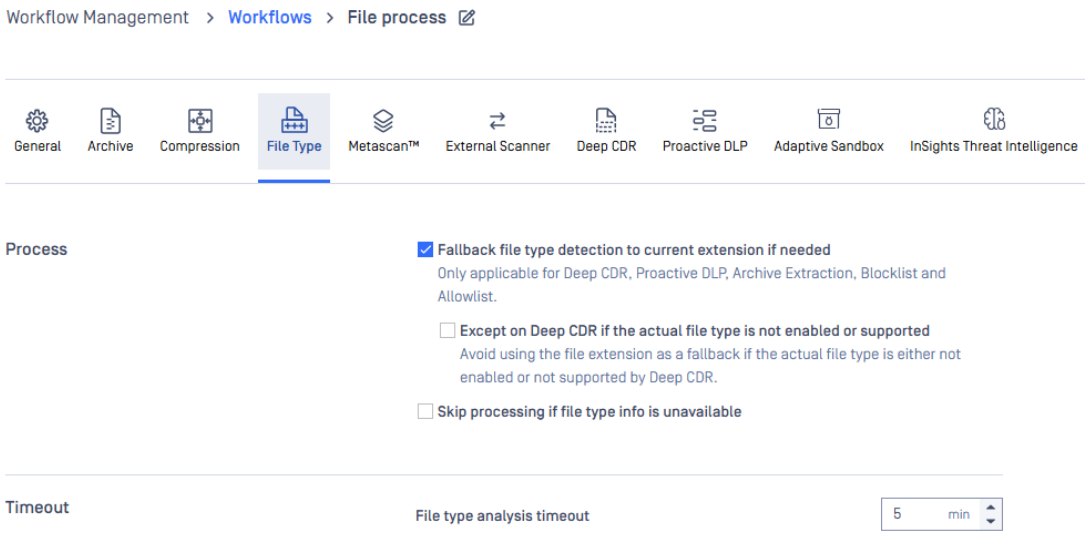
Conclusion MetaDefender Core 5.13.3 brings critical security improvements and greater flexibility for log management. While the upgrade is backward-compatible, reviewing your current configurations and preparing accordingly will help ensure a smooth transition.

Support:
If **Further Assistance** is required, please proceed to log a **support case or chatting with our support engineer**.

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.

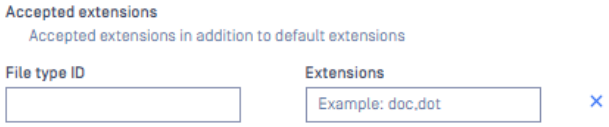
Configuration

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.

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.:

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.

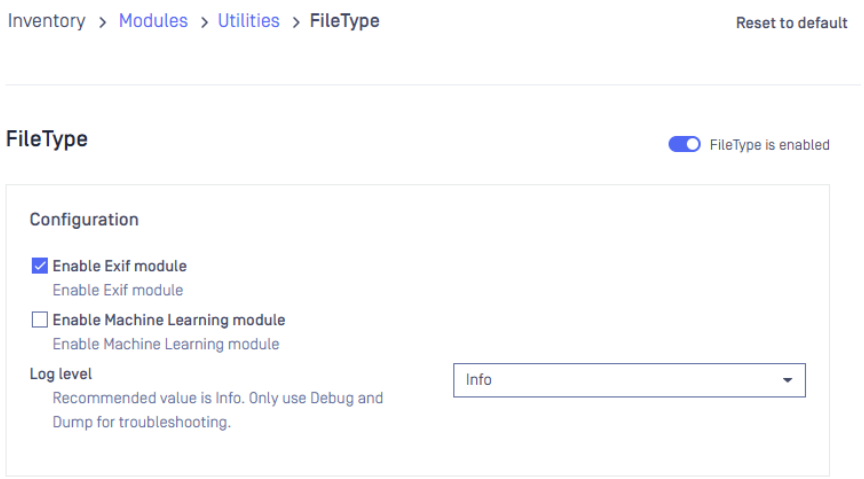
Json javascript

```
{
  "file_type": {
    "detect_encrypted_archive": true,
    "classify_with_ml": false,
    "detect_base64": false
  }
}
```

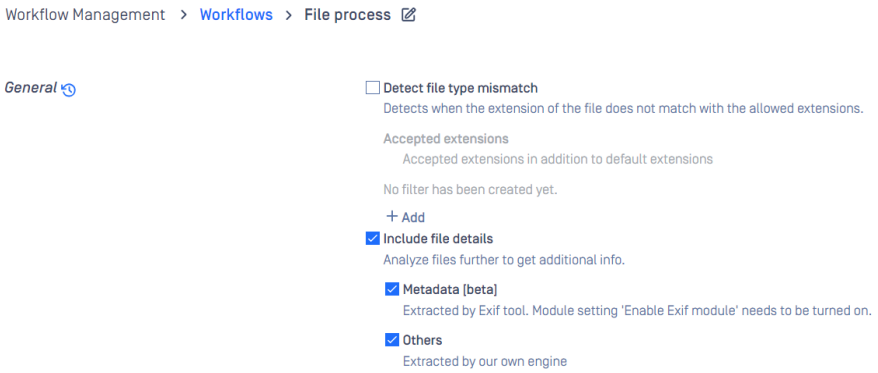
File info details

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**



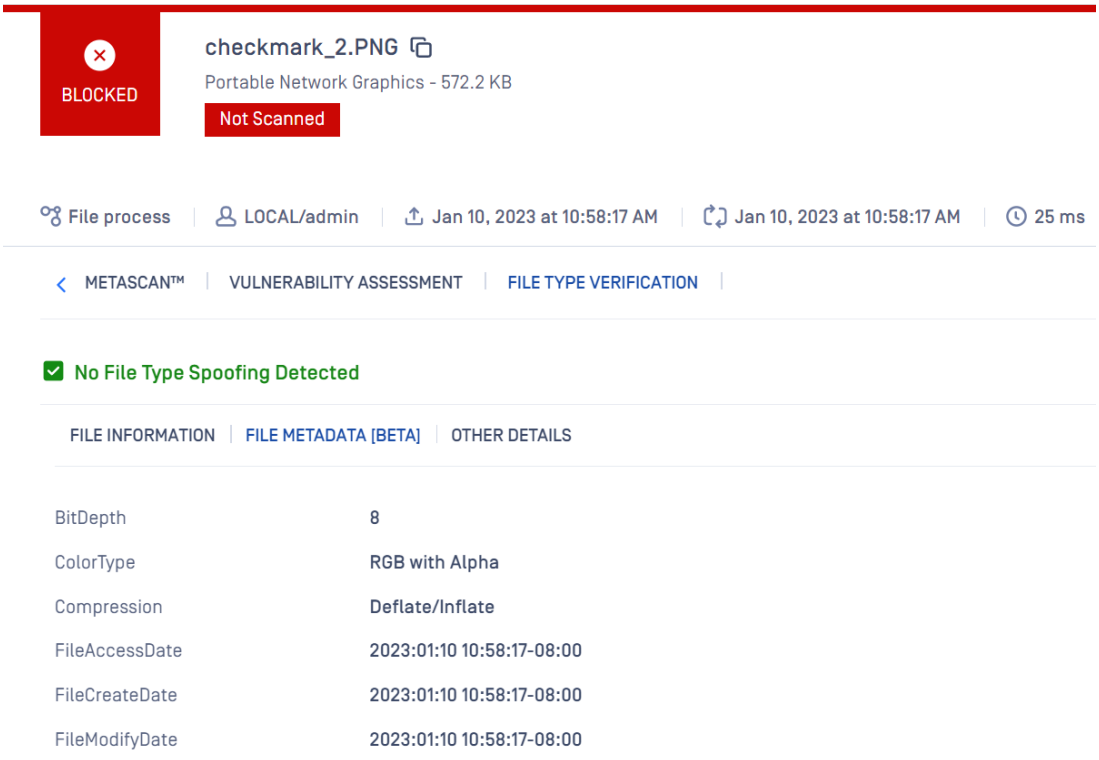
- At **Workflow Management > Workflows > [Workflow name] > File Type > General**, Enable **Include file details > Metadata**



JSON output:

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"
    },
  ..
}
```

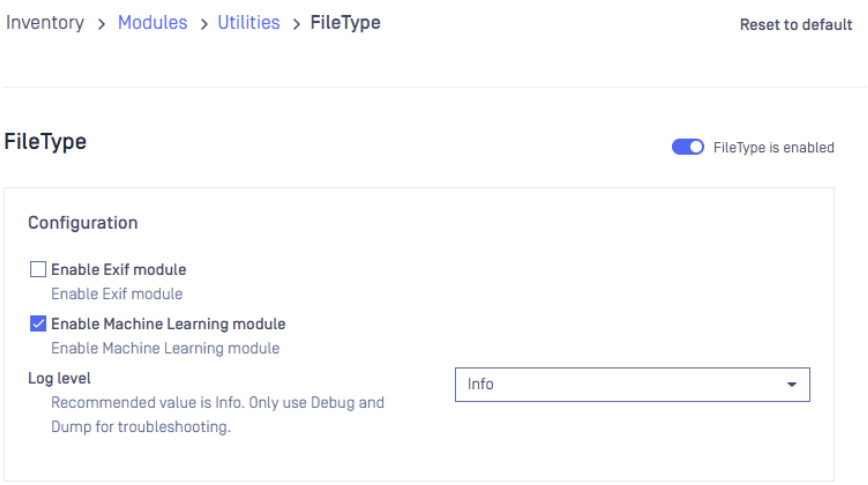
The feature is in BETA, please use the data for information only. The JSON structure and attribute names can be changed without notice

Text detection with AI

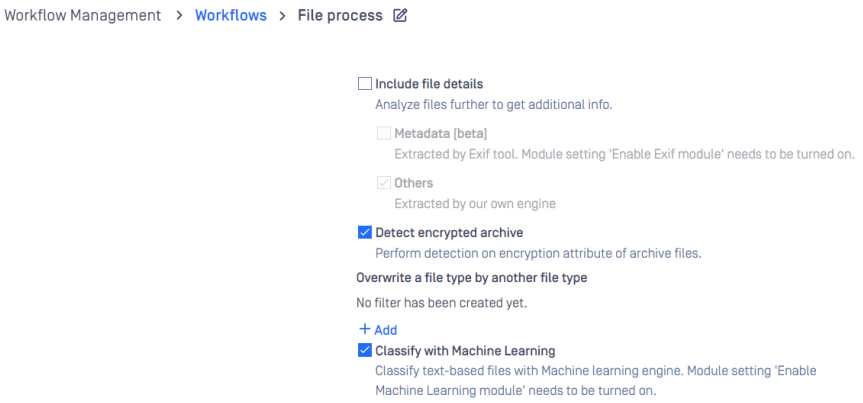
This feature is designed to address the challenge of text-based file detections. These files, such as log files, script files, and readme files, are all simply "text" and lack distinct characteristics revealed by other methods. Analyzing the contents is crucial for accurate classification.

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**



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.

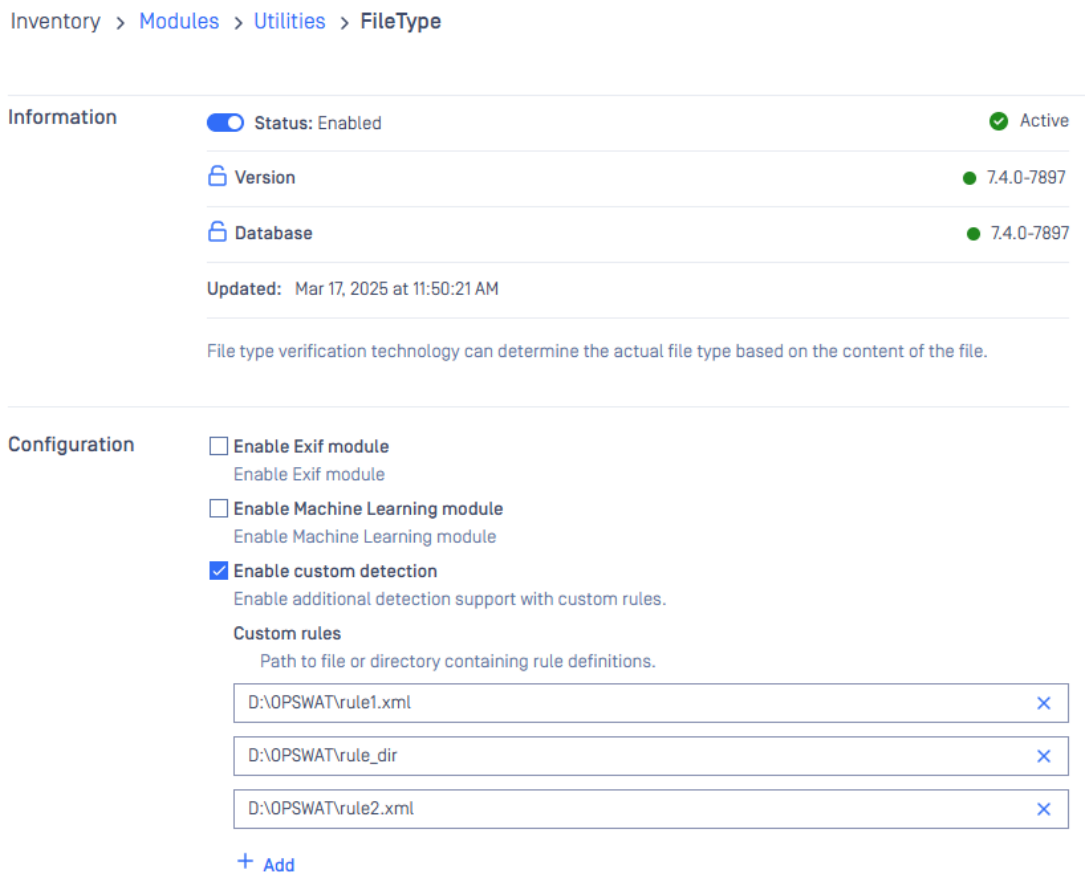
Custom detection

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 XML rule files and/or rule directories that contain XML 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.

Custom rules

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.

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.
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.

Example rules

Below are some XML example rules.

Rule 1 - Pattern 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>0</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>
```

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 6 dependencies, libgdiplus 6.0.5 or above

Supported file types

The archive engine can extract 152 file types:

File type	Note
7z	a password protected file can be extracted with a given password
ACE	
ALZ	
AR	
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	

File type	Note
RAR	all split RAR files can be extracted. A password protected file can be extracted with a given password
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	
XPI	
UUE	
ZSTD	

File type	Note
ASICE	
ASICS	
DOCM	
DOCX	
DOTM	
DOTX	
HPWX	
ODP	
ODS	
ODT	
OTP	
OTS	
OTT	
OXPS	
PBIX	
PBIT	
POTM	
POTX	
PPAM	
PPSM	
PPSX	
PPTM	
PPTX	
RIBC	

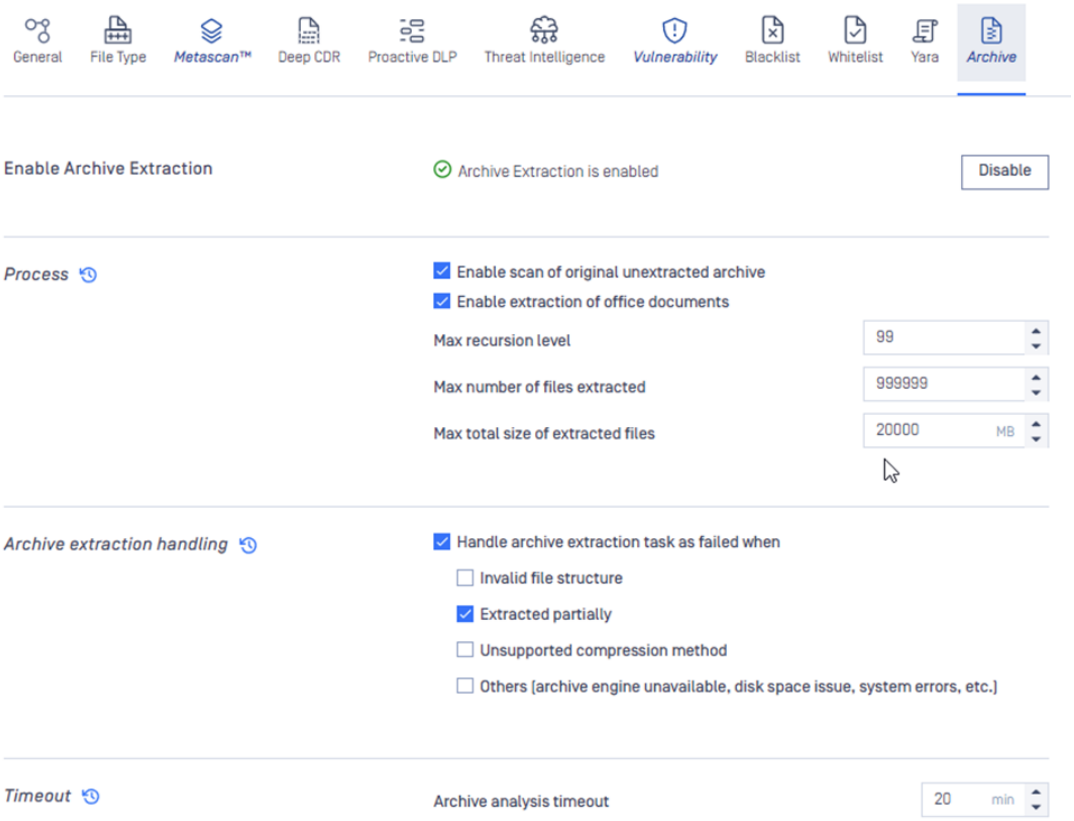
File type	Note
SHOW	
SLDM	
SLDX	
SCDOC	
STC	
STD	
STI	
STW	
SXC	
SXD	
SXI	
SXM	
SXW	
SXG	
TWBX	
UTIB	
VSDM	
VSDX	
VSSM	
VSSX	
VSTM	
VSTX	
XLAM	
XLSB	

File type	Note
XLSM	
XLSX	
XLTM	
XLTX	
XPS	
GPG	please see the details here
APK	
APPX	
DLL	
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	

File type	Note
NTFS	
FAT	
HFS	
OVA	
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	
X12EDI	
ESZ	
MHT	

File type	Note
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

- Encrypted PDF, MS Word (DOC, DOCX, DOCM, DOTX, DOTM, DOT)/Excel (XLS, XLSX, XLSM, XLSB, XLTM)/PowerPoint (PPT, PPTX, PPTM, PPSM, POTX, POTM).
- An encoded base64 file can be decoded by the archive extraction engine.
- Configure the archive extraction in the Workflow:


General
File Type
Metascan™
Deep CDR
Proactive DLP
Threat Intelligence
Vulnerability
Blacklist
Whitelist
Yara
Archive
Enable Archive Extraction
Archive Extraction is enabled
Disable
Process
Enable scan of original unextracted archive
Enable extraction of office documents
Max recursion level
99
Max number of files extracted
999999
Max total size of extracted files
20000
MB
Archive extraction handling
Handle archive extraction task as failed when
Invalid file structure
Extracted partially
Unsupported compression method
Others (archive engine unavailable, disk space issue, system errors, etc.)
Timeout
Archive analysis timeout
20
min

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" }
}
```

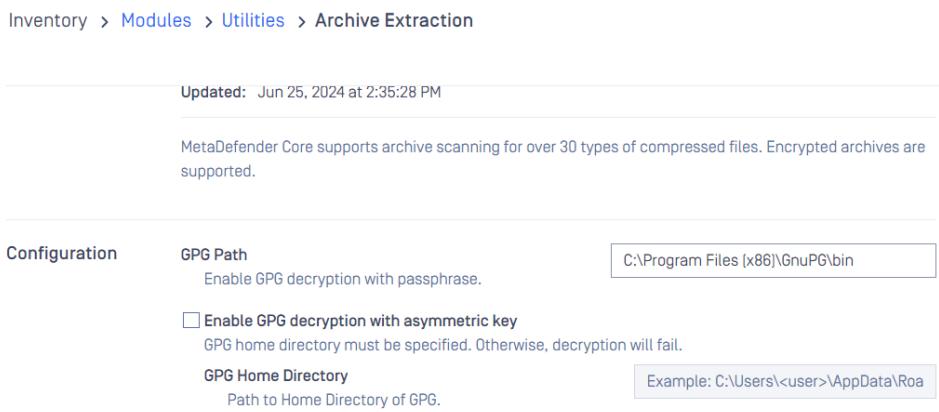
Decrypt GPG files

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

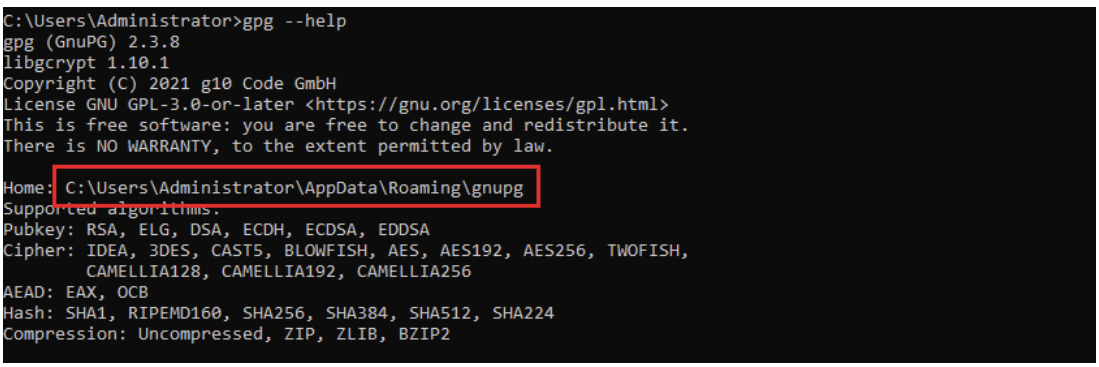
Symmetric encryption (encrypted with password)



With this setting, users can scan a GPG file with a provided password as a regular password-protected archive file.

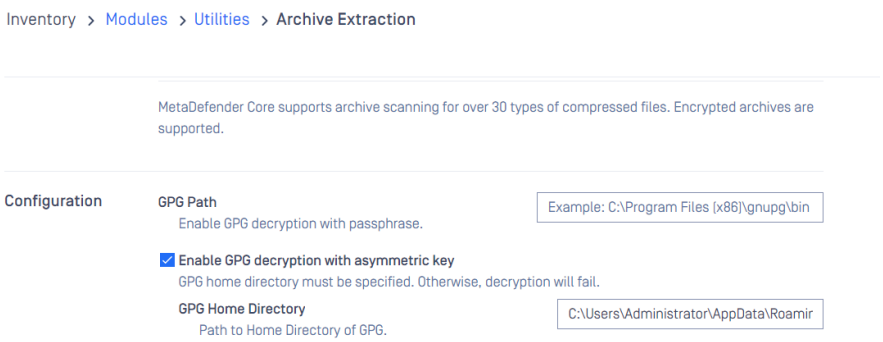
Asymmetric encryption (encrypted with private/public key)

- 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.

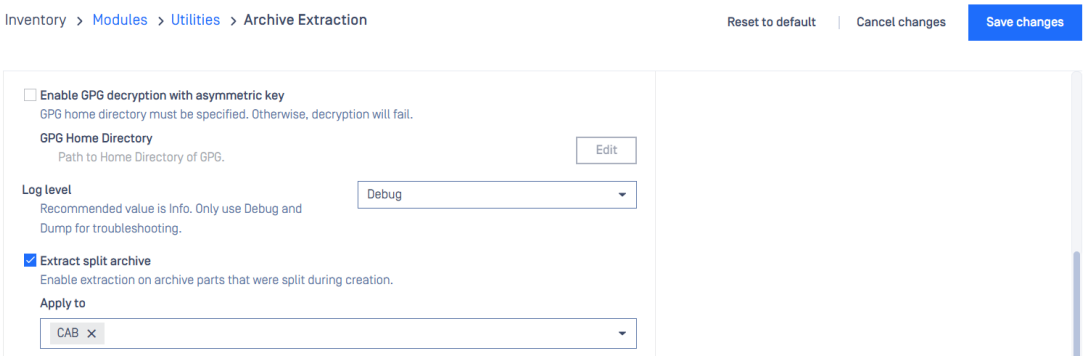
- Set the GPG Home directory in the UI.



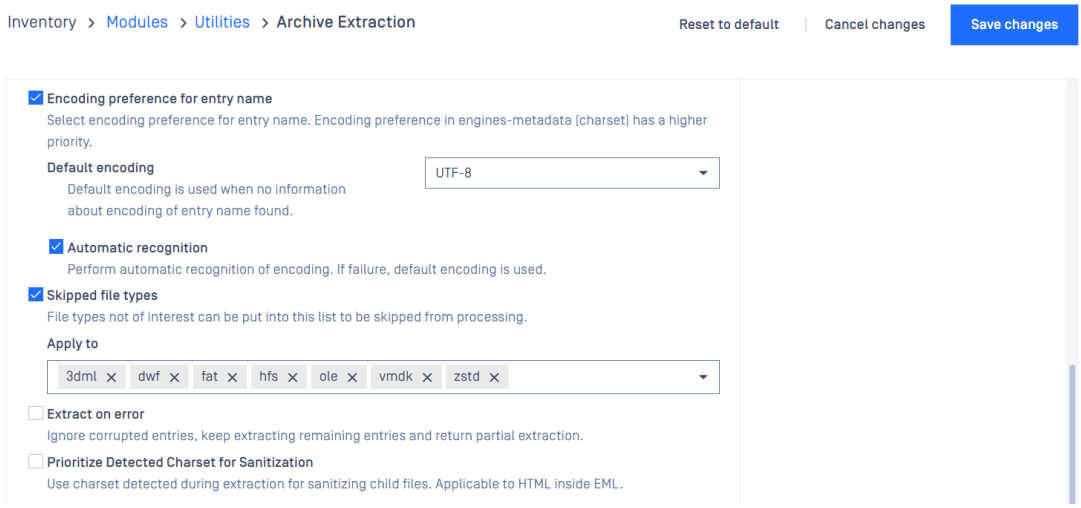
With this setting, users can scan a GPG file with a provided passphrase as a regular password-protected archive file.

Configuration

This guide provides detailed configuration options under **Inventory > Modules > Utilities > Archive Extraction**, enabling users to customize extraction settings effectively.



Archive Extraction Configuration #1



Archive Extraction Configuration #2

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.

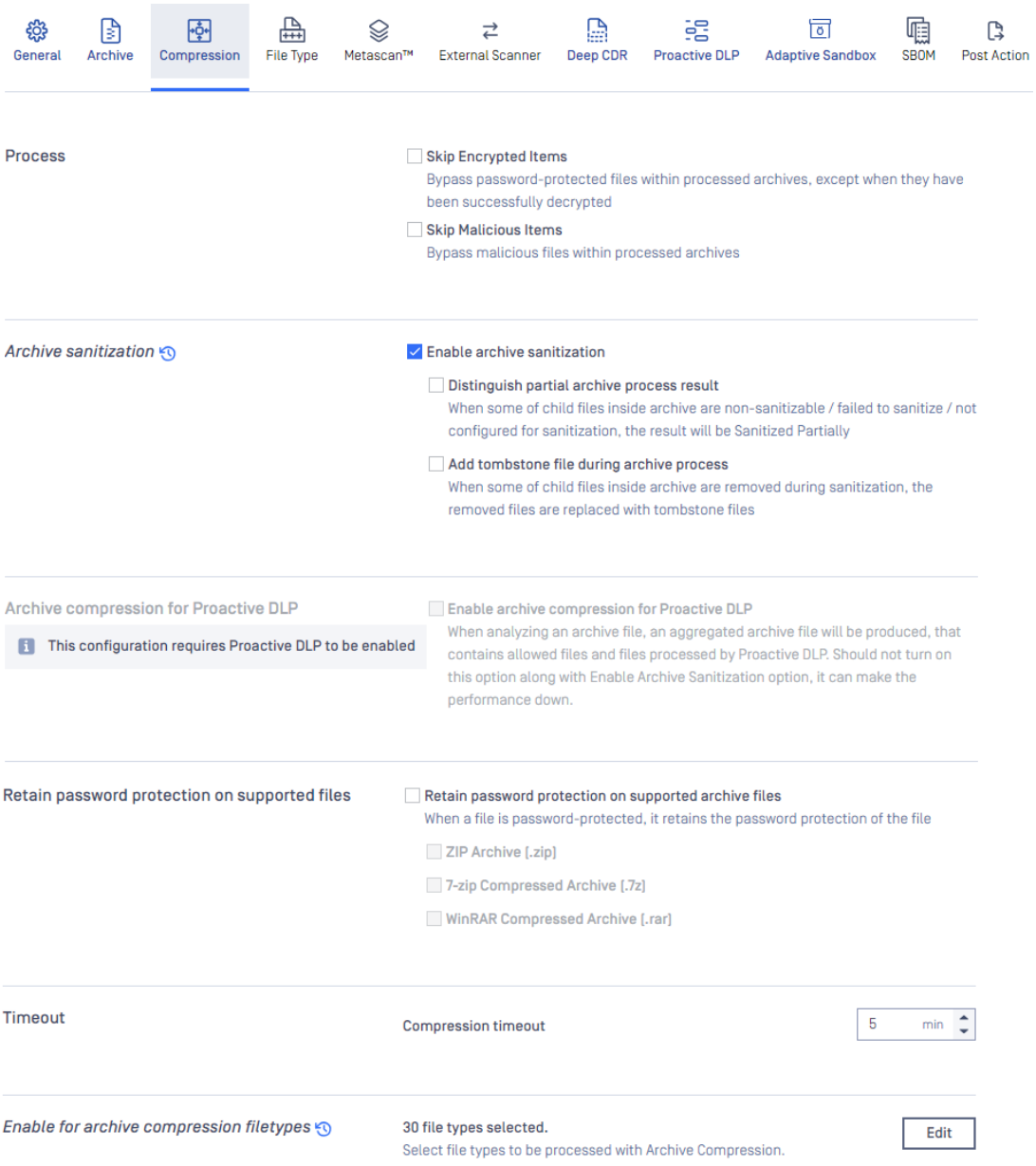
Archive Compression Engine

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.

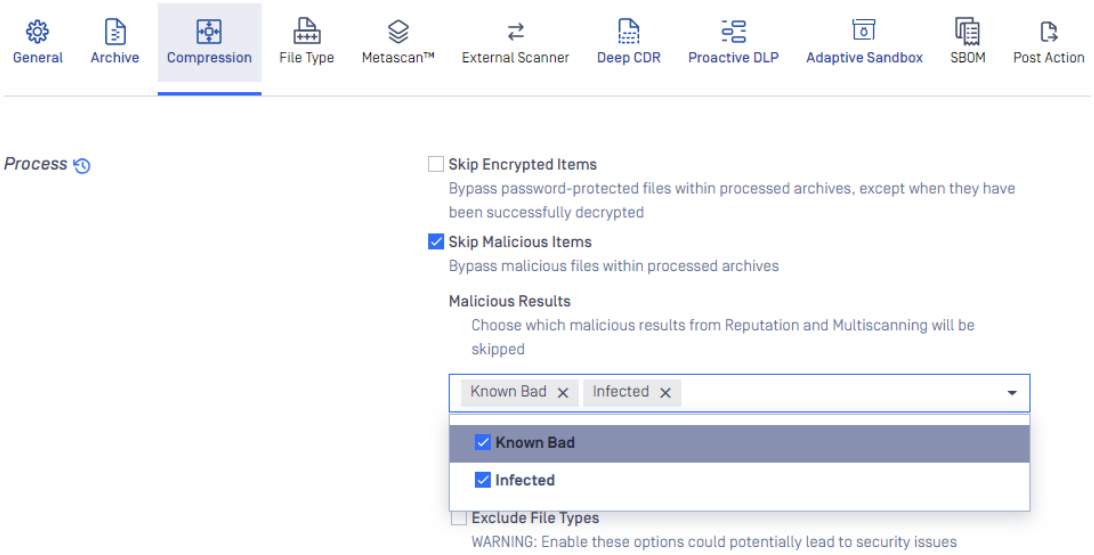
Configuration



Detailed explanations

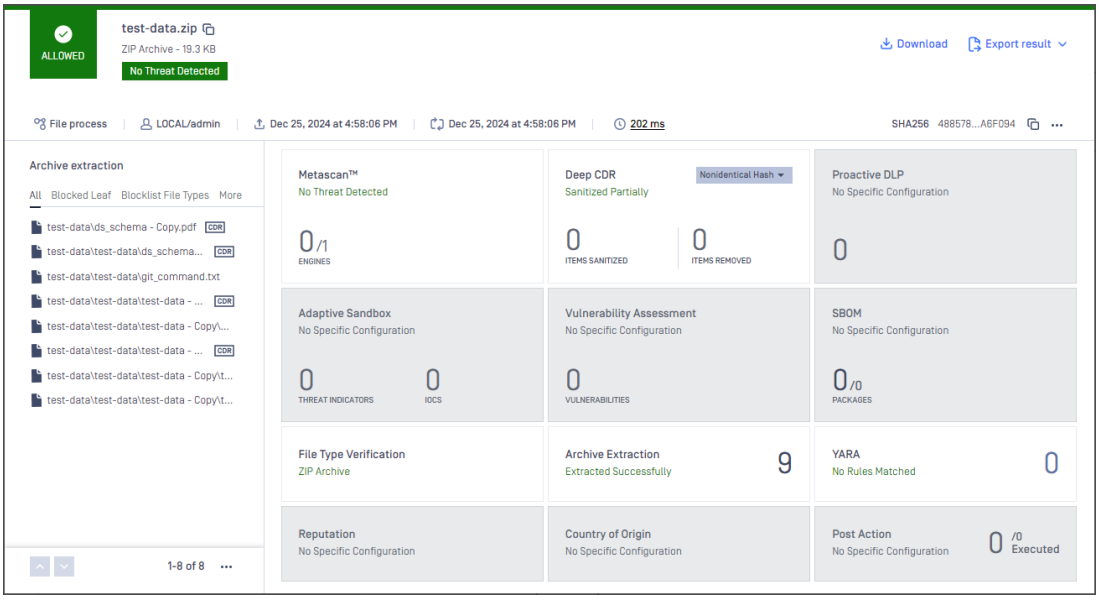
Process

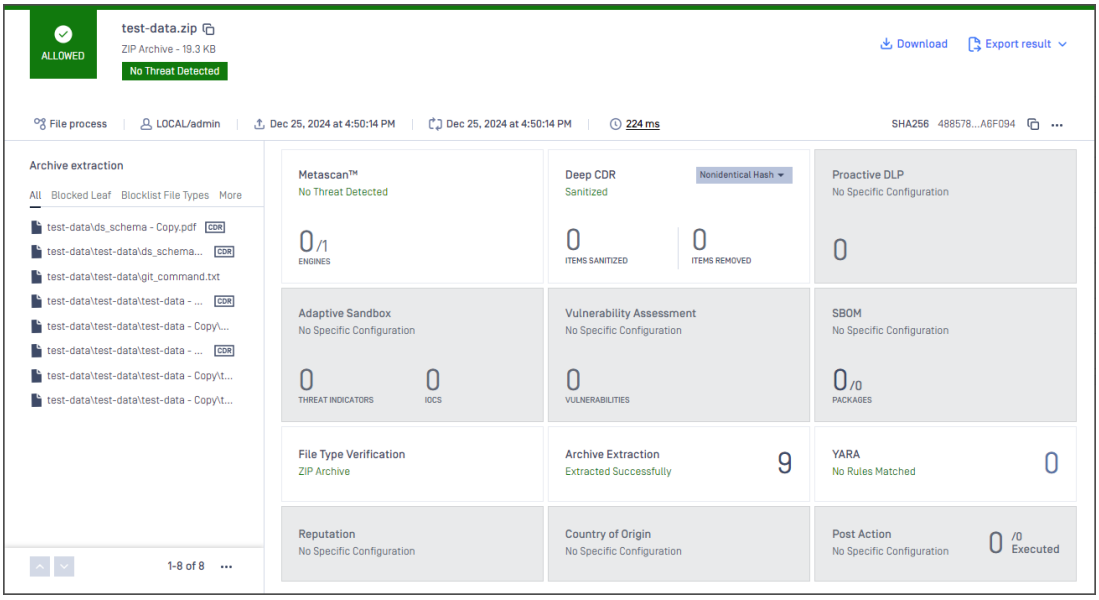
- **Skip Encrypted Items**: When enabled, encrypted files that cannot be decrypted are excluded from the output during the compression of the processed archive.
- **Skip Malicious Items**: When enabled, files identified as malicious are excluded from the output during the compression of the processed 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.

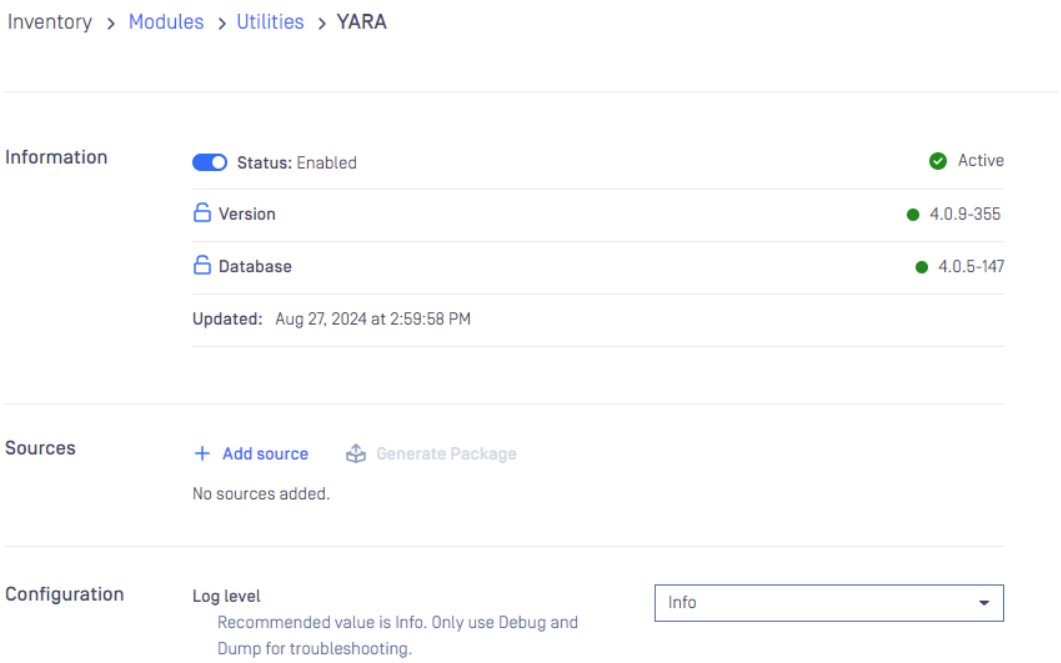
Enable for archive compression filetypes

Select the supported archive file types to apply compression, whether for archive sanitization or DLP processing.

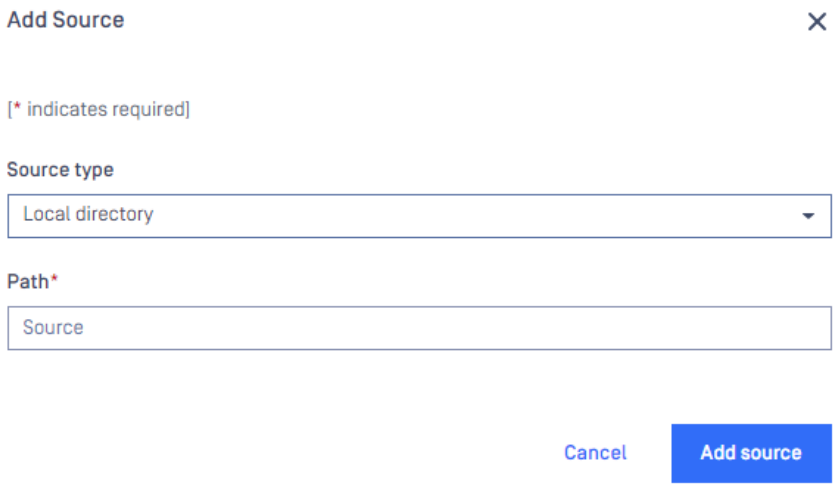
YARA Engine

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:

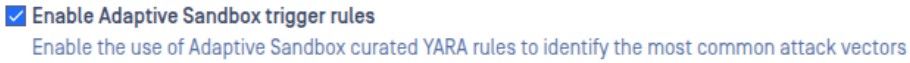
- Magic
- Hash
- Dotnet
- Macho
- Dex
- Cuckoo
- Androguard

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.

Release notes

Version	5.15.0
Release date	03 July 2025
Scope	This major version offers CIS level 2 support, processing efficiency improvements, continued My OPSWAT Central Management integrations, and several feature enhancements.

Before you upgrade MetaDefender Core:
Making sure to check out the Known Limitations

New Features, Improvements and Enhancements

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 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 API.

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

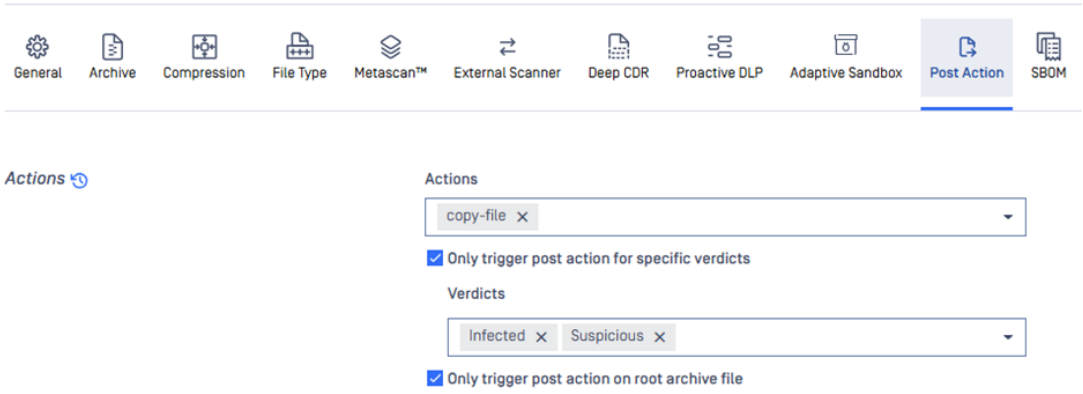
Administrators can now define a file size limit at the NGINX layer. This enables earlier enforcement of policy controls and supports more distributed or containerized environments.

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.

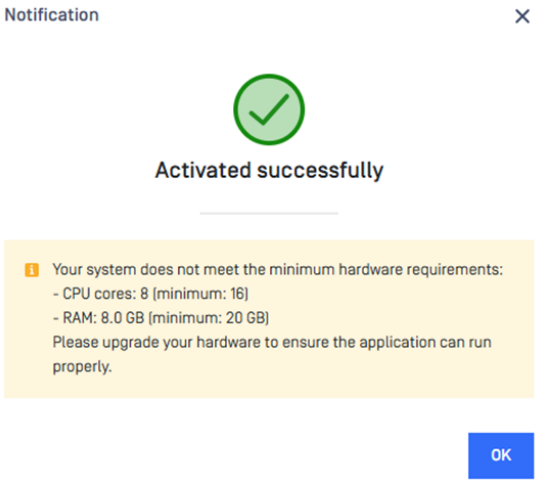
You can find the new option `Only trigger post action on root archive file` under tag Post Action in Workflow Rule.



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

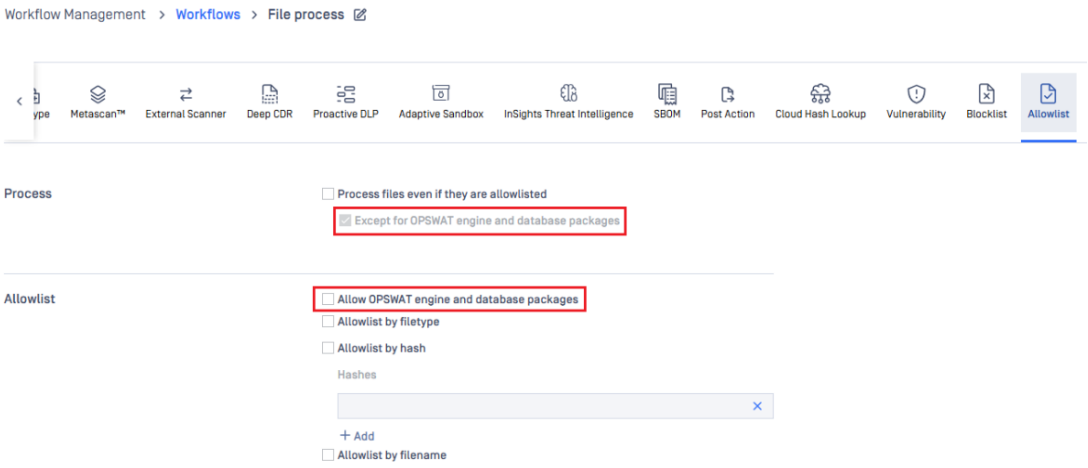
Using the Local Folder Update folder with a UNC path previously consumed significant network bandwidth.

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.

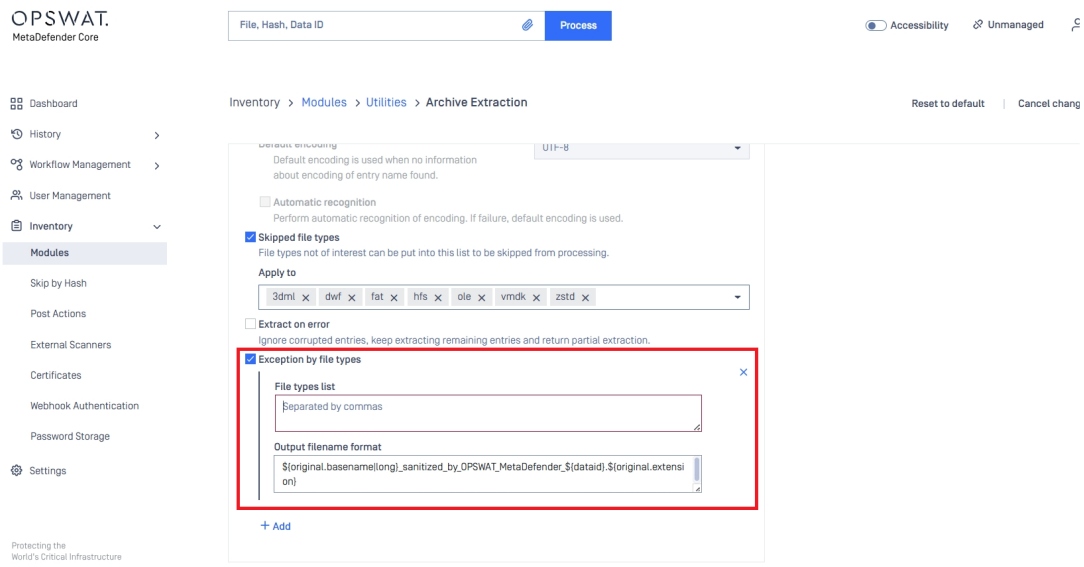
New options have been introduced in the Allowlist workflow configuration to automatically allow the OPSWAT engine and database packages.



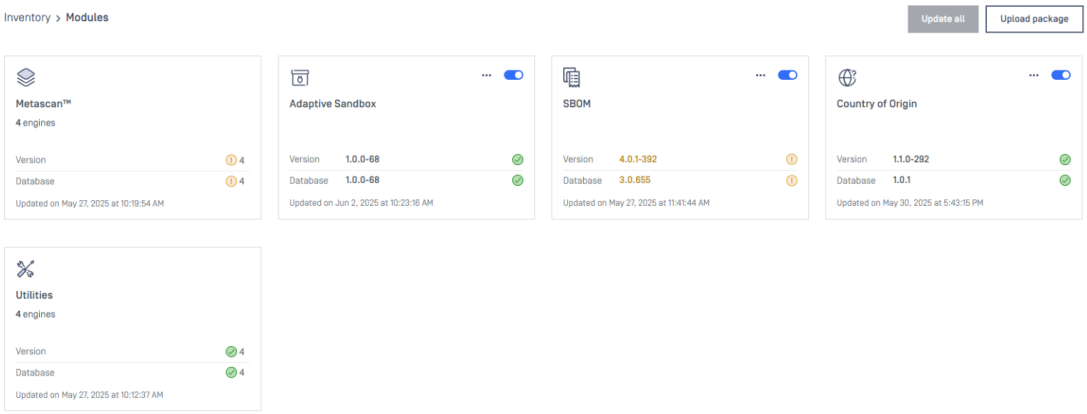
Usability Enhancements & Changes

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.

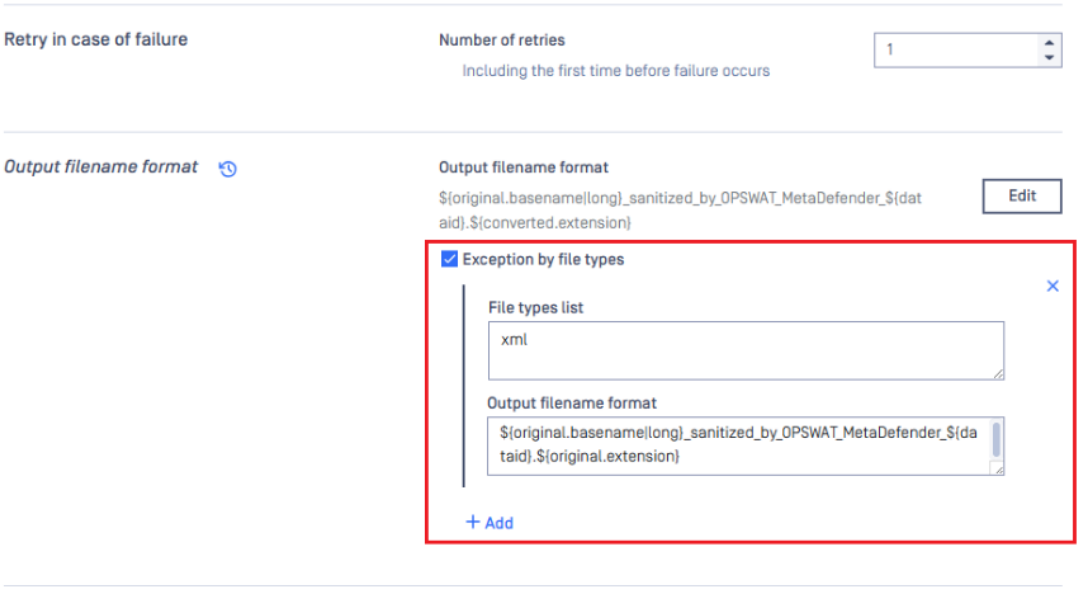
2) Improved the look and feel of multiple text boxes within lists in Workflow Rule.



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 API
- `GET` `/hash` - Fetch Analysis Result By Hash API

2) Supporting customizable parameters with new header `info` in the REST API GET /readyz API 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 API.

4) Introduce a dedicated API endpoint to validate PIN code: POST /admin/pincode/validate API.

5) Provide a new API endpoint to trigger `Local Folder Update` even during business hours or pause update window: POST /admin/update/folder API.

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.

Known Limitations

	Details
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 MetaDefender Core v5.14.0.** 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.

	Details
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 properly cleaned up if the Archive Extraction engine crashes	Starting from MetaDefender Core version 5.10.1, if 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.
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.

	Details
Performance degradation when processing large archive files	**This issue has been resolved in MetaDefender Core version 5.5.0 and the Archive Extraction engine version 6.2.1.** • If you're using MetaDefender Core version 5.4.1, you can update the Archive Extraction engine to version 6.2.1 or newer without waiting for MetaDefender Core version 5.5.0 release. • If you're using MetaDefender Core 5.4.0 or older, you can upgrade it to version 5.4.1 or newer, and update the Archive Extraction engine to version 6.2.1 or newer. If you are not able to upgrade MetaDefender Core, it is recommended to stick with the Archive Extraction engine version 6.0.2 until you are able to upgrade MetaDefender Core.
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 MetaDefender Core v5.11.1** 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.

	Details
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)

Archived release notes

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

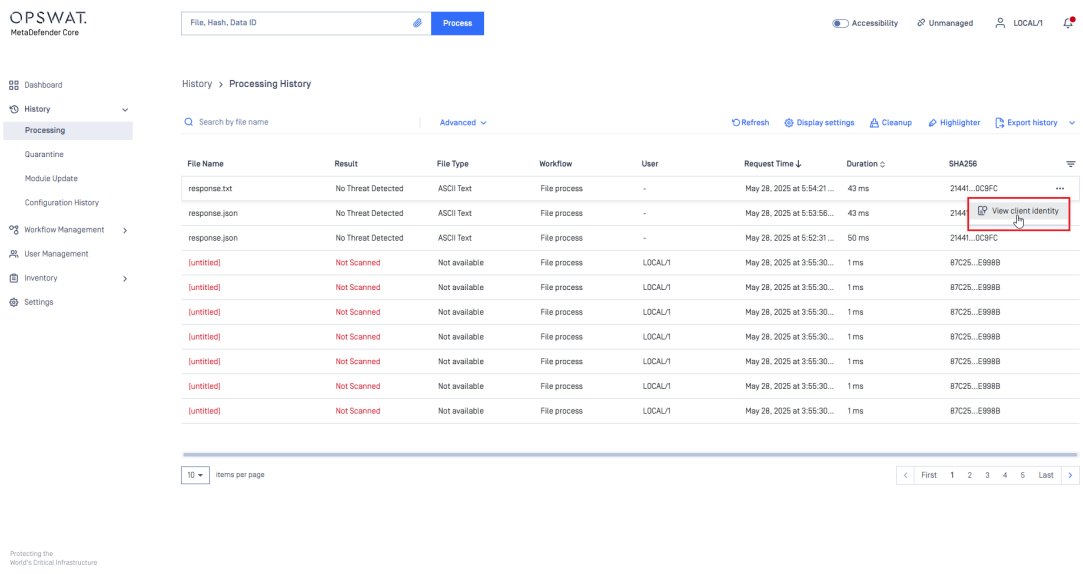
All the following methods support this header: asynchronous mode, synchronous mode, local scan, scan from link, batch scan, multi-part uploading.

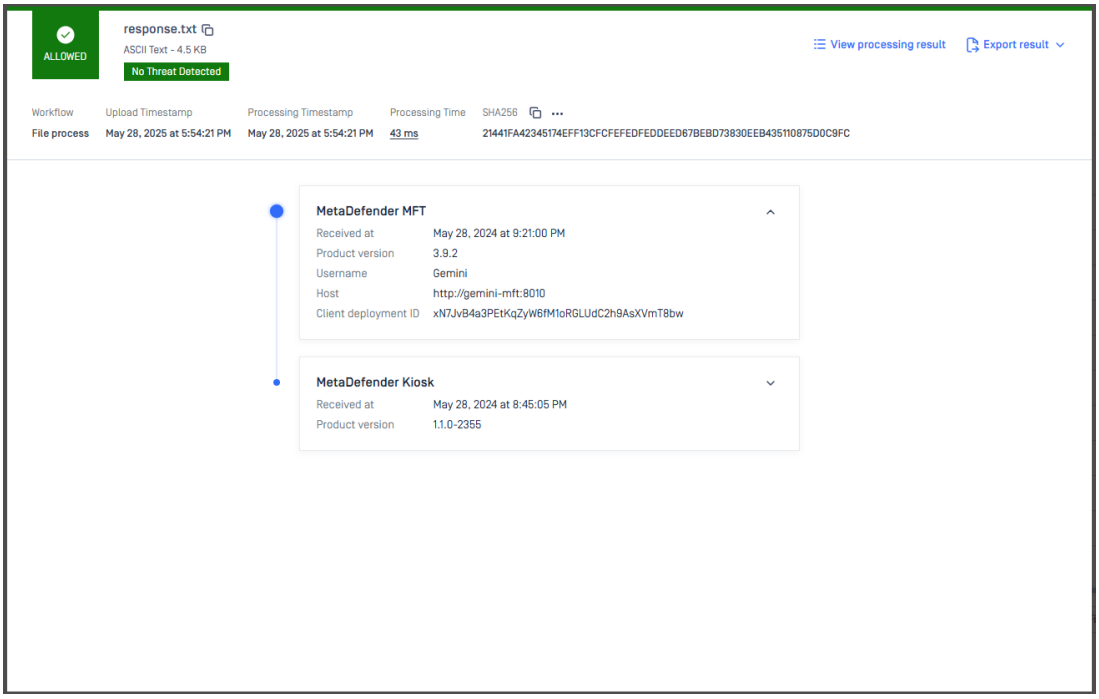
More details at:

- GET - Fetch Analysis Result API
- GET - Fetch Analysis Result By Hash API
- GET - Status of Batch Analysis API
- POST - Analyze File (Asynchronous mode) API
- POST - Analyze File (Synchronous mode) API
- POST - Initiate Batch API
- POST - Create multipart uploading session API

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

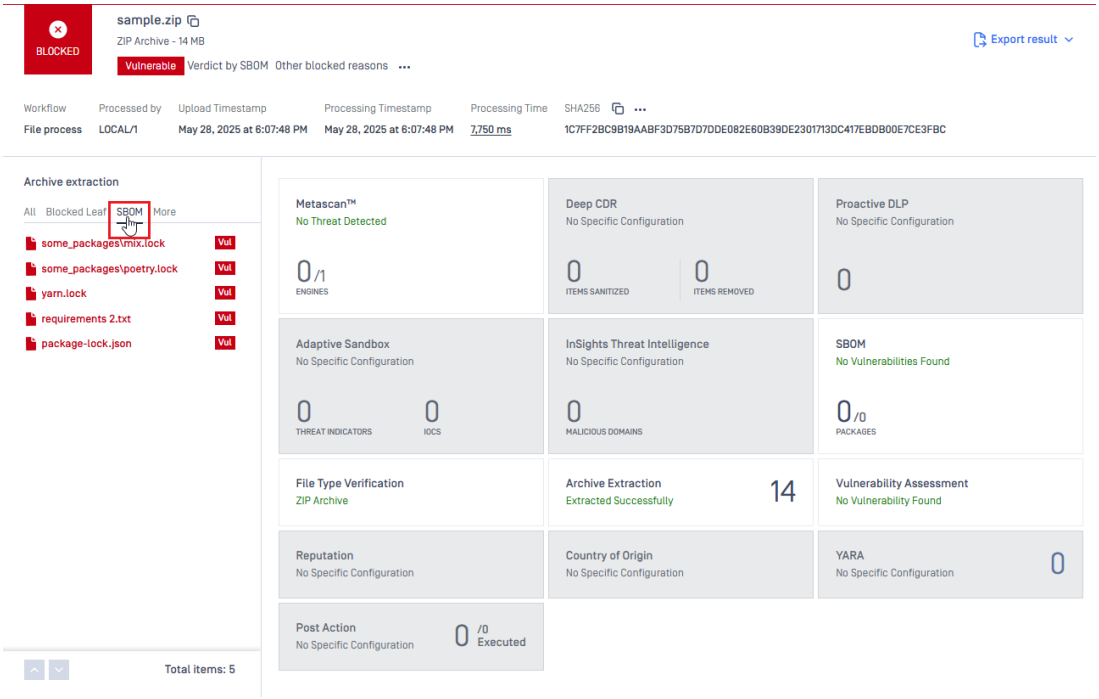
Starting with version 5.14.2, MetaDefender Core introduces support for archive password handling in multipart upload APIs, enhancing functionality for processing password-protected archives.

The passwords are transmitted via header `archivepwd` and are secured through at-rest encryption.

More details at POST - Create multipart uploading session API.

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 API.

Securing syslog communication over TLS

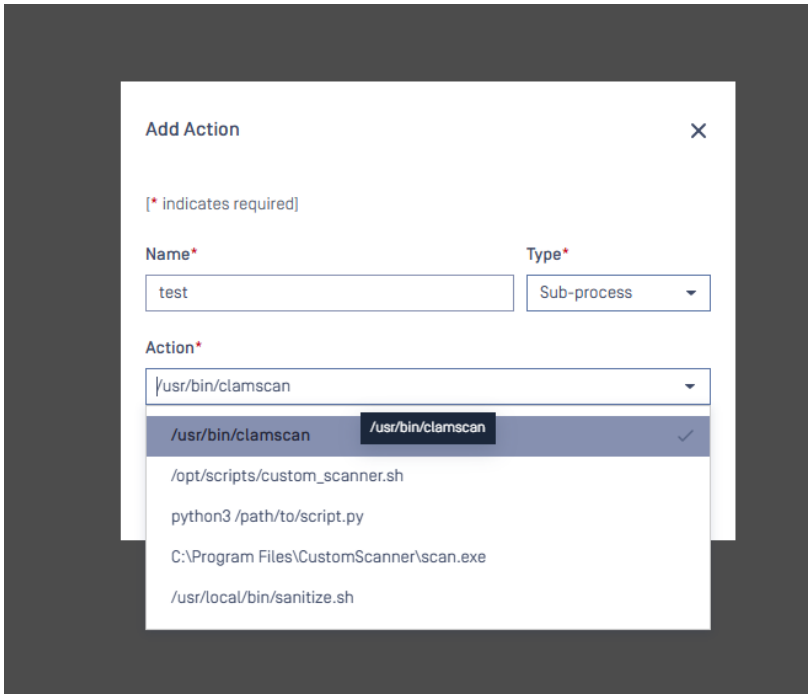
This version introduces support for securing syslog communication over Transport Layer Security (TLS), enhancing the confidentiality and integrity of log data transmitted across networks.

New protocol `tcps` is supported. Documentation at Logging, and API can be found at Apply syslog server configurations API.

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.

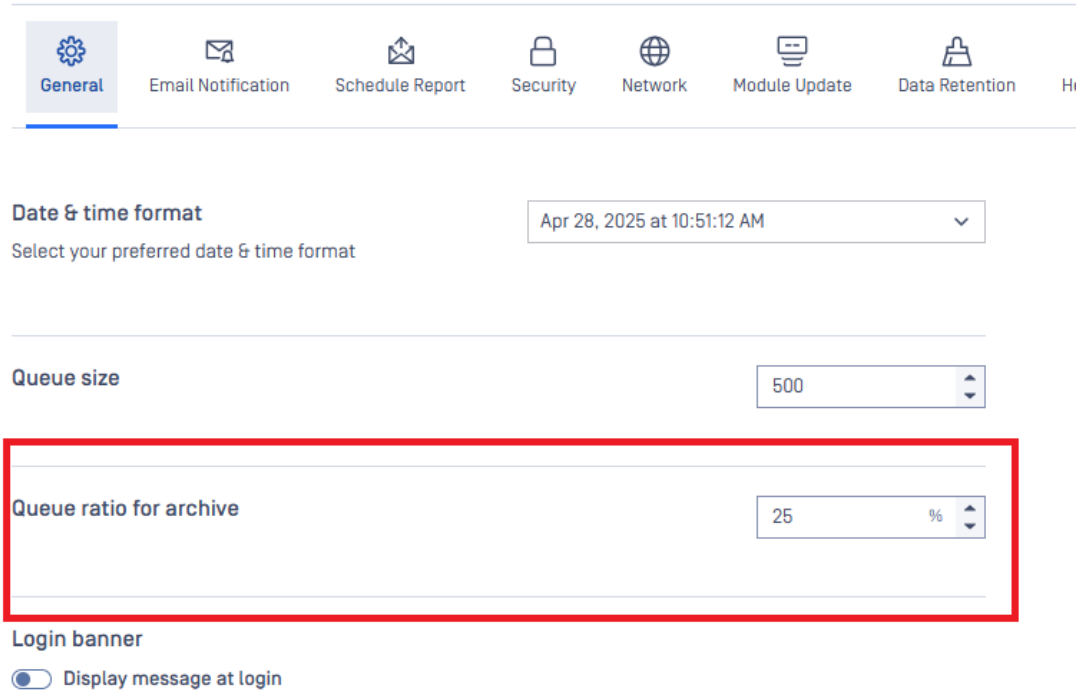


Post Action with permitted-only commands.

An ability to adjust maximum slots of queue capacity for archive processing

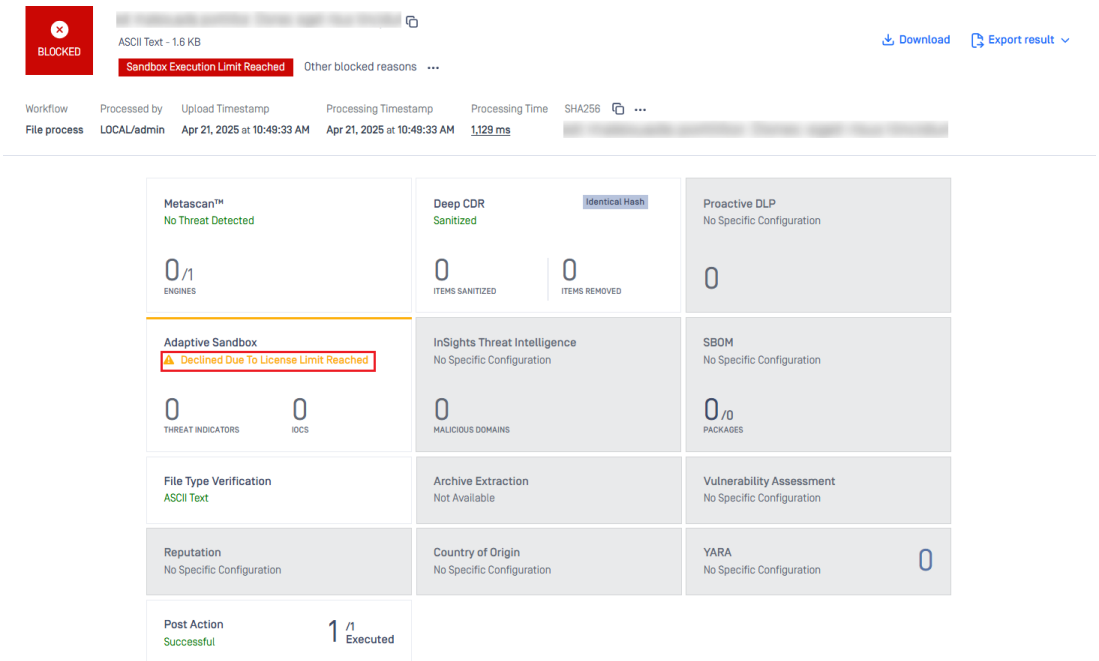
Previously, archive processing was fixed at 25% of the maximum queue size. Now, Administrators can configure this value within a range of 25% to 75%, providing better flexibility to optimize resource allocation for archive processing to improve efficiency in high-throughput environments.

You can find the setting `Queue ratio for archive` at Settings/General.

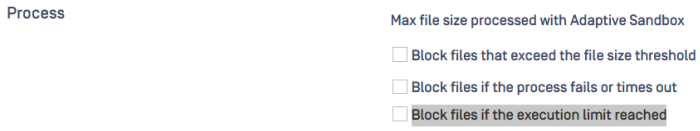


Embedded Adaptive Sandbox engine with Daily Scan Quota

When the daily scan quota is reached, Sandbox engine will reject scan requests and display a warning message `Declined due to license limit reached`.

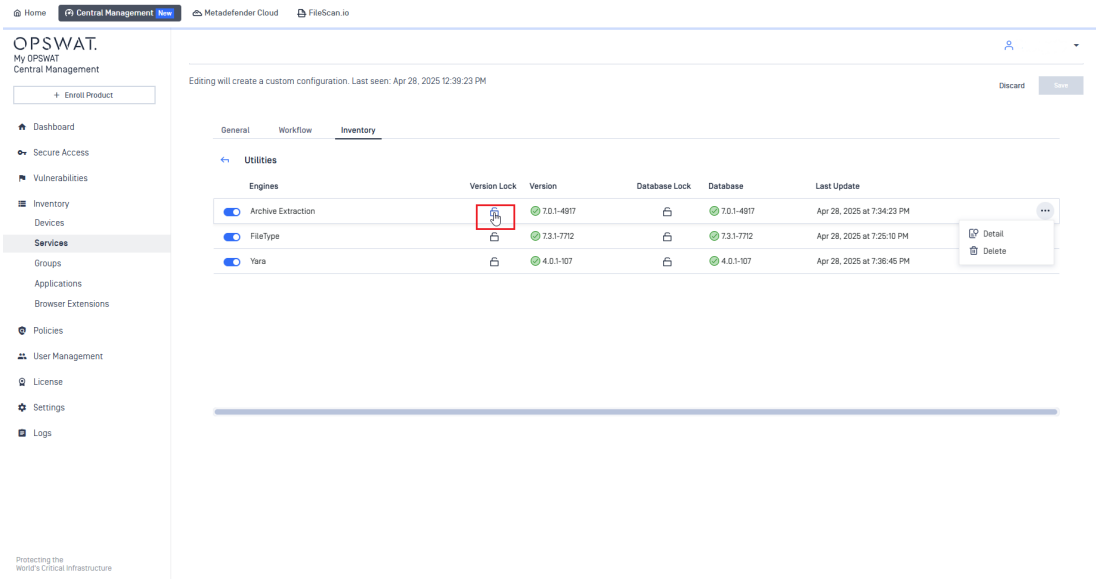


Users can decide to block files due to out-of-quota by ticking a new option `Block files if the execution limit reached` . It can be found in Workflow/Adaptive Sandbox.



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 API.
  - New API for AD-bound password rotation automation. For more details, refer to Change Password for AD/LDAP User Directory API.
  - 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 API
  - Remove syslog servers or clear all syslog configurations API
- 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 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 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 API 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 API.
    - Support a REST API to fetch the product version. The specification is the same as Get Product Version API.
- 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 API
  - Upload individual part of file to the conducted session: POST /file/multipart/{dataID} API
  - Check uploading status: GET /file/multipart/{dataID} API
  - Abort multipart uploading: DELETE /file/multipart/{dataID} API
- 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 API 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) API
  - 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 API 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 API 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 API
  - 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 API, 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 API 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 API, 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 API 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 API
- 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 API
- 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 API
- 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 Distributed Cluster Overview
- 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 API
- 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 API
- 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 API

- 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 API
- 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) API
- 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 API
- 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 PostgreSQL communication
- 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 API
- 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 4.1.7

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

v7.5

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, follow this instruction)
- 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

v3.0.0

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

v7.5

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

Archive Module release notes

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

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

v4.3.0

Release date 9/4/2025

- 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

Warning

The required Java for the embedded 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. Upgrade to Java 17

Known issues

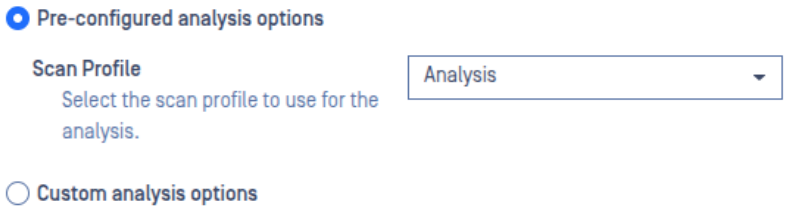
Issue	Solution
MSI certificate validation is not supported on Windows OS	-

V2.3.0

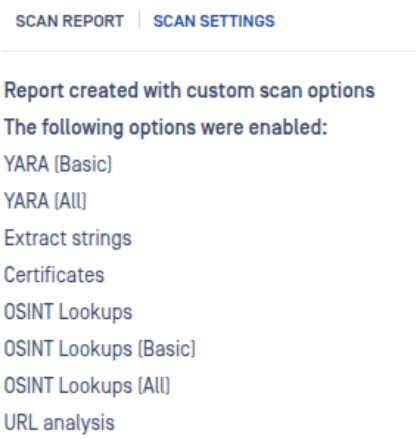
Release date: 06/05/2025

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.



- **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.

New Features and Improvements

- **Phishing DKIM Replay Abuse Detection:** Flags spoofed DKIM attempts, enhancing phishing resilience against 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

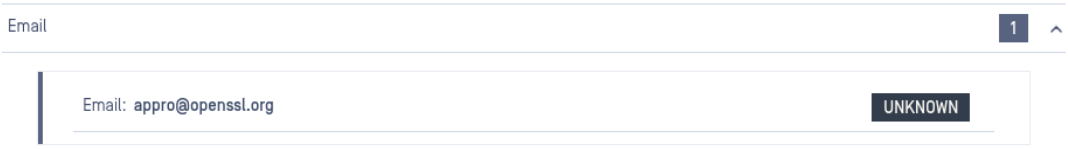
Release date: 02/17/2025

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



- **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.

New Features and Improvements

- **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 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.

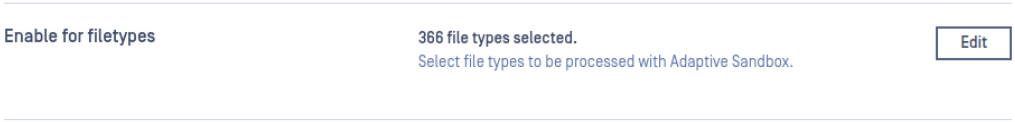
v2.1.0

Release date: 12/02/2024

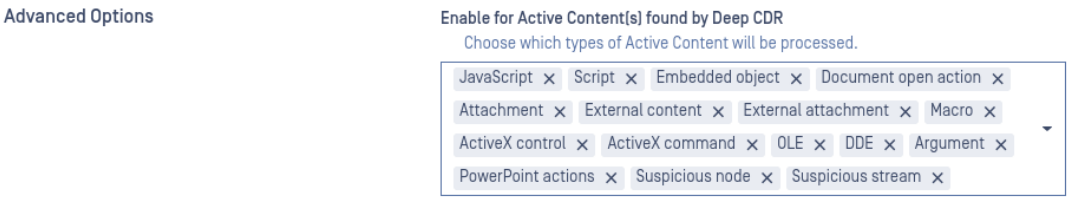
New Features and Improvements

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.



- **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.



Deep CDR Triggers

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



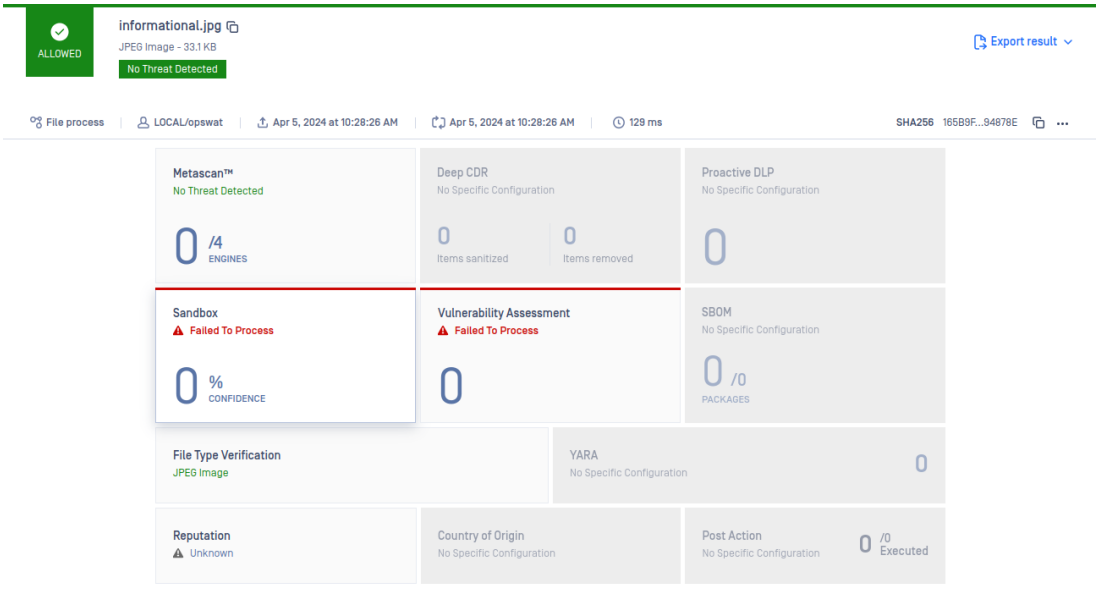
Verdict for IOCs

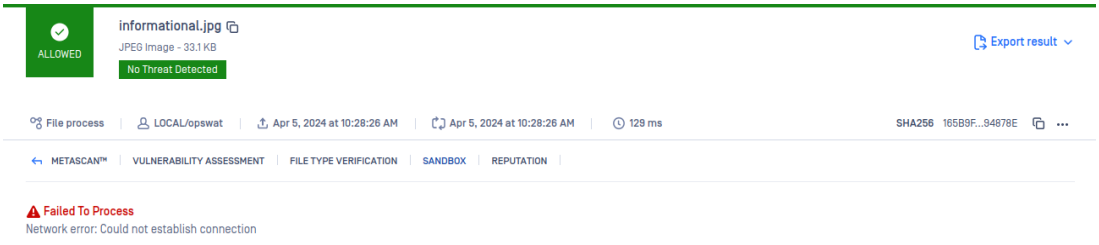
v1.7.0

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
- 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)





Improved error reporting

v1.6.0

Release date: 01/29/2024

New Features and Improvements

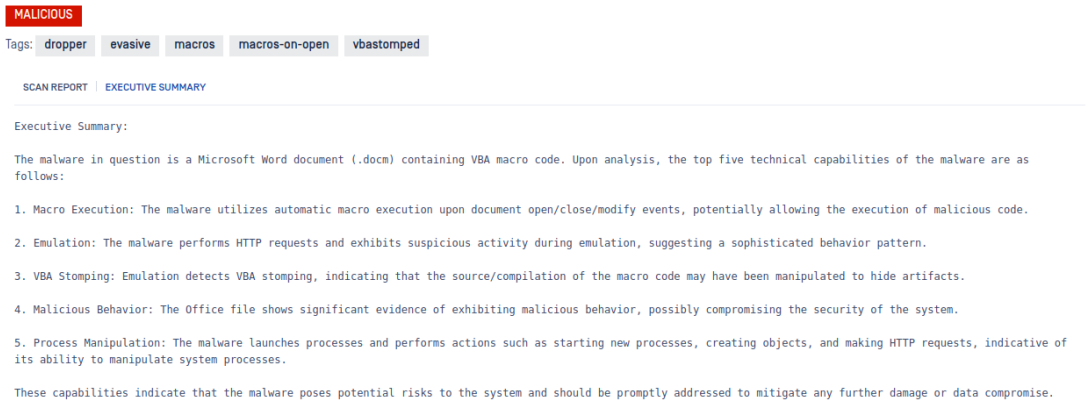
- Improved engine performance and stability
- Implemented configurable OPSWAT Reputation secret in engine global config
- New indicators for Windows APIs related to specific activities
- Implemented flagging for LSASS dump using minidump
- Extracted remote templates inside xTable struct in MS Office documents
- 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)



Executive Summary



Crypto Wallets

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

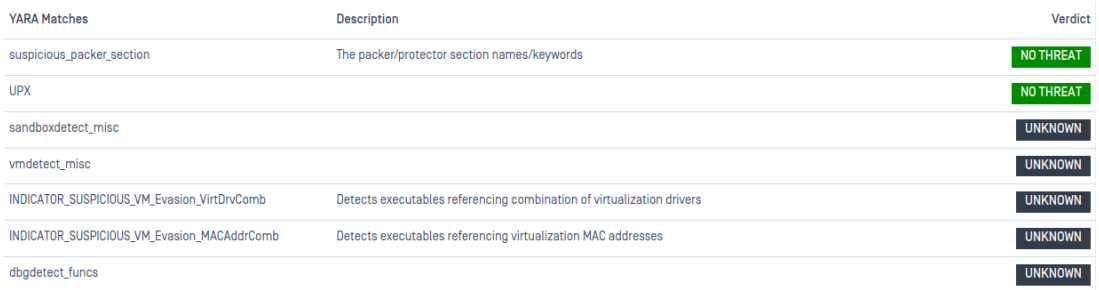
- Enabled XML file support by default
- Updated reputation sources
- Improved verdict calculation
- Fixed global config reset to default values feature
- Fixed report generation for files including Email IOCs

v1.3.0

Release date: 05/26/2023

New Features and Improvements

- Updated YARA rule database
- 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

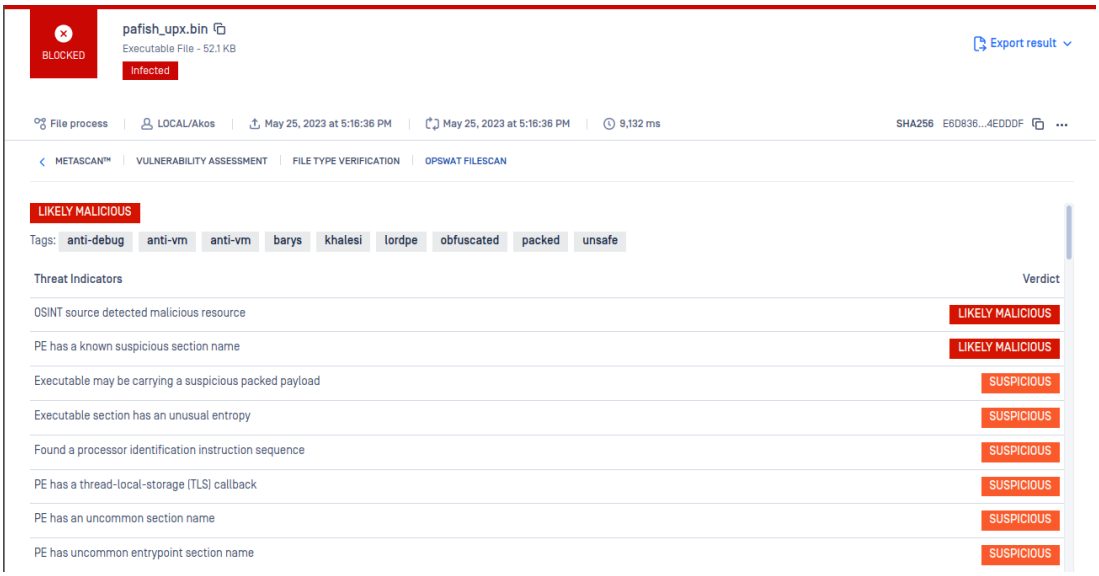
- Improved Microsoft Office file handling
- Security and performance improvements

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 Sandbox standalone instance up to date for the latest features. See the release notes for more information.

Known issues

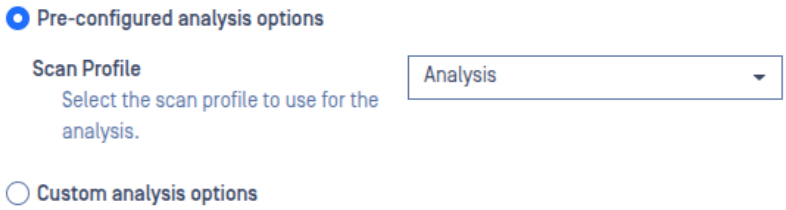
Issue	Solution

v2.3.0

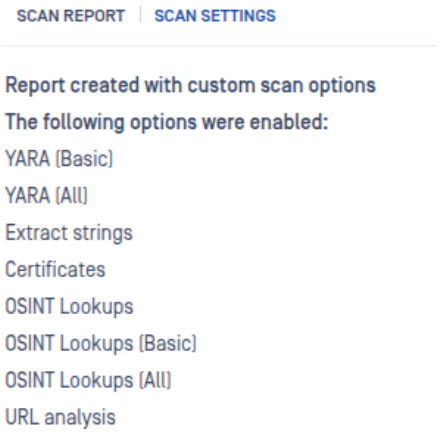
Release date: 06/05/2025

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.



v2.2.0

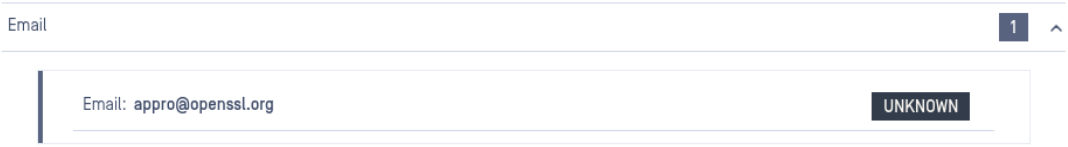
Release date: 12/02/2024

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



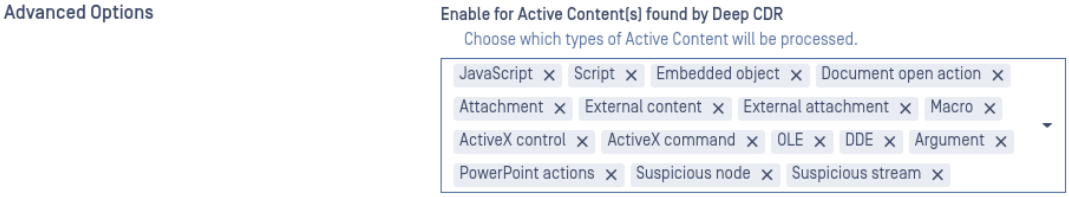
- **Dependency handling**: Visual C++ Redistributable is no longer required, making it easier to install and update the engine on the fly.

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

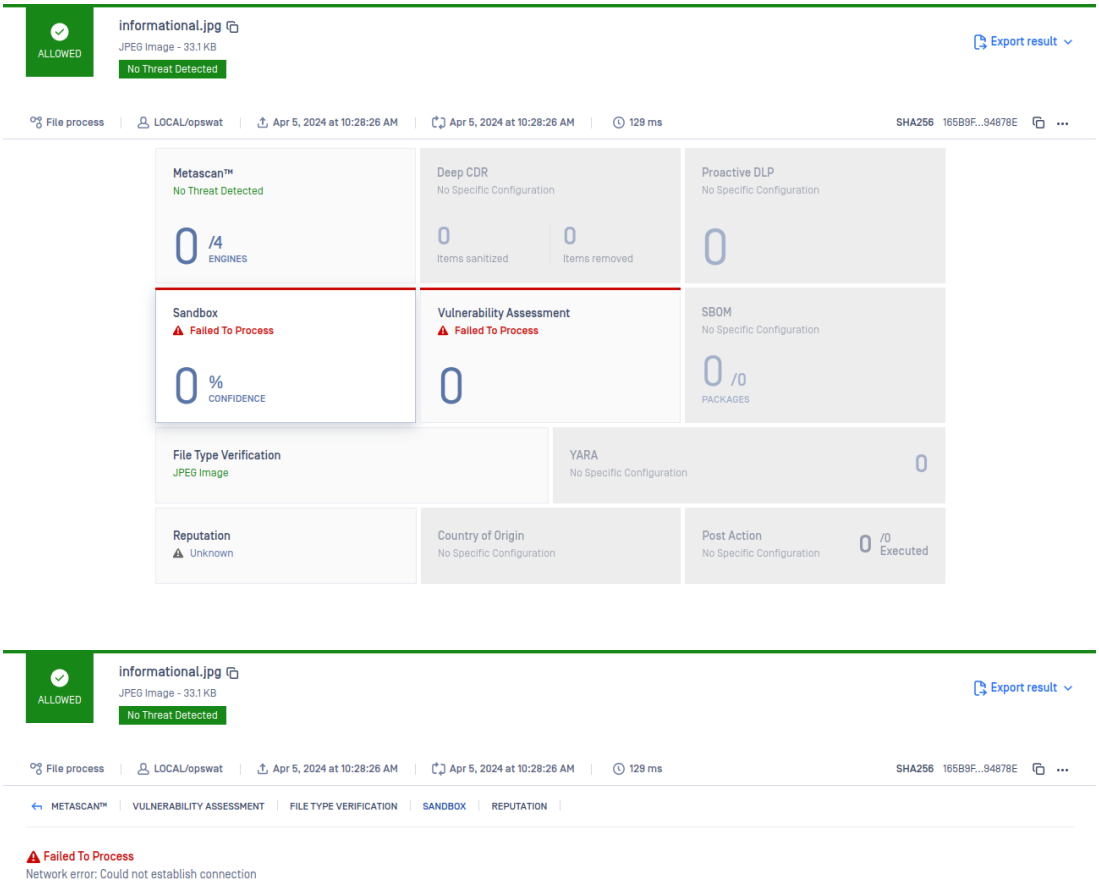
- Ensured support for Ubuntu 22.04
- Added verdict to IOCs on the UI
- Sandbox standalone version 2.0.0 compatibility

v1.7.0

Release date: 04/26/2024

New Features and Improvements

- Improved error reporting
- Added HTTP redirection support



Improved error reporting

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



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
- 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

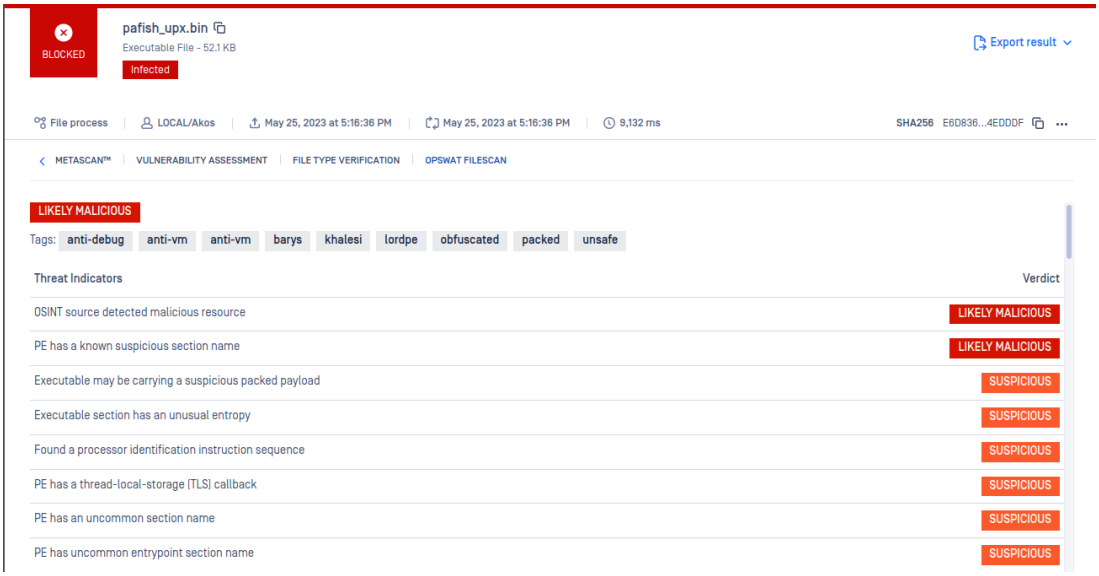
- Improved Microsoft Office file handling
- Security and performance improvements

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

2.1.2

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

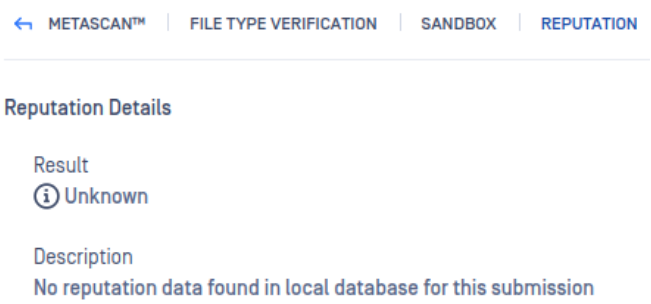


Example for dependency check

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

v2.0

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.

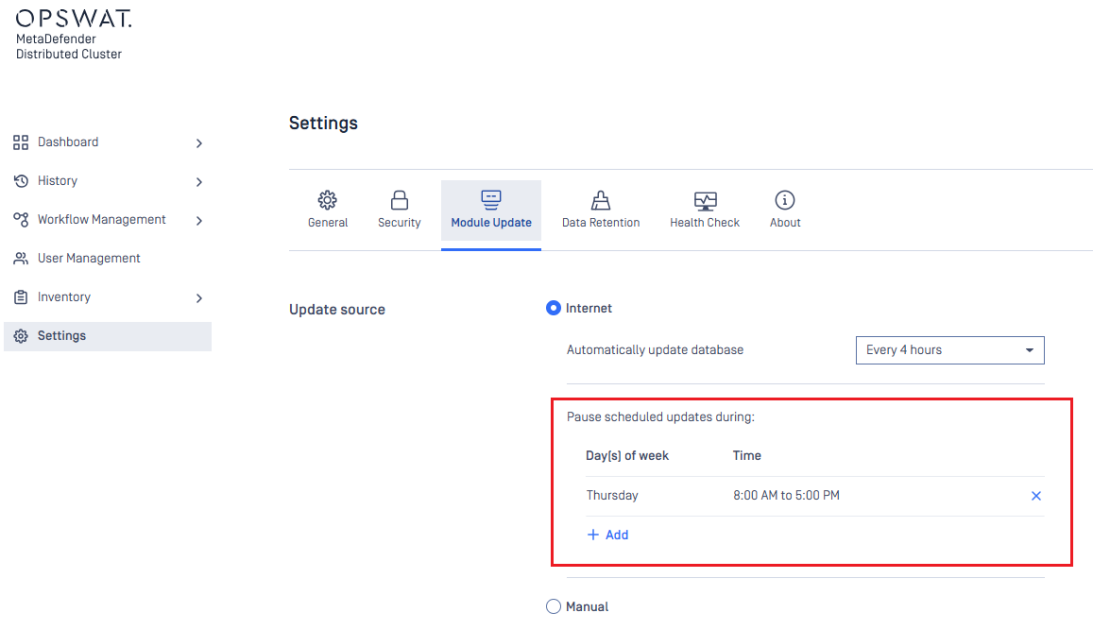
MetaDefender Distributed Cluster release notes

v2.1.0

Release date: 03 July 2025

MetaDefender Distributed Cluster Control Center

- Users can view the details of scan results as the files are being processed in MetaDefender Distributed Cluster Control Center's user interface.
- Users can pause the online scheduled update of modules within MetaDefender Core deployed by MetaDefender Distributed Cluster Worker.



MetaDefender Distributed Cluster API Gateway

- The clients of MetaDefender Distributed Cluster can submit files for scanning with gzip or base64 Content-encoding header. More details at POST - Analyze File (Asynchronous mode) API.
- The clients of MetaDefender Distributed Cluster's client can retrieve in-progress results by hash with API GET /hash/{value}. More details at GET - Fetch Analysis Result By Hash API.

MetaDefender Distributed Cluster File Storage

- The default location is used for saving submitted files if `storage.path` field is ignored in Ignition file.

Security Enhancements

- Upgraded libraries for vulnerability fixes:
  - PostgreSQL v14.18
  - Yaml-cpp v0.8.0

Bug Fixes

- Fixed an issue that prevented user from deploying MetaDefender Core or MetaDefender Distributed Cluster API Gateway following a failure during deployment on MetaDefender Distributed Cluster Worker.
- Fixed an issue that prevented the MetaDefender Distributed Cluster Worker from upgrading to the new version when at least one engine encountered an error.
- Fixed an issue that restricted the hostname to just 64 characters when users attempted to add a new component service to MetaDefender Distributed Cluster.

v2.0.0

Release date: 28 May 2025

Introducing a brand new product, MetaDefender Distributed Cluster, 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.

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?

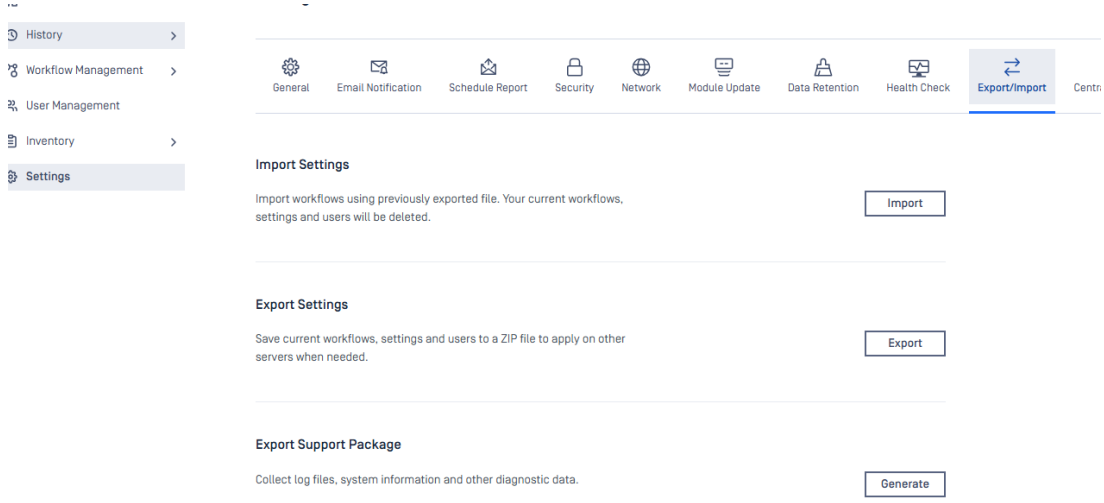
How to Create Support Package With Web Console

> **Check Your Version:**
> This article applies to all MetaDefender Core releases **after version 5.13.0**, deployed on both Windows and Linux systems.

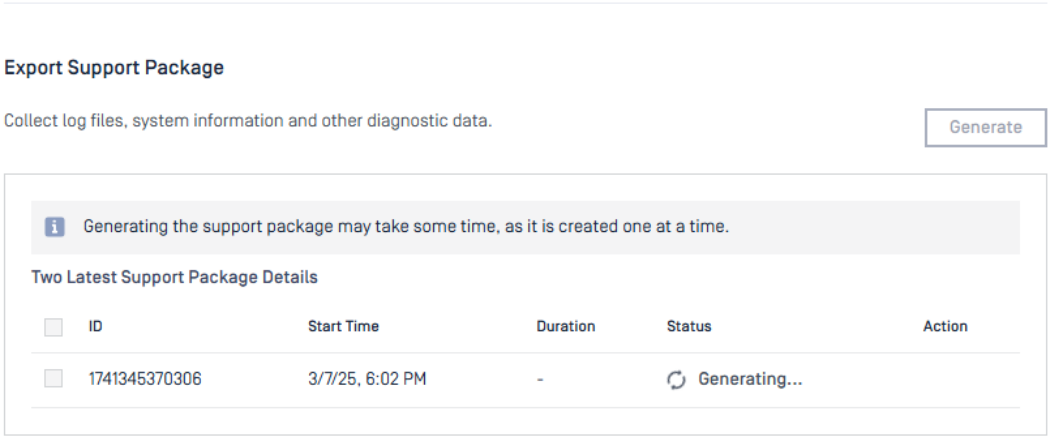
Kindly note that the 'Generate Support Package' button is not available in versions earlier than 5.13.0.

Please go to Settings > Export/Import

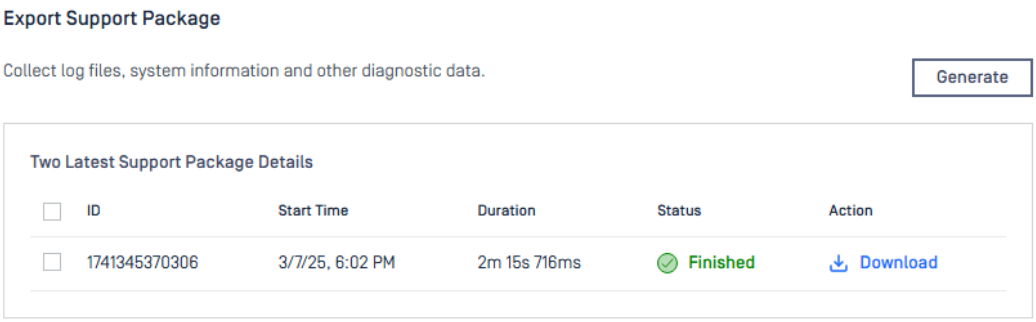
Check the sections: Export Support Package and click Generate



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.

none

```
#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.

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:

none

```
ometascan-<VERSION>-support-<TIMESTAMP>.tar.gz
```

Where the timestamp is the date when the package was generated.

Example:

none

```
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:

none

```
ometascan-<VERSION>-support-<TIMESTAMP>.zip
```

Where the timestamp is the date when the package was generated.

Example:

none

```
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.

Debug logging

> **Warning**
> Enablding debug logging level on MetaDefender Core should only be used when absoluately necessary, and advised by 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 IO bottleneck issue on the system.
>
> **Tips:** We should only enable the debug logging level when:
>
> - We already know steps how to reproduce or trigger the issue symptom.
> - Or, we already know the issue will very likely come back very soon.
> - Or, OPSWAT team checked and confirmed there should be more useful information tracked with 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 MetaDefender Configuration).
2. Next, execute the scenarios requested by the support team, and collect the generated log files from the configured location (see how at How to Create Support Package?).
3. **After that the log level should be set back to `info`. In debug level the size of the logfile will increase significantly.**

For information on how to interpret the log files consult: Log Message Format

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:

none

```
[LEVEL] TIMESTAMP (COMPONENT) MESSAGE [msgid: MESSAGE ID]
```

Example:

none

```
[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.

Open Connection On PostgreSQL Server

Applicable to use-case where users want to use a remote 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 PostgreSQL data directory instead**)

none

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

In 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 new file `C:\Program Files\OPSWAT\MetaDefender Core\postgres\postgresql.conf`

(**If that's a separate PostgreSQL server then you can find it in PostgreSQL data directory instead**)

none

```
listen_addresses = '*'
```

In above example, it means to ask 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 PostgreSQL data directory instead**)

none

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

2.) Create new file `/usr/lib/ometascan/postgres/postgresql.conf`

(**If that's a separate PostgreSQL server then you can find it in PostgreSQL data directory instead**)

none

```
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:

none

```
ulimit -n
```

Check the used file descriptor count on a running process:

none

```
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`

none

```
fs.file-max = 65535
```

Add the following lines to `/etc/security/limits.conf`

none

```
*   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`

none

```
> 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:

none

```
kernel: TCP: request_sock_TCP: Possible SYN flooding on port 8009. Sending
cookies.  Check SNMP counters.
```

Check the `TIME_WAIT` sockets count:

none

```
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:

none

```
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`:

none

```
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`.

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]`

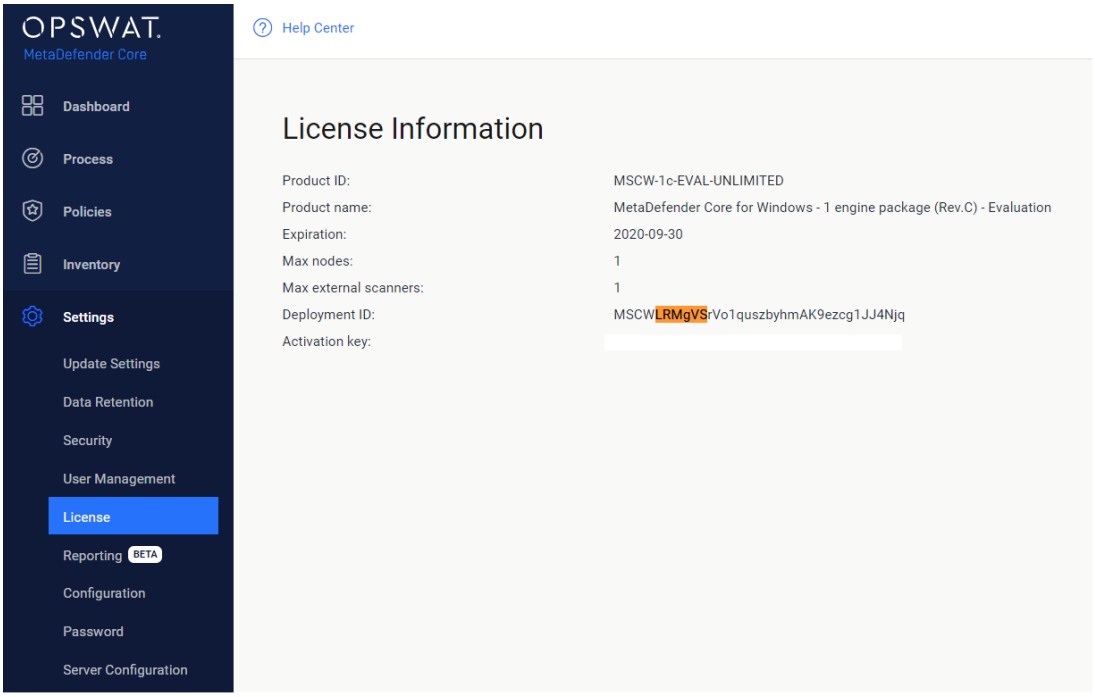
bash

```
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

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:

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)

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:

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)

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:

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

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:

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:

none

```
netsh int ipv4 show dynamicport tcp
```

Example output:

none

```
Protocol tcp Dynamic Port Range
---------------------------------
Start Port      : 1024
Number of Ports : 64511
```

To adjust default values, use command:

none

```
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

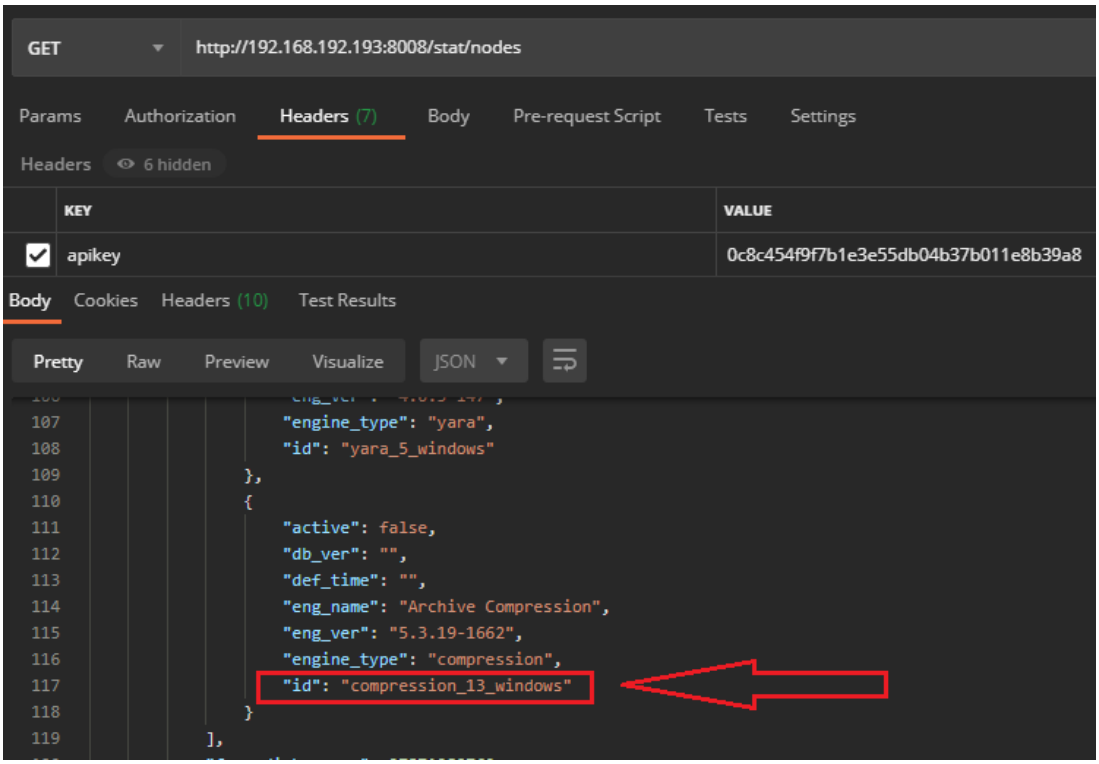
Starting from MetaDefender Core version 5.0.0, we have a new archive compression engine to handle archive sanitization feature on MetaDefender Core v5 whenever you have Deep CDR licensed, and in order to support customers not to use another product license when upgrading to v5, so we had to make a change 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:

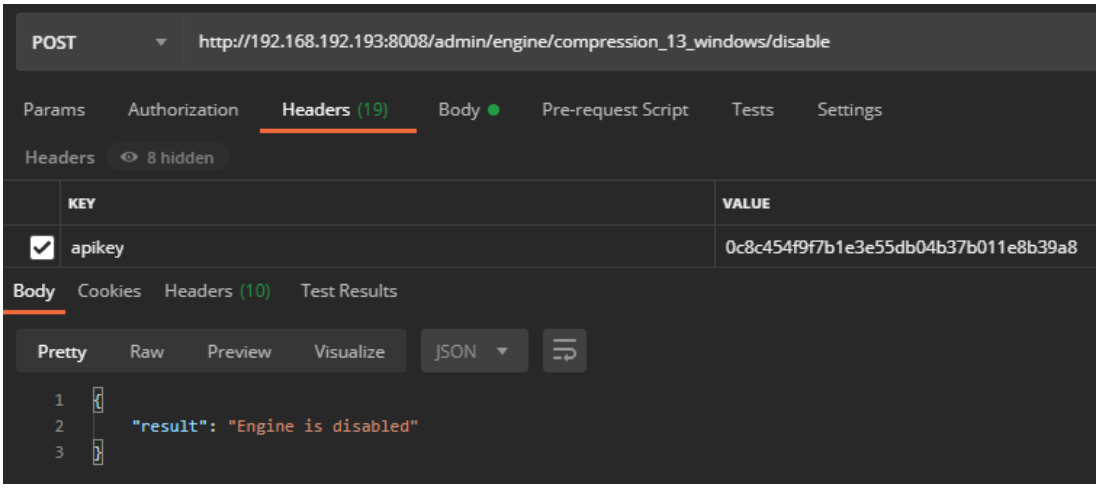
Retrieve exact engine ID of archive compression engine

- `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.

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):

none

```
“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 none

```
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

none

```
<MetaDefender Core installation folder>\ometascan-reset-ssl-conf.exe
```

Restart MetaDefender Core service upon done.

Linux

none

```
export LD_LIBRARY_PATH=/usr/lib/ometascan/

/usr/sbin/ometascan-reset-ssl-conf
```

Restart MetaDefender Core service upon done.

Customize 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.

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).

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:

bash

```
ometascan-custom-internal-pg-user [-c <PATH TO CONFIG DB>] -nu <NEW USERNAME> [-
np <NEW PASSWORD>]
```

or you can drop the current user:

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
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).
	• 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:

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:

none

```
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/

none

```
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.

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.

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.

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**
> The tool must run on the host where PostgreSQL server is running, and Visual C++ Redistribution 2015 or above is installed.

Usage:

bash

```
ometascan-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:

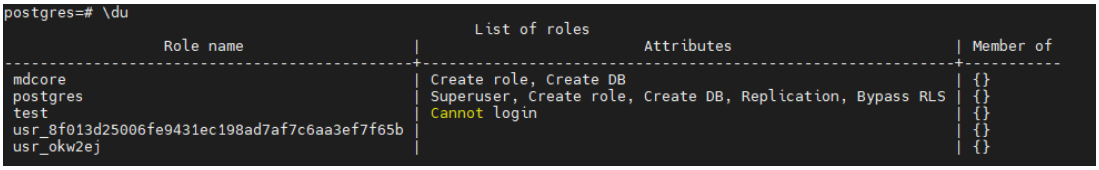
- -h (--help) instruction
- -h (--hostname) database server host or socket directory
- -p (--port) database server port

- -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

bash

```
ometascan-check-db-user-compliance -h 10.40.50.196 -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

bash

```
ometascan-check-db-user-compliance -h 10.40.50.196 -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

bash

```
ometascan-check-db-user-compliance -h 10.40.50.196 -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

```
postgres=# \du
                                   List of roles
                 Role name                  |                         Attributes                         | Member of
--------------------------------------------+------------------------------------------------------------+-----------
 mdcore                                     | Create role, Create DB                                     | {}
 postgres                                   | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 test                                       | Cannot login                                               | {}
 usr_8f013d25006fe9431ec198ad7af7c6aa3ef7f65b |                                                          | {}
 usr_okw2ej                                 |                                                            | {}
```

Example 1: PostgreSQL internal user with LOGIN privilege

bash

```
ometascan-check-db-user-compliance -h 10.40.50.196 -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

none

```
ometascan-check-db-user-compliance -h 10.40.50.196 -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. Otherwise, the tool will delete the instance name with data.
		`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 "fn2xjs"`

`--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

2. a. Default Windows path: `C:\Program Files\OPSWAT\MetaDefender Core\`
   b. 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:
   a. Windows: `ometascan-data-normalizer.exe`
   b. 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

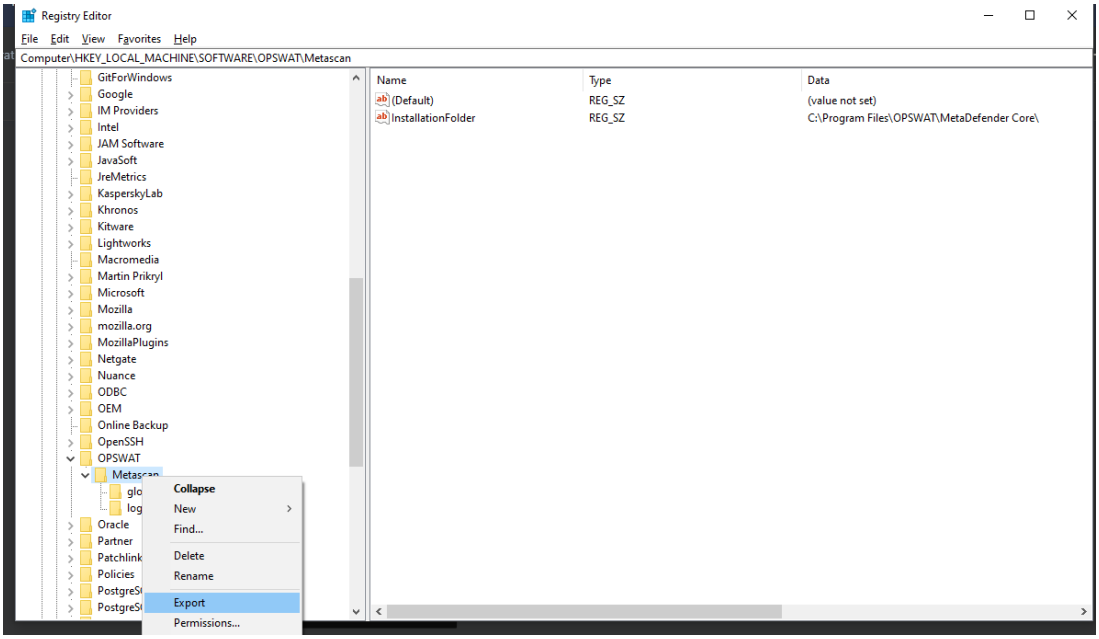
Steps by steps

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):
    a. Open the registry editor, go `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan`
    b. 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

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.

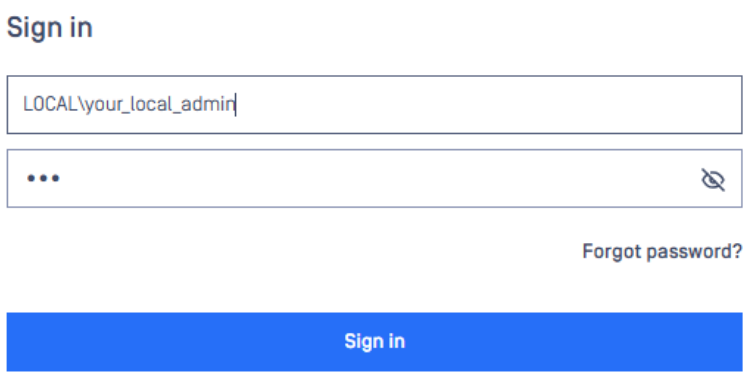
This issue occurs due to an encryption key incompatibility that may have affected your previous upgrade to 5.10.1/5.11.0. If the initial upgrade was completed with incompatible encryption keys, subsequent upgrade will encounter this authentication problem.

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.

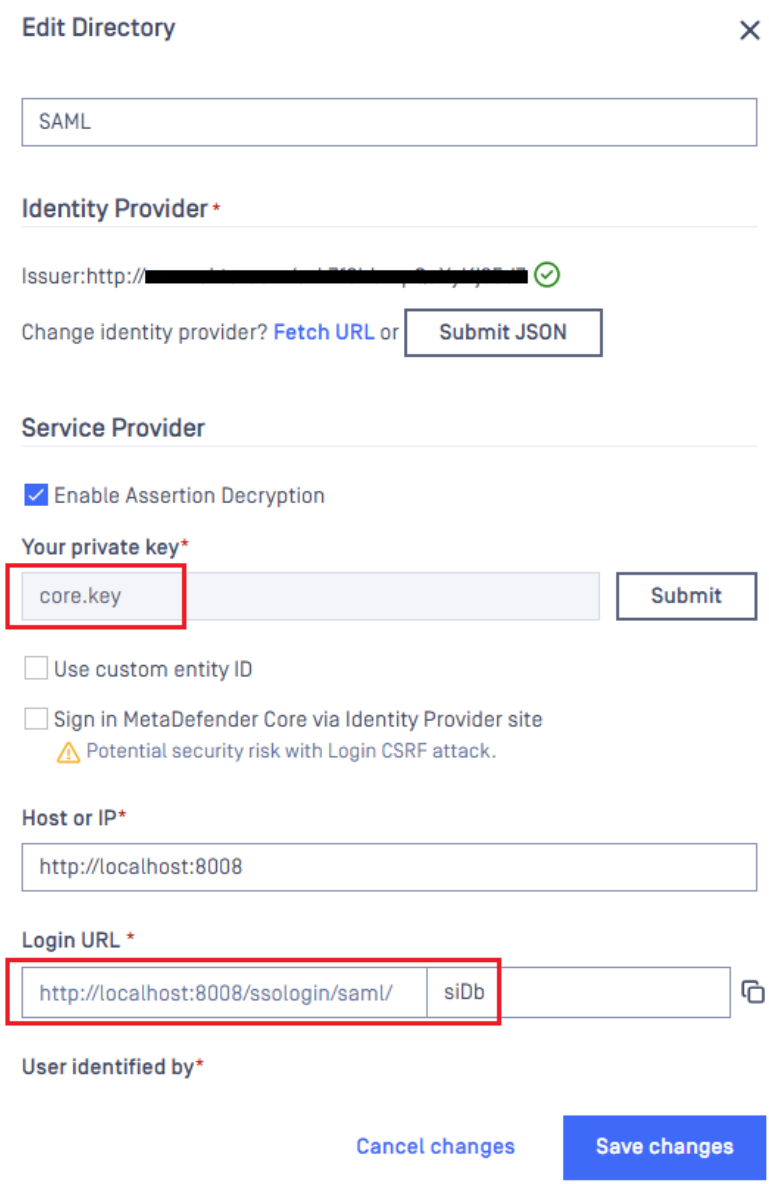
- **Log in using a LOCAL administrator account**.



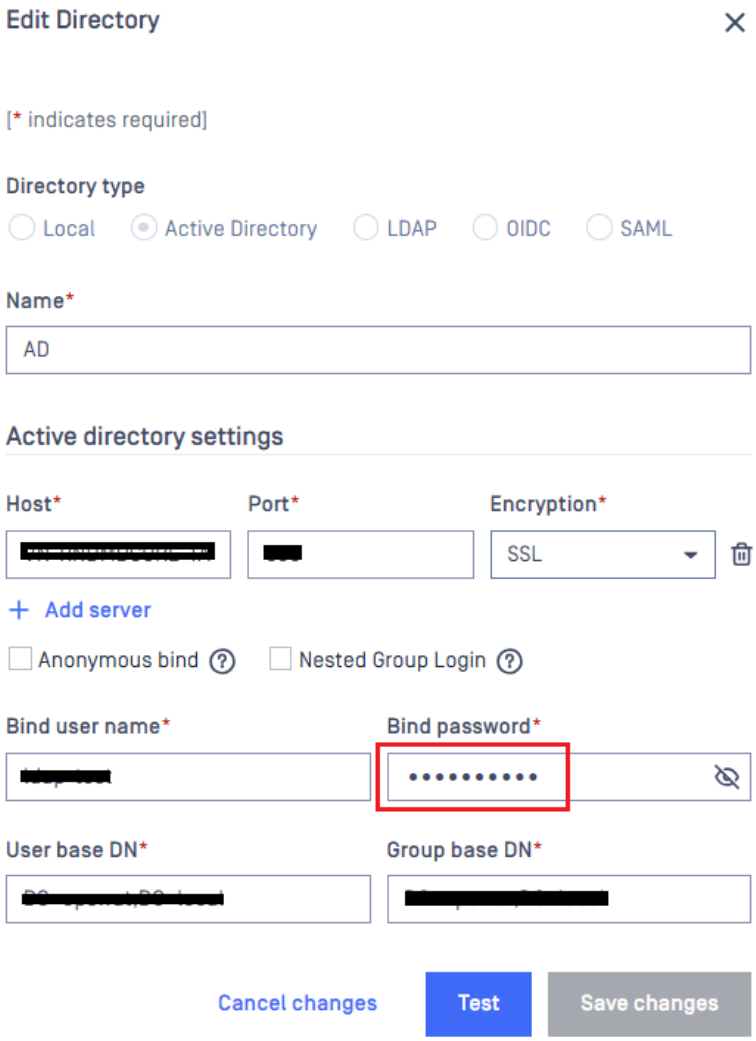
- Reconfigure your identity provider settings
- **OIDC directories**:
  - Re-enter your Client ID
  - Re-enter your Client Secret
  - Verify and update the Login URL



- 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.

API Reference

MetaDefender Core

API Version: v5.15.0

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: File Analysis).

   Note: The performance depends on:

   - number of engines
   - type of file to be scanned
   - MetaDefender Core's hardware

2. You have 2 ways to retrieve the analysis report:

   - **Polling**: Fetch the result with previously received data_id (GET /file/{data_id} resource) until scan result belonging to data_id doesn't reach the 100 percent progress_percentage: (Fetch analysis result)

   Note: Too many data_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.

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. 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

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
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
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

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		
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
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>"
}
```

1.3 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		

NAME	TYPE	DESCRIPTION
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)

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
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
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	
scan	array	
scanlog	array	

NAME	TYPE	DESCRIPTION
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	
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	
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
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		
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		

NAME	TYPE	DESCRIPTION
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
*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. **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 of which result can be retrieved by API Fetch Scan Result.

Note: Chunked transfer encoding (applying header Transfer-Encoding: Chunked) is **not supported** on /file 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.
filename	string		The name of the submitted file
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: 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.

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.
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

NAME	TYPE	EXAMPLE	DESCRIPTION
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>
downloadfrom	uri		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

NAME	TYPE	EXAMPLE	DESCRIPTION
global-timeout	integer		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

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
data_id*	string	Unique submission identifier. Use this value to reference the submission.

EXAMPLE:

```
{
 "data_id": "61dffeaa728844adbf49eb090e4ece0e"
}
```

STATUS CODE - 400: Callbackurl and/or apikey is invalid.

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

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

NAME	TYPE	EXAMPLE	DESCRIPTION
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.
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.

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)
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		
data_id	string	data identifier of the requested file
dlp_info	object	

NAME	TYPE	DESCRIPTION
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`
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]
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.
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`
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`

NAME	TYPE	DESCRIPTION
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_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.

NAME	TYPE	DESCRIPTION
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.
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	
score*	integer	Likelihood score of the file type

NAME	TYPE	DESCRIPTION
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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
opswatfilescan_info	object	
process_info	object	
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).
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.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing 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.

NAME	TYPE	DESCRIPTION
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.
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.

NAME	TYPE	DESCRIPTION
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	
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	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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

NAME	TYPE	DESCRIPTION
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
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

NAME	TYPE	DESCRIPTION
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
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	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: API key is invalid.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
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	
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://.

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/{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

- 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.
  - worst_data_id: data id of the file that has the worst result in the archive

- 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
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.
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		
data_id	string	data identifier of the requested file
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`
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]
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.
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`
redact	object	

NAME	TYPE	DESCRIPTION
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
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_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.

NAME	TYPE	DESCRIPTION
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.
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	
score*	integer	Likelihood score of the file type

NAME	TYPE	DESCRIPTION
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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
opswatfilescan_info	object	
process_info	object	
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).
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.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing 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.

NAME	TYPE	DESCRIPTION
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.
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.

NAME	TYPE	DESCRIPTION
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	
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	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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

NAME	TYPE	DESCRIPTION
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
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

NAME	TYPE	DESCRIPTION
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
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	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		

NAME	TYPE	DESCRIPTION
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>"
}
```

4.4 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	
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.
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	
limit_reached	boolean	Indicates if there're indeed over 100 blocked leaf files.
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>"
}
```

4.5 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		
data_id	string	data identifier of the requested file
dlp_info	object	

NAME	TYPE	DESCRIPTION
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`
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]
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.
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`
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`

NAME	TYPE	DESCRIPTION
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
extracted_files	array	
data_id	string	data identifier of the requested file
dlp_info	object	

NAME	TYPE	DESCRIPTION
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`
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]
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.
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`
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`

NAME	TYPE	DESCRIPTION
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	
data_id	string	data identifier of the requested file
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`
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`

NAME	TYPE	DESCRIPTION
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.
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`
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
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_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.

NAME	TYPE	DESCRIPTION
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.
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

NAME	TYPE	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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
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.
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.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
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.
yara_time	integer	YARA engine's processing time.

NAME	TYPE	DESCRIPTION
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	
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)

NAME	TYPE	DESCRIPTION
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	
scan_results	object	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.

NAME	TYPE	DESCRIPTION
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
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	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.
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.

NAME	TYPE	DESCRIPTION
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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.

NAME	TYPE	DESCRIPTION
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
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.
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.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
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	The waiting time for completely processing all nested files of an archive file.
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.

NAME	TYPE	DESCRIPTION
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	
scan_results	object	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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.

NAME	TYPE	DESCRIPTION
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
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	

NAME	TYPE	DESCRIPTION
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
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	object	
hits	object	

NAME	TYPE	DESCRIPTION
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.
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

NAME	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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
opswatfilescan_info	object	
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.
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.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.

NAME	TYPE	DESCRIPTION
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
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.
vul_time	integer	Vulnerability engine's lookup time.
wait_child_files_time	integer	The waiting time for completely processing all nested files of an archive file.
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	
action	enum	**ALLOWED:** `sanitized, removed` The type of action that was performed

NAME	TYPE	DESCRIPTION
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	
scan_results	object	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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

NAME	TYPE	DESCRIPTION
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
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

NAME	TYPE	DESCRIPTION
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
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	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		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

4.6 GET /hash/{md5|sha1|sha256|sha512}

Fetch Analysis Result By Hash

Retrieve analysis result by hash

REQUEST

PATH PARAMETERS

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)

NAME	TYPE	EXAMPLE	DESCRIPTION
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		
data_id	string	data identifier of the requested file
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`
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.
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`
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
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	

NAME	TYPE	DESCRIPTION
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.

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"
		}
		``

NAME	TYPE	DESCRIPTION
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_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.
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.
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	

NAME	TYPE	DESCRIPTION
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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
opswatfilescan_info	object	
process_info	object	
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).
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.

NAME	TYPE	DESCRIPTION
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing 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.
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.
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	
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.

NAME	TYPE	DESCRIPTION
scan_results	object	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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

NAME	TYPE	DESCRIPTION
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
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	

NAME	TYPE	DESCRIPTION
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
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	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

4.7 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

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.
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	
limit_reached	boolean	Indicates if there're indeed over 100 blocked leaf files.
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>"
}
```

4.8 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
request_time	string	A timestamp when the request has been made.

NAME	TYPE	DESCRIPTION
status_code	integer	What was the returned HTTP status code. * `200` - Callback was sent successfully * `403` - ContentAccessDenied. The access to the remote content was denied (similar to HTTP(S) error 401). * `404` - ContentNotFoundError. The remote content was not found at the server (similar to HTTP(S) error 404). * `408` - TimeoutError. The connection to the remote server timed out. * `503` - HostNotFoundError. The remote host name was not found (invalid hostname). * `520` - RemoteHostClosedError. The remote server closed the connection prematurely, before the entire reply was received and processed. * `444` - Other error types. Since MetaDefender Core 4.21.2, warning-level messages will be logged in case of Webhook failures. Check the following for detailed error descriptions. <details> 1 - the remote server refused the connection (the server is not accepting requests). 2 - the remote server closed the connection prematurely, before the entire reply was received and processed. 3 - the remote host name was not found (invalid hostname). 4 - the connection to the remote server timed out. 5 - operation canceled before it was finished. 6 - the SSL/TLS handshake failed and the encrypted channel could not be established. 7 - temporary network failure. 8 - the connection was broken due to disconnection from the network or failure to start the network. 9 - the background request is not currently allowed due to platform policy. 10 - while following redirects, the maximum limit was reached. 11 - while following redirects, a redirect from a encrypted protocol (https) to an unencrypted one (http) is detected. 99 - an unknown network-related error was detected. 101 - the connection to the proxy server was refused (the proxy server is not accepting requests). 102 - the proxy server closed the connection prematurely, before the entire reply was received and processed. 103 - the proxy host name was not found (invalid proxy hostname). 104 - the connection to the proxy timed out or the proxy did not reply in time to the request sent. 105 - the proxy requires authentication but did not accept any offered credentials (if any). 199 - an unknown proxy-related error was detected. 201 - the access to the remote content was denied. 202 - the operation requested on the remote content is not permitted. 203 - the remote content was not found at the server. 204 - the remote server requires authentication to serve the content but the credentials provided were not accepted (if any). 205 - the request needed to be sent again, but this failed for example because the upload data could not be read a second time. 206 - the request could not be completed due to a conflict with the current state of the resource. 207 - the requested resource is no longer available at the server. 299 - an unknown error related to the remote content was detected. 301 - the protocol is unknown. 302 - the requested operation is invalid for this protocol. 399 - a breakdown in protocol was detected (parsing error, invalid or unexpected responses, etc.). 401 - the server encountered an unexpected condition which prevented it from fulfilling the request. 402 - the server does not support the functionality required to fulfill the request. 403 - the server is unable to handle the request at this time. 499 - an unknown error related to the server response was detected. </details>
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		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
```

```
}
```

4.9 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
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>"
}
```

4.10 GET /file/converted/{data_id}

Download a sanitized file

Retrieve sanitized file based on the data_id

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>"
}
```

4.11 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.

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>"
}
```

4.12 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.

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>"
}
```

4.13 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>"
}
```

4.14 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

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>"
}
```

4.15 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.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*dataid	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`.

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.

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

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>"
}
```

4.16 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

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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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).
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

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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.

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

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).
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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).
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

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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>"
}
```

6.4 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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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).
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	

NAME	TYPE	DESCRIPTION
batch_id	string	The batch unique identifer
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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>"
}
```

6.5 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.
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.

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

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.

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	

NAME	TYPE	DESCRIPTION
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 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 content of file).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
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>"
}
```

7.9 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.

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		

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>"
}
```

7.10 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.

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	

NAME	TYPE	DESCRIPTION
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 headers, invalid request body).

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

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.

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).

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 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.

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	

NAME	TYPE	DESCRIPTION
OPTION:2	object	
number_hashes_removed*	integer	Total number of hashes deleted.

STATUS CODE - 400: Bad Request(e.g. invalid header, invalid request body).

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>"
}
```

7.13 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. 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)
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. 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.

NAME	TYPE	DESCRIPTION
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. 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
*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		

NAME	TYPE	DESCRIPTION
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		
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	

NAME	TYPE	DESCRIPTION
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		
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

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		

NAME	TYPE	DESCRIPTION
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	number	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	number	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	number	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
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

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		
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	

NAME	TYPE	DESCRIPTION
id	string	
display_name	string	
license	boolean	
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	

NAME	TYPE	DESCRIPTION
number_active_av_engines	integer	
required_engines	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: 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	
license	boolean	
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		
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

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/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>`

NAME	TYPE	DESCRIPTION
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>"
```

```
}
```

7.29 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.

NAME	TYPE	DESCRIPTION
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>"
}
```

7.30 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

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		

NAME	TYPE	DESCRIPTION
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 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		

NAME	TYPE	DESCRIPTION
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>"
}
```

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

Deactive 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.
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

NAME	TYPE	EXAMPLE	DESCRIPTION
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.

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	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		

NAME	TYPE	DESCRIPTION
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		
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	

NAME	TYPE	DESCRIPTION
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. STATS

Health check and statistics about MetaDefender Core usage.

13.1 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
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
 "err": "<error message>"
}
```

13.2 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
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

NAME	TYPE	DESCRIPTION
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

13.3 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.

NAME	TYPE	DESCRIPTION
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
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).
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		

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"
}
```

13.4 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.

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	
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 - 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>"
}
```

13.5 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)
info	enum **ALLOWED:** all, database, license, avg_duration_time, cpu_load, scan_queue, number_active_av_engines, engines	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 Note: Takes priority over additional_info parameter but is ignored if verbose is provided.

NAME	TYPE	EXAMPLE	DESCRIPTION
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

NAME	TYPE	DESCRIPTION
engines	array	
engine_id	string	Engine ID
required	boolean	This engine is mandatorily active
status	boolean	Status

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

API Reference

MetaDefender Distributed Cluster

API Version: v2.0.0

Developer Guide

This is the API documentation for *MetaDefender Distributed Cluster 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 Distributed Cluster API Gateway using REST API

MetaDefender Distributed Cluster API Gateway is used to submit files for analysis, retrieve scan results, manage file processing, download processed files, and manage file batches. OPSWAT recommends using the JSON-based REST API. The available methods are documented below.

Note: MetaDefender Distributed Cluster API doesn't support chunk upload, however is recommended to stream the files to MetaDefender Distributed Cluster API Gateway as part of the upload process.

File Analysis Process

MetaDefender Distributed Cluster is a system with multiple components that work together to utilize the power of multiple MetaDefender Core instances. The system is designed to handle large volumes of files and provide high throughput for file analysis. The system can be deployed in a distributed manner, allowing for horizontal scaling and load balancing across multiple MD Core instances.

Below is a brief description of the API integration flow:

1. Upload a file for analysis to MetaDefender Distributed Cluster API Gateway (POST /file), which returns the data_id: File Analysis.

2. The following method can be used to retrieve the analysis report:

   - **Polling**: Fetch the result with previously received data_id (GET /file/{data_id} resource) until scan result belonging to data_id doesn't reach the 100 percent progress_percentage: (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 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. 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-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

API

1. ANALYSIS

File analysis APIs

Submit each file to MetaDefender Distributed Cluster API Gateway 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.

Note: MetaDefender API doesn't support chunk upload. You shouldn't load the file in memory, is recommended to stream the files to MetaDefender Distributed Cluster API Gateway as part of the upload process.

`1.1 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 of which result can be retrieved by API Fetch Scan Result.

Note: Chunked transfer encoding (applying header Transfer-Encoding: Chunked) is **not supported** on /file API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
filename	string		The name of the submitted file
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		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"
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", "gzip". 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.

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)
global-timeout	integer		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

RESPONSE

STATUS CODE - 200: Successful file submission

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id*	string	Unique submission identifier. Use this value to reference the submission.

EXAMPLE:

```
{
 "data_id": "61dffeaa728844adbf49eb090e4ece0e"
}
```

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. 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."
}
```

1.2 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
  - 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.
  - worst_data_id: data id of the file that has the worst result in the archive
- 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
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-distributed-cluster/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.

RESPONSE

STATUS CODE - 200: Entire analysis report generated by MetaDefender Core

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	data identifier of the requested file
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`
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's worflow rule)

NAME	TYPE	DESCRIPTION
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]
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.
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`
redact	`object`	
result	`enum`	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
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

NAME	TYPE	DESCRIPTION
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_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.

NAME	TYPE	DESCRIPTION
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.
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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
opswatfilescan_info	object	
process_info	object	
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).
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.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing 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

NAME	TYPE	DESCRIPTION
parse_dgsg_time	integer	Digital signature analyzing time.
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.
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.

NAME	TYPE	DESCRIPTION
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	
scan_results	object	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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	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>"
}
```

1.3 GET /hash/{md5|sha1|sha256|sha512}

Fetch Analysis Result By Hash

Retrieve analysis result by hash

REQUEST

PATH PARAMETERS

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-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)

NAME	TYPE	EXAMPLE	DESCRIPTION
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		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		
data_id	string	data identifier of the requested file
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`
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.
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`
redact	object	
result	enum	**ALLOWED:** `redacted, not redacted, failed to redact` Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
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	

NAME	TYPE	DESCRIPTION
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.

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"
		}
		``

NAME	TYPE	DESCRIPTION
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_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.
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.
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	

NAME	TYPE	DESCRIPTION
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*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
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.
opswatfilescan_info	object	
process_info	object	
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).
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.

NAME	TYPE	DESCRIPTION
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing 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.
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.
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	
scan_results	object	
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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

NAME	TYPE	DESCRIPTION
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
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

NAME	TYPE	DESCRIPTION
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
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	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

1.4 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-distributed-cluster/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
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>"
}
```

1.5 GET /file/converted/{data_id}

Download Sanitized Files

Retrieve sanitized file based on the data_id

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-distributed-cluster/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>"
}
```

1.6 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.

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-distributed-cluster/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>"
}
```

1.7 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-distributed-cluster/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>"
}
```

2. AUTH

Authentication APIs

User authentication is done via username & password.

2.1 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
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 Distributed Cluster services 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>"
}
```

2.2 POST /logout

Logout

Destroy session for not using protected REST APIs.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/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>"
}
```

3. BATCH

Group the analysis requests in batches. Supported with endpoints: MetaDefender Distributed Cluster API Gateway.

3.1 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-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		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.
user-data	string		Name of the batch (max 1024 bytes, URL encoded UTF-8 string).

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>"
}
```


3.2 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-distributed-cluster/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.
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	

NAME	TYPE	DESCRIPTION
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	**ALLOWED:** `No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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).
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

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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>"
}
```

3.3 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-distributed-cluster/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.
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, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink` 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014` 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).
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

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	**ALLOWED:** No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, 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, Scan 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, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink 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, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 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>"
}
```

3.4 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-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
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:
881d22220c4ca0557d7c7d5c5794d53a8a2780997cd65b27b6e7f1c099a15de03dbcb5edbeaea7aafa6099fab37be
07017b39e3e3a7d66c550f44eb59a096c54d5b9555cb28198546fbec57c33b717751d333a09733d95dd876e2798d0
44c8caef828f4352b91f9a6d057253bb1a9461e0e0e0bf4313a80895998d645bebc81841ff3499589c80ffc4e8a19
0d1ec9b3e4126d86659d303b0e1f22d9289c9c4671d35532b55ad4620e048a78bb405b573897da63efdd5f036692c
934a82d9bdc9b9862e7fea5e8abeeb1444be0689d50373c5c0632484950c0fe0337ed5f91bdf26986f7cff8aa3431
bf4bc948fc127c16ba13ec679fe9f67e7586075c1f467454fa8cf40e9cd501291c95d862eb16f4477c17d1711294f
0ff2b3a1140bd53dbd1fbb0846af6062e9e4e2e1a09af3448503ed11e342164e535fc268bf7d8fbc28ed946cd2bb8
ea075f2295d2fa8392076d41608c3b5decf8fab3a5ec7de190f07583331e0517e5f361735cd59326622dc8b07b10a
464028de781a063e408f918c1d5534329140f4e4dc1a717d808d6784410410b00d36cb9a345f5bbc11fa1c58ee28f
8e7b863f3ea2c923ec5fb2ac29eaa4ddc0d6d9dfd3f16a97f207dc2858410a577c7f4a92ff01bad3229f5fcdb08e2
1df9869a113272aa9d96bfdfe8bfb3a50414c174e16a3504e5780c2718779b0757298546f287ef7ea86e67510d48a
8 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>"
}
```

3.5 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-distributed-cluster/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>"
}
```

OPSWAT.

