OPSWAT Metadefender™

Table of Contents

1. Installing or Upgrading Metadefender Core

1.1. System Requirements

Hardware requirements

Package	System RAM	Free Hard Drive Space	CPU
Metadefender Core 1	4 GB	16 GB + 1.5 GB / one million scan data	4 core
Metadefender Core 4	8 GB	32 GB + 1.5 GB / one million scan data	4 core
Metadefender Core 8	8 GB	32 GB + 1.5 GB / one million scan data	4 core
Metadefender Core 12	16 GB	32 GB + 1.5 GB / one million scan data	4 core
Metadefender Core 16	16 GB	32 GB + 1.5 GB / one million scan data	4 core
Metadefender Core 20	16 GB	32 GB + 1.5 GB / one million scan data	4 core
Metadefender 20+	Consult with OPSWAT Technical Support.		

Software requirements

- Operating System: Windows 7 / 8 / 8.1 / 10 / 2008 R2 / 2012 / 2012 R2
- Bitness: 64-bit only
- Windows Installer 4.5 or higher
- Windows hotfix for Windows Server 2008 R2 or Windows 7
- Beginning with version 3.11.2, Metadefender Core is only supported on Windows 64 bit operating systems.

Browser requirements for the Metadefender Core Management Console

- Internet Explorer 10 or later
- Safari 5.1 or later
- Firefox 3.5 or later
- Chrome

Additional installation of 3rd party framework/components

> The following framework/component may be shared with other applications. Uninstalling may result in unexpected behavior of other applications.

Name	Details	Optional
IIS express	• IIS express 7.5 (Metadefender Core 3.11.0 or older) • IIS express 8.0 (Metadefender Core 3.11.1 or newer)	REQUIRED
.NET framework	• 4 Client Profile • Extended	REQUIRED

Name	Details	Optional
Microsoft Visual C++ redistributable	• Microsoft Visual C++ 2005 Redistributable • Microsoft Visual C++ 2008 Redistributable • Microsoft Visual C++ 2008 Redistributable - x86 • Microsoft Visual C++ 2008 Redistributable - x64 • Microsoft Visual C++ 2010 Redistributable - x86 • Microsoft Visual C++ 2012 Redistributable (x86) • Microsoft Visual C++ 2013 Redistributable (x86)	REQUIRED
VMware Virtual Disk Development Kit	• version 5.1.1.1042608	REQUIRED

Additional installation of Windows services by Metadefender Core

Name	Service Name	Optional
Metadefender Generic Mail Agent	mdfExgEmailAgent	by default
Metascan	Metascan	by default
Metascan Helper	Metascan Helper	by default
Metascan ICAP	omsICAP	by default
Metascan Quarantine	omsQuarantine	by default
Metascan REST	omsRest	by default

Ports that must be available

component/service	port
mail agent service, quarantine service	8001
Quarantine REST	8000
Metascan REST	8008
mongodb	27018

Driver installation by Metadefender Core

- RAM drive (if you choose to install OPSWAT RAM drive)

Whitelisting requirements

- Any process running out of the Metadefender Core install directory should be whitelisted. It is best to exclude the folder from any real time protection

1.2. Installing Metadefender Core

Installation packages

Metadefender Core is available in five packages with different embedded anti-malware engines.

Package	Embedded Anti-Malware Engines
Metadefender Core 1	ClamAV
Metadefender Core 4	AhnLab, ESET, Avira, ClamAV
Metadefender Core 8	Everything in Metadefender Core 4 **PLUS** Total Defense, Quick Heal, Zillya!, Bitdefender
Metadefender Core 12	Everything in Metadefender Core 8 **PLUS** Ikarus, K7, nProtect, AVG

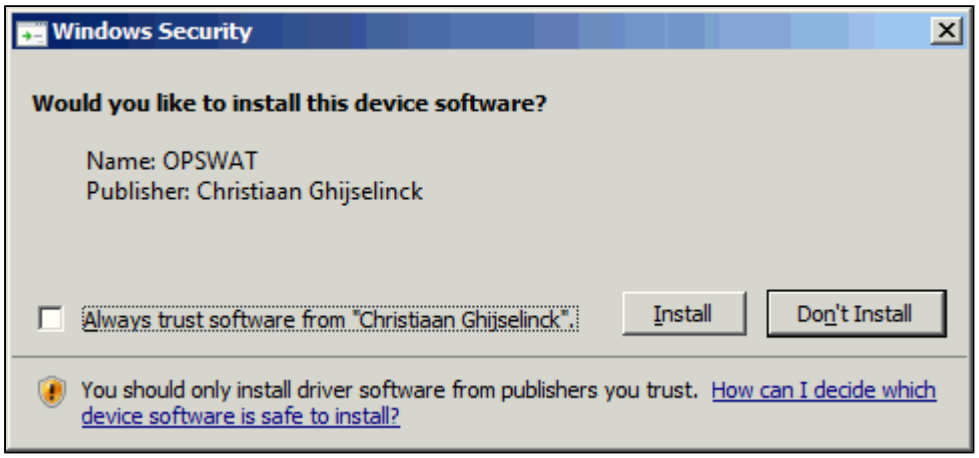
Package	Embedded Anti-Malware Engines
Metadefender Core 16	Everything in Metadefender Core 12 **PLUS** Kaspersky, F-Prot, Emsisoft, Virusblokada
Metadefender Core 20	Everything in Metadefender Core 16 **PLUS** McAfee, Sophos, NANOAV, Vir.IT eXplorer

In addition to the standard Metadefender Core installation, there are optional components that can be installed, based on your requirements.

Optional components

RAM drive

Sample integration code, documentation, and other useful utilities are available under Downloads at https://portal.opswat.com/. RAM drive installation will appear in a separate pop-up window. Click **Install** to continue the installation of the RAM drive component. For more information on installing the RAM drive, click here.



Installing Metadefender Core Using the Command Line

Note: Installing Metadefender Core from the command line requires Windows Installer 3.0 or higher.

Command line options

Important: Command line options for Metadefender Core 3.7 and later are different than the command line options available for versions 3.6 or older. Older command line options will not work with versions 3.7 or later.

The following command line options are available with Metadefender Core.
Note: The Packages column indicates which Metadefender Core package(s) each command line switch is available for: either 1, 4, 8, 12, 16, or ALL. All arguments are case sensitive. “Repair Metascan” is currently not supported for Metadefender Core version 3.7.

Command Line Option	Description	Example	Packages
/install	Install Metadefender Core	C:\Metadefender_Core.exe /install	ALL
/uninstall	Uninstall Metadefender Core and delete all files installed by Metadefender Core	C:\Metadefender_Core.exe /uninstall	ALL
/log <log-file-name>	Create installation log file	C:\Metadefender_Core.exe /log C:\omsinst.log	ALL
/silent	Run Metadefender Core installation silently	C:\Metadefender_Core.exe /silent	ALL
INSTALLLOCATION=<install-path>	Sets the install location for Metadefender Core	c:\Metadefender_Core.exe /i c:\Metascan.msi INSTALLLOCATION="c:\Metascan"	ALL
ADDLOCAL=<comma-separated-feature-list>	Install selected features only. Please see table below for list of features.	c:\Metadefender_Core.exe ADDLOCAL="RamdrvSupport,EsetEng,CaEng"	ALL
SERVICESTOPPED=1			ALL

Command Line Option	Description	Example	Packages
	Install Metadefender Core with the Metadefender Core Windows service in a stopped state. This will install Metadefender Core with automatic engine updates turned off. Not officially documented yet.	C:\Metadefender_Core.exe SERVICESTOPPED=1	
REST=0	This will avoid IIS Express, the webserver and all features that rely on the webserver to work. This includes all REST APIs, Web Management Console, Mail Agent, & Metascan Client.	C:\Metadefender_Core.exe REST=0	ALL
UPDATEOFF=0	This installs Metadefender Core with automatic engine updates turned off. Not officially documented yet.	C:\Metadefender_Core.exe UPDATEOFF=1	ALL

Note: Entries in the Feature Name column are case-sensitive.

Feature Name	Description	Package
RamdrvSupport	RAM Drive	ALL

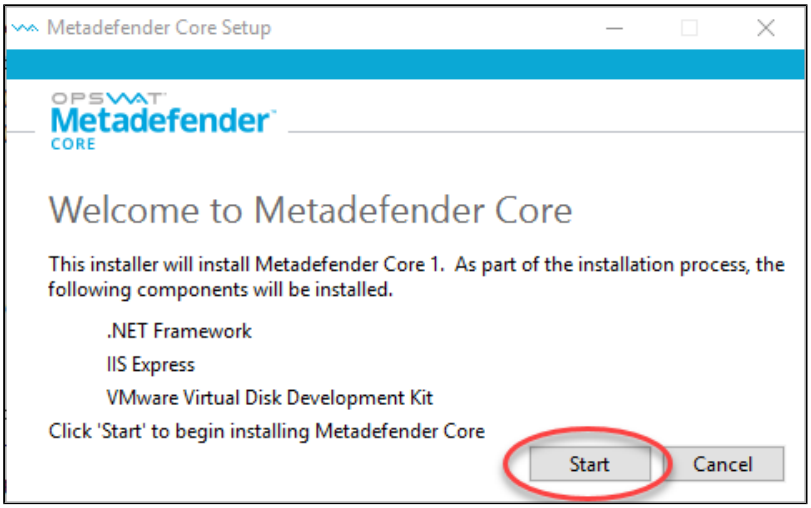
Feature Name	Description	Package
RestSvcE	Metascan REST Server	ALL
Docs	Documentation	ALL
MetascanClient	Metascan Client	ALL
ClamavEng	ClamAV scan engine	ALL
AhnlabEng	Ahnlab scan engine	4, 8, 12, 16, 20
AviraEng	Avira scan engine	4, 8, 12, 16, 20
EsetEng	ESET scan engine	4, 8, 12, 16, 20
CaEng	Total Defensescan engine	8, 12, 16, 20
ZillyaEng	Zillya! scan engine	8, 12, 16, 20
QuickhealEng	Quick Heal scan engine	8, 12, 16, 20
BitDefenderEng	BitDefender scan engine	8, 12, 16, 20
IkarusEng	Ikarus scan engine	12, 16, 20
IncaEng	nProtect scan engine	12, 16, 20
K7Eng	K7 scan engine	12, 16, 20
AVGEng	AVG scan engine	12, 16, 20
VirusBlokAdaEng	VirusBlokAda scan engine	16, 20
EmsisoftEng	Emsisoft scan engine	16, 20
FriskEng	F-Prot scan engine	16, 20
KasperskyEng	Kaspersky scan engine	16, 20

Feature Name	Description	Package
McAfeeEng	McAfee scan engine	20
NanoEng	NanoAV scan engine	20
SophosEng	Sophos scan engine	20
TGSoftEng	Vir.IT eXplorer scan eigne	20

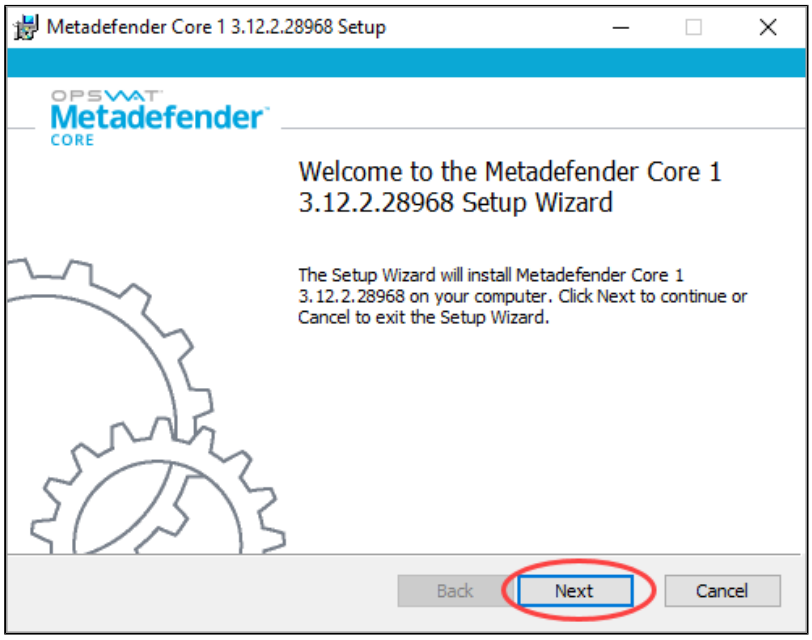
Installing Metadefender Core Using the Install Wizard

To install Metadefender Core using the Install Wizard, follow the steps below:**Note:** If you have a previously installed version of Metadefender Core already installed, you must uninstall it as instructed in Upgrading Metadefender Core.

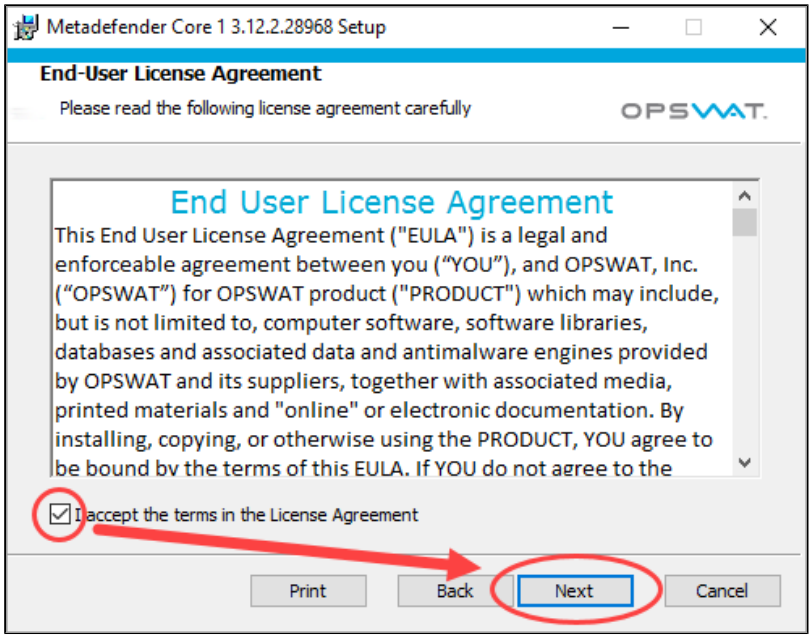
1. Locate the Metadefender Core Installation File. If this was not provided to you by OPSWAT, it can be found by clicking here and scrolling down to the Metadefender Core V3 section.
   Note: This file is named Metadefender_Core___.exe where is the Metadefender Core package you are installing, is the version of Metadefender Core, and is the build number.
2. Double-click on the desired Metadefender Core installation file to launch the Metadefender Core installer. If this does not work, right-click on the file and select **Open** .
3. The Welcome window is displayed. Click **Start** .



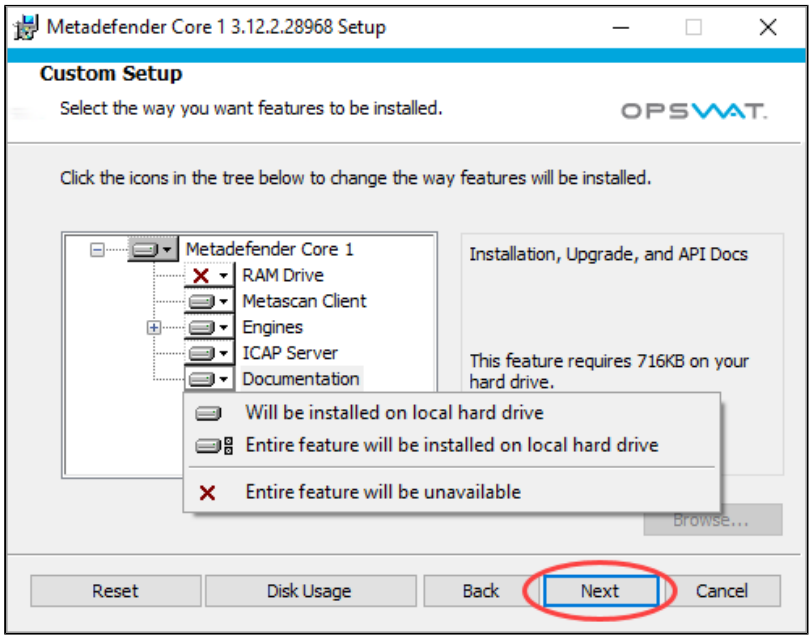
4. The setup Wizard is displayed. Click **Next** .



5. Accept the agreement and click **Next** .



6. Select the components you would like to add or remove from the installation. A drop-down menu is available to select whether to install the component or make it unavailable. After making your selections, click **Next** .



7. Click **Install** .
8. Click **Finish** to exit the Setup Wizard.
9. Click **Close** .

1.3. Metadefender Core Licensing

License types

Metadefender Core has the following types of licenses:

- **Metadefender Core (Required):** The main license, which must be activated in order to use the product and for scanning. Other licenses (below) are not activated until the main license is activated.
- **Metadefender Clients (Optional):** This license increases the maximum number of clients that can connect remotely to the server to perform scanning via REST APIs. With the main license only, you can use up to 1 customer-licensed engine.
- **Customer-Licensed Engines (Optional):** This license increases the maximum number of pre-installed anti-malware engines not embedded in Metadefender Core that can be used for scanning. With the main license only , you can use up to 1 customer licensed engine.
- **Custom Engines (Optional):** This license increases the maximum number of OPSWAT-provided custom engines that can be used simultaneously. With the main license only , you can use up to 1 custom engine.

Trial license

All types of Metadefender Core licenses (Metadefender Core, Metadefender Clients, Customer-Licensed Engines, Custom Engines) are activated through the Metadefender Core Management Console or through the License Manager CLI.

Note: This section applies only to serial key-based licenses. If you have not received a serial key from OPSWAT and have an older form of licensing, please contact support at OPSWAT Portal

Metadefender Core 1, 4 and 8 packages come with a 15-day evaluation license. For Metadefender Core 12, 16, and 20 trial licenses, please contact us at OPSWAT Portal.

Note: When the evaluation period starts, uninstalling and installing Metadefender Core does not extend the evaluation license. Metadefender Core licenses are strongly bound to the Volume ID, MAC address and NetBIOS name of the computer. Changing any of these attributes will disable Metadefender Core, even if you have not yet activated Metadefender Core with a license. If one of these values has changed, please contact support at OPSWAT Portal to receive a new license. For information on purchasing Metadefender Core licenses, click **Contact Sales** here.

Activating and Managing Metadefender licenses

In order to use Metadefender Core, you must have a license. You can activate and manage all types of your Metadefender Core licenses using any of the following methods:

- **Management Console:** You can use the Licenses tab on the Management Console to activate your license online or offline. You can also use it to check your license status.
- **License Manager Command Line Interface :** You can use the License Manager CLI to apply, update, or activate a Metadefender Core license (online or offline), or to check the license status of Metadefender Core or Remote Clients. For more on using this tool, see License Manager Command Line Interface.
- REST API for advanced user.

Online/Offline Activation

Metadefender Core licenses can be activated in two ways:

1. Online activation**:** Metadefender Core licenses can be activated online if the Metadefender Core server is connected to the Internet.
2. Offline activation**:** If the Metadefender Core Server is not connected to the Internet, Metadefender Core licenses can be activated offline through the OPSWAT Activation Portal.

Activating the Management Console

Offline License Activation

The following are required for offline activation:

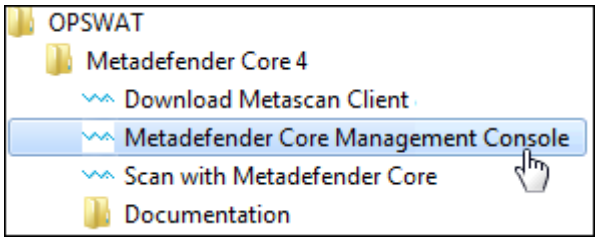
- A Metadefender Core Server
- A serial key that matches the Metadefender Core package installed on that Metadefender Core Server. Metadefender Core licenses can only be used with the package for which they were created (1, 4, 8, 12, 16, 20 and Custom). For example, a serial key for Metadefender Core 4 will not work with Metadefender Core 8.

You can use the following methods for offline license activation:

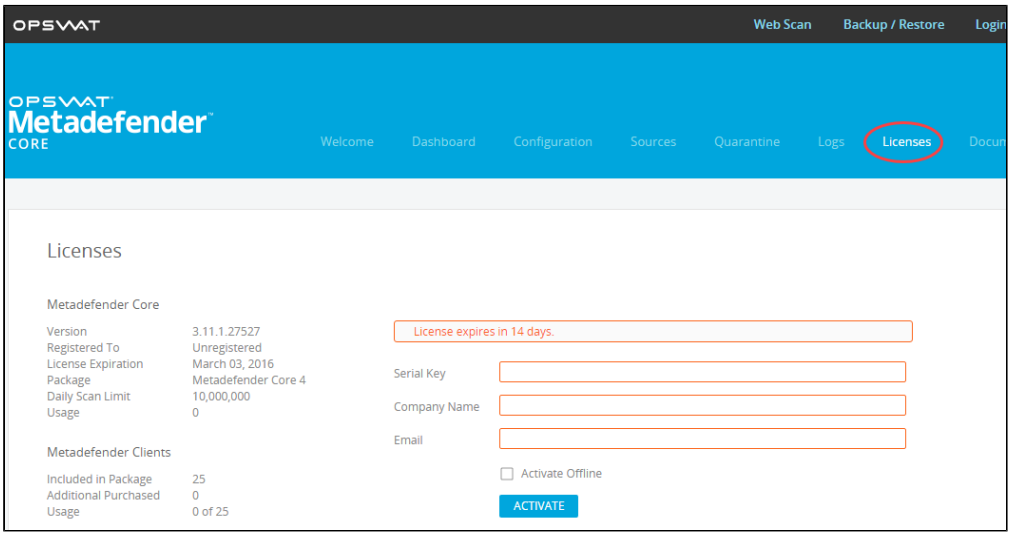
- The Metadefender Core Management Console included with the Metadefender Core installation
- The License Manager Command Line Interface (CLI)

Offline activation of licenses using the Metadefender Core Management Console

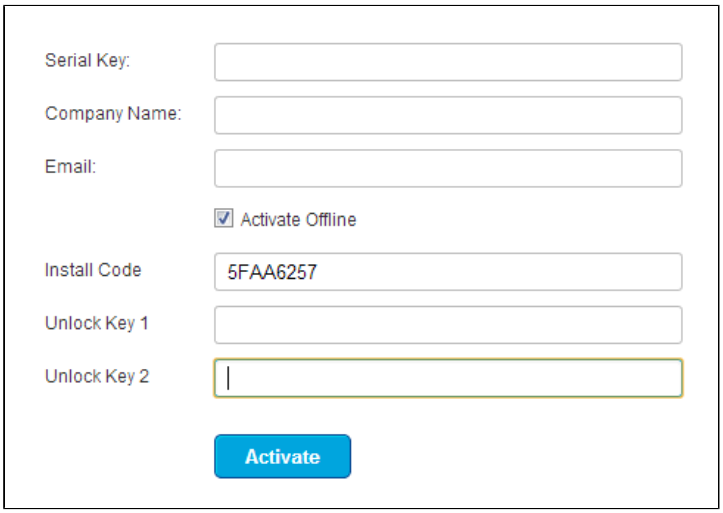
The Metadefender Core Management Console is installed by default and can be launched from the Start menu or in any browser with network access to the Metadefender Core Server.



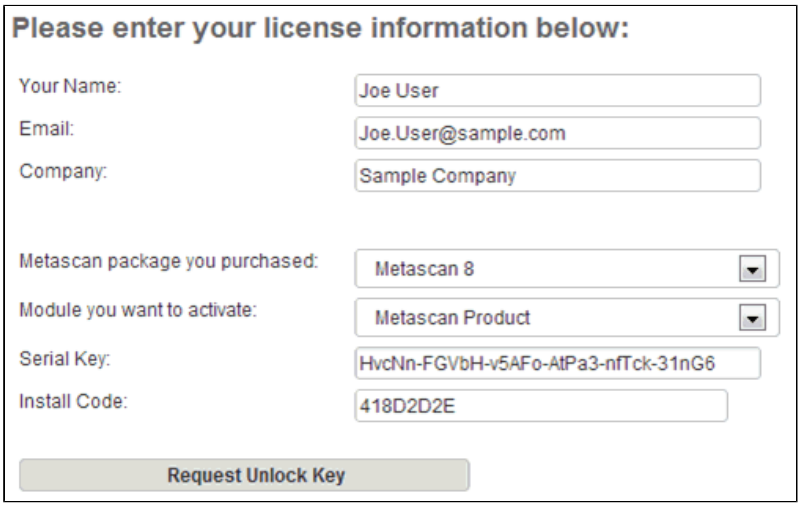
1. In the Metadefender Core Management Console, click the **Licenses** tab to check the current license information.



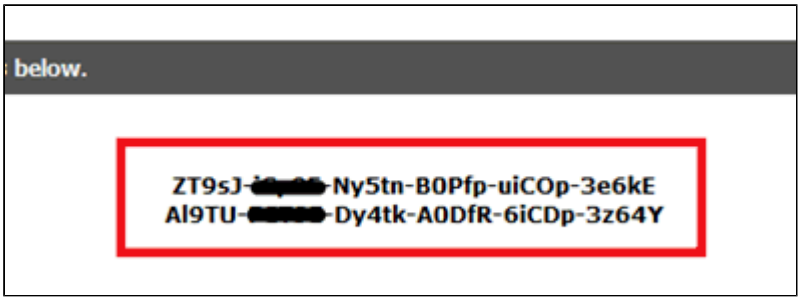
2. Select the **Activate Offline** option and copy or take note of the Install Code, which is needed to obtain the unlock keys.



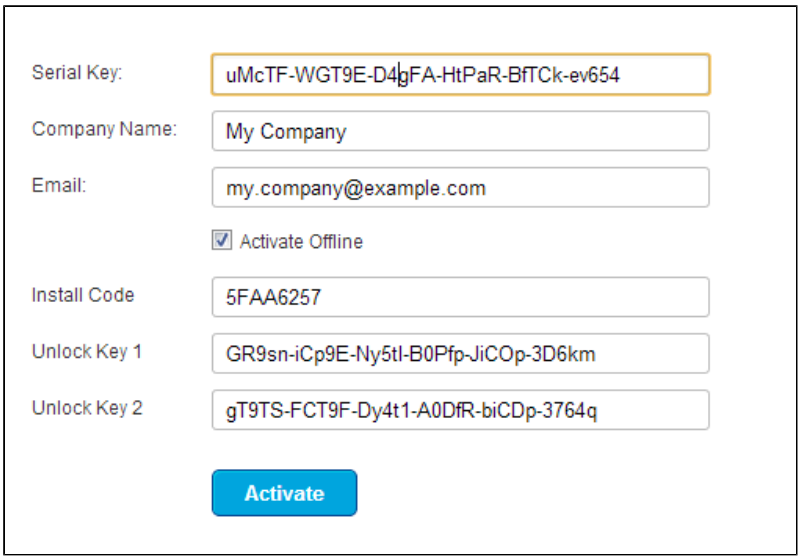
3. To obtain the unlock keys, visit the Metadefender Core offline activation portal at https://portal.opswat.com/activation.



4. Enter your company name and other contact information.
5. Enter your serial key and the Install Code from step 2.
6. Click **Request Unlock Key** , which displays a page with two unlock keys. Copy or take note of both of these.



7. Go back to the Metadefender Core Management Console. Enter the license key and unlock keys in the dialog box along with the company name.



8. Click **Activate** and confirm that the license expiration date has been updated and that it matches the Metadefender Core contract.

Offline activation of licenses using the License Manager Command Line Interface (CLI)

See the "activateoffline" command in License Manager Command Line Interface for more information.

Online License Activation

Note: The Metadefender Core Server must be connected to the Internet for online activation.

The following are required for online activation:

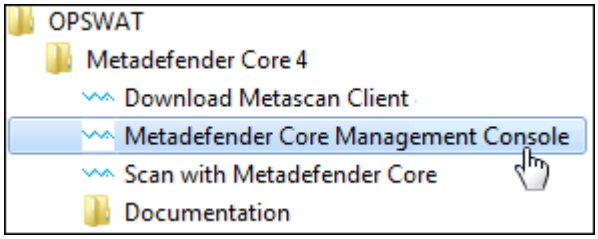
- A Metadefender Core Server
- A serial key that matches the Metadefender Core package installed on that Metadefender Core Server. Metadefender Core licenses can only be used with the package for which they were created (1, 4, 8, 12, 16, 20 and Custom). For example, a serial key for Metadefender Core 4 will not work with Metadefender Core 8.

You can use the following methods for online license activation:

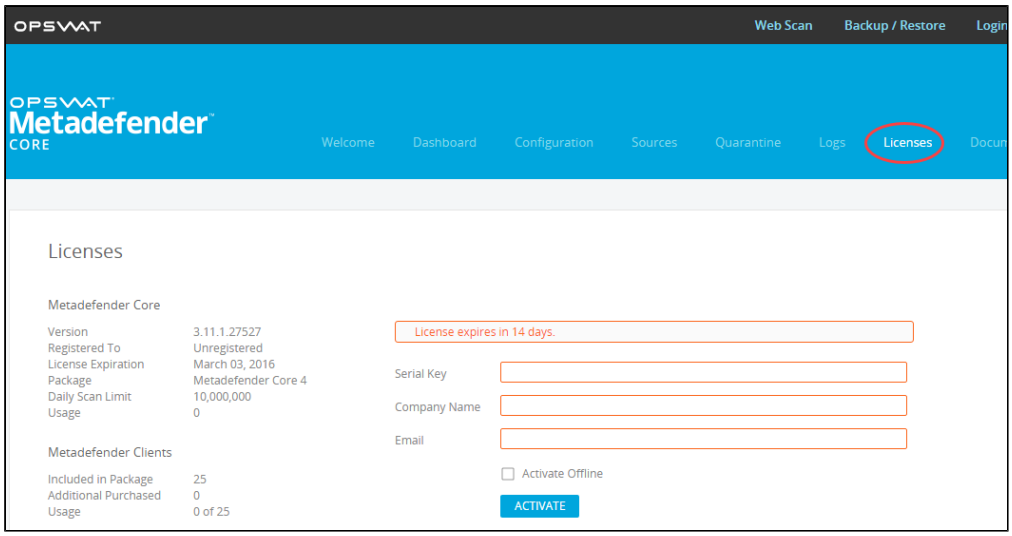
- The Metadefender Core Management Console included with the Metadefender Core installation
- The License Manager Command Line Interface (CLI)

Online activation of licenses using the Metadefender Core Management Console

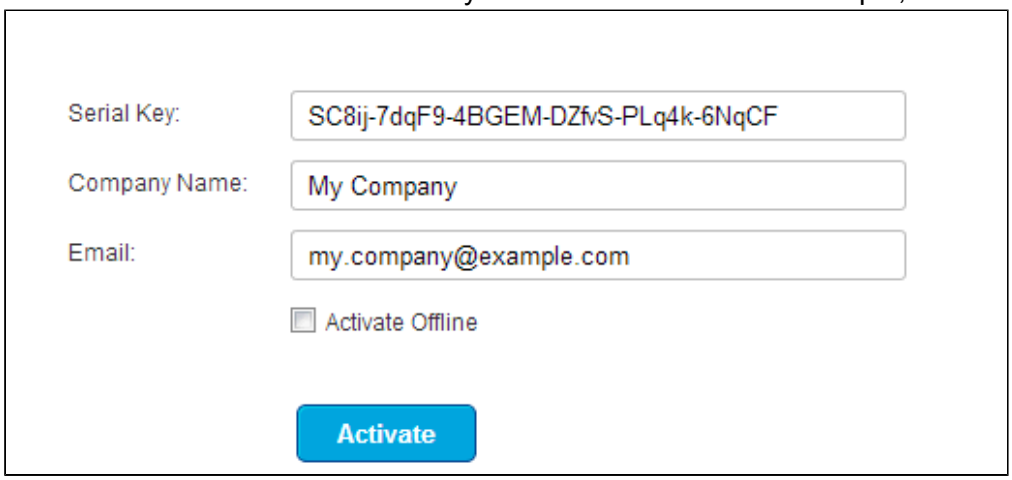
The Metadefender Core Management Console is installed by default and can be launched from the Start menu. You can also access the Metadefender Core Management Console directly from any machine with network access to the Metadefender Core Server at http://<metadefender core server>:8008/management.



1. In the Metadefender Core Management Console, navigate to the **Licenses** tab to check the current license information.



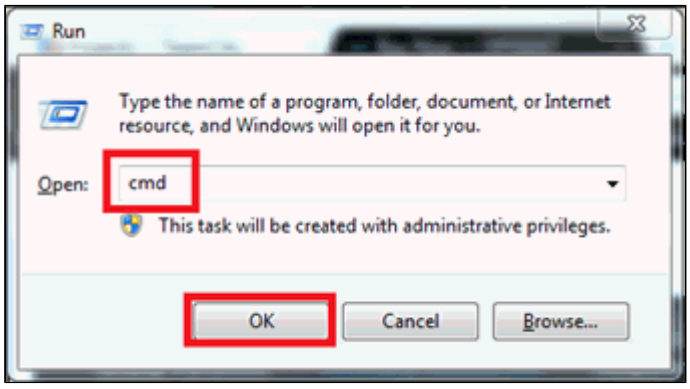
2. Enter the Metadefender Core serial key, company name, and contact email address for the instance of Metadefender Core you wish to activate. For example,



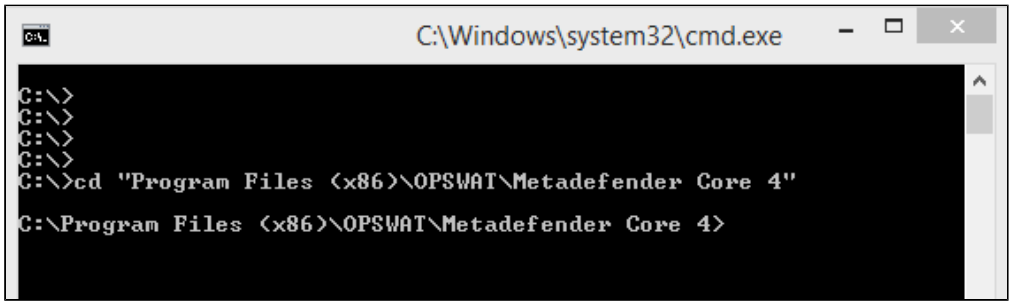
3. Click **Activate** to apply the license.
4. Confirm that the license activation was successful and that the license expiration date matches the expiration date on your Metadefender Core contract.

Online activation of licenses using the License Manager CLI

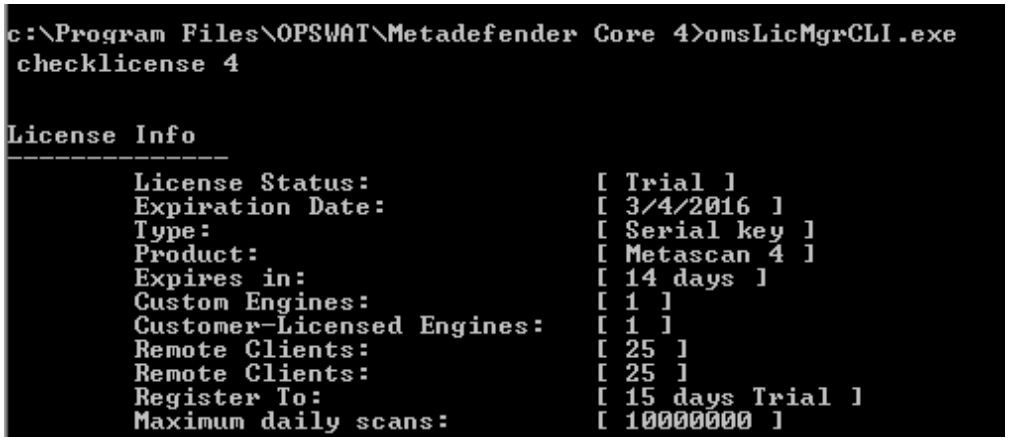
1. Open a command prompt by going to **Start > Run > CMD**.



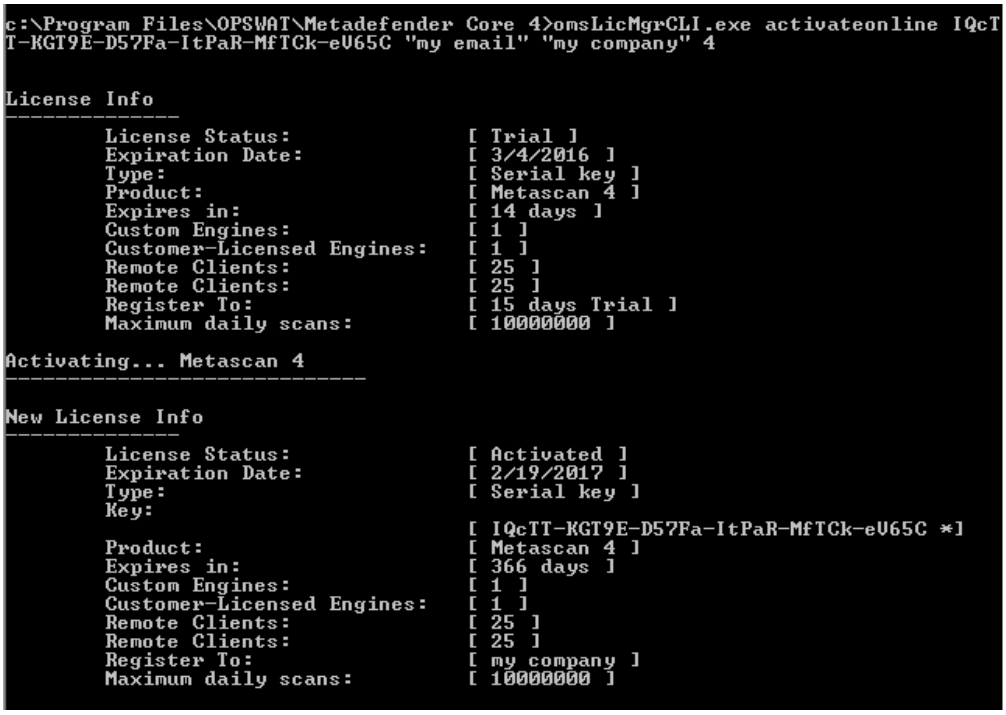
2. Change the working directory to the Metadefender Core install directory by entering cd <Metadefender Core install dir> (e.g., cd “c:\Program Files (x86)\OPSWAT\Metadefender Core 4”).



3. Verify that your installation of Metadefender Core is NOT already activated by entering omsLicMgrCLI.exe checklicense [package] (e.g., omsLicMgrCLI.exe checklicense 4).



4. Activate the license by entering omsLicMgrCLI.exe activateonline <license key> “<email>” “<company>” [package]



5. After Metadefender Core activation completes successfully, the updated license status is displayed. Verify that the license status listed is “Activated” and that the expiration date matches the expiration date on your Metadefender Core contract.

Note: If online activation is not successful, check to ensure the Metadefender Core Server’s Internet connection is working. If online activation is still unsuccessful, follow the steps for offline activation.

For more on using this tool, see License Manager Command Line Interface.

License Manager Command Line Interface

You can use the License Manager CLI to apply, update, or activate a Metadefender Core license. You can also use it to check the license status of Metadefender Core or Remote Clients. This tool (omsLicMgrCLI.exe) is installed in the Metadefender Core installation directory.

To use this tool, open a command prompt by going to **Start > Run > CMD**, and then enter start omsLicMgrCLI.exe

Command	Arguments	Description
help		Display each available command and its parameters.
checklicense	[package]*	Retrieve license status. *Optional.

Command	Arguments	Description
		Will pick a default based on directory that it is running in.
installcode		Retrieve install code which is required for generating unlock keys in order to activate offline.
activateonline	<serial key> <email> <company> [package]	Activate license with a serial key through the Internet. The Metadefender Core Server where the license is being activated must have an Internet connection.
activateoffline	<serial key> <company> <unlock key> <unlock key2> <package>	Activate license on a Metadefender Core Server without a direct connection to the Internet. This command requires a serial key and 2 unlock keys. Obtain the two unlock keys following the steps in Offline License Activation.
finishactivation	<unlock key> <unlock key2> [package]	Finish activation if the license status is “Activation Required”.

Argument values

Arguments	Value	Description
[package]	1 4 8	Optional parameter, if not specified, package of the currently installed Metadefender Core service will be used. 1: Metadefender Core 1 4: Metadefender Core 4

Arguments	Value	Description
	12 16 20 Metadefender Kiosk	8: Metadefender Core 8 12: Metadefender Core 12 16: Metadefender Core 16 20: Metadefender Core 20 Metadefender Kiosk
<serial key>	Serial number consisting of 6 blocks of 5 characters	The serial number is your proof of purchase. It is unique and will look like this: dO8uc-G1iC9-jOGeA-BqgEX-U71lD-0V1VX Although the serial key and unlock key have the same format, their properties are completely different. You must not mix them up and should keep them separate.
<email>	Email address	An email address of the contact who is responsible for Metadefender Core licenses. e.g. myemail@yourcompany.com
<company>	Name of company who owns the license	Used for the “company” property in Metadefender Core.
<unlock key> **<unlock key2>**	Serial number consisting of 6 blocks of 5 characters	Keys tied to a unique machine and serial key. These are obtained from OPSWAT through phone, email, or the web activation portal for offline activation.

Common license activation errors

The following are common errors encountered during activation and their potential causes.

Error	Possible Causes	Notes
INVALID KEY	An unrecognized serial key was entered.	A serial key is specific to the type of Metadefender Core package. For example, a serial key for Metadefender Core 4 cannot be used for Metadefender Core 8 or 12.

invalid unlock keys	An unrecognized unlock key was entered.	Unlock keys are tied to a specific serial key and machine where the Install Code was generated. Using an unlock key on a different system, or with a different serial key will result in this error.
No internet connection	Metadefender Core is unable to connect to the Metadefender Core license server.	Check your Internet connection or follow the steps for offline activation.
Other	In cases of other errors, an error code will be displayed along with this message.	Please contact OPSWAT support (support@opswat.com) with the error code.

1.4. Upgrading Metadefender Core

To upgrade from one Metadefender Core version to another, you must uninstall the version you currently have and install the newer version. Before doing this, you should back up your current configuration so that you can restore the settings into your upgraded version. You can back up these settings either by using your Metadefender Core Management Console, or by exporting the settings using the command line.

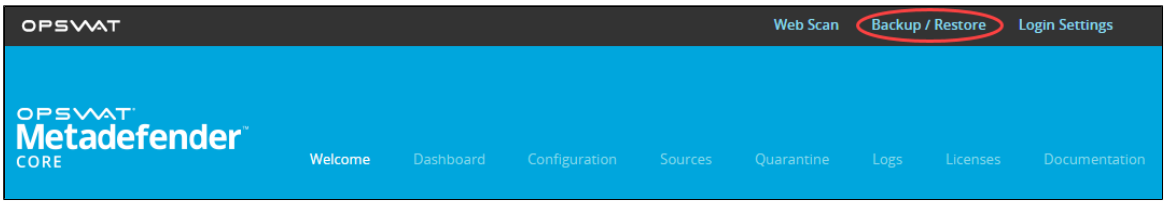
Your Metadefender Core license is automatically retained.

Backing up your current configuration using the Management Console

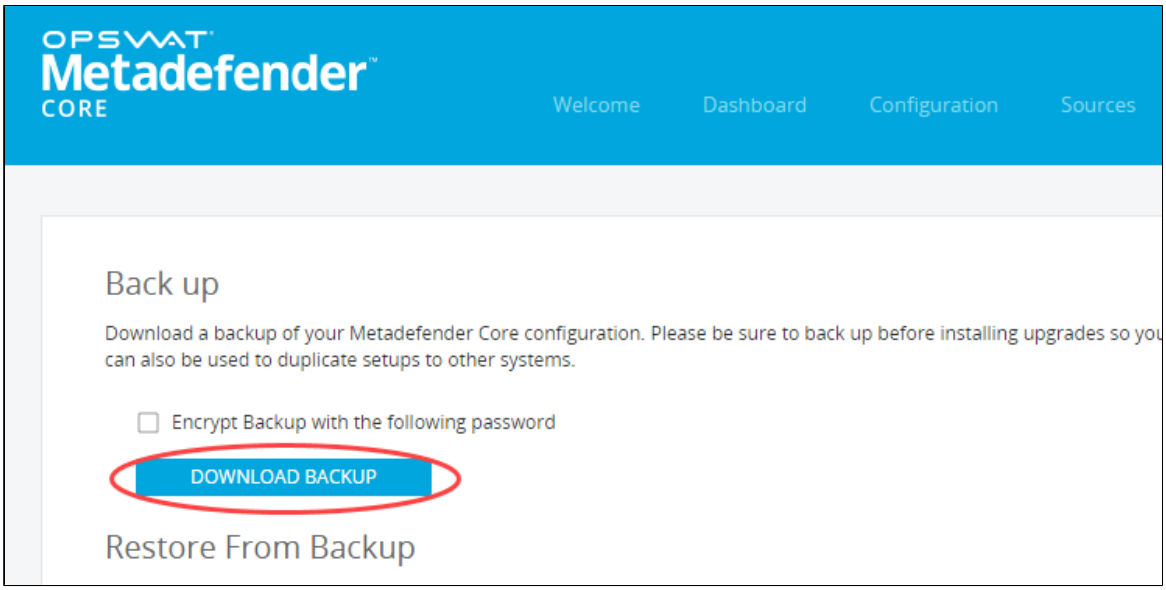
The preferred method of backing up your settings is by using the Metadefender Core Management Console. The following procedure downloads your current configuration settings in a file to a location of your choosing on your local machine.**Note:** The "Maximum File Size for files scanned through the web interface" setting (found on the Configuration > Scan Configuration page of the Metadefender Core Management Console) is NOT backed up and must be manually reconfigured, if necessary, after upgrading Metadefender Core.

1. Access the Metadefender Core Management Console.

2. Click the **Backup/Restore** tab.



3. Optionally, you can select **Encrypt Backup with the following password** to create a password for your download.
4. Click **DOWNLOAD BACKUP** .



5. Save the download to the location where you want to store the backup.

Exporting your current configuration using the command line

Export the Metadefender Core configuration.

```
omsConfig.exe export <directory> [<option>]
```

The following command exports the Metadefender Core Properties (e.g., max archive size or temporary directory), database settings, REST server settings (e.g., API keys or REST port), SYSLOG settings, and MongoDB settings into a zip file.

```
[<option>] is optional, it should be in the format  /pass=password
```

Exporting your scan results history

Optionally, you can export your scan results and then import them after you have installed the upgrade.

1. Open the Windows command line as an administrator.
2. Navigate to <Metascan installation folder>\Mongo\64+ (e.g. C:\Program Files (x86)\OPSWAT\Metascan 16\Mongo\64+).
3. Run the following command:

```
mongoimport.exe --host localhost --port 27018 --db metascan --collection files --file C:\output_files.json
```

"C:\output_files.json" is created, containing all scan recoreds in Mongo DB.

Installing the upgrade

1. Uninstall Metadefender Core using your Windows Control Panel.
2. Reboot your system (required only if you had RAM Drive installed).
3. Double-click the Metadefender Core installer file for the version that you are upgrading to.
4. Follow the prompts to install the upgrade.

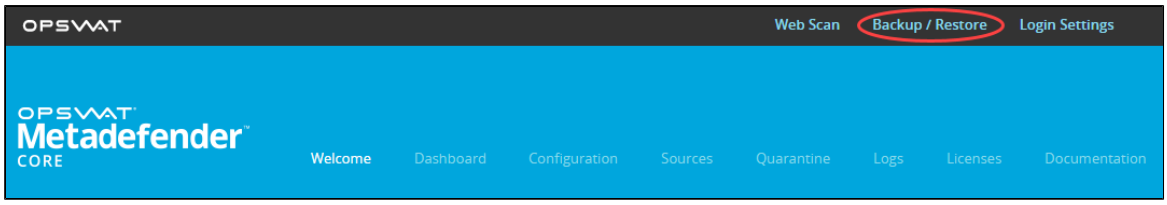
Note: If you are upgrading and your previously-set management console password stops working, perform the following steps:

1. Open the Windows Registry (regedit.exe).
2. Navigate to the Metadefender Core registry: hiveHKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\OPSWAT\Metascan.
3. Double-click on the rest_stat_admin_apikey string item in the list and erase any value data.
4. Restart the Metadefender Core services (Metascan and Metascan REST).

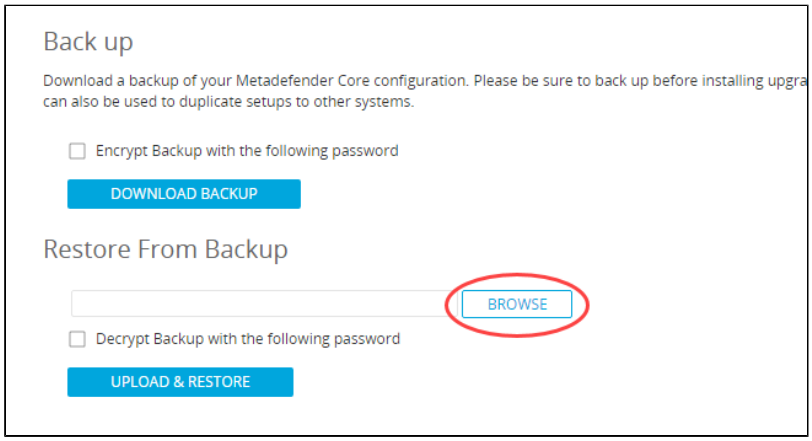
Restoring your previous configuration using the Management Console

After installing your upgrade, you can restore your previous configuration. The preferred method of restoring your settings is by using the Metadefender Core Management Console.

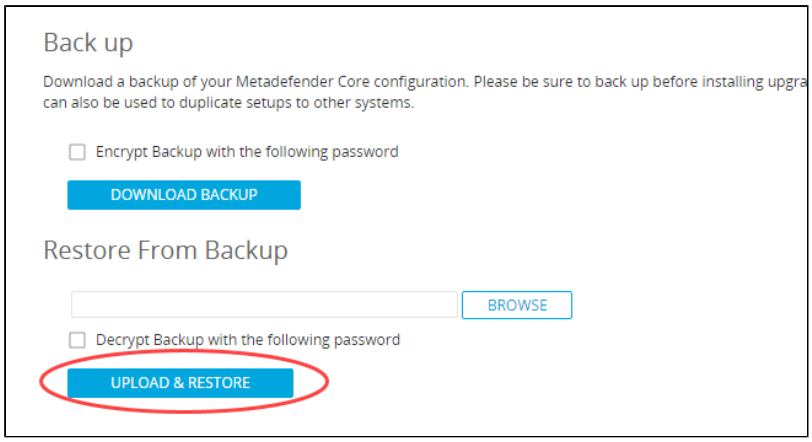
1. Access the Metadefender Core Management Console.
2. Click the **Backup/Restore** tab.



3. Click **Browse** , scroll to the locations where you stored the backup, and select the backup.



4. Click **RESTORE FROM BACKUP** .



Importing your previous configuration using the command line

If you exported your configuration using the command line, you can import those settings into your upgrade.

```
omsConfig.exe import <file path> [<option>]
```

omsConfig.exe is installed in the Metadefender Core Installation directory (e.g., c:\program files (x86)\opswat\Metadefender Core 8).

Importing your scan results history

If you exported your scan results history as described above, you can import the history after you have completed upgrading.

1. Download "Mongo import tool.zip".
2. Extract the zipped file to a "Mongo import tool" folder.
3. Open the Windows command line as an administrator.
4. Navigate to the "Mongo import tool" folder.
5. Run the following command:

```
mongoimport.exe --host localhost --port 27018 --db metascan --
collection files --file C:\output_files.json
```

Upgrading from Metadefender Core 3.5 or earlier

Metadefender Core 3.5 and earlier versions used a different licensing mechanism that is not supported in the current release of Metadefender Core. To obtain a new license key that will work with the current version of Metadefender Core, please contact support@opswat.com.

Backing Up Your Current Configuration

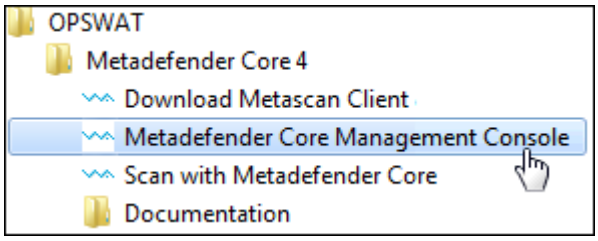
Installing the Upgrade

Restoring Your Previous Configuration

2. Configuring Metadefender Core

Metadefender Core Management Console

You can access the Metadefender Core Management Console through any browser that has access to the Metadefender Core system. A link to the Management Console (named "Metadefender Core Management Console") is added to the Start menu during installation.

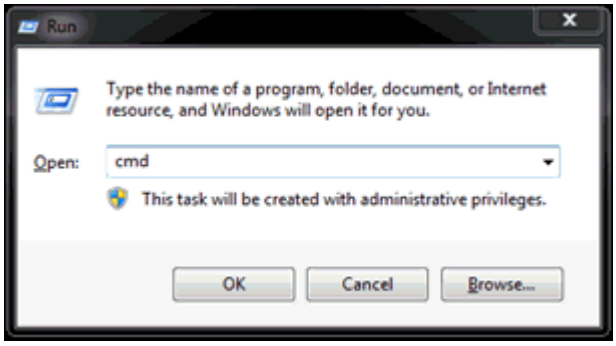


The Management Console can also be accessed directly in any web browser by going to http://<Metadefender Core Server>:8008/management. <Metadefender Core Server> is the name or IP address of the system where Metadefender Core is installed.

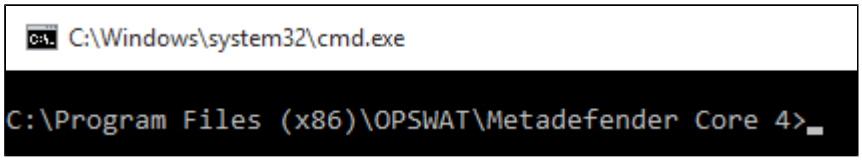
Command line utility

Metadefender Core also provides a command line utility that lets you run Metadefender Core operations, such as scanning a file or setting preferences/properties, etc.

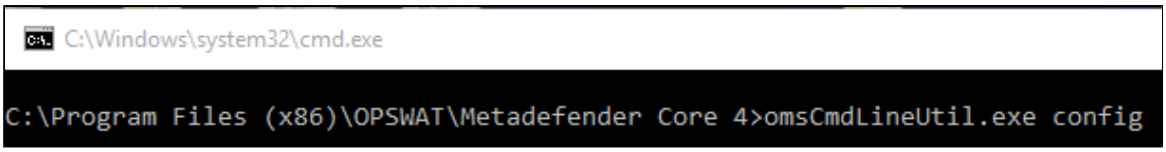
1. Go to **Start > Run > cmd** , and then click **OK** .



2. Change the directory to the directory where Metadefender Core is installed.



3. Run the config command with omsCmdLineUtil.exe to bring up a list of options to configure.



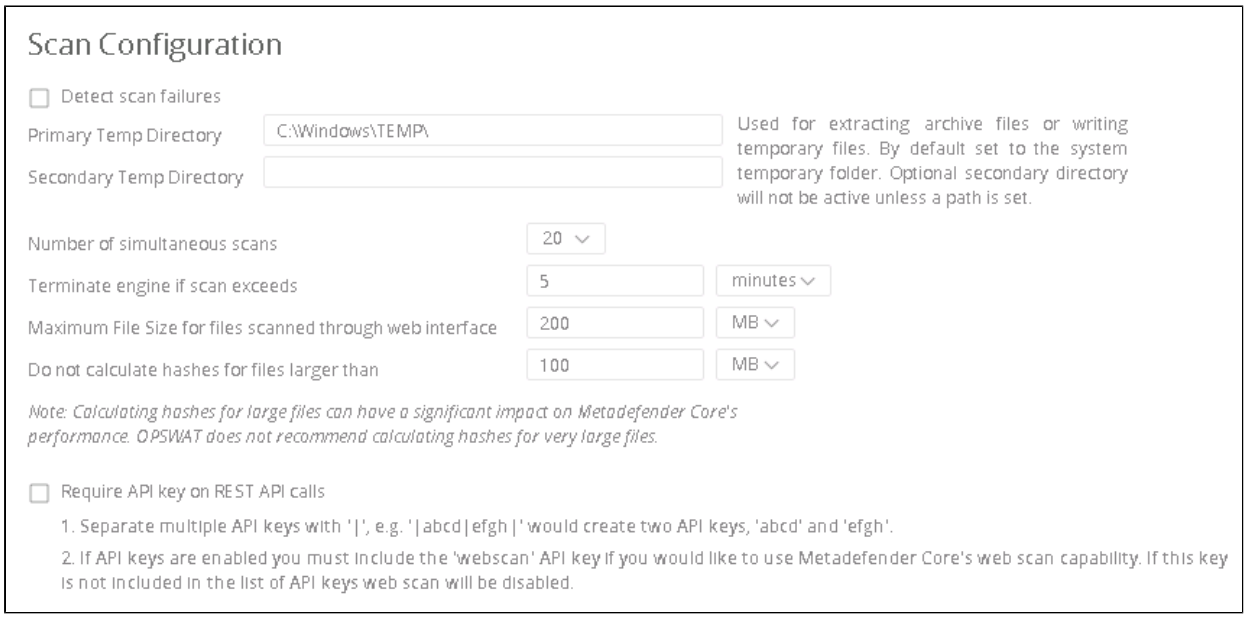
In the sections below, the command line utility commands are provided for each configuration setting.

omsConfig.ini

Additionaly, less commonly used options are set in the omsConfig.ini file located in the installation directory of Metadefender Core. Consult with OPSWAT Support before changing any settings other than proxy settings. You must restart the Metadefender Core service after making any changes to this file.

2.1. Global Scan Configuration

The scan configuration settings define the behavior of Metadefender Core when it is scanning files.



The following settings can be configured through the Scan Configuration page:

Property	Description	Default Value	CLI config
Detect Scan Failures	Specifies the number of engines that can fail to scan a file before the overall result to be 'failed to scan'. For example, if set to 1, then a single engine failing will not result in the file being labeled 'failed to scan' but if two or more engines fail the overall result will be 'failed to scan.'	0	dsf=<0-n-1>, where n is the total number of current AV engines.
Primary Temp Directory	Directory used to copy files for scanning.	RAMDisk (if installed) or the system temporary directory	td=<directory|secondary directory>
Secondary Temp Directory	Optional. Will be used only if the size of an extracted archive exceeds the space available in the primary directory.		
Number of simultaneous scans	The number of scans that can run at the same time.	20	tp=<0-400>
Terminate engine if scan exceeds	The amount of time that Metadefender Core will wait for an engine to complete scanning before it terminates the engine process.	5 minutes	te=<time in seconds>
Maximum File Size for files scanned through the web interface	The largest allowable size for files scanned through the Metadefender Core web interface.		

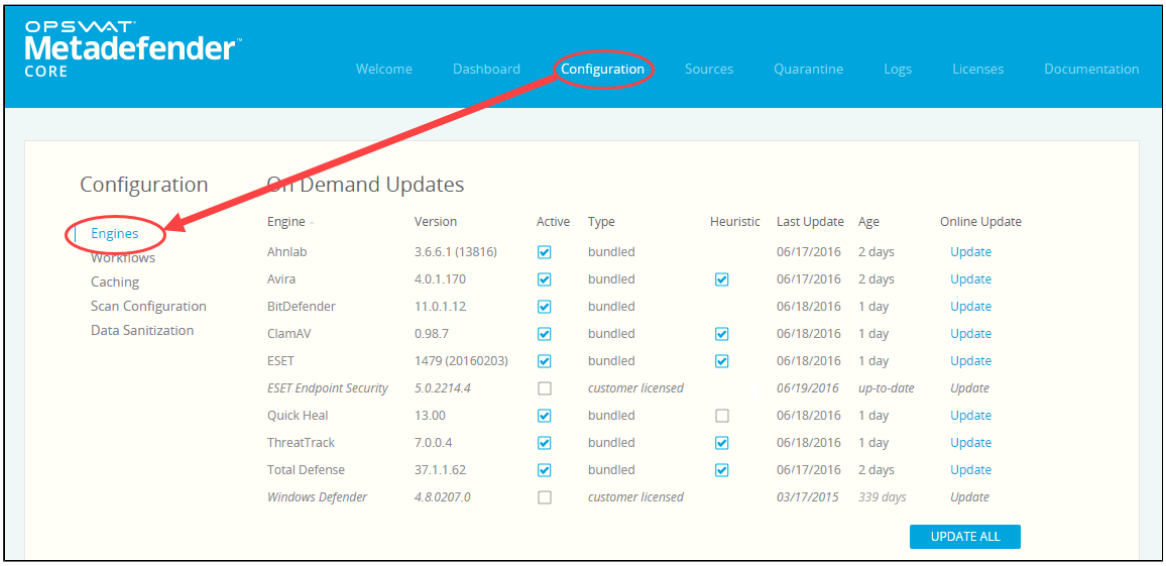
Property	Description	Default Value	CLI config
Do not calculate hashes for files larger than	The maximum size of files to compute the hash. Computing hashes for very large files can take a long time and can significantly affect Metadefender Core's performance.	100 MB	tfi=<size>

2.2. Engine Configuration

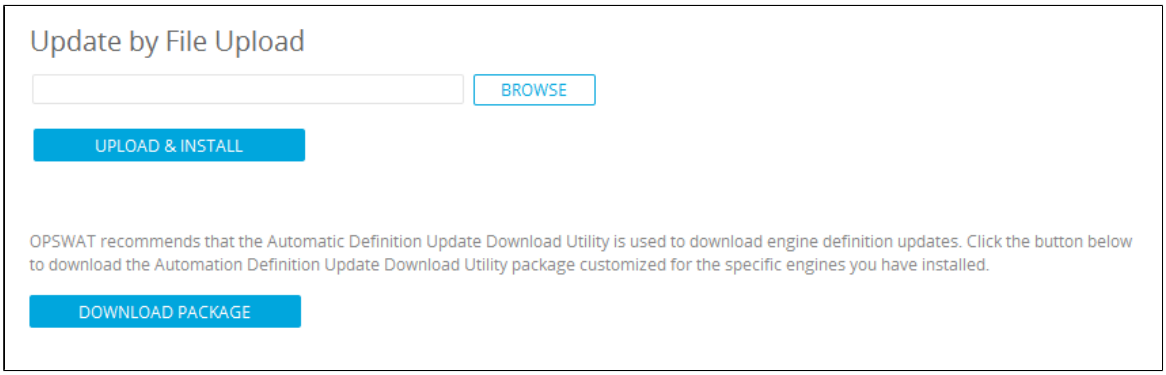
Applying Offline Updates

If your Metadefender Core server is deployed in an offline environment without connectivity to the Internet, you can deploy engine definition updates through the Update by File Upload section of the Engines Configuration page.

1. Access the Metadefender Core Management Console.
2. To configure the Engines section and apply offline updates, click the **Configuration** tab and then click **Engines** .



3. Scroll down to the Update by File Upload section.

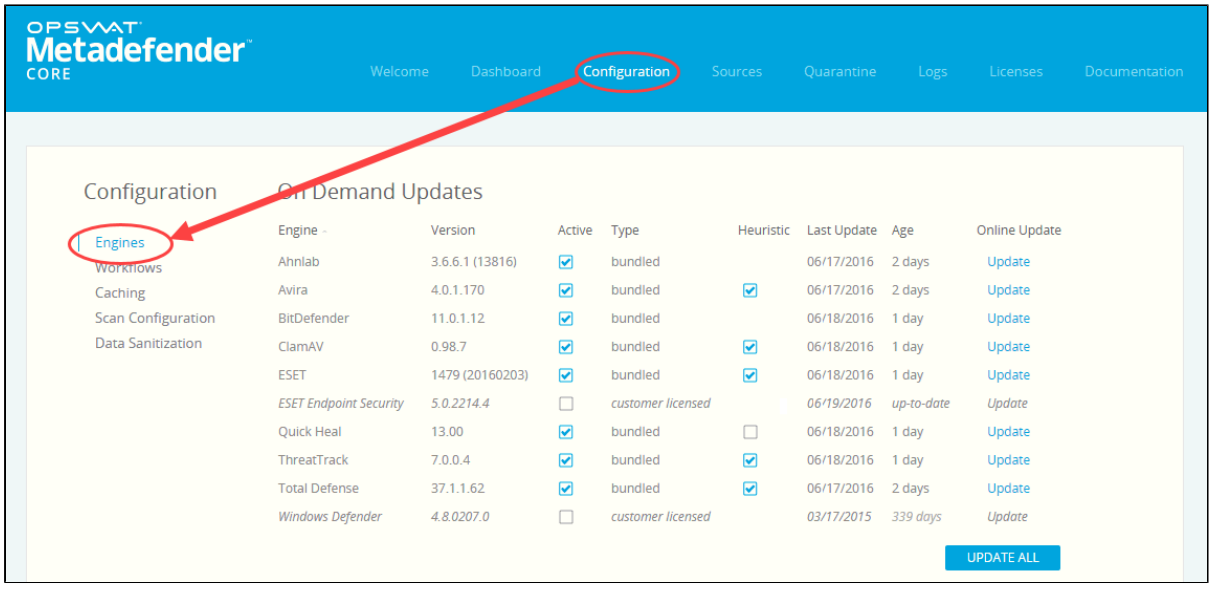


You can download these update packages from the OPSWAT Portal or with the Automatic Definition Update Download Utility. For more information on how to download these updates, please visit the OPSWAT Portal, click **Downloads** , and then click **Download Offline Definition Updates** .

Configuring Scan Engines

Using the Metadefender Core Management Console

1. Access the Metadefender Core Management Console.
2. To configure the Engines section and apply offline updates, click the **Configuration** tab and then click **Engines** .



All bundled engines as well as custom and customer licensed engines that can be used in Metadefender Core are listed on this page. Each engine can be activated or deactivated by selecting the checkbox and then clicking **Apply** at the bottom of the page.

If applicable, an engine's heuristic scanning feature can be activated or deactivated as well. These changes take effect after the Metadefender Core service has been restarted.

Any customer-licensed engine will not be enabled by default for Metadefender Core 4, 8, 12, 16, and 20 packages. You must enable customer-licensed engines before they can be used by Metadefender Core to process scan requests.

Using the command line interface

If you would like to enable or disable engines from the command line interface, run the following commands from the Metadefender Core installation directory.

Retrieving the list of available engines

```
omsCmdLineUtil.exe getinfo
```

Enabling an engine

```
omsCmdLineUtil.exe config in=<engine(s) to be included>
e.g., omsCmdLineUtil.exe config in="eset scan engine|avira scan
engine"
```

Disabling an engine

```
omsCmdLineUtil.exe config ex=<engine(s) to be excluded>
e.g., omsCmdLineUtil.exe config ex="eset scan engine|avira scan
engine"
```

If you are enabling or disabling multiple engines, separate the engine names with a '|' character.

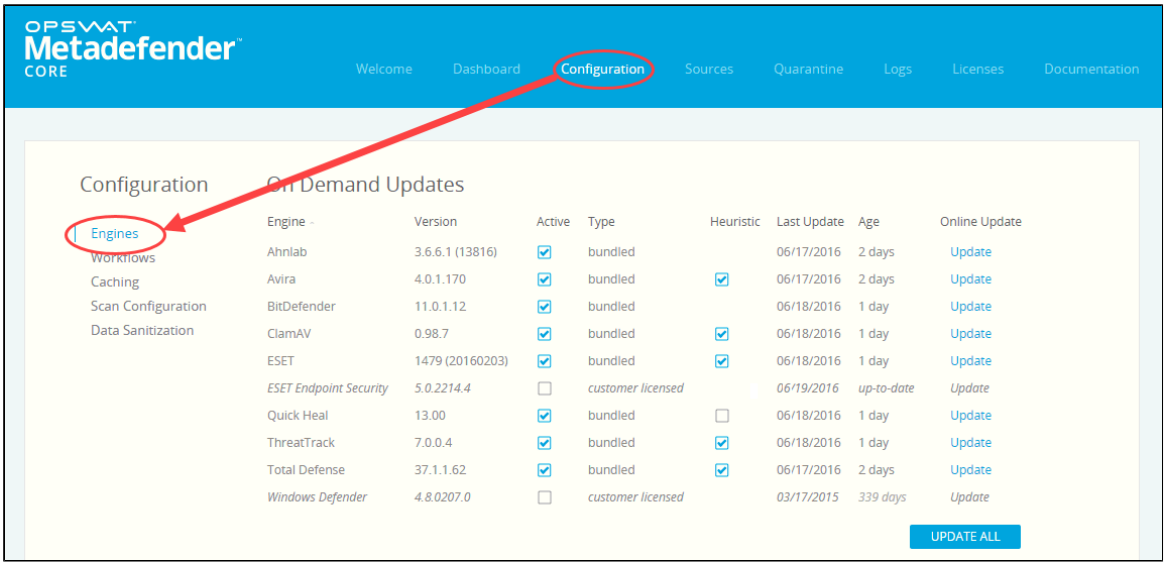
Supporting customer-licensed engines

```
omsCmdLineUtil.exe config sp={0|1}
e.g., omsCmdLineUtil.exe config sp=1
```

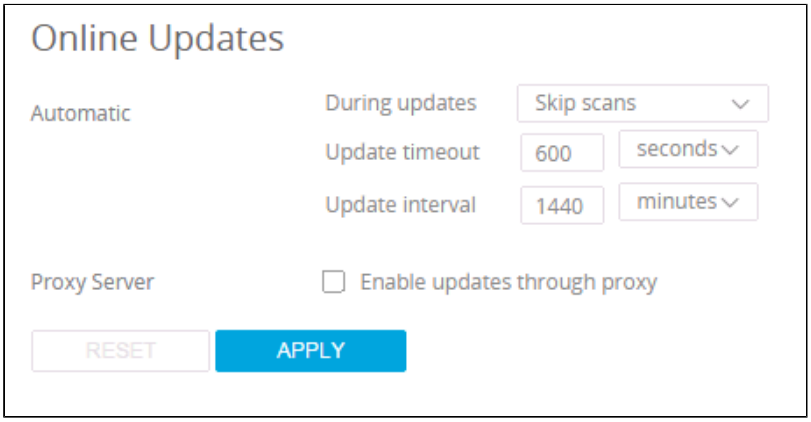
This property determines whether Metadefender Core will attempt to use antivirus engines that are installed on the system outside of Metadefender Core. These engines will often have reduced performance compared to embedded engines and may also reduce overall Metadefender Core performance. By default, customer licensed engines will be enabled in Metadefender Core 1 and disabled in Metadefender Core 4, 8, 12, 16, and 20 packages.

Online Update Configuration

1. Access the Metadefender Core Management Console.
2. To configure the Online Updates section, click the **Configuration** tab and then click **Engines** .



3. Scroll down to Online Updates.



4. Select your settings based on the following tables, and then click **Apply** .

The following settings apply to Metadefender Core's automatic engine definition update functionality:

Property	Description	Default Value	CLI config
During Updates	Specifies scanning behavior when an engine is in the process of having its definitions updated. Scans can configured to either be skipped or paused during updates.	Skip Scans	(Pause) wu=<0|1> (Skip) su=<0|1>
Update Timeout	Specifies how long Metadefender Core should wait for an update to happen before timing out.	600 seconds	to=<time in milliseconds>
Update Interval	Specifies how often engines should be updated.	1440 minutes (1 day)	um=<time in minutes>

Proxy settings can also be defined in the omsConfig.ini configuration file (located in the Metadefender Core installation directory). Proxy settings defined in omsConfig.ini will only be used if the proxy configuration is not set in the Metadefender Core Management Console.

Property	Description	Default Value	omsConfig.ini Property
Enable Updates Through Proxy	Specifies whether the Metadefender Core server will be retrieving engine updates through a proxy server.	disabled	enable_proxy
Server Address	The proxy server to use.		proxy_server
Port	The proxy server port to use.		proxy_port
Username	If authentication is required, the username for the proxy server.		proxy_user
Password			proxy_password

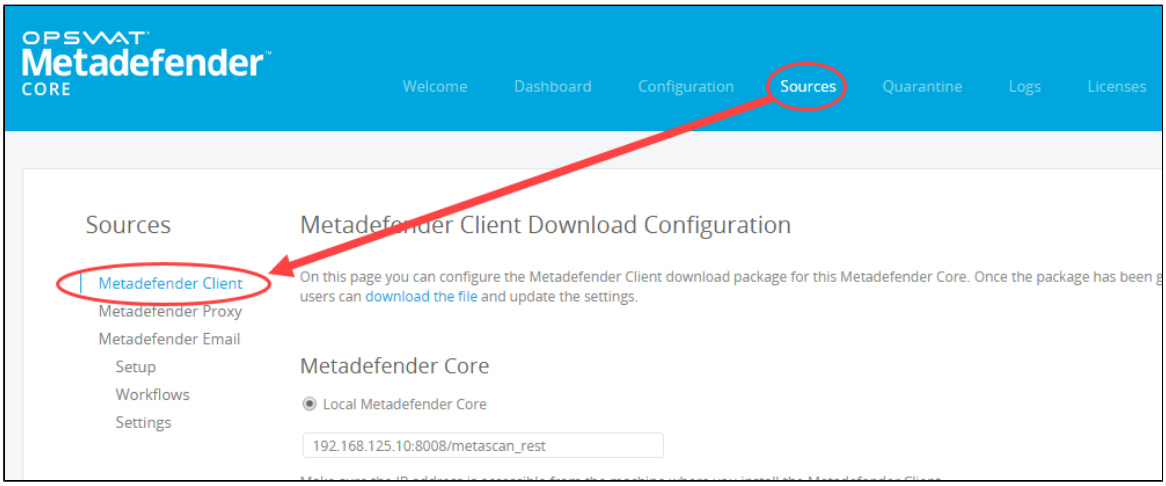
Property	Description	Default Value	omsConfig.ini Property
	If authentication is required, the password for the proxy server.		

2.3. Sources Configuration

Metadefender Client Configuration

The Configuration page specifies the settings for the Metadefender Client package that will be available through the client download page, the location of which is listed at the bottom of the Configuration page. After a client has been generated, the Metadefender Client download page can be shared with anyone who should be able to use Metadefender Client to scan files with this Metadefender Core server.

1. Access the Metadefender Core Management Console .
2. Click the **Sources** tab and then c lick **Metadefender Client** .



Note: The Windows Metadefender Client is supported to run only on endpoints running Windows 7 or later.

The following properties can be set for the Metadefender Client download package. These properties can also be set by modifying the Metadefender Client INI file, which is located by default at C:\Program Files (x86)\OPSWAT\Metadefender Core <1|4|8|12|16|20>\MetascanClient\MetascanClientConf.ini.

Property	Description	Default Value
Metadefender Core	This is the address that Metadefender Client will use to access the Metadefender Core server. This should be an address (IP address or machine name) that will be accessible from every machine where Metadefender Client will be run.	<IP Address>:8008 /metascan_rest
API Key	The API key that is defined on the local server.	N/A
File Size Limit	The maximum size of files to be scanned by Metadefender Client. Files larger than this size will not be scanned.	5 MB
Batch Size	The number of files that can be processed concurrently by Metadefender Client.	100
Thread Count	The number of threads Metadefender Client will use to scan files. This should only be increased if Metadefender Client is expected to run on systems with a high number of processor cores.	10
File Scan Timeout	The number of seconds the Metadefender Client will wait for the server to respond with scan results. After timeout threshold is reach, Metadefender Client will display a "operation time out" error.	300
User Interface	This specifies whether Metadefender Client will allow the end user to select what type of scan they want to run. If Metadefender Client is set to run in automatic mode, then it will run silently without any user interaction.	Allow User Selection
Scan Types	Specifies which scan types will be available for the user to choose from. • **Fast:** Scans all running processes. • **Intermediate:** Scans running processes and loaded libraries. • **Full:** Scans entire system.	Fast

Property	Description	Default Value
	• **Custom:** Scans the files and directories the user selects.	
Local Log Directory	The directory where Metadefender Client will log all scanning activity.	C:\MetascanClient.log

After configuration changes have been made, you can generate a new Metadefender Client downloadable package by clicking **Update Client Download** .

Metadefender client command line options

All of the options specified in the configuration file can also be set on the command line. Options set on the command line override the same options set in the configuration file. The following table specifies additional options that can be set on the command line. Default values are in bold.

Variable	Description
show_ui	Determines whether the Metadefender Client User Interface will be displayed. **Possible values:** • **0:** Metadefender Client will run silently. • **1:** The Metadefender Client user interface will be displayed to the user.
exit_on_clean	Determines whether the Metadefender Client user interface will exit if all files scanned are clean. **Possible values:** • **0:** Metadefender Client will not immediately exit after a scan where all files are clean. • **1:** Metadefender Client will immediately exit after a scan where all files are clean.
exit_on_dirty	Determines whether the Metadefender Client user interface will exit if there are dirty files found during a scan. **Possible values:**

Variable	Description
	• **0:** Metadefender Client will not immediately exit after a scan where one or more dirty files are found. • **1:** Metadefender Client will immediately exit after a scan where one or more dirty files are found.
auto_start	Determines if the Metadefender client will start scanning immediately once application launches. **Possible values:** • **0:** Metadefender Client will not immediately start scanning. • **1:** Metadefender Client will immediately start scanning. **Note:** show_ui=0 is ignored if auto_start is not set to 1.
auto_save_result	Determines if the scan results should be saved automatically. **Possible values:** • **0:** Metadefender Client will not save results automatically. • **1:** Metadefender Client will automatically save scan results at the location where Metadefender Client is located.
key	API key given by OPSWAT when OPSWAT hosted server is used.

Command line usage example

The following is an example command to start a fast scan automatically against a Metadefender Core server with the IP address 10.0.0.1:

MetascanClient.exe server=10.0.0.1:8008/metascan_rest auto_start=1 allowed_scan_levels=1

Metadefender Email Configuration

For Metadefender Email configuration, refer to the Metadefender Email User Guide.

Metadefender Proxy Configuration

For Metadefender Proxy configuration, refer to the Metadefender Proxy User Guide.

2.4. Workflow Profile Configuration

Metadefender Core allows the definition of as many different workflow profiles as are needed to meet security requirements. Defining multiple workflows allows for files from different sources or users to be processed with the appropriate security policies.

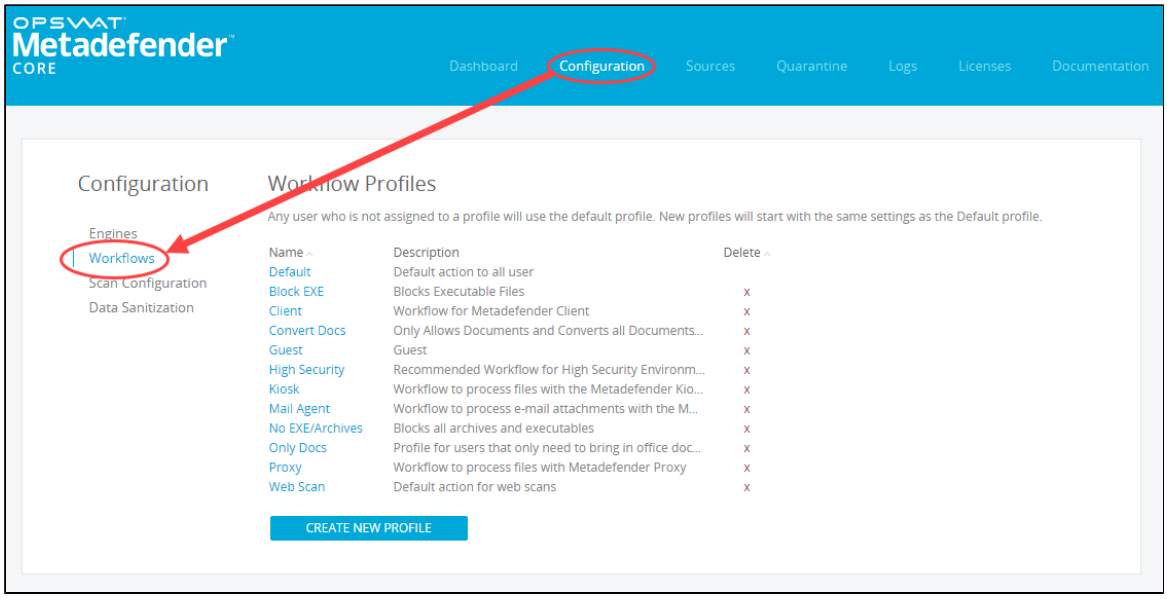
The workflows defined in the Management Console can be used with the process REST API.

Several profiles are included by default in every Metadefender Core installation. The Default profile is used to process requests that are not mapped to any other profiles. The Guest profile is intended to be used to process files by guests that do not have a defined account within an organization.

Core allows the definition of as many different workflow profiles as are needed to meet security requirements. Defining multiple workflows allows for files from different sources or users to be processed with the appropriate security policies. The workflows defined in the Management Console can be used with the process logic of the file REST API.

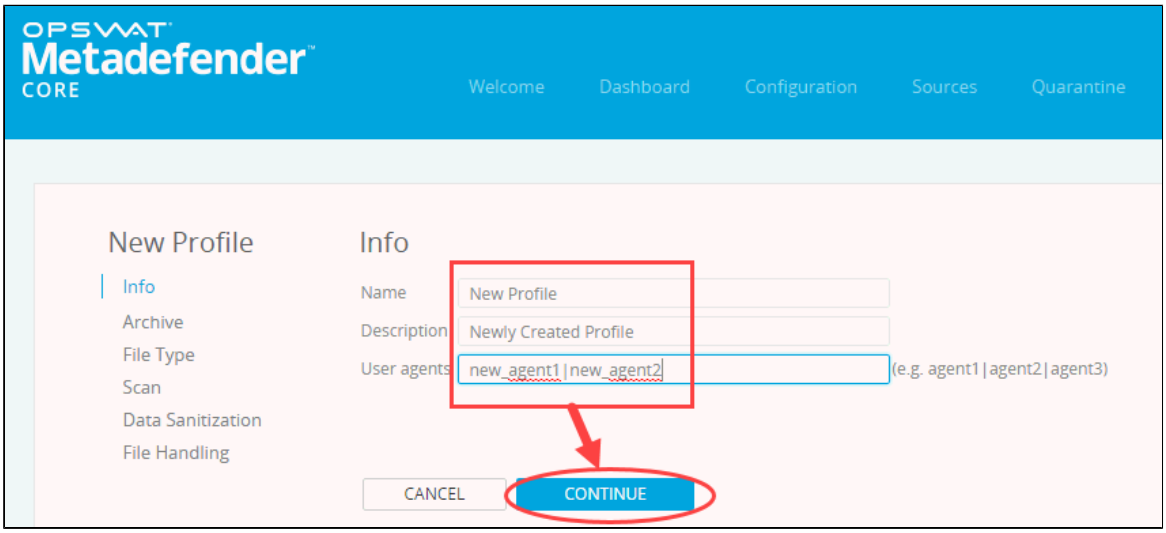
2.4.1. Creating a New Workflow Profile

1. Access the Metadefender Core Management Console .
2. Click the **Configuration** tab and then click **Workflows** .

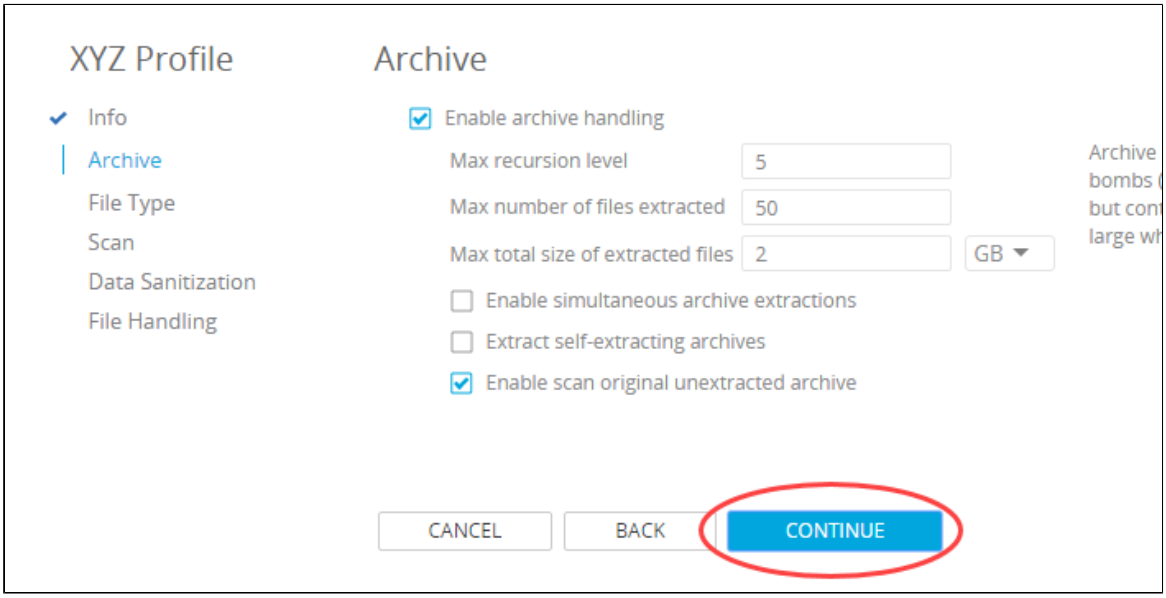


3. Click **CREATE NEW PROFILE** .

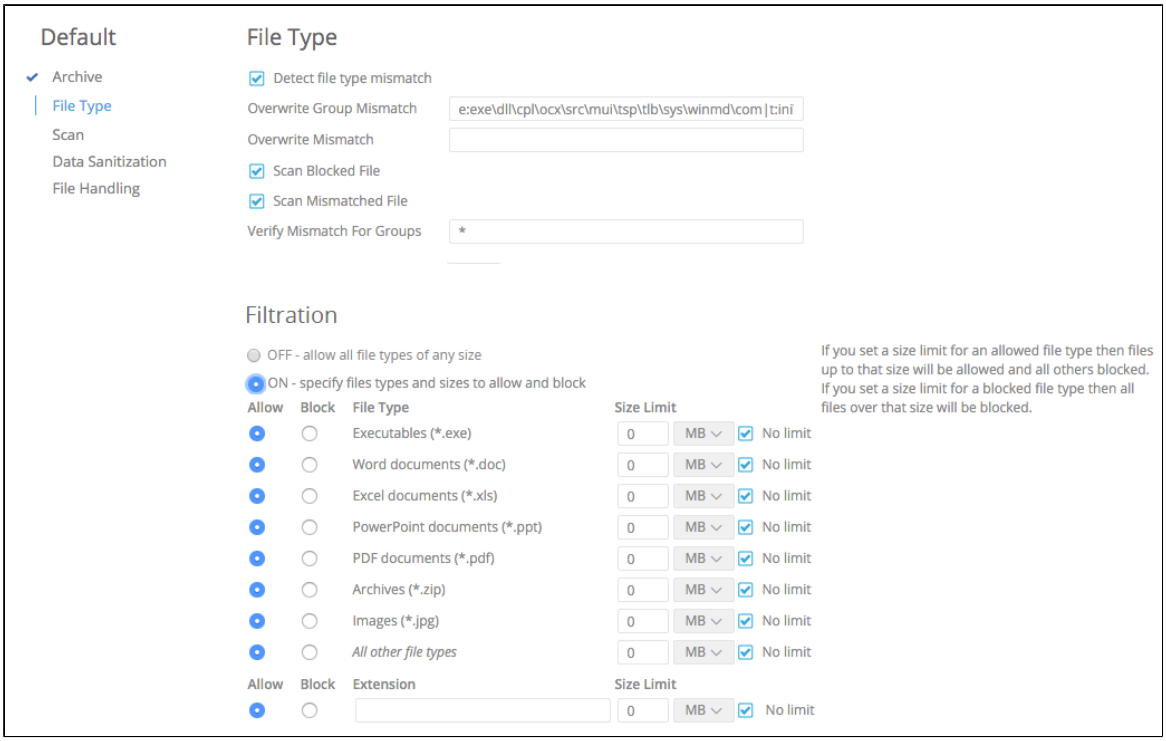
4. Enter a name and description of the new workflow profile, and specify which user agents should use this profile. (Type "|" between the agents.) These values can be changed for all profiles except for the Default profile. Click **CONTINUE** .



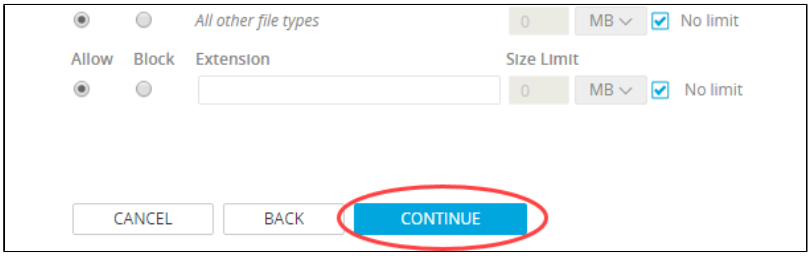
5. If you enable archive handling, Metadefender Core extracts archives and scans the individual files within the archive. For more information about the archive handling options, refer to 2.4.3. Archive Handling. After completing your selections, click **CONTINUE** .



6. The Profile File Type and Filtration screen is displayed. Select how you want file type mismatches to be scanned and file type filters to be applied. For additional information, refer to 2.4.4. File Type Detection And Filtration Overview.

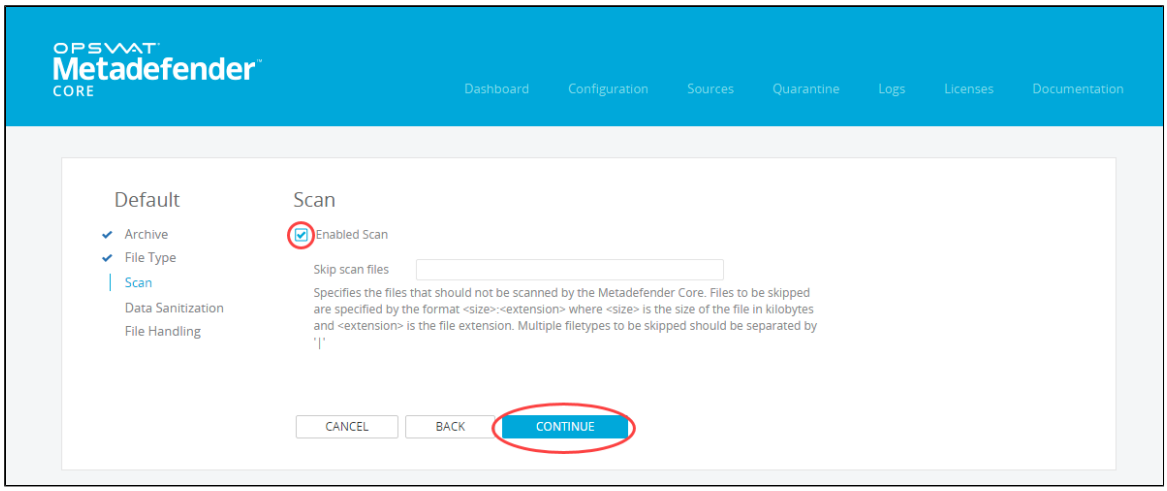


7. After selecting the the file types and filtration, click **Continue** .

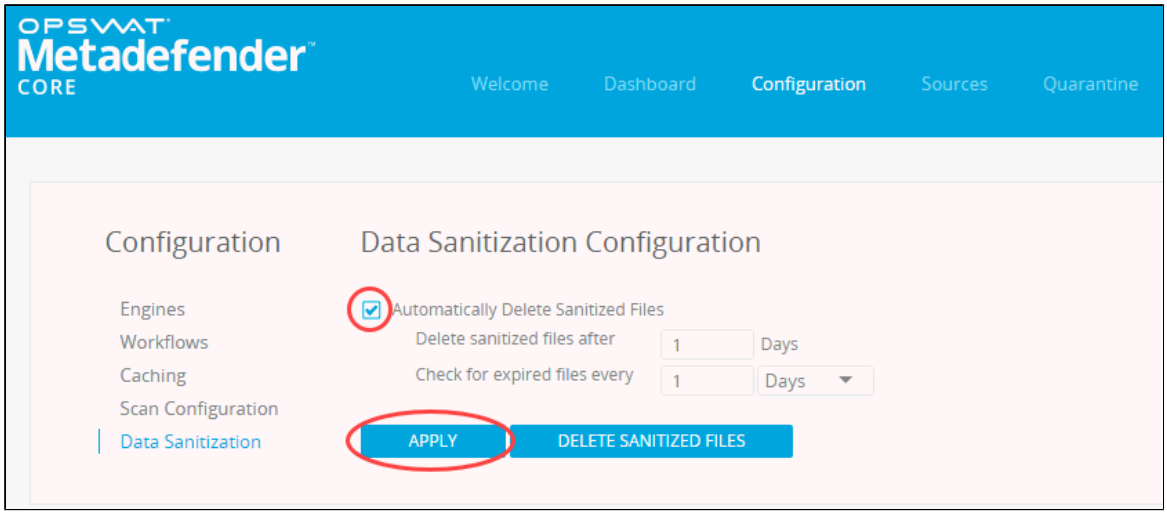


8. The Scan page is displayed, which allows you to enable or disable scanning by the active Metadefender Core engines or specify files that will be skipped. By default, large video files are excluded from scanning for efficiency purposes. Select an option and click

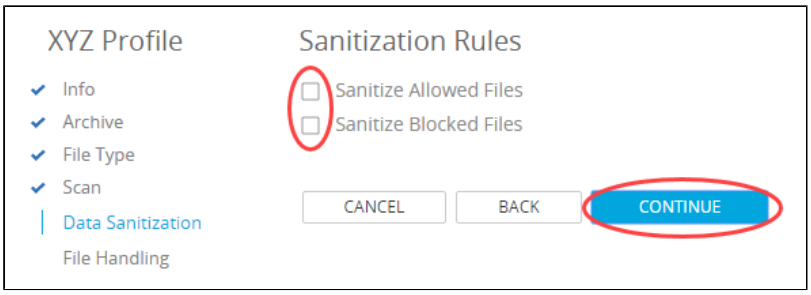
Continue .



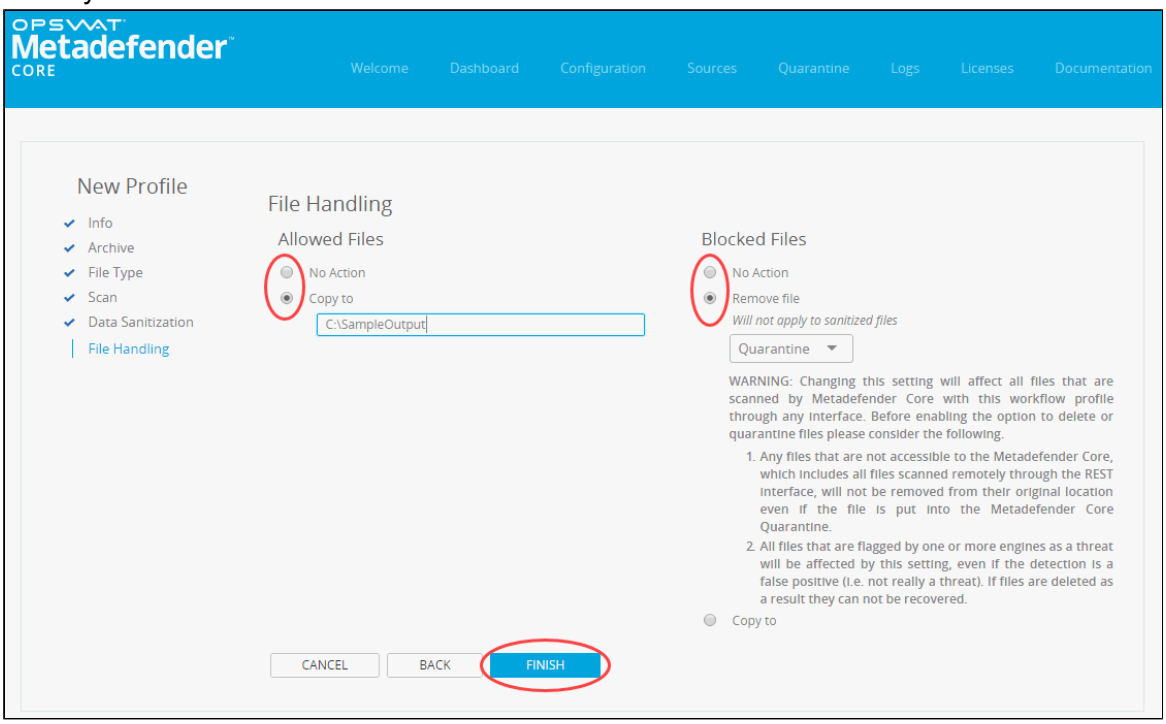
9. The Sanitization Rules page is displayed, allowing you to select if and how long files that have been sanitized by Metadefender Core workflows are available through the REST API. If sanitization is enabled, Metadefender Core automatically deletes the sanitized version of files after the specified time. This configuration determines how files are handled after processing is complete.
   Both allowed and blocked files can be configured to sanitize specific types of files and convert them to other formats. This can remove any threats embedded within the original file. If configured, the original file can be deleted after the sanitization.
   Select your options and click **APPLY** .



10. Select whether to sanitize allowed and blocked files and then click **Continue** .



11. The File Handling screen is displayed, allowing you to select how files are handled after data sanitization is complete. File handling is configured separately for allowed and blocked files.
    You can configure blocked files to be removed, either by quarantining or deleting the blocked file. You can configure both blocked and allowed files to be copied to a specified directory path.
    Make your selections and click **FINISH** .

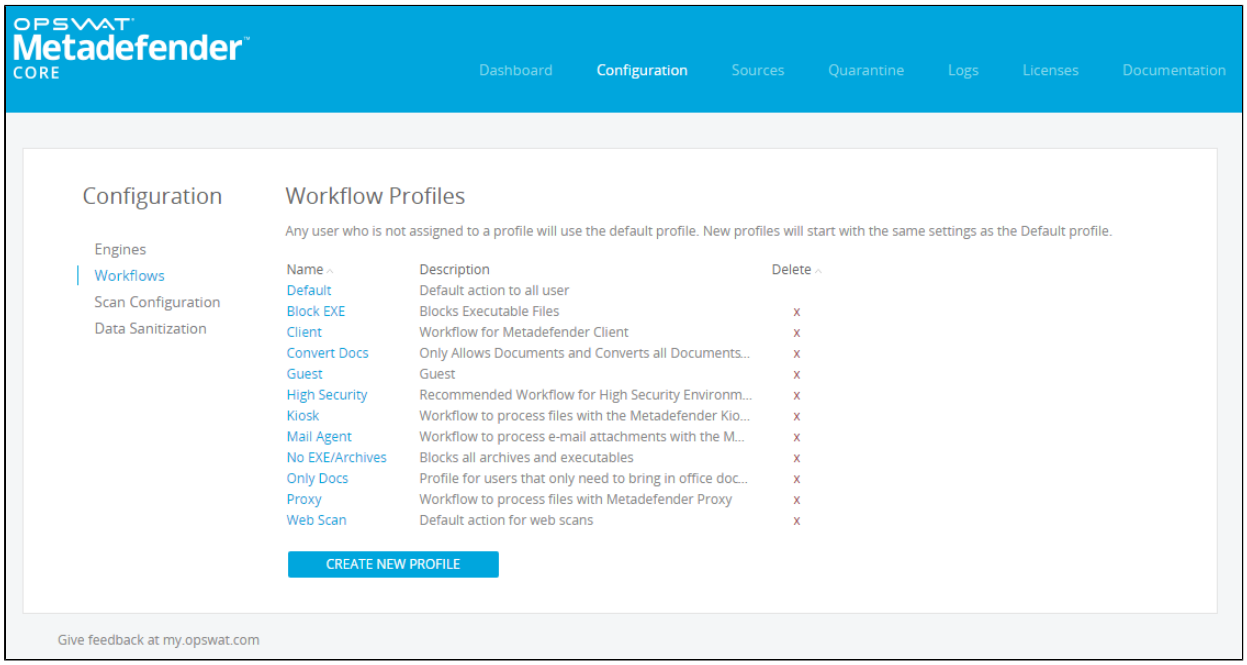


2.4.2. Default Profiles

Several profiles are included by default in every Metadefender Core installation:

Profile Name	Description
Default	Used to process requests that are not mapped to any other profiles

Profile Name	Description
Block EXE	Blocks executable files
Client	Used in the new Metadefender Client
Convert Docs	Only allows documents and converts all documents to safe formats
Guest	Used to process files by guests that do not have a defined account within an organization
High Security	Recommended for usage in high security environments
Kiosk	Used to process files with the Metadefender Kiosk
Mail Agent	Used by the Metadefender Mail Agent
No EXE /Archives	Blocks all archives and executables
Only Docs	Profile for users that only need to bring in office documents
Proxy	Used by Metadefender Proxy
Web Scan	Used to process requests made through the Web Scan page



2.4.3. Archive Handling

The Archive Handling configuration determines how archives are handled within Metadefender Core. If archive handling is enabled, Metadefender Core extracts archives and scans the individual files within the archive.

- Most common archive formats are supported, including Zip, PKLite, 7z, Jar, Jarc, rar, rar5, tar, taz, ISO, Gzip, CAB, ARC, ARJ, LZH, RPM, DEB, LZMA, WIM, SFX, XZ. Metadefender Core can also extract self-extracting archives created by both 7zip and WinRAR.
- MS Office file (from 2007) is treated as an archive file when scanning.

The following settings apply if archive handling is enabled:

Property	Description	Default Value	CLI config	Additional info
Enable Archive Handling	Enables Metadefender Core's archive library handling.	Enabled	le=<0|1>	
Max Recursion Level	The maximum depth that Metadefender Core will continue to extract archives for scanning. After this depth is reached, Metadefender Core	5	rl=<levels>	Maximum value: 2147483646

Property	Description	Default Value	CLI config	Additional info
	does not extract further archives but scans those archives as entire files. If the setting is 0, archives are not extracted.			
Number of Files	The maximum number of files that can be in an archive that Metadefender Core is extracting. If the number of files in an archive exceeds this value, Metadefender Core returns the result as a potential threat.	50	an=<number>	Maximum value: 2147483646
Total Size	The maximum total size of files that can be in an archive that Metadefender Core is extracting. If the total size of files in an archive exceeds this value, Metadefender Core returns the result as a potential threat.	2 GB	as=<size in MB>	Maximum value: Half the current available free space of the Metadefender Core temporary directory. If two temporary directories are set from different drives, the highest available space will be used.
Simultaneous		Disabled	ec=<0|1>	

Property	Description	Default Value	CLI config	Additional info
	Specifies if multiple archive files undergo extraction concurrently. This may improve performance on a multi-core CPU, but means that the ram-drive size should be increased (since more unpacked archives may reside on it at the same time).			
Self-Extracting	Specifies whether self-extracting archives should be extracted and treated as archives.	Disabled	sx=<0|1>	
Scan Original Un-extracted File	In addition to scanning files inside of an archive after extraction, un-extracted archives are sent directly to engines for scanning. **Note:** If “extract_archive” for an engine is enabled, this potentially exposes performance overhead because extraction happens twice, once by Metadefender Core and once by the engine.	Disabled	soa=<0|1>	

Note: Microsoft Office Documents (e.g., DOCX files) are detected as archive files. OPSWAT recommends that you enable the option to scan the original un-extracted archive. These files are then scanned without being extracted.

2.4.4. File Type Detection And Filtration Overview

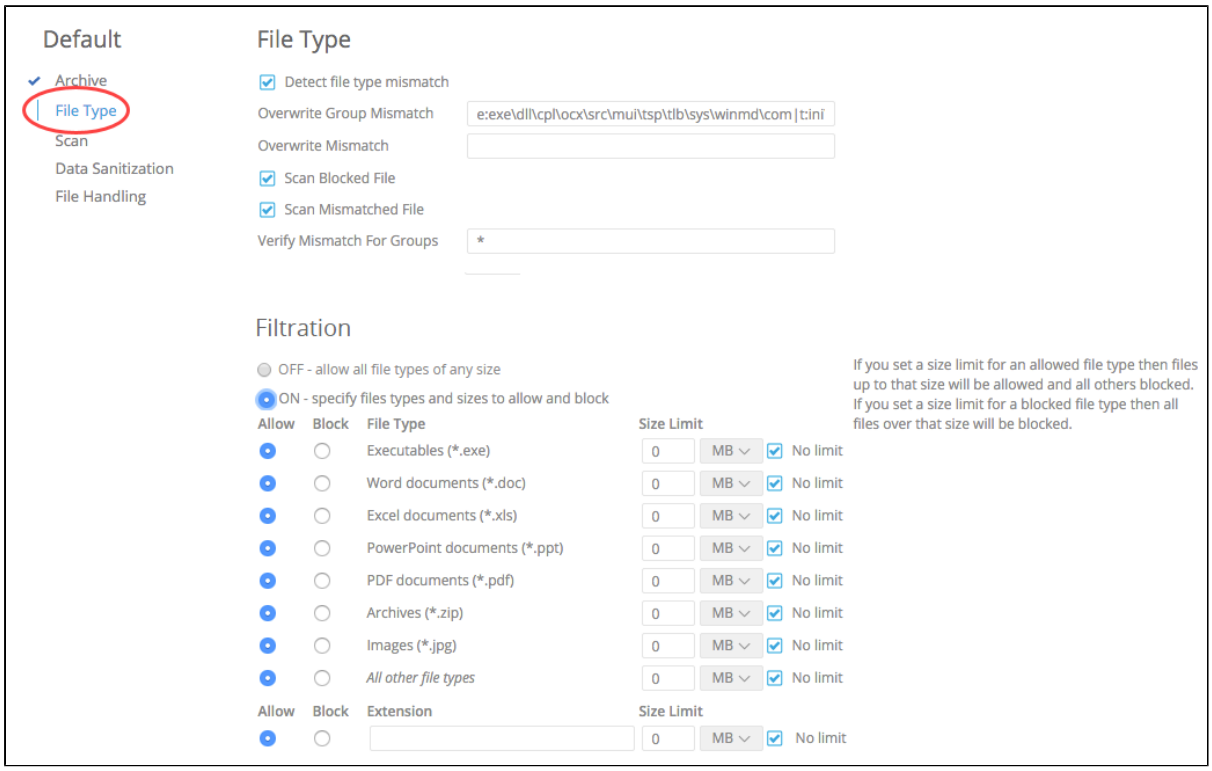
File type detection

The file type analysis configuration allows the administrator to specify whether file type analysis should be performed, and how Metadefender Core should handle file type mismatches (where the detected type of the file differs from its extension).

Common uses of file type detection include:

- Monitoring for discrepancies between a file extension and a detected file type.
- Altering the workflow of files based on certain file types (e.g., blocking files of a certain file type from entering a file system).

You can configure this setting in the Workflow tab via the **File Type** link on the left-side menu.



Metadefender file type detection is driven by an OPSWAT proprietary algorithm that combines the "Magic Number" logic as well as TrID logic. Right now, there are 5837 file types that can be detected and compared to the file extension; a complete listing can be found at the TrID site .

File type detection/analysis is not as accurate as other file metadata analysis. There will be cases where the Metadefender file identification engine will not be able to correctly analyze the file type. In these cases, you can submit a ticket with the file to OPSWAT Support for more investigation. However, we cannot guarantee that we will be able to fix the underlying issue and we cannot provide an expected turnaround time to provide an answer. Please do not open an express ticket for false file type detection.

Note that while file type detection functionality is based on the logic above, file scanning functionality is not limited to these file types.

File type detection is also referred to as 'file type analysis', 'file type mismatch', and 'file mismatch analysis'.

Filtration

The Filtration configuration allows a Metadefender Core administrator to specify that certain file types should be blocked, or that only certain file types should be allowed.

If File Type Detection is turned on, filtering will happen based on the type returned. However, only the following types will be checked for mismatches: D, P, A, E, G. (Documents, PDFs, Archives, Executable & Images)

If File Type Detection information is not available, filtering is only based on the file extension.

File Type (Mismatch and Filtration)

The file type mismatch and filtration depends on Metascan file type detection API (or REST API).

Overwrite group mismatch

- Specifies file types that should be added to file type groups for file type matching purposes.
- Available file type groups:

```
“E” – Executable (EXE, DLL, …)
“D” – Document (MS Office Word document, MS Office Excel sheet)
“A” – Archive (Zip, Rar, Tar, …)
“G” – Graphical format (Jpeg, GIF, TIFF, BMP, …)
“F” – Folder
“Y” – Logical drive
“I” – Disk image
“T” – Text
“P” – PDF format
“M” – Audio or video format
“Z” – Mail messages (MSG, …)
```

```
“O” – Other (anything that is not recognized as one of the above)
Note: An ISO is treated as an archive file type (type “A”) and
not as a disk image (type “I”).
```

- Format for a overwriteGroupMismatch value is:
  - X:EXT1\EXT2\EXT3|Y:EXT
- Example
  - E:OCX\SCR => Files with the extensions OCX and SCR will be added to the group “E” (Executables). When a file with one of these extensions is detected as an executable, it will not be reported as a mismatch.

Overwrite mismatch

- Specifies file types where a file type mismatch should be ignored.
- The format for a overwriteMismatch value is:
  - EXT_A:EXT_A1\EXT_A2\EXT_A3|EXT_B:EXT_B1
  - * means that this file type will never be reported as a mismatch.
- Example:
  - DOCX:DOC\ZIP|XXX:* => Files with the extension DOCX will not be reported as a file type mismatch if MD4M detects that this is a DOC or ZIP file. Files with the extension XXX will never be reported as a file type mismatch.

Block file type filtration

Files that are blocked are indicatied. Through workflows these blocked files can then be moved, copied, deleted, etc..

Block file type mismatch

After File Type Analysis completes, if it is determined that the file's extension does not match the true file type, Metadefender blocks this file.

Verify mismatch for groups

- Specifies which files will be checked for a mismatch according to the group they belong to.
- Format for a verifyMismatchForGroups value is:
  - X|Y|Z
- Default Value is *, which means check for a mismatch for all groups

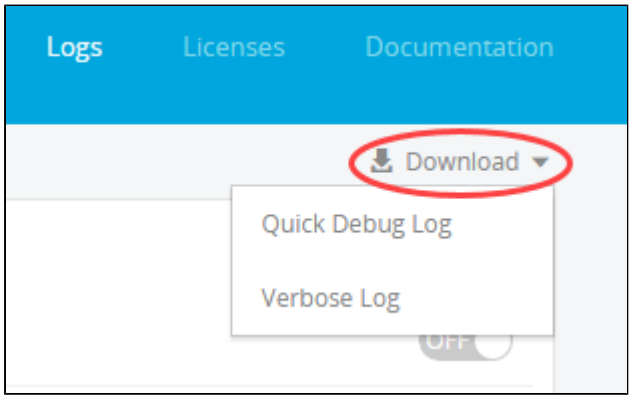
- Example:
  - E|D|A => Check for mismatches only for files that were detected as executables (“E”), documents (“D”) or archives (“A”).

2.5. Metadefender Core Logs

Metadefender Core stores the results of file scans for a period of time determined by its Log Retention settings.

Debug logs

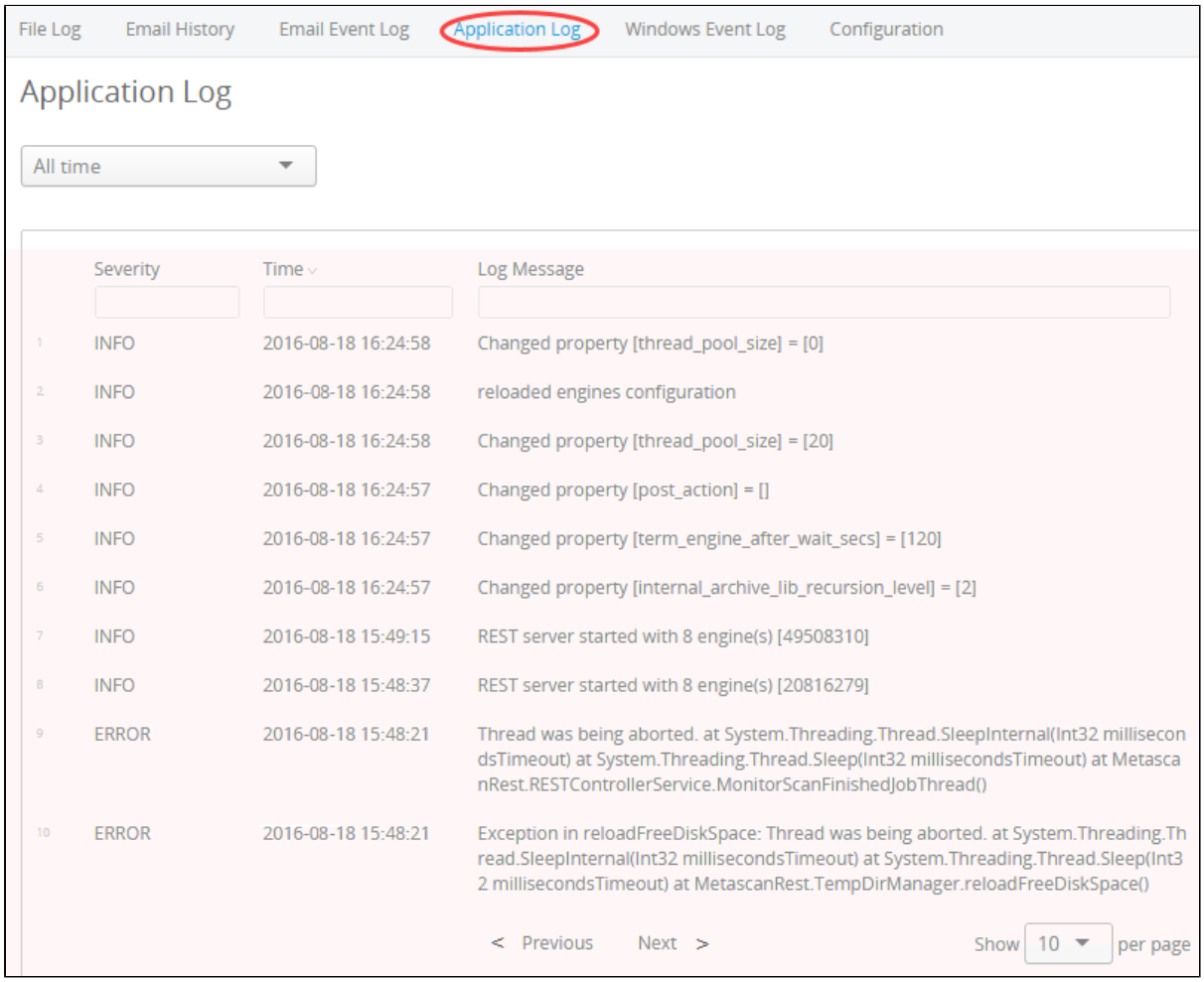
On each log page, you can generate debug information if needed for troubleshooting purposes. To generate the debug log packages, click **Download** , and then click one of the links to either export a quick or verbose debug log.



Application Log

This section of the Metadefender Core logs contains internal Metadefender Core DB event logging.

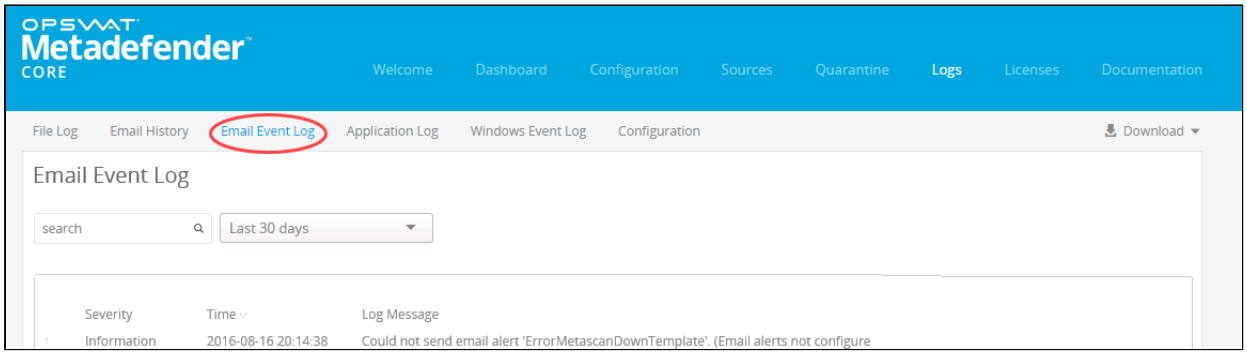
To access the Application Log from the Logs tab, click **Application Log** .



Email Event Log

The Email Event Log contains email event logs. Information includes the severity of the event, the time the event was logged, and a description of the event.

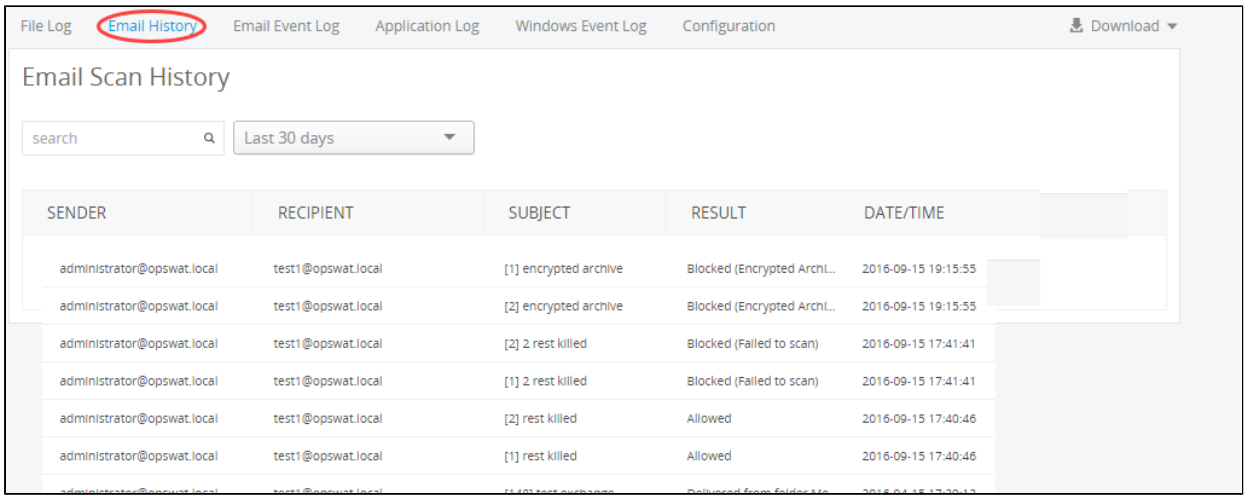
To access the Email Event Log from the Logs tab, click **Email Event Log** .



Email History

The Email Scan History page allows you to view the history of processed emails.

To access the Email Scan History from the Logs tab, click **Email History** .



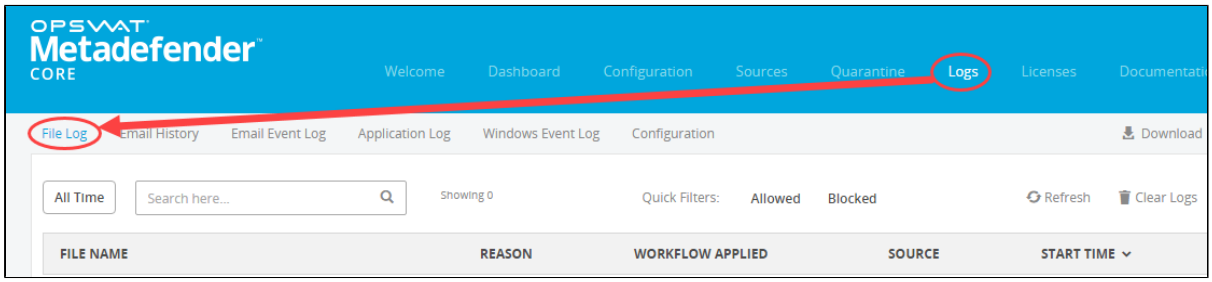
Description of fields

Field name	Description
SENDER	The sender of the email.
RECIPIENT	The recipient(s) of the email.
SUBJECT	The subject of the email.
DATE/TIME	The date and time when email was processed.
RESULT	The result of processing the email.

File Log

To access Metadefender Core's file scan history, do the following:

1. Access the Metadefender Core Management Console.
2. Click the **Logs** tab and then the **File Log** tab.



You can filter file results to display a specific time period. To view detailed scan results, select a file name from the list.

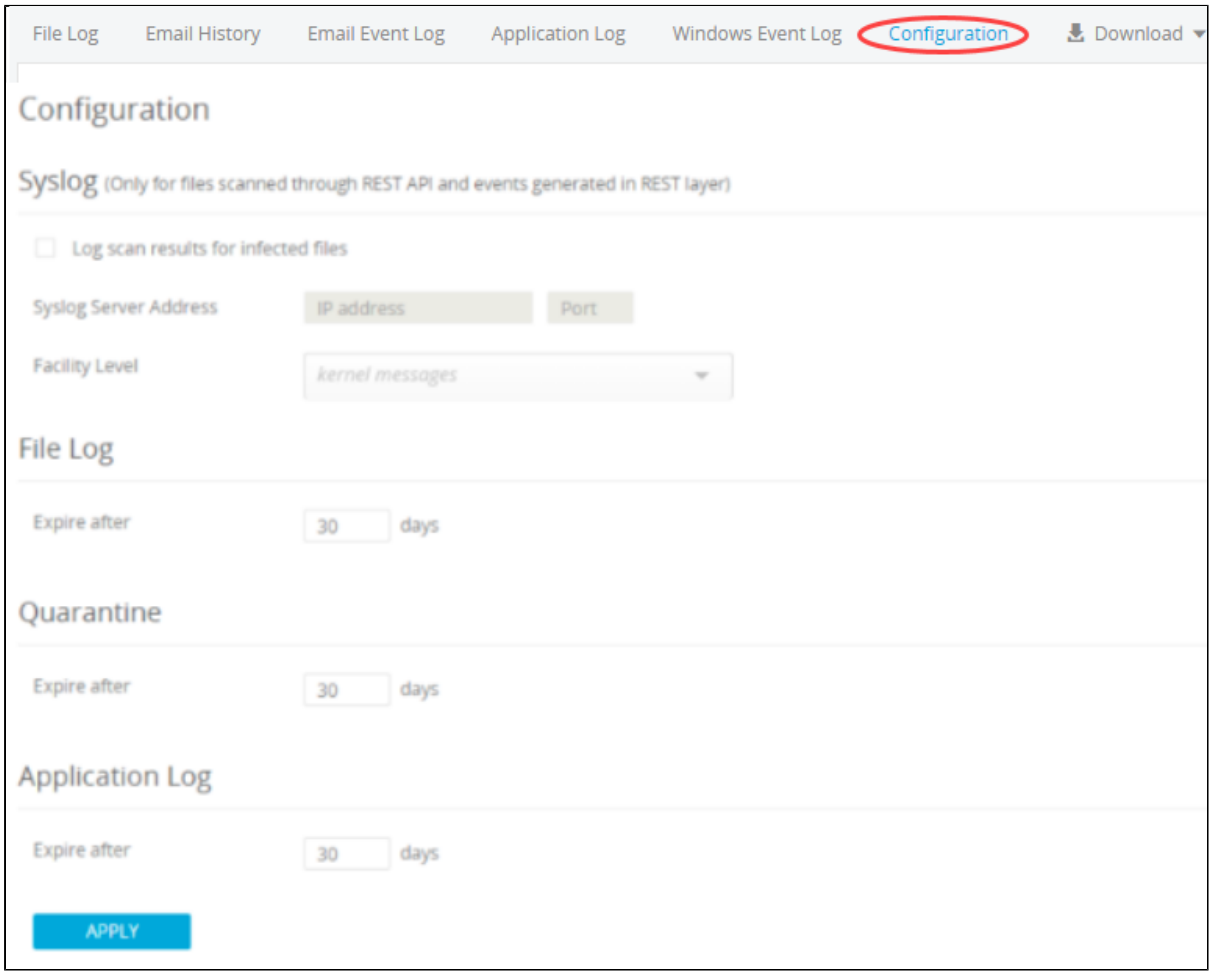
The following values display for each file in the file scan history:

Field Name	Description
File Name	The filename specified at the time of the scan request. This is determined by the following for the different Metadefender Core interfaces: • REST: The value specified in the 'filename' header in the scan request. • COM: The name of the file from where it is scanned on the Metadefender Core system. • ICAP: The value of the 'filename' header in the ICAP request. If this is not specified, then this will be the portion of the URL after the final forward slash (/).
Reason	Provides detailed information why the file is blocked by Metadefender Core. For a complete list of possible block reasons, refer to Callback for Processing a File (COM).
Workflow Applied	Indicates which workflow profile is used for the specific file. Click the workflow name to go to the workflow profile detail page and view the profile's configurations.
Source	The client IP address where the process initiated from.
Start Time	The timestamp of when the scan started.

Logging Configuration

You can configure all events in the Event Log section to be sent to a Syslog server.

To access the Configuration section from the Logs tab, click **Configuration** .



The following settings apply to Metadefender Core's Syslog logging functionality:

Property	Description	Default Value
Enable syslog messages	Indicates whether the following settings will be used to send messages to a Syslog server.	Off
IP Address	The IP address of the Syslog server.	
Port	The port that the Syslog server is listening on.	514
Facility Level	The facility level can be configured for an additional level of filtering of messages on the Syslog server.	User-Level

Example of Metadefender Core event and dirty result log in syslog:

2014-06-19 15:05:15 User.Notice 10.0.3.101 Metascan: Changed property [thread_pool_size] = [20]

2014-06-19 15:05:15 User.Notice 10.0.3.101 Metascan: Changed property [enable_cache_scan] = [1]

Note: Only infected files scanned through Metadefender Core's REST API result in syslog messages. Files scanned through any other interface do not produce syslog messages. Files scanned using Metadefender Core's workflows are not logged to syslog.

Log Retention

Logging can use a significant amount of disk space while the server is running. By default, Metadefender Core retains 30 days of logging for the File Log, Application Log, and Quarantine. You can change this default value in the Metadefender Core Management Console.

Log Type	Default	Minimum	Maximum
File Logs	30 DAYS	0 days (removes all records when check is performed)	10675199 days
Application Logs	30 DAYS	0 days (removes all records when check is performed)	10675199 days
Quarantine	30 DAYS	0 days (removes all records when check is performed)	10675199 days

Changing the number of days Metadefender Core stores log files

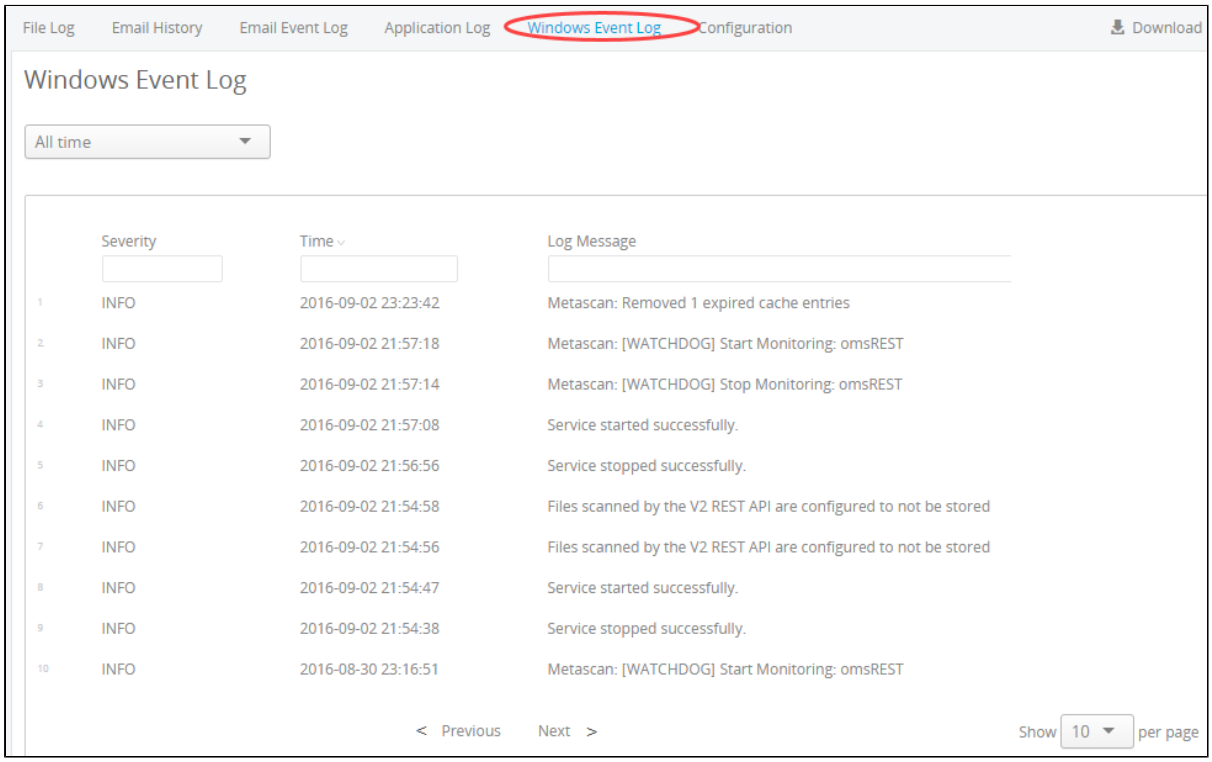
You can change the number of days Metadefender Core stores the log files for File Logs, Application Logs, and Quarantine by going to **Logs > Configuration** in the Metadefender Core Management Console.

To make changes to this page, move the slider to **On**. After making changes to the default number of days for any of the logs, click **Apply**.

Windows Event Log

The Windows Event Log contains system application event logging of all Metadefender Core-related components.

To access the Windows Event Log from the Logs tab, click **Windows Event Log** .



2.6. Quarantine Management

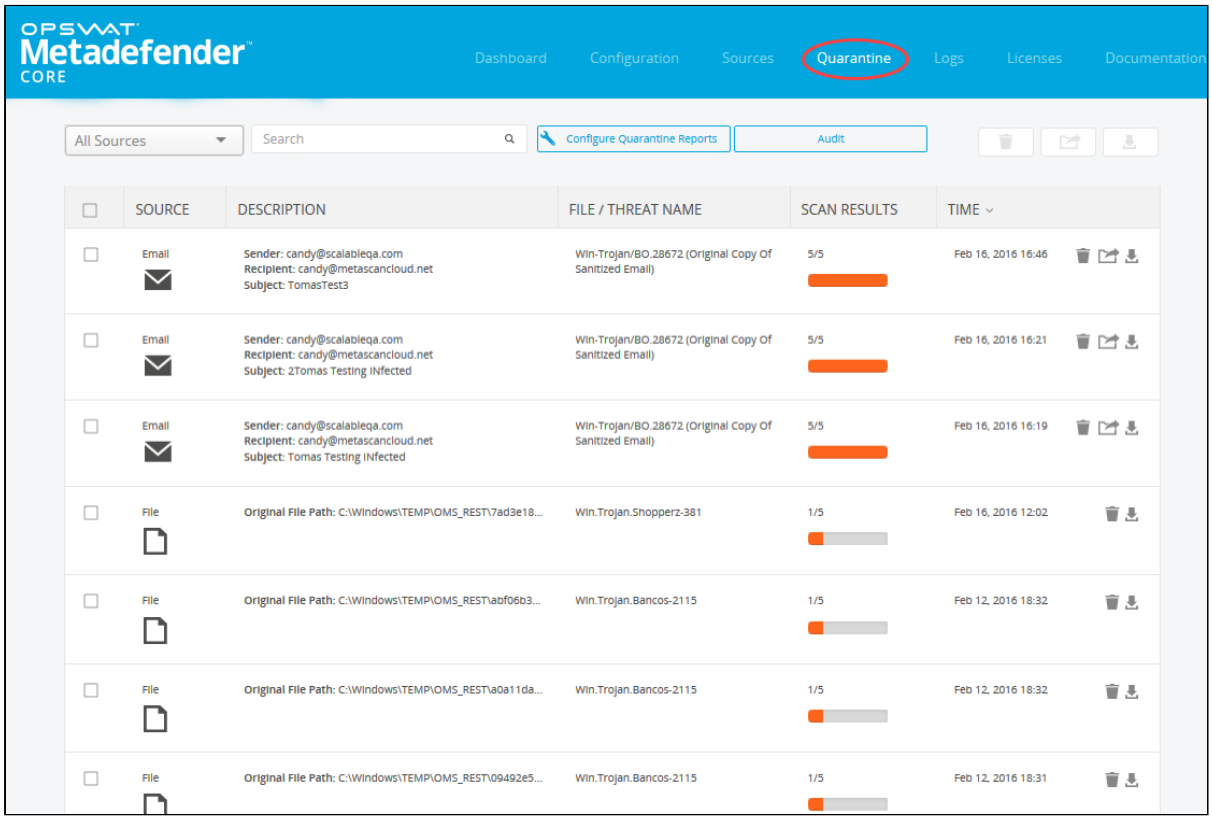
Email messages or files that have been quarantined by Metadefender Core are listed in the Quarantine tab. From this tab, an administrator can choose to delete, download, or release files and emails that have been quarantined. An administrator can also schedule and send quarantine reports.

Quarantine reports contain emails which have been quarantined by Metadefender Core. Quarantine reports include a summary of quarantined email information (subject, quarantine reason, and sending time). They also contain a link to detailed scan results in the Metadefender Core Management Console. Administrators can choose further action for the quarantined email (such as delete, release email, and download.).

Quarantine reports are the only way to be notified about quarantined emails, other than viewing the Quarantine tab. If activated, quarantined reports are sent daily by default.

Accessing the Quarantine tab in the Metadefender Core Management Console

1. Access the Metadefender Core Management Console .
2. Click the **Quarantine** tab .

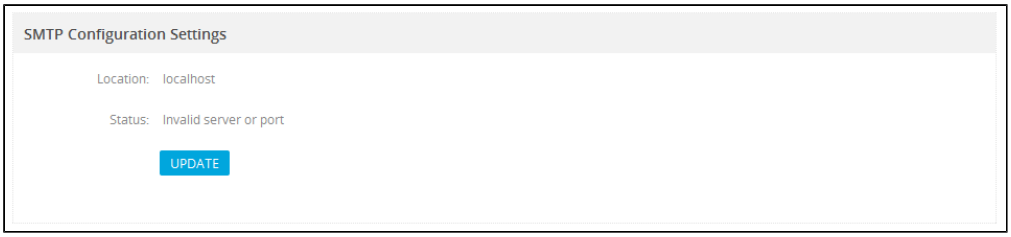


Note: Port 8000 must be open on the Metadefender Core server for the Quarantine history to be displayed correctly.

Scheduling quarantine reports

To set up and schedule quarantine reports, do the following:

1. From the Metadefender Core Management Console, click the **Quarantine** tab .
2. Click **Configure Quarantine Reports** .
3. To update SMTP settings, click **Update** .

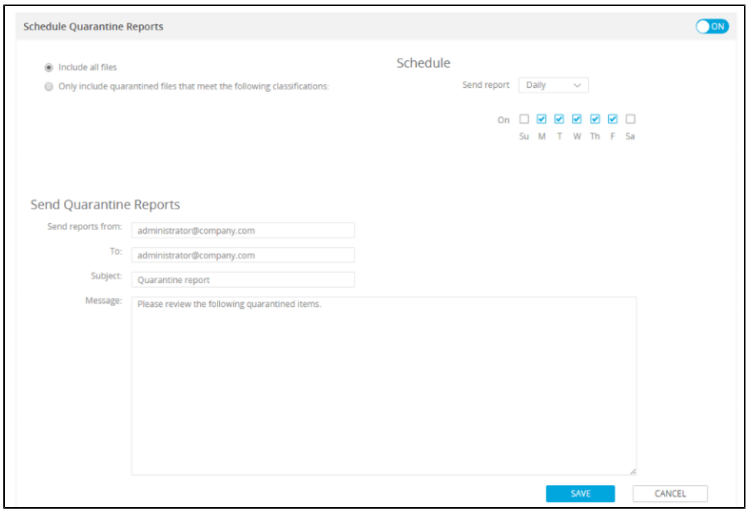


4. Enter the following SMTP server information in the SMTP Settings section:
    a. **Host:** The SMTP server's IP address or hostname.
    b. **Port:** The SMTP server's connection port.

c. **Enable SSL:** If the SMTP server enables SSL, select this checkbox to allow Metadefender Core to send the quarantine report email via SSL.

d. **Username/Password:** The username and password for the SMTP server.

5. Click **Save** .
6. Move the Scheduled Quarantine Reports On/Off slide to **On** .
7. Enter which files to include, the quarantine schedule, and email information in the **Schedule Quarantine Reports** section.



Note: Separate multiple email addresses in the **To** field with a comma.

8. Click **Save** .

2.7 Filter CLI

Title	Description	Syntax
omsFilter - remove	remove specific file from the lists	remove <sha256>

Title	Description	Syntax
omsFilter - list	retrieve whitelist and blacklist	list <type>
omsFilter - addtowhite	add the file to the whitelist	addtowhite <path> <type>
omsFilter - addtoblack	add the file to the blacklist	addtoblack <path> <type> <threat>
omsFilter - about	Shows the info of the current version	about

```
omsFilterCLI.exe

C:\Program Files (x86)\OPSWAT\Metadefender Core 4>omsFilterCLI.exe

    Metascan(R) Filter Command Line Utility 3.12.3.28579
    (C) OPSWAT, Inc. 2002-2016

    Command                                 Description
    -------
-----------------------------------
    addtoblack <path> <type> <threat>    add the file to the
blacklist
    addtowhite <path> <type>             add the file to the
whitelist
    remove <sha256>                      remove specific file
from the lists
    list <type>                          retrieve whitelist and
blacklist
    about                                shows the info of the
current version

    Arguments                            Description
    ----------
-------------------------------------
    <type>                               0: all, 1: archive
library, 2: engines
    <threat>                             Name of threat for
reference
    <path>                               Absolute file path
```

omsFilter - about

Syntax	about
Description	Shows the info of the current version

```
about

C:\Program Files (x86)\OPSWAT\Metadefender Core 4>omsFilterCLI.
exe about

    Metascan(R) Filter Command Line Utility 3.12.2.28579
    (C) OPSWAT, Inc. 2002-2016
```

omsFilter - addtoblack

Syntax	addtoblack <path> <type> <threat>
Description	add the file to the blacklist

```
addtoblack

C:\Program Files (x86)\OPSWAT\Metadefender Core 4>omsFilterCLI.
exe addtoblack C:\Files\200\10.txt 0 sampleThreatName
File 'C:\Files\200\10.txt' successfully added to blacklist
Restarting Metascan is required for changes to take effect
```

omsFilter - addtowhite

Syntax	addtowhite <path> <type>
Description	add the file to the whitelist

```
addtowhite

C:\Program Files (x86)\OPSWAT\Metadefender Core 4>omsFilterCLI.
exe addtowhite C:\Files\200\10001.txt 0
```

```
File 'C:\Files\200\10001.txt' successfully added to whitelist
Restarting Metascan is required for changes to take effect
```

omsFilter - list

Syntax	list <type>
Description	retrieve whitelist and blacklist

```
list

C:\Program Files (x86)\OPSWAT\Metadefender Core 4>omsFilterCLI.
exe list 0


<list><black_list><object><sha256_hash>1C403526E3A858775C88D113DF4
6ECEA85269964<
/sha256_hash><filter_type>0</filter_type><threat_name>Archbomb.ZIP
trojan</threa
t_name><
/object><object><sha256_hash>753690994A18CF58ED0FE3749D16448B76304
7B8</s
ha256_hash><filter_type>0</filter_type><threat_name>Archbomb.ZIP
trojan</threat_
name><
/object><object><sha256_hash>75F2667CCACCE7C7947C186DCA5029FFEE720
C01</sha
256_hash><filter_type>0</filter_type><threat_name>Archbomb.ZIP
trojan</threat_na
me><
/object><object><sha256_hash>541DF8E3BD1A55BCF3DC9839A0A1BBB7E7737
EB7</sha25
6_hash><filter_type>0</filter_type><threat_name>Archbomb.ZIP
trojan</threat_name
><
/object><object><sha256_hash>12CF28C37E9BC65D67F30ED24154D1DE4503D
7BD</sha256_
hash><filter_type>0</filter_type><threat_name>Archbomb.ZIP trojan<
/threat_name><
/object><object><sha256_hash>54E65A7B23E3E5C3771C041BFA3198E877015
925</sha256_ha
sh><filter_type>0</filter_type><threat_name>Archbomb.ZIP trojan<
/threat_name></o
bject><object><sha256_hash>C2AAB6A66924B29828C52E27AF9E8A8038E54EE
0</sha256_hash
><filter_type>0</filter_type><threat_name>yoda</threat_name><
/object></black_lis
```

```
t><white_list></white_list></list>
```

omsFilter - remove

Syntax	remove <sha256>
Description	remove specific file from the lists

> **Version supported**
>
> Supported in Core version 3.12.3+

```
removefromblack

C:\Program Files (x86)\OPSWAT\Metadefender Core 4>omsFilterCLI.
exe remove
D7B0FD22C8741846EC35E43B224DED709A9A5E558BA6536F9CAC0BA5BB608658

File with hash of
'D7B0FD22C8741846EC35E43B224DED709A9A5E558BA6536F9CAC0BA5BB608658'
successfully removed
Restarting Metascan is required for changes to take effect
```

2.8. Advanced Configuration Options

CORS Configuration

You can harden the Metadefender Core's cross-origin resource sharing (CORS) configuration to only allow access from a restricted list of systems.

The following edits can be made in C:\Program Files (x86)\OPSWAT\Metadefender Core X\REST\Web\web.config.

To restrict access to the local system, the line

```
<add name="Access-Control-Allow-Origin" value="*"/>
```

can be changed to

```
<add name="Access-Control-Allow-Origin" value="http://localhost"/>
```

Then add a new rule to <system.webServer><rewrite><outboundRules>

```
<rule name="Allow CORS on specify ip/subnet" >
    <match serverVariable="RESPONSE_Access-Control-Allow-Origin"
pattern=".+" />
    <conditions>
        <add input="{REMOTE_ADDR}" pattern="^(192.168.200.*|192.
168.201.102)$" />
    </conditions>
    <action type="Rewrite" value="*" />
</rule>
```

Enabling HTTPS

By default, communication with the RESTful web server is not encrypted. By setting up an HTTPS server, the server can enforce secure connections between client and server on an SSL channel. Steps to configure IIS Express to host an HTTPS server are outlined in the sections below.

Requirements

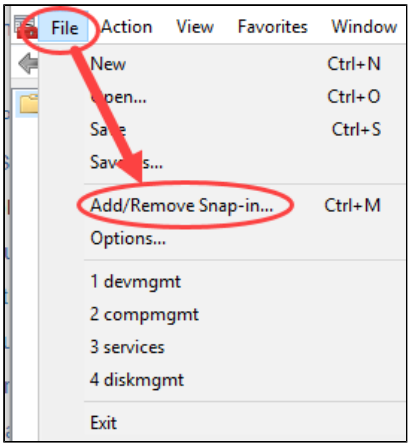
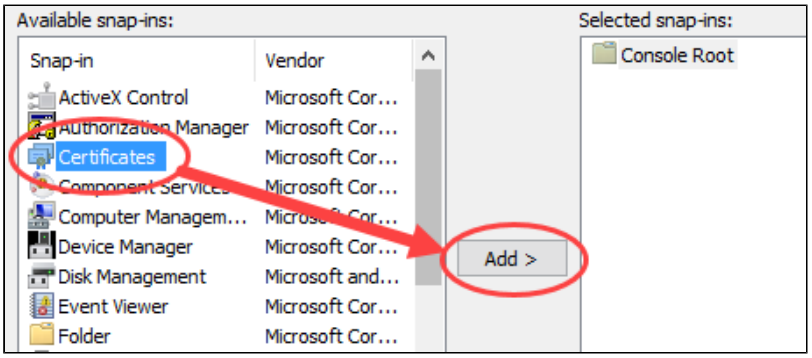
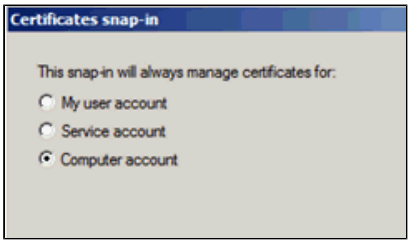
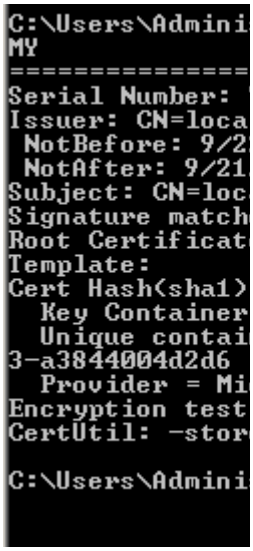
- To enable HTTPS, you must install a trusted certificate issued by a certificate authority OR a self-signed certificate used for development testing. The procedures detailed below cover how to install self-signed server certificate.
- To install a CA (Certificate Authority) signed server certificate, go to Microsoft's TechNet Install a Server Certificate website. If you enable HTTPS and are using a self-signed certificate, you MUST install the self-signed certificate.

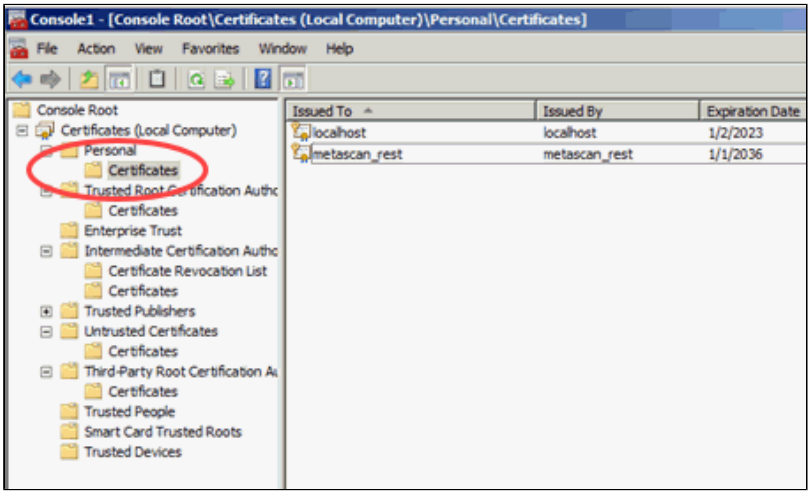
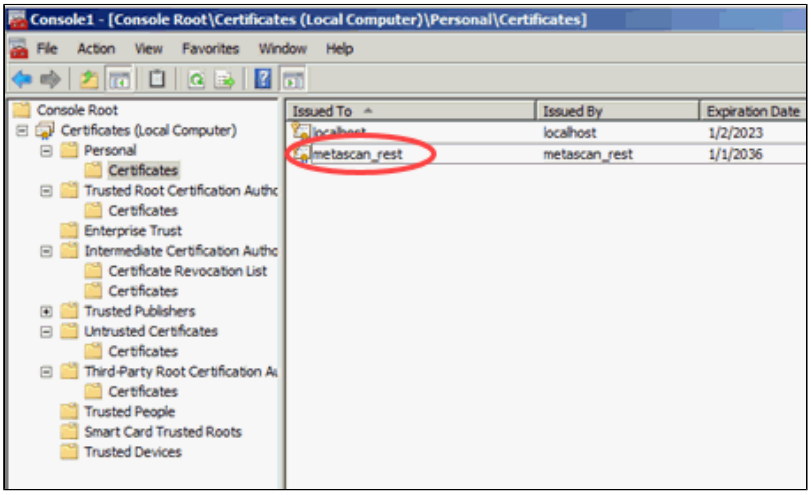
Overview

The following high-level steps are covered on this page.

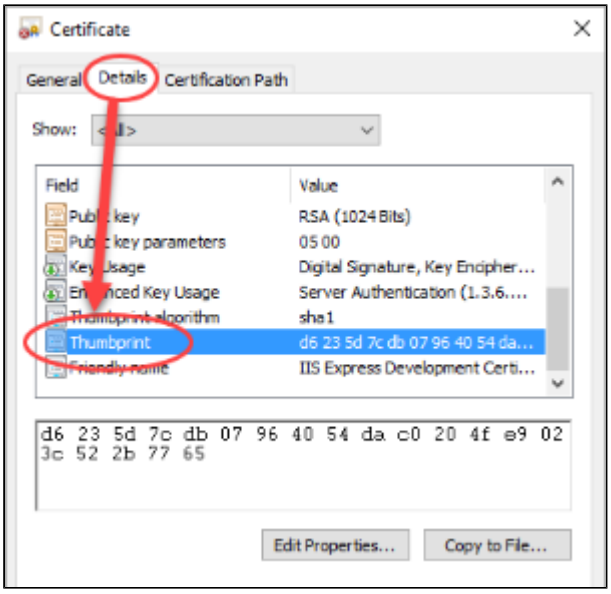
1. Obtain a certificate (we will use self-signed certificate in this documentation).
2. Install the certificate.
3. Configure IIS with HTTPs.
4. Restart Metadefender REST service.

Generating a self-signed certificate

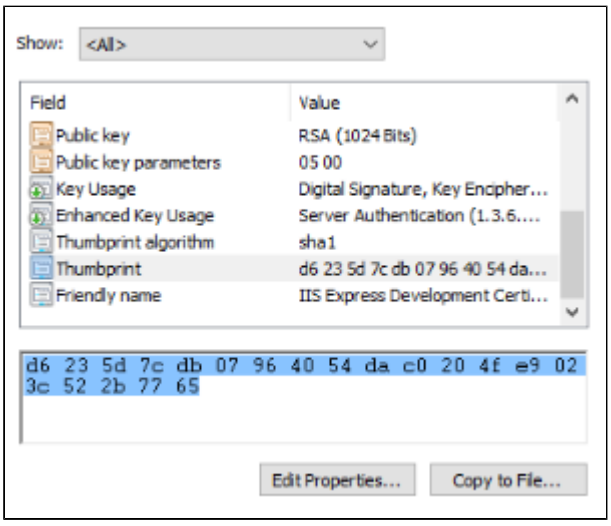
Using Certificates MMC	Using certutil
1. Click the Windows Start menu and enter “MMC.exe” in the Search box. 2. In the MMC window, click on **File > Add/Remove Snap-In...** .  3. In the Available snap-ins list, select **Certificates** and then click **Add** .  4. Select **Computer Account.**  5. Click **Next > Finish** .	1. Run certutil -store M  2. Copy the Cert Hash ef8a0fc5620b621a 3. Create a new GUID {CDA52389-5954-4

Using Certificates MMC	Using certutil
6. Click **OK** to load the certificates snap-in. 7. Expand the Certificates (Local Computer) menu and browse to **Personal > Certificates** . The example below uses ‘metascan_rest’ certificate. Your certificate may have any name.  8. Double-click the certificate name you want to use for the Metadefender Core REST Server.  9. Click the **Details** tab and select **Thumbprint** .	

Using Certificates MMC	Using certutil



10. Copy the value into a text editor for later use.



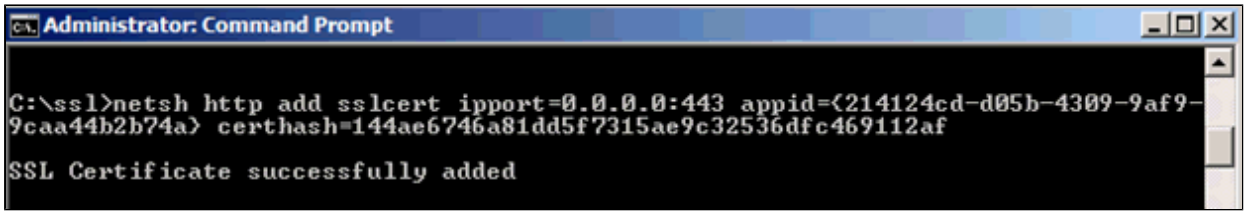
Note: The spaces in the thumbprint should be removed for the command to execute properly.

Install Certificate

1. Open a command prompt.
2. Run the following command:

```
netsh http add sslcert ipport=0.0.0.0:443 appid={214124cd-
d05b-4309-9af9-9caa44b2b74a} certhash=<certificate thumbprint
retrieved from previous step>
```

3. Confirm that the SSL Certificate is successfully added, as indicated by the example below.



Enabling HTTPS on IIS Express

The following procedure enables HTTPS on IIS Express. To set up quarantine and mail agent, see the sections that follow.

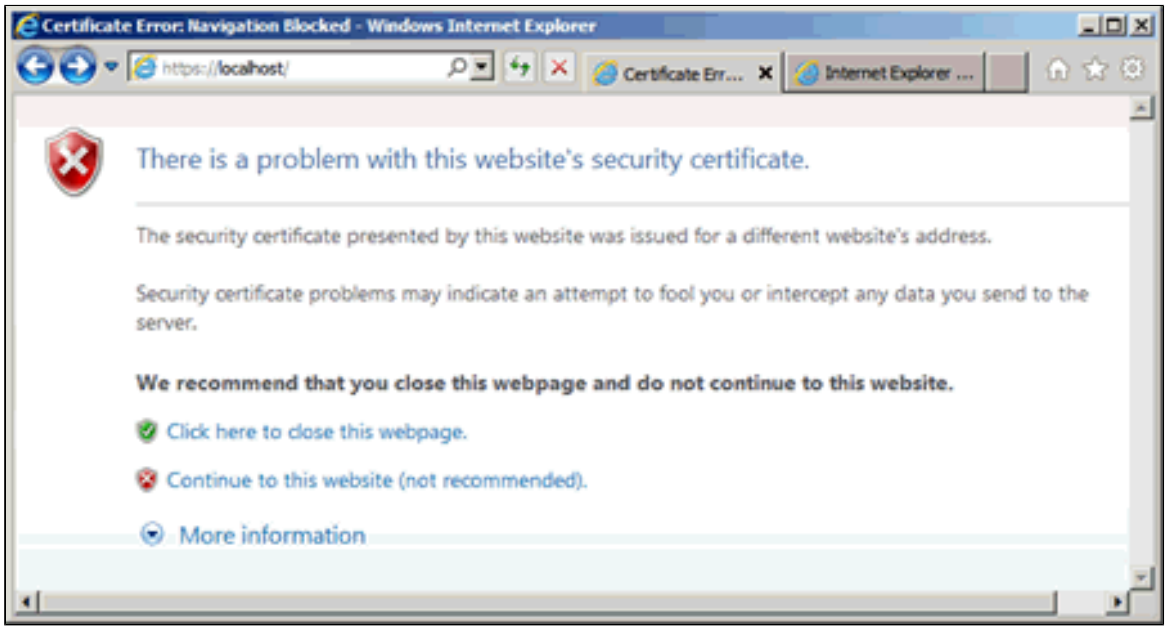
1. Open the <Metadefender Core installation directory>\REST\Config folder (e.g., C:\Program Files (x86)\OPSWAT\Metadefender Core X\REST\Config).
2. Open the applicationhost.config file in a text editor.
3. Go to the <sites> tag and add the HTTPS binding to the metascan_rest website as shown in the example below. This port can not be in use by any other applications.

```
<site name="metascan_rest" id="1" serverAutoStart="true">
    <application path="/" applicationPool="Clr4ClassicAppPool">
        <virtualDirectory path="/" physicalPath="C:\Program Files (x86)\OPSWAT\Metascan 16\REST\Web\"/>
    </application>
    <bindings>
        <binding protocol="http" bindingInformation=":8008:"/>
        <binding protocol="https" bindingInformation=":443:" />
    </bindings>
</site>
```

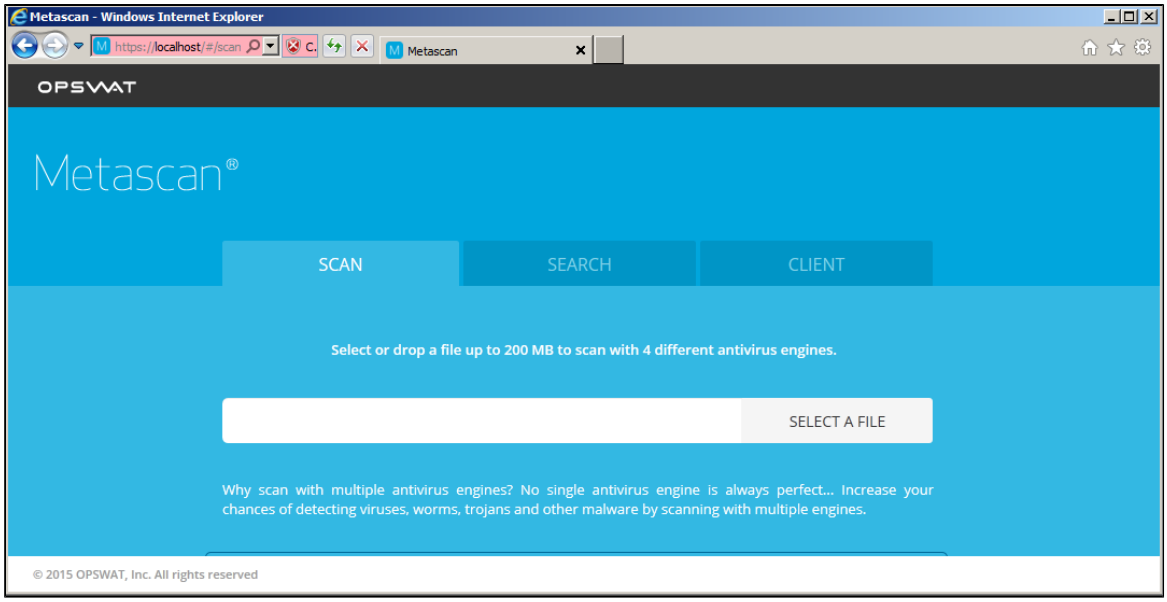
4. Optionally, to keep REST HTTPS only, remove the <binding protocol="http" bindingInformation=":8008:" /> line from applicationhost.config.
5. Save and close the ‘applicationhost.config’ file.
6. Navigate to the Quarantine folder (by default, this is C:\Program Files (x86)\OPSWAT\Metadefender Core X\Metascan Quarantine).
7. Open Metadefender.Quarantine.Service.exe.config and change the following section:

Original	New
<setting name="RestBaseUrl" serializeAs="String"> <value>http://localhost:8000</value> </setting> <setting name="QuarantineBaseUrl" serializeAs="String"> <value>http://localhost:8000</value> </setting> <setting name="QuarantineProtocol" serializeAs="String"> <value>REST</value> </setting> <setting name="MetascanUrl" serializeAs="String"> <value>http://localhost:8008/metascan_rest/</value> </setting> <setting name="WebBaseUrl" serializeAs="String"> <value>http://localhost:8008/management/#</value> </setting>	<setting name="RestBaseUrl" serializeAs="String"> <value>https://localhost</value> </setting> <setting name="QuarantineBaseUrl" serializeAs="String"> <value>https://localhost</value> </setting> <setting name="QuarantineProtocol" serializeAs="String"> <value>REST</value> </setting> <setting name="MetascanUrl" serializeAs="String"> <value>https://localhost/metascan_rest/</value> </setting> <setting name="WebBaseUrl" serializeAs="String"> <value>https://localhost/management/#</value> </setting>

8. Restart the Metadefender Core Quarantine Service.
9. Test that the site works by going to https://localhost. The following webpage should be displayed:



10. Click **Continue to this website** .



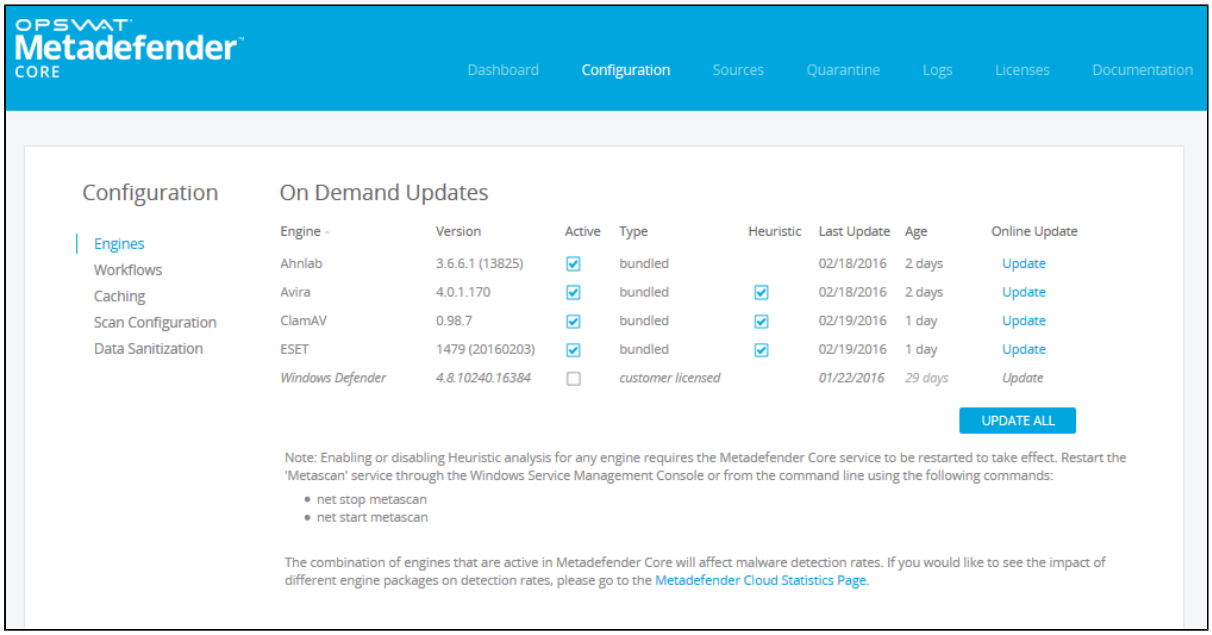
Engine Specific Configuration

Configuration specific to each built-in engine should be done via the configuration file: omsConfig.ini. This file is located in the directory where Metadefender Core is installed. The following table shows which configurations are available for each engine.

Important: The Metadefender Core service must be restarted to apply any changes.

Heuristic scan / extract archive

Some engines allow you to scan files using heuristics, which can detect viruses that are not yet in the engine's virus definition database. Enabling heuristic scanning for engines increases detection of potential new viruses, but also increases the false positive rate.



Note: Enabling heuristic scanning can also increase the engine's scan time.

Engine [CATEGORY]	Configuration [ATTRIBUTE]	Value
Ahnlab [ahnlab]	Heuristic scan [heuristic_scan]	1: Enable 0: Disable No entry: default level
	Archive library [extract_archive]	
AVG [avg]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
Avira [avira]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
Bitdefender [bitdefender]	Archive library [extract_archive]	
Clam AV [clamav]	Heuristic scan [heuristic_scan]	

Engine [CATEGORY]	Configuration [ATTRIBUTE]	Value
	Archive library [extract_archive]	
Emsisoft [emsisoft]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
ESET [eset]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
F-Prot [f-prot]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
Ikarus [ikarus]	Archive library [extract_archive]	
K7 [k7]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
Kaspersky [kaspersky]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
McAfee [mcafee]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
NANOAV [nanoav]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
nProtect [nprotect]	Heuristic scan [heuristic_scan]	
Quick Heal [quickheal]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	

Engine [CATEGORY]	Configuration [ATTRIBUTE]	Value
Sophos [sophos]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
Vir.IT eXplorer [vir.itexplorer]	Archive library [extract_archive]	
Total Defense [totaldefense]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	
Zillya! [zillya!]	Heuristic scan [heuristic_scan]	
	Archive library [extract_archive]	

Import/export configuration

You can import and export your Metadefender Core configuration using the omsConfig.exe command line utility. This can be useful for preserving configurations during an upgrade or when deploying multiple Metadefender Core installations with the same configuration.

Help

Syntax: help

This function prints the available command line options.

Import

Syntax: import <full file path> [<option>]

This command imports configurations from a file. All settings will take effect after a restart of the Metadefender Core services.

Export

Syntax: export <directory> [<option>]

This command exports current configurations to a file.

Options

The following argument can be used with the import and export commands.

Option	Description
/pass	Password to encrypted & extract configuration.

Post Action Script

> Using Post Action Script is not recommended. If you plan to use post action script, please contact OPSWAT Technical Support to see if there are different ways of achieving the same goal. For example, if you want to copy files after scan, use the Workflow - copy to directory feature instead.

Running scripts on clean/infected files

Creating a post action script provides the capability to handle simple or advanced application logic. Once a file is scanned by engines, the script will be executed on a file based on scan results.

XML format for post action script

```
<post_action>
    <scripts>
        <if type="[scan result]">[script]</if>
        …
        <if type="[scan result]">[script]</if>
    </scripts>
    <user_vars>
        <user_var name="[user variable]">[user variable string]</u
ser_var>
        …
        <user_var name="[user variable]">[user variable string]</u
ser_var>
    </user_vars>
</post_action>
```

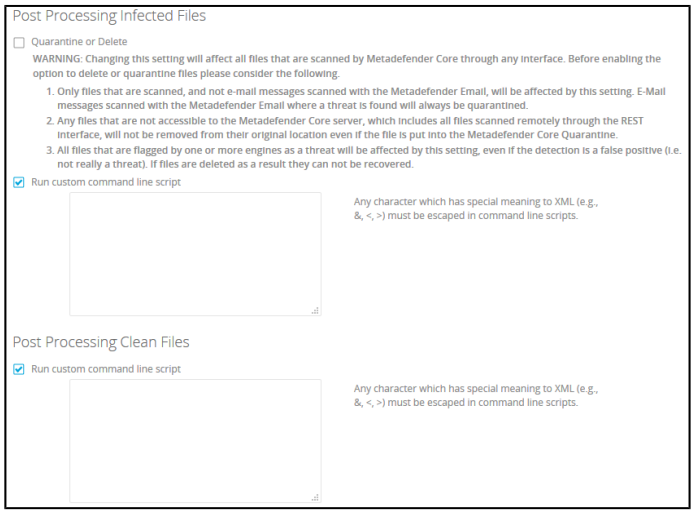
Item	Description	Example
[script]	Batch script supported by cmd.exe. For multiline scripts,use “&&”. (“&&” for xml)	DEL c:\eicar.com
[scan result]	Refer to asynchronous scan API in Software Developer Guide	for clean files, 0
[user variable]	Variables that you define	data_folder
[user variable string]	Value for user variables	C:\ data_folder_for_archiving

Pre-defined variables

Variables	Description	Note
%%%file_path%%%	Absolute path to the file to be scanned.	This variable is supported only on local COM scans.
%%% threat_name%%%	Name of threats found by engines.	Applies only to infected(“1”) scan result.
%%% scan_finished%%%	The time when scan is finished.	

Applying post action script

Property Name	post_action
Description	XML-formatted scripts applied per scan result with support of pre-defined variables and user-defined variables.
CLI	omsCmdLineUtil.exe config pa=<xml file path>

Property Name	post_action
Management Console	On the Metadefender Core Management Console, use only the script in the text box, instead of the XML format for Post Action Script described in the above section. 
Remark	Any character that has special meaning to XML (e.g., &, <, >) must be escaped.

Example of post action script XML

Note: In the following examples, any variable surrounded with %%% that is not already defined in pre-defined variables, need to either be defined in your post action XML or replaced with the actual values you intend to use.

Moving clean file and infected files to different folder

```
<post_action>
            <scripts>
                        <if type="0">move %%%file_path%%% c:
\archive_data\clean</if>
                        <if type="1">move %%%file_path%%% c:
\archive_data\infected</if>
            </scripts>
</post_action>
```

Uploading infected file to ftp server

```
<post_action>
    <scripts>
        <if type="1">
echo OPEN %%%server%%%&gt; ftp.scr&amp;&amp;
echo %%%username%%%&gt;&gt; ftp.scr&amp;&amp;
echo %%%password%%%&gt;&gt; ftp.scr&amp;&amp;
echo cd test_post_action&gt;&gt; ftp.scr&amp;&amp;
echo put %%%file_path%%%&gt;&gt; ftp.scr&amp;&amp;
echo CLOSE&gt;&gt; ftp.scr&amp;&amp;
echo quit&gt;&gt; ftp.scr &amp;&amp;
ftp -s:ftp.scr
        </if>
    </scripts>
    <user_vars>
        <user_var name="server">127.0.0.1</user_var>
        <user_var name="username">ftp_user</user_var>
        <user_var name="password">1234abcd</user_var>
    </user_vars>
</post_action>
```

RAM Disk Configuration

Overview

You can configure the RAM Disk (or RAM Drive) from the Windows device property dialog box.

The RAM Disk turns a chunk of your system's RAM into a disk drive. The RAM Disk is more secure than a fixed disk because the contents of the disk will be wiped clean on a system reboot. It provides speed optimization because the AV scanners being used do not have to fetch the data from a physical disk. The file is already stored in RAM.

Below are two use cases:

1. Temporary directory for extracting archive files before scanning
    - Set the temp_dir property of Metadefender Core to RAM Disk
2. As a temp directory for your application
    - You could use the RAM Disk if your application accepts files which are uploaded by your customers and you want to perform some temporary actions on that content before scanning.

RAM Disk as Temporary Directory

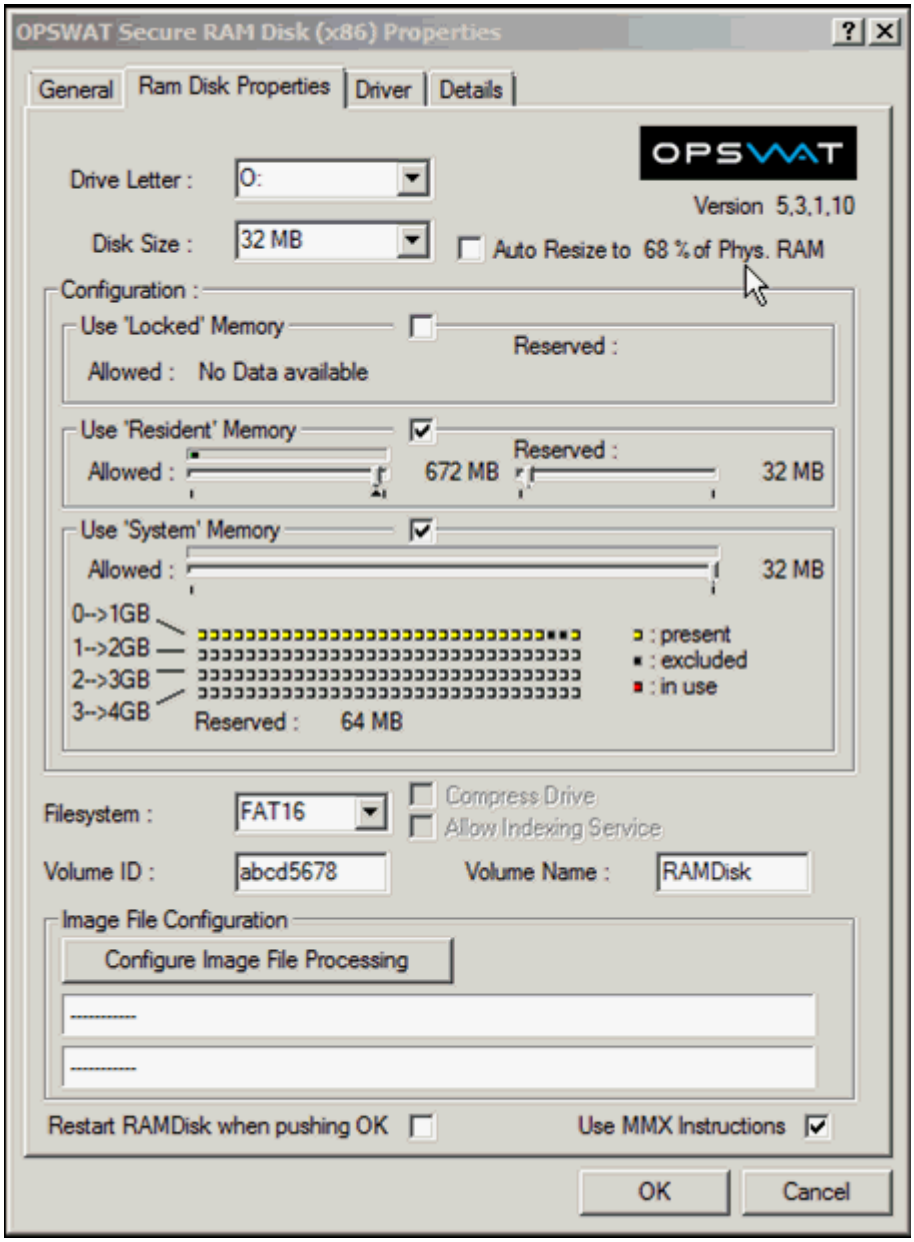
When you use the RAM Disk as a temporary directory, Metadefender Core uses this directory for the following purposes:

- Extracting archive files (when “internal_archive_lib” is set to “1” (true))
- Writing data to a file in this directory if using any of the Metadefender Core Buffer (stream) APIs
  Note: If the buffer is bigger than the available RAM Disk space, scan APIs will return E_FAIL.
- If the RAM Disk is used as the primary temp directory, the maximum extracted archive file size will be half the size of the RAM Disk for a single archive. If an archive exceeds that size, an error will be returned unless a secondary temp file is configured. If a secondary temp file is configured, then the archive will use it to extract the archive. The size of the secondary temp files is always half the available space on the drive.

Device Property

In order to change the RAM Disk properties, open the Device Manager using the following steps:

1. Go to **Start > Run** and enter devmgmt.msc, and then press **Return**.
2. Select the **Ram Drive** category.
3. Right-click on OPSWAT Secure RAM Disk and select **Properties**.



REST Server Configuration

Enabling IIS Logging

By default, Internet Information Services (IIS) logging is disabled in the Metadefender Core REST server. In order to enable IIS logging, please follow these steps:

1. Go to the the REST\Config folder inside Metadefender Core's installation directory and open the applicationhost.config file. For example: C:\Program Files(x86)\OPSWAT\Metadefender Core X\REST\Config\applicationhost.config.

2. Uncomment these 2 properties in the applicationhost.config file.

```
<! --   <add name="HttpLoggingModule" image="%IIS_BIN%\loghttp.
dll" />   -->
<! --   <add name="HttpLoggingModule" lockItem="true" />   -->
```

Restart the Metadefender Core REST server using the following steps:

Using the Windows Control Panel:

1. Go to **Start > Administrative Tools** and click **Services** . (In Windows 10, open the Control Panel and click **System and Security** > **Administrative Tools** and then double-click **Services** .)
2. In the Services panel, right-click on Metascan REST Service and click **Start** .

Using the command line:

1. Open a command prompt.
2. To stop the service, execute the ‘net stop omsrest’ command.
3. To restart the service, execute the ‘net start omsrest’ command.

Load Balancing With Multiple Instance

When multiple instances of Metadefender Core are serving multiple clients with higher load than one Metadefender Core can handle, clients should make progress (file\<data_id>) on a specific server. Setting this value results in initial file upload response to provide rest_ip information so that the client can make follow up requests to this specific server. You can configure this in the Metascan registry key (HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan on x86, HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\OPSWAT\Metascan on x64).

String value	Value	Description	
rest_ip	integer	<IP address or URL> that client uses	

Storing File on the REST server

The server can be configured so that any data requested for scanning can be stored. This should be configured on the Metascan registry key (HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan on x86, HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\OPSWAT\Metascan on x64). You must restart the Metadefender Core REST service after this key has been updated for the change to take effect.

String value	Value	Description	
rest_filedb_type	NONE or mongoDB	Set to mongoDB to enable storing files, the default is NONE. This key does not exist by default and must be created on the Metadefender Core system.	
rest_store_file	0 or 1	Set to 1 to enable storing files, the default is 0.	

Enabling storing file on the server side will consume disk spaces rapidly depending on the load. You must plan for enough disk spaces depending on the load. You can monitor disk space usage on management console dashboard.

Whitelisting/Blacklisting

You can specify a whitelist and/or blacklist for Metadefender Core to use in scanning via the omsFilterCLI.exe tool. The default path of this tool is C:\Program Files (x86)\OPSWAT\Metadefender Core 1/4/8/12/16\omsFilterCLI.exe.

Whitelist: Files that will not be scanned and will automatically be marked as clean.

Blacklist: Files that will not be scanned and will automatically be marked as a threat.

Command	Description
addtoblack <path> <type> <threat>	Add the file to the blacklist
addtowhite <path> <type>	Add the file to the whitelist
list <type>	Retrieve whitelist and blacklist
about	Shows the info of the current version

Argument	Description
<type>	0: all, 1: archive library, 2: engines
<threat>	Name of threat for reference
<path>	Absolute file path

2.9. Non-Workflow Configuration (Deprecated)

Caching and ScanEx configurations are no longer recommended by OPSWAT, but may be used under certain circumstances. Please contact OPSWAT customer support (https://www.opswat.com/support#support-levels) before using one of these methods.

There is no caching needed in the workflow API, because each process request is treated as new. There are a lot data processing in the workflow (e.g., file analyze, data sanitization, etc.). The workflow covers all ScanEx functions as well.

Caching

The caching configuration determines the size and behavior of the Metadefender Core scan result cache. The cache allows Metadefender Core to return results from a previous file scan with Metadefender Core if those scan results were cached and have not yet expired. Caching can only be configured through omsCmdLineUtil.exe.

The following settings apply if caching is enabled:

Property	Description	Default Value	CLI config
Enable Caching	This enables Metadefender Core's scan result caching.	Disabled	cs=<0|1>
Reset cache on engine update	Specifies whether the cache should be reset (and all cached results discarded) whenever there has been an engine definition update.	Disabled	rc=<0|1>
Cache Expiration		72 hours	

Property	Description	Default Value	CLI config
	Specifies how long (in hours) cached results remain available after a scan. After this time the cached results will be deleted.		kc=<time in seconds>
Maximum Cache Size	The maximum size of the cache in RAM. This is only the amount of space required to store the cached scan results, not the files themselves.	100 MB	ms=<size in MB>

To manually reset the cache, do the following.

1. Stop the Metadefender Core service (net stop metascan).
2. Delete the cache database (<installation directory>\Data\omsCK_KN.db3).
3. Restart the Metadefender Core service (net start metascan).

ScanEx Configuration

This configuration is used when calling the Metadefender Core APIs that do not use the Metadefender Core workflows, and only scan files. This includes files scanned through Metadefender Core's ICAP interface, files scanned with Metascan Client, files scanned through the Java API, and the sample code provided on the OPSWAT Portal.

Archive handling

The archive handling configuration options determine how archives are handled.

If archive handling is enabled, Metadefender Core will extract archives and scan the individual files within the archive.

Most common archive formats are supported, including 7z, XZ, BZIP2, GZIP, TAR, ZIP, WIM, ARJ, CAB, CHM, CPIO, CramFS, DEB, DMG, FAT, HFS, ISO, LZH, LZMA, MBR, MSI, NSIS, NTFS, RAR, RPM, SquashFS, UDF, VHD, WIM, XAR and Z. Metadefender Core can also extract self-extracting archives created by both 7zip and WinRAR.

The following settings apply if archive handling is enabled:

Property	Description	Default Value	CLI config	Additional info
Enable Archive	This enables Metadefender Core's archive library handling.	Enabled	le=<0|1>	
Max Recursion Level	The maximum depth that Metadefender Core will continue to extract archives for scanning. Once this depth is reached, Metadefender Core will not extract further archives but will scan those archives as entire files. If this is set to 0, archives will not be extracted.	5	rl=<levels>	Maximum value: 2147483646
Number of Files	The maximum number of files that can be in an archive that Metadefender Core is extracting. If the number of files in an archive exceeds this value, Metadefender Core will return the result as a potential threat.	50	an=<number>	Maximum value: 2147483646
Total Size	The maximum total size of files that can be in an archive that Metadefender Core is extracting. If the total size of files in an archive exceeds this value, Metadefender Core will return the result as a potential threat.	2 GB	as=<size in MB>	Maximum value: Half the current available free space of the

Property	Description	Default Value	CLI config	Additional info
				Metadefender Core temporary directory. If two temporary directories are set from different drives, the highest available space will be used.
Simultaneous	Specifies if multiple archive files undergo extraction concurrently. This may improve performance on a multi-core CPU, but means that the RAM-drive size should be increased (since more unpacked archives may reside on it at the same time).	Disabled	ec=<0|1>	
Self-Extracting	Specifies if self-extracting archives should be extracted and treated as archives	Disabled	sx=<0|1>	
Scan Original Un-extracted File	In addition to scanning files inside of an archive after extraction, un-extracted archives are sent directly to engines for scanning. **Note:** If “extract_archive” for an engine is enabled, this potentially exposes	Disabled	soa=<0|1>	

Property	Description	Default Value	CLI config	Additional info
	performance overhead since extraction happens twice, once by Metadefender Core and once by the engine.			

Note: DOCX and DOCM files can be detected as archive files. OPSWAT recommends that the option to scan the original un-extracted archive is enabled so that these files are properly scanned.

Post processing configuration

The Post Processing configuration allows an administrator to specify a script that should be executed for all allowed and/or blocked files. By clicking on 'Run custom command line script', a message can be put in place for both infected and clean files.

An administrator can also choose to delete or quarantine infected files.

3. Developer Guide

OPSWAT is committed to making the Metadefender Core API backward compatible to any older version of Metadefender Core. With the growing capabilities and features of Metadefender Core, it is necessary to add new APIs, new COM Connection Points, and new return types. When you upgrade your integration to Metadefender Core to a newer version, please keep the following factors in mind:

C/C++ application: By the design of COM, all Connection Points are required to be implemented in your application. Please check Metascan.idl for newly added Connection Points.

Building Custom Engines

You can develop your own custom engine integration. We provide documentation on how to do this as well as sample code in C & C#.

What you need to do

1. Install Metadefender Core 3.8.0 or newer.
2. Download the custom engine template package here. This download includes the following templates:
   • Template for C custom engine (vc10)
   • Template for C# custom engine (vc10)
3. Obtain Metadefender Core custom engine handler (omsCEHandler.exe). This file is included in the template package's bin folder.
4. Implement the custom engine interface.
5. Build the custom engine library (DLL).
6. Deploy the custom engine library (DLL) with the Metadefender Core custom engine handler (omsCEHandler.exe).

Installation and configuration

This section covers the configuration of Metadefender Core in order to load and use your custom engine. The following properties must be defined and configured in the omsConfig.ini, located in the Metadefender Core installation directory.

Section or attribute	Value	Description
[custom_engine_*]**	Replace *** with a unique identifier for your engine. (e.g., [custom_engine_myav] or [custom_engine_mycompany])	A unique section name to group the settings for your custom engine. This section must be unique and begin with “custom_engine”. If you do not name the section appropriately, Metadefender Core will not detect your custom engine.
custom_engine_dir	Absolute path to the directory. Do not end this path with a trailing separator. e.g., c:\my Custom Engine AV	The directory where your custom engine binaries reside. At a minimum, this includes omsCEHandler.exe and the C interface DLL.
custom_unique_id	4-10 characters starting with “ce” prefix e.g., ce01	This is used for communication between Metadefender Core and the omsCEHandler processes. If you have multiple custom engines, this must be unique for each engine.
custom_dll_name	Custom engine DLL name e.g., sampleCustomEng.DLL	The sample custom engine project generates “sampleCustomEng. DLL” by default, but you can modify as you wish.
custom_engine_type	Numerical value 0 – Indicates the engine is a scan engine	
support_multithread	Numerical Value. 0 – No 1 – Yes	You must set this attribute to 0 if your custom engine is not thread-safe. If you are not sure, set to 0.

Example

The following is an example of the properties that need to be configured for the omsConfig.ini file.

```
custom_engine_myav
custom_engine_dir= c:\my Custom Engine AV
custom_dll_name=sampleCustomEng.dll
custom_engine_type=0
support_multithread=0
custom_unique_id=ce01
```

When uninstalling or upgrading Metadefender Core

Any configuration pertaining to the custom engines will not be preserved if you uninstall and reinstall Metadefender Core. You must backup and then restore the following changes after your installation or upgrade is complete:

- omsConfig.ini (changes made pertaining to the custom engines)
- Any custom engine directories located under the Metadefender Core installation folder

C# Custom Engine Template

Title/API	Function
NReloadDatabase	This method provides a way for Metadefender Core to explicitly reload the definition database of the engine, especially for a malware engine. During this call, any requests to scan should return ‘Not Scanned.’ If there is no specific step required for reloading the database, it should still return "0" to indicate that the engine is ready.
NGetDefinitionInfo	This method is used to retrieve additional attributes about the definition files in use by the custom engine.
NGetDefinitionTime	This is used to retrieve the timestamp of the definition files currently in use by the custom engine. Memory is allocated by Metadefender Core.
NPerformUpdate	This is called to initiate an update of the custom engine. This call should be a synchronous call and wait until the update completes.

Title/API	Function
NScan	This is the main function of the custom engine. The scan result returned from this function call is used by Metadefender Core's scan-related API.
NGetProductName	This method provides the name of the custom engine. Metadefender Core uses this for display purposes.
NDeinitCE	This is called when Metadefender Core is shut down. Any resources acquired by your engine will be released and any unsaved data will be stored.
NInitCE	This is the first method called. It allows you to initialize your engine. Metadefender Core will recognize your engine only if this method returns successfully (returns "0").

Additional steps for VB/C# custom engine

1. Place C wrapper DLL (Dynamic Link Library) with the custom engine DLLs.

If you compile the VB sample solution or C# solution, you get 2 DLLs, one for the cwrapper provided by OPSWAT and the other for your custom engine.

NDeinitCE

API	NDeinitCE
Function	This is called when Metadefender Core is shut down. Any resources acquired by your engine will be released and any unsaved data will be stored.

Function prototype

```
int NDeinitCE
(
)
```

Arguments

No arguments are required for this function.

NGetDefinitionInfo

API	NGetDefinitionInfo
Function	This method is used to retrieve additional attributes about the definition files in use by the custom engine.

Function prototype

```
int NGetDefinitionTime
(
        ref string reserved,
        ref string defVer,
        ref string defSig,
        ref string engVer,
)
```

Arguments

Argument	Description
defVer	Definition version you would like to report. Metadefender Core does not use this information for any logic. It is used only for display purposes.
defSig	Definition signature count you would like to report. Metadefender Core does not use this information for any logic. It is used only for display purposes.
engVer	Engine version you would like to report. Metadefender does not use this information for any logic. It is used only for display purposes.
reserved	Reserved for future use.

Return value

Value	Description
0	Information successfully retrieved.

Value	Description
1	Information unavailable.

NGetDefinitionTime

API	NGetDefinitionTime
Function	This is used to retrieve the timestamp of the definition files currently in use by the custom engine. Memory is allocated by Metadefender Core.

Function prototype

```
int NGetDefinitionTime
(
        ref int defYear,
        ref int defMon,
        ref int defDay,
        ref int defHour,
        ref int defMin,
)
```

Arguments

Argument	Description
defYear	Year portion of definition time.
defMon	Month portion of definition time.
defDay	Day portion of definition time.
defHour	Hour portion of definition time.
defMin	Minute portion of definition time.

Return value

Value	Description
0	Information successfully retrieved.
1	Information unavailable.

NGetProductName

API	NGetProductName
Function	This method provides the name of the custom engine. Metadefender Core uses this for display purposes.

Function prototype

```
int GetProductName
(
ref string productName
)
```

Arguments

Argument	Description
productName	The name of the custom engine.

Return value

Value	Description
0	To indicate that the product name is successfully returned.
Other value	Not supported. Reserved for future usage.

NInitCE

API	NInitCE
Function	This is the first method called. It allows you to initialize your engine. Metadefender Core will recognize your engine only if this method returns successfully (returns "0").

Function prototype

```
public int NInitCE
(
)
```

Arguments

No argument.

Return value

Value	Description
0	To indicate that the engine initialized correctly.
1	To indicate initialization failed and the engine should not be used.

NPerformUpdate

API	NPerformUpdate
Function	This is called to initiate an update of the custom engine. This call should be a synchronous call and wait until the update completes.

Function prototype

```
int PerformUpdate
(
```

```
)
```

Return value

Value	Description
0	Update completed successfully.
1	Failed to start or finish updating.
2	Engine is already up to date.
3	Failed to update due to network problem.
4	Engine is already updating.
5	Failed downloading update.
6	Failed reloading database.

NReloadDatabase

API	NReloadDatabase
Function	This method provides a way for Metadefender Core to explicitly reload the definition database of the engine, especially for a malware engine. During this call, any requests to scan should return ‘Not Scanned.’ If there is no specific step required for reloading the database, it should still return "0" to indicate that the engine is ready.

Function prototype

```
int ReloadDatabase
(
ref string srcDir,
)
```

Arguments

Argument	Description
srcDir	If specified, path to the folder where new definition files are placed.

Return value

Value	Description
0	Database is successfully reloaded or nothing needs to be done to reload database.
1	Fail to reload database (same result as fail to init).

NScan

API	NScan
Function	This is the main function of the custom engine. The scan result returned from this function call is used by Metadefender Core's scan-related API.

Function prototype

```
int NScan
(
        int             reserved,
        string          filepath,
        ref int         scanResult,
        ref string      threatName,
)
```

Arguments

Argument	Description
reserved	Reserved for future.
filepath	Path to the file needed to scan.

Argument	Description
scanResult	Scan result to be returned. • 0: clean • 2: suspicious • 1: found threat • 3: failed to scan • 10: not scanned (during update)
threatName	Threat name to be returned.

Return value

Value	Description
0	Scan completed successfully.
1	Failed to start or finish scanning.

Custom Engine C Interface

Interface

API	Function
FreeString	This is called to free allocated memory for wchar **. Metadefender Core calls this after Metadefender Core is finished with the values allocated by your functionality.
GetDefinitionInfo	This method is used to retrieve additional attributes about the definition files in use by the custom engine.
GetDefinitionTime	This is used to retrieve the timestamp of the definition files currently in use by the custom engine. Memory is allocated by Metadefender Core.
GetEngineProperty	

API	Function
	This method is used to get engine properties such as extract archive, heuristic.
GetExpirationDate	This method is used to retrieve the expiration date of engine.
GetProductName	This method provides the name of the custom engine. Metadefender Core uses this for display purposes.
InitCE	This is the first method called. It allows you to initialize your engine. Metadefender Core recognizes your engine only if this method returns successfully (returns 0).
PerformUpdate	This is called in order to initiate an update of the custom engine. The call is synchronous and waits until the update completes.
ReloadConfig	This is used to reload any or all specific engine configurations. If there is no specific engine configuration that needs to be reloaded, this function returns "0" without any re-initialization of engines.
ReloadDatabase	This method provides a way for Metadefender Core to explicitly reload the definition database of the engine, especially for a malware engine. During this call, any requests to scan return "Not Scanned." If there is no specific step required for reloading the database, it still returns "0" to indicate that the engine is ready.
Scan File	This is the main function of the custom engine. The scan result returned from this function call is used by Metadefender Core's scan-related API.
SetEngineProperty	This method is used to set engine properties such as extract archive and heuristic.

DeinitCE

API	DeinitCE
Function	This is called when Metadefender Core is shut down. Any resources acquired by your engine are released and any unsaved data is stored.

Function prototype

```
int DeinitCE ()
```

Arguments

No arguments are required for this function.

Return value

Value	Description
0	Indicates that the engine deinitialized correctly.
1	Indicates that deinitialization has failed.

FreeString

API	FreeString
Function	This is called to free allocated memory for wchar **. Metadefender Core calls this after Metadefender Core is finished with the values allocated by your functionality.

Function prototype

```
void FreeString
(
     wchar ** stringToFree
```

```
)
```

Arguments

Argument	Description	Other
stringToFree	Double pointer to wchar	Must be deallocated in the same way that memory is allocated.

GetDefinitionInfo

API	GetDefinitionInfo
Function	This method is used to retrieve additional attributes about the definition files in use by the custom engine.

Function prototype

```
int GetDefinitionTime
(
     wchar_t** reserved,
     wchar_t** defVer,
     wchar_t** defSig,
     wchar_t** engVer,
     void** reserved2
)
```

Arguments

Argument	Description	Other
defVer	Definition version you would like to report. Metadefender Core does not use this information for any logic. It is used only for display purposes.	Implementer must allocate the memory required.
defSig	Definition signature count you would like to report. Metadefender Core does not use this information for any logic. It is used only for display purposes.	Implementer must allocate the memory required.

Argument	Description	Other
engVer	Engine version you would like to report. Metadefender Core does not use this information for any logic. It is used only for display purposes.	Implementer must allocate the memory required.
reserved, reserved2	Reserved for future use.	

Important: defVer, defSig, engVer are double pointers to wchar_t or NULL. They must be allocated by the custom engine in a way that can be deallocated by the FreeString function unless NULL pointer is passed.

Return value

Value	Description
0	Information successfully retrieved.
1	Information unavailable.

GetDefinitionTime

API	GetDefinitionTime
Function	This is used to retrieve the timestamp of the definition files currently in use by the custom engine. Memory is allocated by Metadefender Core.

Function prototype

```
int GetDefinitionTime
(
     int * defYear,
     int * defMon,
     int * defDay,
     int * defHour,
     int * defMin,
     void** reserved
)
```

Arguments

Argument	Description
defYear	Year portion of definition time
defMon	Month portion of definition time
defDay	Day portion of definition time
defHour	Hour portion of definition time
defMin	Minute portion of definition time

Return value

Value	Description
0	Information successfully retrieved.
1	Information unavailable.

GetExpirationDate

API	GetExpirationDate
Function	This method is used to retrieve the expiration date of engine.

Function prototype

```
int GetExpirationDate
(
     int *licYear,
     int *licMon,
     int *licDay,
     void** reserved
)
```

Arguments

Argument	Description
licYear	Year portion of license time.
licMon	Month portion of license time.
licDay	Day portion of license time.
reserved	Reserved for future use.

Return value

Value	Description
0	Information successfully retrieved.
1	Information unavailable.

GetProductName

API	GetProductName
Function	This method provides the name of the custom engine. Metadefender Core uses this for display purposes.

Function prototype

```
int GetProductName
(
    wchar_t ** productName
    void ** reserved
)
```

Arguments

Argument	Description	Other
productName	The name of the custom engine	Implementer must allocate the memory required.
reserved	Reserved for future use	

Return value

Value	Description
0	To indicate that product name is successfully returned.
Other value	Not supported. Reserved for future use.

InitCE

API	InitCE
Function	This is the first method called. It allows you to initialize your engine. Metadefender Core recognizes your engine only if this method returns successfully (returns 0).

Function prototype

```
int InitCE
(
    void **reserved
)
```

Arguments

Argument	Description
reserved	Reserved for future usage

Return value

Value	Description
0	To indicate that the engine initialized correctly.
1	To indicate initialization failed and the engine should not be used.

PerformUpdate

API	PerformUpdate
Function	This is called in order to initiate an update of the custom engine. The call is synchronous and waits until the update completes.

Function prototype

```
int PerformUpdate
(
    void ** reserved
)
```

Arguments

Argument	Description
reserved	Reserved for future use.

Return value

Value	Description
0	Update completed successfully.
1	Failed to start or finish updating.
2	Engine is already up to date.
3	Failed to update due to network problem.
4	Engine is already updating.
5	Failed downloading update.
6	Failed reloading database.

ReloadConfig

API	ReloadConfig
Function	This is used to reload any or all specific engine configurations. If there is no specific engine configuration that needs to be reloaded, this function returns "0" without any re-initialization of engines.

Function prototype

```
int ReloadConfig
(
     const wchar_t * reservedI,
     void** reservedIO
)
```

Arguments

Argument	Description
reservedI, reservedI O	Reserved for future use

Return value

Value	Description
0	Successfully reloaded configuration.
1	Failed to reload configuration.

ReloadDatabase

API	ReloadDatabase
Function	This method provides a way for Metadefender Core to explicitly reload the definition database of the engine, especially for a malware engine. During this call, any requests to scan return "Not Scanned." If there is no specific step required for reloading the database, it still returns "0" to indicate that the engine is ready.

Function prototype

```
int ReloadDatabase
(
	const wchar_t * srcDir,
	void ** reserved
)
```

Arguments

Argument	Description
srcDir	If specified, path to the folder where new definition files are placed.
reserved	Reserved for future use.

Important: If srcDir is used, the folder and its contents must be consumed and deleted by engine.

Return value

Value	Description
0	Database is successfully reloaded or nothing needs to be done to reload the database.
1	Failed to reload database (same result as fail to init).

Scan File

API	Scan
Function	This is the main function of the custom engine. The scan result returned from this function call is used by Metadefender Core's scan-related API.

Function prototype

```
int Scan
(
    int dataType,
    const void * data,
    int * scanResult,
    wchar_t ** threatName,
    void ** reserved
)
```

Arguments

Argument	Description
dataType	0: if data is a file path string (wide characters) Other dataType values are not supported with 3.2.5
data	When dataType indicates a file path, you must cast to a const wchar_t pointer. e.g., wstring filePathToScan = (const wchar_t *)data;
scanResult	Scan result to be returned: • 0: clean • 1: found threat • 2: suspicious • 3: failed to scan • 10: not scanned (during update)
threatName	Threat name to be returned.
reserved	Reserved for future.

Important: threatName is a double pointer to wchar_t. It must be allocated by the custom engine in a way that can be deallocated by the FreeString function.

Return value

Value	Description
0	Scan completed successfully.
1	Failed to start or finish scanning.

COM Interface

The Metadefender Core COM server externalizes COM (Microsoft's Component Object Model) interfaces. The interfaces are all based on the IDispatch automation interface making integration possible from many scripting languages.

When the Metadefender Core Server starts (this happens when you log in or upon the first invocation of the server from a client), it starts updating all the supported antivirus engines on the system. The server repeats the update process at configurable intervals.

Before any method is called, a client instance must be initialized by calling Init or InitEx (deprecated). For APIs via callbacks, please refer to COM Connection Points.

Title	Note	Method	Description
Stop ongoing scan (COM)		StopScan	This method purges all scan requests and ongoing scans. Method returns only when all requests are purged.
Sanitize a file (COM)		ConvertFileType	This method converts the specified input file into a different format.
Analyze file type III (COM)		AnalyzeEx	This method is an extensible file type analysis API that allows the user to set various analyze file type options that can be configured for each file type analysis request. It can be used as a wrapper of the other file type analysis APIs. It also supports specifying custom ticket ID and session ID to associate file type analysis and scan.
Analyze file type II (COM)		AnalyzeFileTypeEx2	This method analyzes a file (or buffer) and attempts to guess its type, detects files of which the

Title	Note	Method	Description
			file extensions may have been altered, and recursively analyzes contents of archive files.
Get an engine info (COM)		GetUpdateStatusEx	This method returns information about the virus definition files of the requested product.
Set a property value (COM)		SetProperty	Modify one of Metadefender Core's internal properties.
Get a property value (COM)		GetProperty	Reads one of the internal properties.
Update an anti-malware engine (COM)		UpdateEngine	This method is called to initiate an update of a specific anti-malware engine on the local machine.
Update anti-malware engines (COM)		UpdateAntiViruses	This method is called to initiate an update of all anti-malware engines at the same time on the local machine. It is recommended to perform updates via automatic periodic update, which is performed at predetermined intervals as configured in “Update Interval” property.
Scan a file (COM)		ScanEx	This method is an extensible scan API which allows the setting of various scan options that can be configured for each scan request.

Title	Note	Method	Description
Process a file (COM)		Process	This method processes a file through a workflow based on the specified user agent.
Initialize (COM)		Init	This method is called to initialize the connection to the Metadefender Core service. This method needs to be called before any other method can be called.
Callback for Processing a File (COM)	Connection Points	OnProcess	This callback is fired after each Process method when a pre-defined event is found.
Callback For Archive File Info (COM)	Connection Points	OnFileInArchiveInfoAvailable	This callback method is invoked when Metadefender Core's archive library is enabled and a file from within an archive is analyzed (not a scan result). This callback can be used to obtain more discrete information of a file contained within an archive.
Callback For Scan Started Report (COM)	Connection Points	ScanStartReport	This callback method is invoked when each engine begins the scan. It is used when a scan request begins. This callback is fired on if “report_on_scan_start” is set to “1” (true).
Callback For Update Completion (COM)	Connection Points	MyOnUpdateComplete	This callback method is invoked when an update completes. It is fired to all Metadefender Core client objects that are registered to this connection point. However, if there is already an

Title	Note	Method	Description
			update request that has not completed, only the Metadefender Core client that requested the update will get a callback with the update result of “updateAlreadyRequested(2)”.
Callback For Update Progress Report (COM)	Connection Points	MyOnUpdateProgressReport	This callback method is invoked when an update for each engine completes. It is fired to the Metadefender Core client object that initiates the update.
Callback For Additional Scan Progress Report (COM)	Connection Points	MyScanProgressReport	This callback method is invoked when each engine completes the scan. The callback can be used for a scan progress report when more than one engine is used. If an engine is not used for any reason such as encrypted archive or scan request on a known file inside the folder, no progress report will be fired for the specific file.
Callback For Scan Progress Report (COM)	Connection Points	MyScanProgressReport	This method is called to initialize the connection to the Metadefender Core service. This method needs to be called before any other method can be called.
Callback for Scan Completion (COM)	Connection Points	MyOnScanComplete	This connection point is fired whenever a scan is completed. This event is not global, but is specific to the client that makes this scan request. In other words, when scanning is

Title	Note	Method	Description
			requested by a client, other client objects will not get a callback when the scanning is completed.
Unsubscribe from global events (deprecated)	deprecated	UnsubscribeGlobalEvents	This method is called to unsubscribe to specific or all events. If you subscribe to a specific type of event, you must unsubscribe to that type of event. For example, if you subscribe to scan event type and update event type, you must unsubscribe to both scan event type and update event type. Calling to unsubscribe to events to which you are not subscribed has no effect.
Subscribe to global events (deprecated)	deprecated	SubscribeGlobalEvents	If you want to receive global callbacks such as OnLogAvailable, you must subscribe to global events with an appropriate event type through this call. If you subscribe to a specific type of event, you must unsubscribe to that type of event. For example, if you subscribe to a scan event type and update event type, you must both unsubscribe to the scan event type and update the event type.
Analyze file type (deprecated)	deprecated	AnalyzeFileType	This method analyzes the contents of a file and attempts to guess its type.

Title	Note	Method	Description
Get updated virus definition files (deprecated)	deprecated	GetUpdateStatus	This method returns information about the virus definition files of the requested product.
Put in scan-and-clean queue (deprecated)	deprecated	PutToScanAndCleanQueue	This method places a file to the scan-and-clean queue. In other words, the scan request will be processed asynchronously.
Put in scan queue (deprecated)	deprecated	PutToScanQueue	This method is called to place a file to the scan-queue. In other words, the scan request will be processed asynchronously.
Scan and clean file (deprecated)	deprecated	ScanAndClean	This method is called to scan and clean a file. This is done in a synchronous (blocking) fashion - the method returns only after the scan is complete.
Scan file (deprecated)	deprecated	Scan	This method is called to scan a file or a memory buffer. This is done in a synchronous (blocking) fashion - the method returns only after the scan is complete.
Initialize object (deprecated)	deprecated	InitEx	This method can still be used for the purpose of initializing an instance of a Metadefender Core object. However, with more flexible and extensive DB controller, the ability to have its own log writer per instance is disabled.

Important Return Type

Scan outcome Return Type

Return value	Description	Note
0	No threat found	No threat detection or the file is empty.
1	Infected /Known	Threat is found.
2	Suspicious	Classified as possible threat but not identified as specific threat.
3	Failed To Scan	Scanning is not fully performed (For example, invalid file or no read permission). If no engine is included and scan is enabled, this will be the final result.
4	Cleaned	Threat is found and file is cleaned (repaired or deleted).
5	Unknown	Unknown scan result.
6	Quarantined	File is quarantined.
7	Skipped Clean	Scan is skipped because this file type is in whitelist.
8	Skipped Dirty	Scan is skipped because this file type is in blacklist.
9	Exceeded Archive Depth	Threat is not found, but there are more archive levels that were not extracted. This is affected by the Metadefender Core property, ‘internal_archive_recursive_level’.
10	Not Scanned	Scan is skipped by the engine, either due to update or other engine specific reason. If scan is disabled, this is the final result.
11	Aborted	All ongoing scans are purged by StopScan API call.

Return value	Description	Note
12	Encrypted	File/buffer is not scanned because the file type is detected as encrypted (password-protected). If the Internal Archive Library is ON, *encrypted* return type is not going to be returned through Metadefender Core scan progress callbacks because the engines do not perform any scan operations. If the Internal Archive Library is OFF, Metadefender Core passes the *encrypted* files to the engines directly, bypassing the detection.
13	Exceeded Archive Size	The extracted archive is larger than set in the maximum file size for archive. (For more details, refer to “archive_lib_max_size” properties in the Config topic.)
14	Exceeded Archive File Number	There are more files in the archive than set in the maximum number of files extracted. (For more details, refer to “archive_lib_max_num_files” properties in the Config topic.)
15	Password Protected Document	Only workflow has this result.

Others

Return Type	Values	API/Connection points which use this return type
ThreatList	A threat list found on the scanned object, otherwise null (Threat name should NOT be used in a way that affects the application logic. For example, a threat name can be an empty string)	Scan ScanEx ScanAndClean PutToScanQueue PutToScanAndCleanQueue
FileTypeShort	“E” – Executable (EXE, DLL, …)	

Return Type	Values	API/Connection points which use this return type
	“D” – Document (MS Office word document, MS Office excel sheet) “A” – Archive (Zip, Rar, Tar, …) “G” – Graphical format (Jpeg, GIF, TIFF, BMP, …) “F” – Folder “Y” – Logical drive “I” – Disk image “T” – Text “P” – PDF format “M” – audio or video format “Z” – mail messages (MSG, …) “O” – Other (anything that is not recognized as one of the above) **Note:** An ISO is treated as an archive file type (type “A”) and not as a disk image (type “I”).	

COM Connection Points

The Metadefender Core server provides callback methods by allowing client objects to register connection points. In order to use Metadefender Core API in C/C++ development, all connection points must be implemented. For other languages such as C# or ASP.net , only desired methods can be implemented.

Callback For Additional Scan Progress Report (COM)

Method	MyScanProgressReport
Description	This callback method is invoked when each engine completes the scan. The callback can be used for a scan progress report when more than one engine is used. If an engine is not used for any reason such as encrypted archive or scan request on a known file inside the folder, no progress report will be fired for the specific file.
Note	Connection Points

Function prototype

```
void MyScanProgressReport
(
    VARIANT ticket,
    VARIANT result,
    VARIANT threatList,
    VARIANT productName,
    VARIANT scanStartTime
    VARIANT scanEndTime
    VARIANT filePath
)
```

Arguments

Argument	Description	Data Type
ticket	The ticket which is generated when this scan has been requested.	UINT32
result	A UINT32 value that specifies the scan outcome.	UINT32
threatList	An array of strings(String []). Each string element in the array is the name of the virus which has been detected.	Array of strings

Argument	Description	Data Type
productName	The description of the product.	string
scanStartTime	The time when the scan is started.	date
scanEndTime	The time when the scan is finished.	date
filePath	The path to the file which is scanned.	string

Callback For Archive File Info (COM)

Method	OnFileInArchiveInfoAvailable
Description	This callback method is invoked when Metadefender Core's archive library is enabled and a file from within an archive is analyzed (not a scan result). This callback can be used to obtain more discrete information of a file contained within an archive.
Note	Connection Points

Function prototype

```
void OnFileInArchiveInfoAvailable
(
       VARIANT SessionID,
       VARIANT Ticket,
       VARIANT FileName,
       VARIANT MD5,
       VARIANT SHA1,
       VARIANT SHA256,
       VARIANT FileSize,
       VARIANT FileTypeDesc,
       VARIANT Reserved
)
```

Arguments

Argument	Description	Data Type
SessionID	Reserved for future use.	
Ticket	The ticket that is generated when this scan has been requested.	UINT 32
FileName	The name of the file contained in the archive with the file path of the root archive preceding it.	string
MD5	MD5 signature of the file.	string
SHA1	SHA1 signature of the file.	string
SHA256	SHA256 signature of the file.	string
FileSize	Reserved for future use.	
FileTypeDesc	Reserved for future use.	
Reserved	Reserved for future use.	

Callback for Processing a File (COM)

Method	OnProcess
Description	This callback is fired after each Process method when a pre-defined event is found.
Note	Connection Points

Function prototype

```
void OnProcess
(
    [out] VARIANT *OutDetail,
    [out, retval] VARIANT *OutArgsArray
)
```

Arguments

Argument	Description	Data Type
OutDetail	The details from this event. The JSON schema depends on the callback type. See ‘‘callback types’’ for more details.	string
OutArgsArray	0: CallBackType: Since OnProcess is a general callback that is fired for different events, this specifies the event type. See the callback types for details of the type. 1: Ticket: The ticket returned from the Process API. For example 974989300. 2: The final result of processing the specified file through a workflow 1 = The file was allowed. JSON equivalent is "Allowed" 2 = The file was blocked. JSON equivalent is "Blocked" 3 = The file is currently being processed. JSON equivalent is "Processing"	Safearray of variants with the following order 0: UINT32 1: string 2: UINT32

Callback types

Callback Type	**Description**	Example JSON
2	This callback method is invoked whenever each engine completes the scan.	**Example**

Callback Type	**Description**	Example JSON

	This callback can be used for a progress report of scanning when more than one engine is used. If engines are not used for any reason such as encrypted archive or scan request on a known file, no progress report is fired.	{
		"type": "ScanProgressReport",
		"ticket": 974989300,
		"scan_result": "1",
		"threat_list": [
		"eicar",
		"eicar"
		],
		"product_name": "Clamwin Scan Eng
		"scan_start_time": "6/5/2014 2:32
		"scan_end_time": "6/5/2014 2:32 P
		"scan_time_(ms)": "1",
		"file_path": "C:\\some\\file.txt"
		"process_result": "Processing"
		}

4	This connection point is fired whenever a scan is completed. This event is not global but specific to the client that makes this scan request. In other words, when scanning is requested by a client, other client objects will not get a callback when the scanning is completed.	**Example**
		{
		"type": "OnScanComplete",
		"ticket": 974989300,
		"scan_result": "1",
		"threat_list": [
		"eicar",
		"eicar"
		],
		"scan_start_time": "6/5/2014 2:32
		"scan_end_time": "6/5/2014 2:32 P
		"scan_time_(ms)": "2",
		"file_path": "C:\\some\\file.txt"
		"process_result": "Processing"
		}

7	This callback method is invoked when Process' archive handling is enabled and an archive was successfully extracted.	**Example**

Callback Type	**Description**	Example JSON

	This callback can be used to determine if a processed file is an archive.	{
		"type": "ArchiveFileInfo",
		"ticket": 783676241,
		"file_path": "2files_dirty.zip",
		"process_result": "Processing"
		}

8	This is fired after a file has been fully processed through each stage of a workflow.
	
	FTC Details
	If FTC is enabled and copy/move is disabled, the converted file appears next to the original file.
	If FTC and copy /move are enabled, the converted file location will be reported in "post_processing. copy /move_destination"
	Collisions are handled by incrementing the file name: file_name.txt , file_name_1.txt , file_name_2.txt , ..
		Example
		{
		"type": "ProcessComplete",
		"ticket": "974989300",
		"process_result": "Allowed",
		"file_path": "C:\\some\\file.txt"
		"user_agent": "metascan_rest",
		"profile": "Default",
		"file_type_skipped_scan": false,
		"blocked_reason": ""
		}
		If Post Processing is ran:
		Example
		{
		"type": "ProcessComplete",
		"ticket": "974989300",
		"process_result": "Allowed",
		"file_path": "C:\\some\\toConvert
		"user_agent": "metascan_rest",
		"profile": "Default",
		"file_type_skipped_scan": false,
		"blocked_reason": "",
		"post_processing": {
		"actions_ran": "Converted | C
		"actions_failed": "Convert Fa
		"converted_to": "pdf",

Callback Type	**Description**	Example JSON

	Possible Blocked Reasons
	• Failed to start the processing job: "Process Failed"
	• Mismatch detected: "Mismatch"
	• File type disallowed: "Filtered"
	• Scan Result: "Known | ("Dirty" or "Infected") | Suspicious | Failed | Cleaned | Unknown | ("Skipped Dirty" or "Skipped Infected") | Not Scanned | Aborted | Password Protected Document"
	• Archive: "Encrypted Archive | Missing File During
		<pre>"copy_destination": "C:\\
		}
		}</pre>

Callback Type	**Description**	Example JSON
	Extraction | Insufficient Space For Archive Extraction | Exceeded Archive Size | Exceeded Archive File Number | Archive Extraction Error | Exceeded Archive Depth | Archive Error" • Post Action: "Sanitization Failed" (Copy or Quarantine failure will not cause blocked result) **Possible Process Results** • "Allowed": the file is allowed	

Callback Type	**Description**	Example JSON
	• "Blocked": the file is blocked • "Processing": the file is currently being processed **Possible Action Ran Results** • Sanitized • Copied • Moved • Quarantined • Deleted • Ran Script *Copied and Moved will never exist together *Quarantined and Deleted will never exist together *(Copied/Moved) and (Quarantined /Deleted) will never exist together	
9		**Example**

Callback Type	**Description**	Example JSON

	This callback is fired for AnalyzeFileType. It will be fired for a single file as well as for individual files within an archive.
	
	For a single file it will be a fully qualified path, such as: "C:\\level1\\file.txt"
	
	For a file inside an archive it will be prefixed with \\, such as: "\\level1\\file.txt"
		<pre>{
		"type": "FileTypeAnalyze",
		"ticket": "974989300",
		"file_path": "C:\\some\\toConvert
		"file_size": 5134
		"file_type_category" : "E",
		"file_type_description" : "Win64
		"file_type_extension" : "EXE/DLL"
		"md5": "20A2DFB5EC69BCEA66203CF09
		"sha1": "6D5052FC0ED06C0D975F19D7
		"sha256": "4A7C7077F929041D2D2A18
		"process_result": "Processing"
		}</pre>

Callback for Scan Completion (COM)

Method	MyOnScanComplete
Description	This connection point is fired whenever a scan is completed. This event is not global, but is specific to the client that makes this scan request. In other words, when scanning is requested by a client, other client objects will not get a callback when the scanning is completed.
Note	Connection Points

Function prototype

```
void MyOnScanComplete
(
    VARIANT ticket,
```

```
    VARIANT * result,
    VARIANT * threatList,
    VARIANT * scanStartTime
    VARIANT * scanEndTime
)
```

Arguments

Argument	Description	Data Type
ticket	The ticket that is generated when this scan has been requested.	UINT32
result	A UINT32 value that specifies the scan outcome.	UINT32
threatList	An array of strings (String []). Each string element in the array is the name of the virus which has been detected.	Array of strings
scanStartTime	The time when the scan is started.	Date
scanEndTime	The time when the scan is finished.	Date

Callback For Scan Progress Report (COM)

Method	MyScanProgressReport
Description	This method is called to initialize the connection to the Metadefender Core service. This method needs to be called before any other method can be called.
Note	Connection Points

Function prototype

```
void MyScanProgressReport
(
    VARIANT ticket,
    VARIANT result,
```

```
    VARIANT threatList,
    VARIANT productName,
    VARIANT productNum,
    VARIANT totalNumberOfProducts,
    VARIANT scanStartTime
    VARIANT scanEndTime
)
```

Arguments

Argument	Description	Data Type
ticket	The ticket that is generated when this scan has been requested.	UINT32
result	A UINT32 value that specifies the scan outcome.	UINT32
threatList	An array of strings(String []). Each string element in the array is the name of the virus which has been detected.	Array of strings
productName	The description of the product.	string
productNumber	The order of scan completion among the usable AVs.	UINT32
totalNumberOfProducts	The number of products scan is requested for this ticket.	UINT32
scanStartTime	The time when the scan is started.	date
scanEndTime	The time when the scan is finished.	date

Callback For Scan Started Report (COM)

Method	ScanStartReport
Description	This callback method is invoked when each engine begins the scan. It is used when a scan request begins. This callback is fired on if “report_on_scan_start” is set to “1” (true).
Note	Connection Points

Function prototype

```
void ScanStartReport
(
    VARIANT ticket,
    VARIANT productName,
    VARIANT filePath
)
```

Arguments

Argument	Description	Data Type
ticket	The ticket which is generated when this scan has been requested.	UINT32
productName	The description of the product.	string
filePath	The path to the file which is scanned.	string

Callback For Update Completion (COM)

Method	MyOnUpdateComplete
Description	This callback method is invoked when an update completes. It is fired to all Metadefender Core client objects that are registered to this connection point. However, if there is already an update request that has not completed, only the Metadefender Core client that requested the update will get a callback with the update result of “updateAlreadyRequested(2)”.
Note	Connection Points

Function prototype

```
void MyOnUpdateComplete
(
    VARIANT updateResult,
    VARIANT updateReport
)
```

Arguments

Argument	Description	Data Type
updateResult	A UINT32 value which contains the update result: • 0: updateFinished • 1: updateFailed • 2: updateAlreadyRequested • 3: updateUnknown	UINT32
updateReport	XML-formatted string consisting of engine names and engine update results. Engine update result is one of the following: • updateOk • updateUpToDate	string

Argument	Description	Data Type
	• updateFailed • updateUnknown • updateNetworkProblem • updateTimeOut • updateNotImplemented • updateNotSupported • updateAlreadyInProgress*	

* updateAlreadyInProgress - If any update on the antivirus product is already running, no additional updates will be queued. This means that the update that has previously been started will return the UpdateResult through the connection point after completion.

Callback For Update Progress Report (COM)

Method	MyOnUpdateProgressReport
Description	This callback method is invoked when an update for each engine completes. It is fired to the Metadefender Core client object that initiates the update.
Note	Connection Points

Function prototype

```
void MyOnUpdateProgressReport
(
    VARIANT productName,
    VARIANT updateState,
    VARIANT updateReport,
    VARIANT productNum,
    VARIANT totalNumberOfProducts,
    VARIANT startTime,
    VARIANT elapsedMiliSec,
    VARIANT reserved
)
```

Arguments

Argument	Description	Data Type
productName	The description of the product.	
updateState	The progress in percentage for update with the specified engine (0 indicates update started while 100 indicates update complete).	
updateReport	XML-formatted string consisting of engine names and engine update results. Engine update result is one of the following: • updateOk • updateUpToDate • updateFailed • updateUnknown • updateNetworkProblem • updateTimeOut • updateNotImplemented • updateNotSupported • updateAlreadyInProgress* * updateAlreadyInProgress - If any update on the antivirus product is already running, no additional updates will be queued. This means that the update that has previously been started will return the UpdateResult through the connection point after completion.	string
productNum	The order of update among the current AVs.	
totalNumberOfProducts	The number of products scanned is requested for this ticket.	
startTime	Time when update for the engine is initiated.	
elapsedMiliSec	Time elapsed for update complete in milliseconds.	

Argument	Description	Data Type
reserved	Reserved	

Deprecated COM interface

The following methods are no longer recommended by OPSWAT, but may be used under certain circumstances. Please contact OPSWAT customer support (https://www.opswat.com/support#support-levels) before using one of these methods.

Analyze file type (deprecated)

Method	AnalyzeFileType
Description	This method analyzes the contents of a file and attempts to guess its type.
Note	deprecated

Function prototype

```
HRESULT AnalyzeFileType
(
    [in] VARIANT FileNameOrBuffer,
    [out] VARIANT * FileTypeLong,
    [out, retval] VARIANT * FilteTypeShort
)
```

Arguments

Argument	Description	Data Type
FileNameOrBuffer	If FileNameOrBuffer is of type string then it is treated as a file name to be analyzed. Otherwise, if it is a byte array (Byte []) it is treated as a memory buffer that should be analyzed.	File name: string

Argument	Description	Data Type
		Buffer: byte array
FileTypeLong	If the pointer that is passed to the method is not null, then the VARIANT will be filled with a string describing the file type in detail.	string
FileTypeShort	A series of letters that summarize the file type.	string

The string returned in FileTypeLong is a series of types that match the file. If the file matches more than one type (sometimes it's impossible to tell the difference), the returned string will contain all the possible file types separated by ','.

The string returned in FileTypeShort is a string comprised of letters that give the rough classification of the file.

Get updated virus definition files (deprecated)

Method	GetUpdateStatus
Description	This method returns information about the virus definition files of the requested product.
Note	deprecated

Function prototype

```
HRESULT GetUpdateStatus
(
    [in] VARIANT ProductNamePart,
    [out] VARIANT * ProductDescription,
    [out] VARIANT * VirusDefinitionDate,
    [out] VARIANT * VirusDefinitionVersion
)
```

Arguments

Argument	Description	Data Type
ProductNamePart	A part of the product name of the desired package. Definition file information of the product containing this value is returned.	string
ProductDescription	The description of the product that matched ProductNamePart.	string
VirusDefinitionDate	The dates of virus definition files for the product. "0" is returned if the data file time is unavailable, for example when the product does not support the data file time.	Date
VirusDefinitionVersion	The virus definition file version of the product.	string

Initialize object (deprecated)

Method	InitEx
Description	This method can still be used for the purpose of initializing an instance of a Metadefender Core object. However, with more flexible and extensive DB controller, the ability to have its own log writer per instance is disabled.
Note	deprecated

Function prototype

```
HRESULT InitEx
(
    [in] VARIANT * argsArray,
    [in] VARIANT argXML,
    [out, retval] VARIANT *UsableAntiViruses
)
```

Arguments

Argument	Description	Data Type
argsArray		empty
argXML	This parameter is reserved for later use. Pass null as its value.	empty
UsableAntiViruses	The list of AVs will be used for this instance of Metadefender Core client.	Array of strings

Put in scan-and-clean queue (deprecated)

Method	PutToScanAndCleanQueue
Description	This method places a file to the scan-and-clean queue. In other words, the scan request will be processed asynchronously.
Note	deprecated

Note: This is a deprecated method and we suggest using ScanEx instead.

Function prototype

```
HRESULT PutToScanAndCleanQueue
(
    [in] VARIANT FileNameOrBuffer,
    [out, retval] VARIANT* Ticket
)
```

Arguments

Argument	Description	Data Type
FileNameOrBuffer		

Argument	Description	Data Type
	If FileNameOrBuffer is of type string then it is treated as a file or directory name to be scanned. Otherwise, if it is a byte array (Byte []), it is treated as a memory buffer that should be scanned.	File name: string Buffer: byte array
Ticket	A ticket for this scan job. This ticket is passed to the callback methods registered via the Callback for Scan Completion (COM) connection point.	UINT32

Note: To get the result of this call, you must implement the Callback for Scan Completion (COM) connection point.

Put in scan queue (deprecated)

Method	PutToScanQueue
Description	This method is called to place a file to the scan-queue. In other words, the scan request will be processed asynchronously.
Note	deprecated

Note: This is a deprecated method and we suggest using ScanEx instead.

Function prototype

```
HRESULT PutToScanQueue
(
    [in] VARIANT FileNameOrBuffer,
    [out, retval] VARIANT* Ticket
)
```

Arguments

Argument	Description	Data Type
FileNameOrBuffer	If FileNameOrBuffer is of type string then it is treated as a file / directory name to be scanned. Otherwise, if it is a byte array (Byte []) it is treated as a memory buffer that should be scanned.	File name: string Buffer: byte array
Ticket	A ticket for this scan job. This ticket will be passed to the callback methods registered via the Callback for Scan Completion (COM) connection point.	UINT32

Note: To get the result of this call, you must implement the Callback for Scan Completion (COM) connection point.

Scan and clean file (deprecated)

Method	ScanAndClean
Description	This method is called to scan and clean a file. This is done in a synchronous (blocking) fashion - the method returns only after the scan is complete.
Note	deprecated

Note: This is a deprecated method and we suggest using ScanEx instead.

Function prototype

```
HRESULT ScanAndClean
(
    [in] VARIANT FileName,
    [out] VARIANT* ThreatList,
    [out, retval] VARIANT* ScanOutcome
)
```

Arguments

Argument	Description	Data Type
FileName	If FileNameOrBuffer is of type string then it is treated as a file / directory name to scan. Otherwise, if it is a byte array (Byte []) it is treated as a memory buffer that should be scanned.	File name: string Buffer: byte array
ThreatList	A threat list found on the scanned object, otherwise null.	Array of strings
ScanOutCome	The scan result.	UINT32

Scan file (deprecated)

Method	Scan
Description	This method is called to scan a file or a memory buffer. This is done in a synchronous (blocking) fashion - the method returns only after the scan is complete.
Note	deprecated

Note: This is a deprecated method and we suggest using ScanEx instead.

Function prototype

```
HRESULT Scan
(
[in] VARIANT FileNameOrBuffer,
[out] VARIANT* ThreatList,
[out, retval] VARIANT* ScanOutcome
)
```

Arguments

Argument	Description	Data Type
FileNameOrBuffer	If FileNameOrBuffer is of type string, then it is treated as a file or directory name to scan. Otherwise, if it is a byte array (Byte []), it is treated as a memory buffer that should be scanned.	File name: string Buffer: byte array
ThreatList	A threat list found on the scanned object, otherwise null.	array of strings
ScanOutCome	The scan result.	UINT32

Subscribe to global events (deprecated)

Method	SubscribeGlobalEvents
Description	If you want to receive global callbacks such as OnLogAvailable, you must subscribe to global events with an appropriate event type through this call. If you subscribe to a specific type of event, you must unsubscribe to that type of event. For example, if you subscribe to a scan event type and update event type, you must both unsubscribe to the scan event type and update the event type.
Note	deprecated

Function prototype

```
HRESULT SubscribeGlobalEvents
(
    [in] VARIANT EventType
    [out, retval] VARIANT * Reserved
)
```

Arguments

Argument	Description	Data Type
EventType	Determines the type of events to subscribe. • 0: All events • 1: Update events • 2: Log events • 3: Scan events	UINT32
Reserved	Reserved	

Unsubscribe from global events (deprecated)

Method	UnsubscribeGlobalEvents
Description	This method is called to unsubscribe to specific or all events. If you subscribe to a specific type of event, you must unsubscribe to that type of event. For example, if you subscribe to scan event type and update event type, you must unsubscribe to both scan event type and update event type. Calling to unsubscribe to events to which you are not subscribed has no effect.
Note	deprecated

Function prototype

```
HRESULT UnsubscribeGlobalEvents
(
    [in] VARIANT EventType
    [out, retval] VARIANT * Reserved
)
```

Arguments

Argument	Description	Data Type
Reserved	Reserved	

Methods

This section lists all the methods available via Metadefender Core COM object. Before you can call any of the methods listed here, you need to create Metadefender Core COM object called 'Metascanner'. When you install Metadefender Core, Metascanner is registered on the system as part of the type library *MetascanLib*.

Metascanner Program ID: Metascan.Metascanner.1

Analyze file type I (COM)

Method	AnalyzeFileTypeEx
Description	This method analyzes a file (or buffer) and attempts to guess its type, detects files of which the file extensions may have been altered, and recursively analyzes contents of archive files.

Function prototype

```
HRESULT AnalyzeFileTypeEx
(
    [in] VARIANT FileName,
    [out] VARIANT * SuggestedExtension,
    [out] VARIANT * FileTypeLong,
    [out, retval] VARIANT * FileTypeShort
)
```

Arguments

Argument	Description	Data Type
FileName	If FileNameOrBuffer is of type string then it is treated as a file name to be analyzed. Otherwise, if it is a byte array it is treated as a memory buffer that should be analyzed	String (file) OR Byte array (buffer)
FileTypeLong	If the pointer that is passed to the method is not null then the VARIANT will be filled with a string describing the file type in detail.	string
SuggestedExtension	If the pointer that is passed to the method is not null then the VARIANT will be filled with the suggested extension type of the file.	string
FileTypeShort	A series of letters that summarize the file type. More details are described in UOther Important Return Types.	string

The string returned in SuggestedExtension is the three letter extension of the filename. The AnalyzeFiletypeEx detection is based on the FileTypeLong returned from the AnalyzeFileType. The SuggestedExtension returns the actual extension of the file, even if it has been altered before scanning the file. If the file comprises of more than one extension, such as executable plus archive, the function returns only the first extension type detected.

Analyze file type II (COM)

Method	AnalyzeFileTypeEx2
Description	This method analyzes a file (or buffer) and attempts to guess its type, detects files of which the file extensions may have been altered, and recursively analyzes contents of archive files.

Function prototype

```
HRESULT AnalyzeFileTypeEx2
{
    [in] VARIANT FileNameOrBuffer,
    [out] VARIANT * InArgsArray,
    [out, retval] VARIANT * OutArgsArray
}
```

Arguments

Argument	Description	Data Type
FileNameOrBuffer	If FileNameOrBuffer is of type string then it is treated as a file name to be analyzed. Otherwise, if it is a byte array, it is treated as a memory buffer that should be analyzed.	String (file) OR Byte array (buffer)
InArgsArray	A list of input arguments in the following order: [**0**] Detect file extension mismatches. Result can be read from OutArgsArray. Set this to one of the following: a. True - detect mismatches if the input file's extension doesn't match any of our suggested extensions b. False - do not detect mismatches [**1**] If mismatch detection and recursive analysis is enabled, stop analyzing upon a mismatch and return immediately with results up until the file that contained the mismatch. Set this to one of the following: a. True – stop analyzing on the first mismatch b. False – analyze all the files within an archive	********** Variant Array [0] Boolean [1] Boolean [2] Boolean **********

Argument	Description	Data Type
	[**2**] Analyze the full contents of archive files along with the archive file itself. Set this to one of the following: a. True – recursively analyze archive files b. False – only analyze the input file	
OutArgsArray	A list of output arguments in the following order: [**0**] If detection of file extension mismatch is enabled, this flag will indicate whether a mismatch was detected. This will be set to one of the following: a. 0 – no mismatch is detected b. 1 – a file extension mismatch was found while analyzing the input file or the contents of the archive c. 2 –unknown because of no detection of file type and / or the input was a buffer [**1**] Array containing the analyzed file(s) and file type information – consisting of the file name, file description, a short type, suggested extension, and extension mismatch flag	********** Variant Array [0] UINT32 (mismatch flag) [1] Variant Array (files) ---------- Variant Array (files) [0][0] String (file name) [0][1] Variant Array (file info) [0][2] UINT32 (mismatch flag) ---------- Variant Array (file info) [0][0] Long type [0][1] Short type [0][2] Suggested Extension **********

Comparison with AnalyzeFileTypeEx()

In addition to returning similar results as AnalyzeFileTypeEx, this method analyzes a file and the contents if it is an archive file. It also gives one or more suggestions for the file type information (description, category, and suggested extension) when available. Additionally, it can raise a flag if the extension of a file does not match any suggested extensions. The caller can specify to abort further analysis on the first mismatch. There are three levels of VARIANT arrays, the last two levels consisting of 2-dimensional SafeArrays.

Upon analysis with AnalyzeFileTypeEx(), if specific details of the file type could be not retrieved, the 'FileTypeShort' result will be set to the letter 'O', throwing it into the "others" category. In contrast, AnalyzeFileTypeEx2() will return an empty array, indicating that details could not be retrieved for the file

Analyze file type III (COM)

Method	AnalyzeEx
Description	This method is an extensible file type analysis API that allows the user to set various analyze file type options that can be configured for each file type analysis request. It can be used as a wrapper of the other file type analysis APIs. It also supports specifying custom ticket ID and session ID to associate file type analysis and scan.

Function prototype

```
HRESULT AnalyzeEx
{
    [in] VARIANT ContentsToScan,
    [in] VARIANT * InArgsArray,
    [out, retval] VARIANT * OutArgsArray
}
```

Arguments

Argument	Description	Data Type
ContentsToScan		File name: string

Argument	Description	Data Type
	This argument may hold one of two types, a file path to be scanned or a memory buffer to be scanned based on the data type of this parameter and input argument (*InArgsArray*) value • A file path: if the type of this parameter is a string • Memory buffer: if the type of this parameter is a byte array(byte[])	Buffer: byte array
InArgsArray	A list of input arguments in the following order. **0**. File Type Library choice • 0: Use default • 1: Magic File Type • 2: TriD (Not supported for now) **1**. Custom ticket # (maximum 9 digits from 1-999999999) or (0 for not using custom ticket #) **2**. Session ID a. To associate multiple file scan as a group (e.g., multiple requests for single folder)	Array of variants with the following order: 1: UINT32 1: UINT32 2: String
OutArgsArray	A list of outputs with the following order: **0**: Ticket number (if custom ticket # is specific in InargsArray, same ticket number is returned). **1**: Suggested File Type Extension **2**: File Type Long **3**: File Type Short	Array of variants with the following order: 0: UINT32 1: String 2: String 3: String

Check license (COM)

Method	Checklicense - COM
Description	This method checks if the license installed on the machine is valid and when it expires. Metadefender Core processes, scans, and updates requests only if LicenseType is one of the“valid” types below.

Function prototype

```
HRESULT CheckLicense
{
    [out] VARIANT * LicenseType,
    [out] VARIANT * ExpiredDate,
    [out, retval] VARIANT * IsValid
}
```

Arguments

Argument	Description	Data Type
LicenseType	The type of license: • **0**: Expired (invalid) • **1**: Activated (valid) • **2**: Trial (valid) • **3**: Activate Required (invalid) • **4**: Activated but expires in 30 days (valid)	UINT32
ExpiredDate	The date when the license is expired in the following format: MM/DD/YYYY (MM: month, DD: day, YYYY: year)	string
IsValid	**True**: if the license is valid **False**: if the license is invalid	bool

Get an engine info (COM)

Method	GetUpdateStatusEx
Description	This method returns information about the virus definition files of the requested product.

Function prototype

```
HRESULT GetUpdateStatusEx
{
    [in] VARIANT ProductNamePart,
    [out] VARIANT * ProductDescription,
    [out] VARIANT * VirusDefinitionDate,
    [out] VARIANT * VirusDefinitionVersion,
    [out] VARIANT * VirusDefinitionSignature,
    [out] VARIANT * ProductEngineVersion
}
```

Arguments

Argument	Description	Data Type
ProductNamePart	A part of the product name of the desired package. Definition file information of the product containing this value will be returned.	string
ProductDescription	The description of the product that matched ProductNamePart.	string
VirusDefinitionDate	The dates of virus definition files for the product. 0 will be returned if the data file time is unavailable, for example when the product does not support the data file time.	date
VirusDefinitionVersion	The virus definition file version of the product.	string

Argument	Description	Data Type
VirusDefinitionSignature	The virus definition file signature of the product.	string
ProductEngineVersion	The product (engine) version.	string

Get a property value (COM)

Method	GetProperty
Description	Reads one of the internal properties.

Function prototype

```
HRESULT GetProperty
(
    [in] VARIANT PropertyName,
    [out, retval] VARIANT* PropertyValue
)
```

Arguments

Argument	Description	Data Type
PropertyName	The name of the requested property. For valid names, please refer to the Metadefender Core Configuration Guide.	string
PropertyValue	The value of the property	string

Initialize (COM)

Method	Init
Description	This method is called to initialize the connection to the Metadefender Core service. This method needs to be called before any other method can be called.

Function prototype

```
HRESULT Init
(
    [out, retval] VARIANT *UsableAntiMalwares
)
```

Arguments

Argument	Description	Data Type
UsableAntiMalwares	The list of anti-malware engine names available to be used with this instance of Metadefender Core	array of strings (VT_BSTR)

Process a file (COM)

Method	Process
Description	This method processes a file through a workflow based on the specified user agent.

Function p rototype

```
HRESULT Process
(
    [in] VARIANT ContentsToBeProcessed,
    [in] VARIANT ContentsType,
```

```
    [in] VARIANT UserAgent,
    [in] VARIANT InArgsArray,
    [out] VARIANT *OutDetail,
    [out, retval] VARIANT *OutArgsArray
)
```

Arguments

Argument	Description	Data Type
ContentsToBeProcessed	Full path to the file to be processed	string
ContentsType	0: holds a file path (Currently this is ignored since only 0 is supported)	UINT32
UserAgent	A string to identify the user. Used for mapping to a specific workflow.	string
InArgsArray	0: Sync Flag - Reserved for future use. 1: Custom Ticket - Currently, only the integer range value is supported. 2: Password for Archives 3: Additional Logging Parameter - This is passed to the user description argument in ScanEx if scanning is enabled on the workflow. Reserved for future use. 4. File Display Path - The full path of the file from the source machine to display to the user after processing. 5. Disable Logging - Disallows Metadefender Core to log any file type analysis and scan results. 6. Caching Option • 0: disable to cache for the scan request • 1: enable to cache for the scan request • 2: disregard the existing cache • 3: use the global setting	Safearray of variants with the following order 0: boolean 1: string 2: string 3: string 4: string 5. boolean 6. uint32 7. string

Argument	Description	Data Type
	7. Customized data sanitization location	
OutDetail	JSON result details string Example JSON { "Ticket": "974989300" } In case of error [e.g. "Server is too busy. Try again later." | "Invalid License" | "Daily scan limit reached" | "Failed to continue. unknown error" | "Invalid file path"] : { "Error": "Invalid License" } In case of too many simultaneous scans: { "Error": " Exceeded maximum thread count" }	string
OutArgsArray	0: Custom Ticket - to be used to associate callback results	Safearray of variants with the following order 0: string

Note: When the Metadefender Core server is too busy, the HRESULT value returned will be E_OUTOFMEMORY

Creating workflow profiles

Many configurations are supported via workflow profiles. To access the Workflow tab on the Metadefender Core Management Console, go here: http://localhost:8008/management/workflow .

Sanitize a file (COM)

Method	ConvertFileType
Description	This method converts the specified input file into a different format.

Function prototype

```
HRESULT ConvertFileType
{
    [in] VARIANT ContentsToConvert,
    [in] VARIANT ConvertTo,
    [in] VARIANT TargetFolderPath,
    [in] VARIANT * ReservedInArgsArray,
    [out] VARIANT * ReservedOutArgsArray,
    [out, retval] VARIANT * ConvertedFilePath
}
```

Arguments

Argument	Description	Data Type
ContentsToConvert	UNC of the input file to convert.	String
ConvertTo	Identifier for targeted conversion. Usually file extension (e.g., pdf, docx…)	String
TargetFolderPath	UNC of the destination folder path. The converted file will be created in that folder.	String
ReservedInArgsArray	Reserved	
ReservedOutArgsArray	Reserved	
ConvertedFilePath	Path to the converted file if conversion was successful.	String

Scan a file (COM)

Method	ScanEx
Description	This method is an extensible scan API which allows the setting of various scan options that can be configured for each scan request.

Function prototype

```
HRESULT ScanEx
(
    [in] VARIANT ContentsToScan,
    [in] VARIANT* InArgsArray,
    [out, retval] VARIANT* OutArgsArray
)
```

Arguments

Argument	Description	Data Type
ContentsToScan	This argument may hold one of four types: a file path to be scanned, a file signature to be scanned, a boot sector to be scanned, or a memory buffer to be scanned, based on the data type of this parameter and input argument (*InArgsArray*) value • **File path:** if the type of this parameter is a string and the fifth argument of *InArgsArray* is set to 0 or the size of *InArgsArray* is 3 • **File signature:** if the type of this parameter is a string and the fifth argument of *InArgsArray* is set to 1 • **Boot sector scan:** if the type of this parameter is a string and the fifth argument of *InArgsArray* is set to 2 • **Memory buffer:** if the type of this parameter is a byte array(byte[]) • Virtual Machine image folder path	File name: string Buffer: byte array
InArgsArray	A list of input arguments in following order: 1. (boolean) Sync flag (true: synchronous scan , false or invalid value: asynchronous scan) 2. (boolean) Clean flag (true: Action on dirty file, false: keep)	Array of variants

Argument	Description	Data Type
	3. (UINT32) Custom ticket # - maximum 9 digits from 1-999999999 or 0 for not using custom ticket # 4. (UINT32) Analyze before scan 0: disable analyze file before scan 1: enable analyze file before scan 2: use global setting (“analyze_before_scan”) 5. (UINT32) Contents Type 0: ContentToScan argument holds file path or memory buffer 1: ContentToScan argument holds SHA1-based signature. 2: ContentToScan argument holds driver letter for a boot sector scan 4: ContenToScan argument holds Virtual Machine image folder path (Virtual machine disk must contain a Windows based filesystem) 6. (UINT32) Clean Action Type 0: take no action 1: quarantine (ignore global setting, “clean_action”) 2: delete (ignore global setting, “clean action”) 3: follow global setting 7. (UINT32) Caching option 0: disable to cache for this scan request 1: enable to cache for this scan request 2:rescan (disregard existing cache and update with new scan result) 3: use global setting(“enable_cache_scan”) 8. (String) Password for encrypted archive.	

Argument	Description	Data Type
	- Use empty string to when no password is required. (“enable_cache_scan”) 9. (String) User agent - This is used to associate scan history with the source of scan request. Empty string is allowed. 10. (String) User description - This will be logged along with scan request for the caller’s purpose. Empty string is allowed. - Pass the following JSON formatted string to control the file name and file path to be logged: {"file_info.original_file_path":"<original file path not including file name", "file_info.display_name":"<original file name>"} 11. (UINT32) Configure logging 0: disable logging this particular scan request 1: enable logging this particular scan request 2: use global setting (omsConfig.ini) 12. (UINT32) Archive File Handling 0: do not extract archive file for this particular request 1: extract archive file for this particular scan request 2: use global setting (“internal_archive_lib_enable”)	
OutArgsArray	A list of outputs with the following order: 1. (UINT32) Ticket number (if custom ticket # is specific in InargsArray, same ticket number is returned 2. (UINT32) Scan result as described in ScanOutCome Return Type 3. (Array of strings) List of threats (String) Scan detail per engine	

Argument	Description	Data Type
	<scan_details> <objects> <object name="[object name]"> <engine_result> <engine_name>[engine name]</name> <scan_result>[scan outcome for the engine]</scan_result> </engine_result> </object> </objects> </scan_details> [object_name]: file name or signature depends on type of scan requested [scan outcome for the engine]: Scan result per engine as described in ScanOutCome Return Type 4. (String) File type information this is returned only if analyze_before_scan is enabled orthe third argument of *InArgsArray* is set to 1 <file_type> <objects size=""> <object> <sha1></sha1> <type_infos size=""> <type_info> <long></long> <short></short> <extension></extension> </type_info> </type_infos> </object> </objects> </file_type>	

Set a property value (COM)

Method	SetProperty
Description	Modify one of Metadefender Core's internal properties.

Function prototype

```
HRESULT SetProperty
{
    [in] VARIANT PropertyName,
    [in] VARIANT PropertyValue
}
```

Arguments

Argument	Description	Data Type
PropertyName	The name of the requested property.	string
PropertyValue	New value of the property	string

Stop ongoing scan (COM)

Method	StopScan
Description	This method purges all scan requests and ongoing scans. Method returns only when all requests are purged.

Function prototype

```
HRESULT StopScan
{
    [in] VARIANT Reserved
}
```

Arguments

Argument	Description	Data Type
Reserved	Reserved	

Update an anti-malware engine (COM)

Method	UpdateEngine
Description	This method is called to initiate an update of a specific anti-malware engine on the local machine.

Function prototype

```
HRESULT UpdateEngine
(
    [in] VARIANT EngineName,
    [in] VARIANT ReservedArg
)
```

Arguments

Argument	Description	Data Type
EngineName	The name of the anti-malware product that needs to be updated (case in-sensitive).	string
ReservedArg	Reserved. Any type of Variant is allowed except NULL pointer.	

Note: The result of this call is provided via the OnUpdateComplete connection point.

Update anti-malware engines (COM)

Method	UpdateAntiViruses
Description	This method is called to initiate an update of all anti-malware engines at the same time on the local machine. It is recommended to perform updates via automatic periodic update, which is performed at predetermined intervals as configured in “Update Interval” property.

Function prototype

```
HRESULT UpdateAntiViruses
(
)
```

Note: The result of this call is provided via the OnUpdateCompleteconnection point.

Sample Code

You can download Metadefender Core Sample Code for COM interface from OPSWAT portal. This document describes how to set up and use the sample code you have downloaded.

ASP Sample

1. Install IIS.
2. Install Web Development support for Visual Studio.
3. Enable web sharing on the sample dir.
    a. Go to sample dir.
    b. Right-click on **Properties** .
    c. Click **Web Sharing > Share this folder** .
    d. Give all permissions to folder (including **Execute for scripts**).
4. Enable advanced file sharing options – Open an explorer window -> Tools -> Folder Options -> View -> uncheck “Use simple file sharing”.

5. Ensure the following ASP accounts have permissions on the project folder:
   • ASP.Net machine account
   • Internet guest account (IUSR)
   • Launch IIS Process account (IWAM)
6. To change the permissions, right-click the folder and select **Properties > Security** .
7. Add the accounts. For each account, add read and execute permissions.
8. Open Internet Explorer, and change security settings to custom. Reset them to Medium low, and change the “Download unsigned ActiveX controls” and “Initialize and script ActiveX controls not marked as safe” options to “prompt”.
9. Open the project. You may get the following message "The specified Web server is not running ASP.NET version 1.1 ..." If so. follow the steps in: http://support.microsoft.com/kb/817267/en-us.
10. In the IIS, ensure access is allowed to the sample folder by setting the folder's "Directory Security".
11. Assign "Launch and Activation" permission on the Metascan to the account running ASP.Net in DCOM Configuration (DCOMCNFG).

PHP Sample Code for IIS 7

1. Install IIS 7 by adding a role to the server.
2. Add CGI/Fast CGI as an IIS feature (from cmd.exe):
   dism /online /enable-feature /featurename:IIS-CGI
3. Download the latest PHP 5 from http://www.php.net/downloads.php.
4. Install PHP as IIS FastCGI to “C:\PHP” (be sure to select script executable option to associate *.php with PHP).
5. Edit C:\PHP\php.ini.
   short_open_tag = On
   cgi.force_redirect = 0
6. Configure IIS on the system.
   a. Go to **Start > Control Panel > Administrative Tools > Internet Information Services (IIS) Manager** .
   b. Choose **Sites > Default Web Site** .
   c. Double-click on **Handler Mappings** in the right pane.
   d. Double-click on **PHP_via_FastCGIModule** .
   e. Click **Request Restrictions**.

f. Select the **Verbs** tab.

g. Select radial button **One of the following verbs** , enter “GET,POST,HEAD”.

h. Select the **Access** tab.

i. Select the **Script** radio button.

j. Click **OK** all the way out and accept changes.

7. Place scan.php, userInput.html and uploads folder into C:\Inetpub\wwwroot folder.
8. Allow necessary security permission to both the files and the folder (for testing please allow to everyone).
9. Configure Metadefender Core COM settings to allow IIS_USRS and IUSR to launch, access, and configure Metadefender Core (local permissions only).
10. Restart Metadefender Core service.
11. Go to http://localhost/userInput.html.
12. Upload the file to be scanned. HTML calls PHP script and prints the scan results on the web page.

Command Line Utility

Metadefender Core provides a command line utility (omsCmdLineUtil.exe) that enables a user to run Metadefender Core operations such as scanning files, getting or setting preferences /properties, etc.

For more information, run "help" command from the directory in which the command line utility is located.

Exit code (for scan command)

Metadefender Core provides a process exit code for each scan to the command console. After you scan a file from CLI, you can enter “echo %errorlevel%” from the command prompt to get the scan result code. For detailed scan result return value information, refer to COM Interface.

Note: If omsCmdLineUtil.exe is unable to establish a connection to the Metadefender Core service, it returns the message "Failed to initialize Metadefender Core".

Process a File Command

syntax	process <file path> <user_agent>

This function allows a user to process a file on the command line with a specific user agent using all available AV products on the machine. This effectively allows the user to choose which workflow to use when processing their file.

COM API

Calls Metascan COM Process asynchronously.

Process parameters

input parameters	
<file path>	user entered file path
<user_agent>	the User Agent corresponding to desired Workflow

Example

```
c:\Program Files (x86)\OPSWAT\Metascan 4>omsCmdLineUtil.exe
process C:\TestFiles\eicar.com default


Ticket:[ 2772276979 ]

Process Details
---------------


File: [ C:\TestFiles\eicar.com ]
   MD5: [ 44D88612FEA8A8F36DE82E1278ABB02F ]
   SHA1: [ 3395856CE81F2B7382DEE72602F798B642F14140 ]
   SHA256: [
275A021BBFB6489E54D471899F7DB9D1663FC695EC2FE2A2C4538AABF651FD0F ]
   File Size: [ 68 bytes ]
   File Type  Category: [ T ]
   File Type: [ unknown ]
   File Type Description: [ ASCII text, with no line terminators ]

[ Dirty ] Avira scan engine [ 1 ms ] | Eicar-Test-Signature
[ Dirty ] ESET scan engine [ 2 ms ] | Eicar test file
[ Dirty ] Ahnlab scan engine [ 3 ms ] | EICAR_Test_File
[ Dirty ] ClamAV scan engine [ 14 ms ] | Eicar-Test-Signature

Scan Completion
---------------
```

```
        [ Dirty ]
        Ticket: [ 2772276979 ]
        File path: C:\TestFiles\eicar.com
        Found threats:
                EICAR_Test_File
        Scan time: 14 ms [01/25/2016 16:01:44:835]


Process Completion
------------------
        Ticket: [ 2772276979 ]
        User agent: default
        Profile: Default
        Result: [ Blocked ]
        Blocked reason: [ Dirty ]
        File processed: C:\TestFiles\eicar.com
```

Limitations

Does not support processing a folder.

Java Sample

Below are the steps to use for the Java sample with Eclipse IDE running or by running on the command line. Java sample is not installed with Metadefender Core. You must download this separately from the OPSWAT portal. Go to https://portal.opswat.com/en/content/metadefender-core-sample-code, and click **Download** .

With the Eclipse IDE running:

1. Create a new Java project from existing source code. (Choose the Java sample code folder you have downloaded).
2. Add Metasscan_java_interface.jar to the build path of the project. By default, Metascan_java_interface.jar file is installed in the directory where Metadefender Core is installed under the JNI folder.
3. Set java argument of Native library path to “omsJInterfaceImple.dll”

Example: *-Djava.library.path=C:/Program Files (x86)/OPSWAT/Metadefender Core 8/JNI* (in vm arguments of “Run Configuration”)

Running on the Command Line:

1. Go to bin directory.

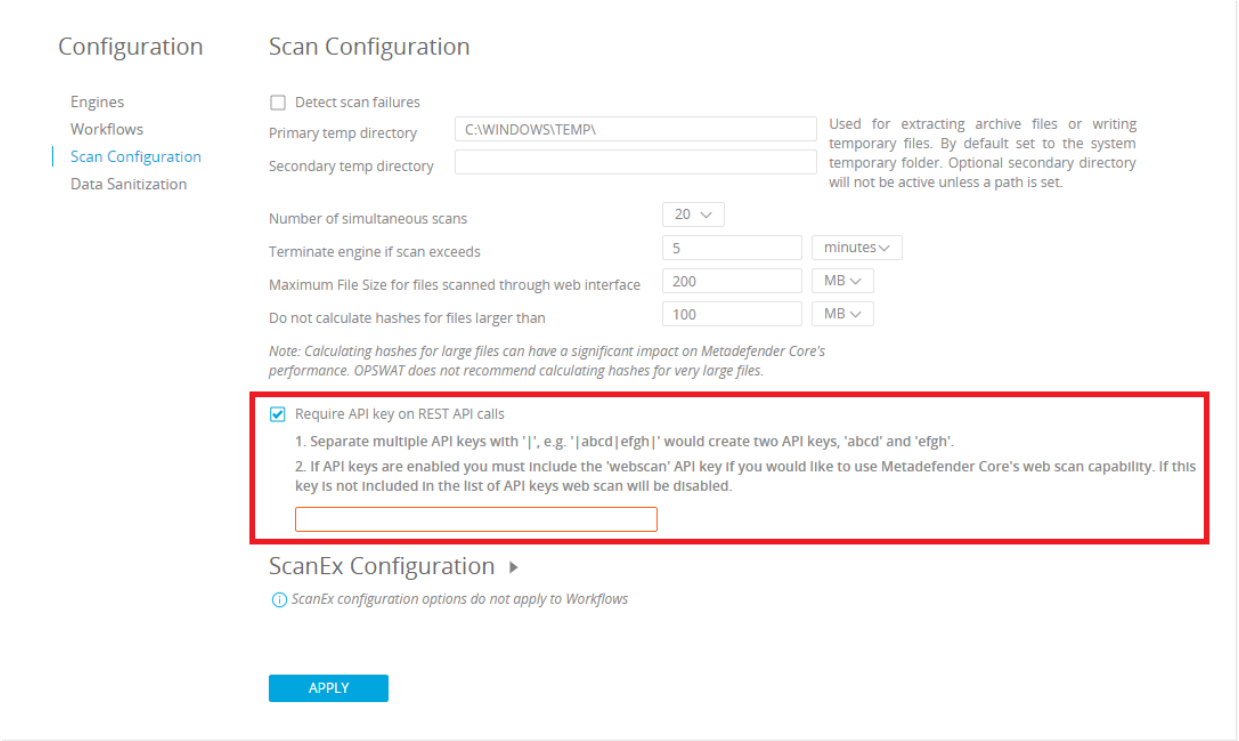
2. Run the following command:
   Java.exe -cp .;<absolute path to Metascan_java_interface.jar file> -Djava.library.path=<absolute path to omsJInterfaceImpl.dll> com.opswat.metascan.Sample

REST API (v2)

Beginning with Metascan 3.8.1, OPSWAT supports only using the JSON-based (v2) REST API and XML-based (v1) REST API is deprecated.

API key-based authentication

Metadefender Core allows you to limit access to the REST API only to users that have a defined API key. If no such API keys are defined then the 'apikey' parameter should be excluded from the REST API calls. If one or more API keys are defined, the 'apikey' parameter is required. To define API keys configure them in the management console.



Getting Started

- Look Up Hash and Process (Scan / Sanitize) A File

Compatibility with Core v4 and Metadefender.com

We have designed REST API in a way that most common functions on Core v3, Core v4, and Metadefender.com public/private API share same structures and mechanism.

If you are moving from Metadefender.com to Core or vise versa, the change you need to make on your application should be minimal unless there is specific function that is only available to either one. If you notice any incompatible API, please contact OPSWAT support for consulting.

Advanced Usages

Load handling

Maximum Number of Simultaneous Workflow Clients Through REST

Overview

Configuring many systems to perform a full system scan using MClient.

Queuing Mechanism

We will have 2 queues in REST.

1. **Pending**: Pending scans
    a. Scan not yet submitted to the COM service.
    b. Maximum 20,000 files here before server returns 500 too busy.
    c. Files go pass through here first, then if the ongoing scan queue has space they are moved there.
2. **Ongoing**: Scan in progress
    a. Scan already submitted to the COM service.
    b. Maximum 1000 files here.
    c. If this queue is full the files stay in pending scan queue.

Disk Check

We have 2 types of disk checks based on how much free space is left.

1. If < 3.5 GB free disk space:
    a. With every file upload we will do a disk space check.
2. If > 3.5 GB free disk space:

a. We used the cached value from our already periodic check.

The reason we need to checks is because there is a known issue about Mongo not being able to start if there is enough free space on the disk. (3379MB)

```
Mongo Error

2016-09-06T16:12:15.514-0700 [initandlisten] ERROR: Insufficient
free space for journal files
2016-09-06T16:12:15.514-0700 [initandlisten] Please make at least
3379MB available in C:\Program Files (x86)\OPSWAT\Metadefender
Core 4\Mongo\data\journal or use --smallfiles
```

Recommend REST integration in multiple client environment

1. REST users should only submit ~20 files at a time and wait until processing is complete before uploading more.
2. /file/inqueue reports how many files are in the queues (adding both pending on ongoing)
3. If client gets 503 server too busy they should wait a bit and try again.

Number of pending jobs

Description	Number of pending jobs
URL Path	/file/inqueue
Method	GET

Summary

Returns the number of pending jobs on the server

Request URL

http://localhost:8008/metascan_rest/file/inqueue

HTTP header parameters

apikey	API key assigned by user	OPTIONAL

Method: GET

URL path	/file/inqueue	REQUIRED

Response Code

200	**OK**	Request is successful
400	**Bad request**	Not supported HTTP method or invalid http request .
401	**Invalid API key / Exceeded usage**	Either missing API key or invalid api is passed.
500	**Internal server error**	Server temporary unavailable. Try again later.

Response

Example of when request is successful :

```
{
    "in_queue" : "0"
}
```

Descriptions of response:

in_queue	Number of jobs inqueue.

Multiple Hashes Lookup

Description	Retrieve scan report by multiple hashes.
URL Path	/hash
Method	POST

Summary

- Get a list of available hashes for an API key.
- Dues to business stuff, we need to remove scan result of Kaspersky in response json (MCL-3016) (Only applicable for Metascan Online)

HTTP header parameters

apikey	API key assigned by user	OPTIONAL
include_scan_details	include scan details Value: 0, 1	OPTIONAL Default: 0

"scan_details" exists in response when **"scan_result"** = 0|1|2 in response and header **"include_scan_details"**=1

Method: POST

URL path	/hash	REQUIRED
Body content	{"hash":["hash_value_1","hash_value_2","hash_value_3",..]}	

Response Codes

200	**OK**	Found or Not Found
400	**Bad request**	Not supported HTTP method or invalid http request .
401	**Invalid API key**	Either missing API key or invalid api is passed.
403	**Signature lookup limit reached, try again later**	User exceeds limit
503	**Internal server error**	Server temporary unavailable. Try again later.

Response

Example of when input body is empty :

```
{
    "err" : "hashes in body content is null or empty"
}
```

Example of when input body is wrong format :

```
{
    "err" : "Could not parse body content."
}
```

Example of maximum hashes input.

```
{
    "err" : "Exceeded maximum allowed. Allow maximum 1000 hashes"
}
```

Example of when request is successful :

```
[
    {
        "hash" : md5_1,
          "scan_result" : 0,
        "data_id" : data_id_1
    },
    {
        "hash" : md5_2,
          "scan_result" : 1,
        "data_id" : data_id_2
    },
    {
        "hash" : "abc",
          "scan_result" : 5,
          "data_id" : null
    }
]
```

Example of when request is successful and "include_scan_details"=1:

```
[
    {
        "hash" : md5_1,
          "scan_result" : 0,
        "scan_details":{
            "Agnitum":{
                "scan_result_i":0,
                "threat_found":"",
                "def_time":"2014-08-07T07:00:00Z",
                "scan_time":78.0
            }...{}},
        "data_id" : data_id_1
    },
    {
        "hash" : md5_2,
          "scan_result" : 1,
        "scan_details":{
            "Agnitum":{
                "scan_result_i":0,
                "threat_found":"",
                "def_time":"2014-08-07T07:00:00Z",
                "scan_time":78.0
            }...{}},
```

```
        "data_id" : data_id_2
    },
    {
        "hash" : "abc",
          "scan_result" : 5,
          "data_id" : null
    }
]
```

Example of when the number of lookup hashes i s more than remaning lookup number:

e.g: user can only scan 2 hashes but posts 3 hashes.

```
[
    {
        "hash" : md5_1,
          "scan_result" : 0,
        "data_id" : data_id_1
    },
    {
        "hash" : md5_2,
          "scan_result" : 1,
        "data_id" : data_id_2
    },
    {
        "hash" : md5_3,
          "scan_result" : 403,
          "data_id" : null
    }
]
```

Descriptions of response:

scan_result	the newest final scan result of hash 5: unknown scan result - may be hash does not exist in mongo
data_id	the newest data_id of hash
scan_details	Array of scan results for engines

For more details of response, refer to Response Description.

Statistics

Check Server health

Description	Server health
URL Path	/stat/serverhealth
Method	GET

Summary

Retrieve server health including free disk space, RAM and CPU usage, disk queue length. These are average values based on sampling.

Request URL

http://localhost:8008/metascan_rest/stat/serverhealth

Use Cases (used by)

Management Console (to retrieve server statistics of memory, CPU usage, and average disk queue length to monitor server health).

HTTP header parameters

Header	Description	Required
apikey	API key assigned by OPSWAT or admin	REQUIRED

Method: GET

/stat /serverhealth	Returns all entries in server health recorded.
/stat /serverhealth/ {N}	n is hours. It will return entry between(inclusive) HH-n:00 and HH:00. For example: call /stat/serverhealth/1 at 13:13, server will return entry between 12:00 and 13:00.

Response Codes

Code	Status Description	Notes
200	**OK**	successful response
400	**Bad request**	Not supported HTTP method or invalid http request.
401	**Permission Denied for statistic**	Either missing API key or invalid API is passed.
500	**Internal server error**	Server temporary unavailable. Try again later.

Response

Example of successful scan request:


```
[
  {
    "usage_check_localtime": "2016-02-03T12:00:00-08:00",
    "avg_ram_usage": 84.92316500345866,
    "avg_cpu_usage": 13.39331203699112,
    "avg_disk_q_len": 0.2144404633436352,
    "avg_disk_free_space": 25.00000000000000
  },
  {
    "usage_check_localtime": "2016-02-03T13:00:00-08:00",
    "avg_ram_usage": 82.8503189086914,
    "avg_cpu_usage": 2.4295437335968018,
    "avg_disk_q_len": 0.0027925085742026567,
    "avg_disk_free_space": 25.0000000000000000
  }
]
```


Descriptions of response:

Response	Description
usage_check_localtime	The local time when the information is queried.

Response	Description
avg_ram_usage	Average of RAM usage of Metadefender Core server as a percentage of total RAM.
avg_cpu_usage	Average of CPU usage of Metadefender Core server as a percentage of total CPU.
avg_disk_q_len	Tracks the number of requests that are queued and waiting for a disk during the sample interval, as well as requests in service.
avg_disk_free_space	Average free disk space on Metadefender Core server as a percentage of total space.

Get enginedef stat

Description	Get enginedef stat
URL Path	/stat/enginedef
Method	GET

Summary

Gets definition time of all current anti-malware engines. This is recommended API to check engine definition rather than using Get engines stat because it returns cached definition info and relatively cheap API.

Request URL

http://localhost:8008/metascan_rest/stat/engines

Response Codes

200	**OK**	
400	**Bad request**	Not supported HTTP method or invalid http request.
500	**Internal server error**	Server temporary unavailable. Try again later.

Response

Notice that engine name will be shortened to single word.

Example of successful scan request:

```
{
  "Lavasoft" : "2014-11-24T00:00:00",
  "TotalDefense" : "2013-02-12T00:00:00",
  "ESET" : "2014-11-25T00:00:00",
  "AVG" : "2013-08-12T00:00:00",
  "Avira" : "2014-10-01T00:00:00"
}
```

Get engines stat

Description	Get engines stat
URL Path	/stat/engines
Method	GET

Summary

Returns engine information for all usable anti-malware engines. This is relatively expensive API call on the server side. If you want to retreive definition info, use Get enginedef stat.

Request URL

http://localhost:8008/metascan_rest/stat/engines

HTTP header parameters

apikey	API key assigned by OPSWAT or admin	REQUIRED

Request Error

400	**Bad request**	Not supported HTTP method or invalid HTTP request.
500	**Internal server error**	Server temporary unavailable. Try again later.

Response

Example of successful scan request:

```
[{
    "eng_name" : "ClamWin scan engine",
    "eng_ver" : "0.97.8",
    "def_time" : "11/10/2013 12:00:00 AM",
    "eng_type" : "Bundled engine",
    "active" : true
},
{
    "eng_name" : "ESET NOD32 Antivirus",
    "eng_ver" : "4.2.71.2",
    "def_time" : "11/11/2013 12:00:00 AM",
    "eng_type" : "Customer licensed engine",
    "active" : false
},
{
    "eng_name" : "ESET scan engine",
    "eng_ver" : "1412 (20131023)",
    "def_time" : "6/1/2012 12:00:00 AM",
    "eng_type" : "Bundled engine",
    "active" : true
},
{
    "eng_name" : "Norman scan engine",
    "eng_ver" : "7.2.6",
    "def_time" : "10/13/2013 12:00:00 AM",
    "eng_type" : "Bundled engine",
    "active" : true
},
{
    "eng_name" : "Total Defense scan engine",
    "eng_ver" : "37.0.1.55",
    "def_time" : "11/8/2013 12:00:00 AM",
    "eng_type" : "Bundled engine",
    "active" : true
}]
```

Descriptions of response:

def_time	The last time updated of engine.
active	True: engine is used to scan file.

Summary of scan

Description	Retrieve scan history
URL Path	/stat/scanhistory/{n_hour}
Method	GET

Summary

Retrieve history of scanning in last n hours.

HTTP header parameters

apikey	API key assigned by OPSWAT or admin	REQUIRED

Response Codes

200	**OK**	successful response
400	**Bad request**	Not supported HTTP method or invalid http request
401	**Permission Denied for statistic**	Either missing API key or invalid api is passed
500	**Internal server error**	Server temporary unavailable. Try again later.

Response

Example of successful scan request:

```
[  {
    "total_scanned_files" : 101,
    "total_infected" : 0,
    "avg_scan_time" :  19.712871287128714,
    "localtime" :  "2013-11-11T11:00:00+07:00"
  },
  {
    "total_scanned_files" : 3,
    "total_infected" : 0,
```

```
    "avg_scan_time" :  113.66666666666667,
    "localtime" :  "2013-11-11T10:00:00+07:00"
  },
  {
    "total_scanned_files" : 5,
    "total_infected" : 0,
    "avg_scan_time"  :25.455566656565212,
    "localtime" :  "2013-11-11T09:00:00+07:00"
} ]
```

Descriptions of response:

avg_scan_time	Average scanning time
total_infected	all logs with scan_result_i of 1, 2, 4, 6, 8

Look Up Hash and Process (Scan / Sanitize) A File

Recommended Flow Of API usage

1. Calculate hash value of a file.
2. Hash Lookup to see if known instead of sending whole file to server for scan.
3. If unknown, Initiate Process File call
4. Retrieve process result Retrieve process result by data_id
5. if file is sanitized, Download Sanitized File.

User Provided Info

Certain data processing requires of accurate metadata for better process results. Refer to Initiate Process File for more information.

Download Sanitized File

Description	Download a file which was converted by the /file workflow API
URL Path	/file/converted/{data_id}
Method	GET

Summary

Downloading file scanned from database using data_id

HTTP header parameters

apikey	Only required if REST API keys have been defined	OPTIONAL

Method: GET

URL path	/file/converted/{data_id}	REQUIRED
data_id	Returned by Initiate Scan/Process a File	REQUIRED

Example: http://localhost:8008/metascan_rest/file/converted/012e1a8945c041d181e79ab8d711555d

Request Codes

200	**OK**	
400	**Bad request**	Not supported HTTP method or invalid http request (e.g., empty body).
401	**Invalid API key**	Either missing API key or invalid api is passed.
401	**Exceeded usage**	
500	**Internal server error**	Server temporary unavailable. Try again later.

Response

Example of when request is not successful :

```
{
    err: "DataID not found in the database"
}
```

Example of when request is successful :

```
It will return the content of that file if using rest client tool.
If using browser, a pop up save file will appear with the file
name is the display_name in database. (This API sets the filename
using the Content-Disposition)
```

Hash Lookup

Description	Retrieving previous scan reports using hash value
URL Path	/hash/{hash value}
Method	GET

Metadefender provides multipe ways of looking up previously processed results using hashes or known data_id. If data_id is unknown, hash value (md5, sha1, or sha256) can be used to look up known scan results. While single hash lookup provides full scan results related to hash if found, Multiple Hashes Lookup will return condensed results with links (data_ids) to full scan result.

HTTP header parameters

apikey	If API key is configured by the administrator	OPTIONAL

Method: GET

URL path	/hash/{hash value}	REQUIRED

hash value: any of the followings.

- SHA1
- MD5
- SHA256

Response Codes

code	status description	note
200	OK	either result is found or not found.
400	Bad request	Not supported HTTP method or invalid http request
401	Invalid API key	Either missing API key or invalid api is passed
404	Not found	The scan result for the given data_id was not found.
503	Internal server error	Server temporary unavailable. Try again later.

Response

Only when server returns 200 OK, proper response is returned.

Example of when result is not found :

When scan results are not found, a 200 response status code appears.

```
Status: 200

{
    "BED12FDA073BB386B54700138FB47EEA": "Not Found"
}
```

Response

When result is not found (example) :

```
Status: 200 (Should be 404)

{
    "4f84e3b096d54908963d037b5457bf27": "Not Found"
}
```

When result is found :

It will return process result. Refer to Response Description.

Initiate Process File

Description	Initiate scan request
URL Path	/file
Method	POST

Summary

There are two ways of initiate processing file. One is to upload file and the other is passing the file path (local to the server).

There are several important parameters should be passed along the request so that administrators can track such histories on the server side without relying on additional application logic on client sides.

HTTP header parameters

Parameter	Value	Required	Description	Notes
apikey	API key configured by administrator.	OPTIONAL		
filename	Name of files to preserve extension during scan and meta data.	RECOMMENDED	filename should be encoded with URLencode and should not contains characters "<" and ">" Otherwise, it will be removed from filename.	

Parameter	Value	Required	Description	Notes
			• This will be sanitized and all invalid file_name characters will be removed. • If file type mismatch is enabled, needs filename to know original file extension. If not providing this header, it will cause mismatch.	
filepath	Path of files that are accessible to Core server	OPTIONAL	• filepath should be URLEncoded. • filepath should be accessible to core server. • When this head is set, no content need to be put in HTTP request body. Core server will try to process /scan using filepath directly	
user_agent		OPTIONAL		

Parameter	Value	Required	Description	Notes
	A string to identify the client application		Administrator configures to map user agents to specific workflow.	
archivepwd	password of encrypted file	OPTIONAL	If submitted file is password protected archive.	
original_file_path	File path of file scanned on local to the Metascan	OPTIONAL	original_file_path should not contains characters "<" and ">". Otherwise, it will be removed from original_file_path.	This will be sanitized and all invalid path and file name characters will be removed.
source	IP, hostname, or other client ID	RECOMMENDED	Helps identify where the file is originated from. This should be end-user information to trace where the file came from.	
X-File-Size	Total size of the original file (before splitted to chunks)	OPTIONAL	Used for chunk uploading. • Both MO and REST have already had these headers before. It still be used now.	
X-File-ID	ID number is retrieved after uploading the 1st chunk	OPTIONAL		

Parameter	Value	Required	Description	Notes
			• Should expose, a big file can be uploaded as many parts. • Case insensitive. • This is data_id returned from 1st request and following request will return same data_id. Any further upload more than x-file-size will result in error (not affecting the scan). Checking progress until all file contents is uploaded will return "not found" Limitation: returned data_id **can be changed** after last chunk uploaded, see this case: database already had scan result for file A, customer uses chunk upload and posts this file again, for the first chunk we return	

Parameter	Value	Required	Description	Notes
			data_id_1 (since don't know that file already exist in DB with only first chunk), when all chunks are uploaded, we calculated hash and found the result in DB, then we return old data_id (different with data_id_1) without scanning.	

Method: POST

HTTP body	Contents of the file for scan	REQUIRED

Response Codes

Code	Status Description	Notes
200	OK	File is uploaded to the server for initiation.
400	Bad Request	Invalid request format or server is not configured correctly for Metascan.
401	Invalid API key	Either missing API key or invalid API is passed.
403	scan limit reached, try again later	
403	exceeded number of allowed clients exceeded number of simultaneous clients	
403	Invalid License	

Code	Status Description	Notes
500	Not enough disk space	Not enough free space in the temp directory to upload the file.
500	License expired	
500	Internal error	
503	Failed to request scan. Try again later.	
503	Server is too busy. Try again later.	

Response

Example of successful scan request:

```
{
     "data_id":"4f84e3b096d54908963d037b5457bf27"
    "rest_ip": "scan01.metascan-online.com/v2"
}
```

Example of failed to upload:

```
{
    "err": "Failed to upload - <error message>"
}
```

Descriptions of response:

Response	Description
data_id	Data ID used for retrieving scan results. Since there are potentially multiple scans for same files when any engine has different definition time or when there is an additional engine, this is identifier for per scan other than per file.

Response	Description
rest_ip	Requests for the scan progress using 'data_id' should be made to this address instead of the original address. Once a scan is finished, future requests for this 'data_id' can be made to the original address.

Retrieve process result by data_id

Description	Query process result
URL Path	/file/{data_id}
Method	GET

Summary

Query a previously submitted file's scanning result. Scan can be initiated the API below. Calling this API is required to retrieve scan results since the initiate API does not return results.

Initiate Process File

You can retrieve scan results using the Data ID. The scan is done asynchronously and each scan request is tracked by the ID. You need to use two separate API calls to initiate a file scan and retrieve the scan results. This request needs to be made multiple times until the scan is complete. You can trace the scan completion using the “scan_results.progress_percentage” value from the response.

Use the following URL to initiate the scan request: http://localhost:8008/metascan_rest/file

Use the following URL to retrieve scan results: http://localhost:8008/metascan_rest/file/{data_id*}

HTTP header parameters

apikey	Only required if REST API keys have been defined	OPTIONAL

Method: GET

URL path	/file/{data_id}	REQUIRED
data_id	Returned from Initiate Process File	REQUIRED

Response Codes

200	**OK**	Found the scan result for given data_id
400	**Bad request**	Not supported HTTP method or invalid HTTP request.
401	**Invalid API key**	Missing apikey header or invalid key passed
404	**Not Found**	Unable to find a scan result for the given data_id. This is an error.
500	**Internal Error**	Server temporary unavailable. Try again later.

Response

Example of when result is not found :

```
Status: 200 (Should be 404)

{
    "4f84e3b096d54908963d037b5457bf27": "Not Found"
}
```

When result is found :

It will return process result. Refer to Response Description.

Response Description

Top levels of Process Result

data_id	Metadefender provides asynchronous API calls (stateless) and data_id is used for identifying a specific process job. It is obtained from initial data (e. g., file) upload and can be used any time as long as scan history is preserved on the server side. It is generated randomly from the server and guaranteed to be unique so it can be used without worrying about having duplicate data_id. Not applicable for hash look up.
scan_results	scan result from each engines along with other metadata for scanning job.
file_info	static file information and analyzed file type based on contents
process_info	overall process status and information including applied workflow and action results such as sanitization.
extracted_files	list of files inside of an archive file. If file is not an archive file or not extracted, this block will not be present.

Minimum Required Parsing Process Results (process_info)

This only applies if you are using workflow-based process.

There should always process_info field for process JSON message.

1. Clients should use **process_info.progress_percentage** to know whether the process has been finished or still in progress.
2. Clients should use **process_info.result** to check whether the file processed is **allowed** or **blocked.**
3. When file is allowed, **process_info.blocked_reason** is empty string**.** If file is blocked, clients should use the field to retrieve detailed block reasons
4. Clients should use **process.profile** to retrieve which profile is used when processing.

Clients should use **process_info.file_type_skipped_scan** to know if the input file's detected type is configured to skip scanning

If post processing is configured and applied, there will be **process.post_processing** field in JSON result. Clients should always check whether this field exists or not to find out if post_processing has been applied or not.

1. If **process_info.post_processing** exists, clients could use **process.post_processing. action_ran** to check what actions have been applied and these actions are separated by "|". See details in [Description of Response]
2. If **process_info.post_processing** exists, clients could use **process.post_processing. action_failed** to check which post_processing failed.
3. If "Move" is in "action_ran", clients should check **process.post_processing. converted_to** to retrieve information of the target file type. And clients should call /file /converted/{data_id} to retrieve link to download converted files
4. if "Copied" or "Moved" is in "action_ran", clients should check **process.post_processing. copy_or_move_destination** for the target directory info

Response Details For process_info

process_info	Process node	Only applied if user scanned with user_agent
progress_percentage	Percentage of processing completed	Scale of 0-100
user_agent	A string which identifies the user. This is used for mapping to a specific workflow.	
profile	The profile name. This is what the user_agent mapped to.	
result	The final result of processing the file	Possible values: Possible Process Results (COM) "Not processed": default value set in REST before the file is processed by Core
blocked_reason	The reason for a file being blocked	Possible list values: • Failed to request a process: "Process Failed" • Possible Blocked Reasons (COM) "null": default value set in REST before the file is processed by Core
file_type_skipped_scan	Indicates if the input file's detected type was configured to skip scanning	JSON Boolean type

post_processing	Post processing node	Exists only if post processing was configured
actions_ran	List of post processing actions ran Delimited by " | "	Possible list values: "Sanitized | Copied | Moved | Quarantined | Deleted | Ran Script" *Copied and Moved will never exist together *Quarantined and Deleted will never exist together *(Copied/Moved) and (Quarantined /Deleted) will never exist together
actions_failed	List of post processing actions that failed Delimited by " | "	Possible list values: "Sanitization Failed | Copy Failed | Move Failed | Quarantine Failed | Delete Failed | Ran Script Failed" *Copy Failed and Move Failed will never exist together *Quarantine Failed and Delete Failed will never exist together *(Copy/Move Failed) and (Quarantine /Delete Failed) will never exist together
converted_to	Extension the file was converted to	If the string contains a file type, the converted file can be downloaded
copy_move_destination	Destination to where the file was copied/moved	
converted_destination	Location where the converted file exists	

Response Details For scan_results and others

file_id	Populated only when files are configured to be stored. Not applicable for hash look up. This is unique per file so if same file is scanned twice, file_id would be identifical.
scan_result	Global scan results, meta data, and scan reports from all engines.
file_info	File metadata, hash value, analyzed file types
scan_details	Array of scan results for engines
scan_result_i	See table below for descriptions
threat_found	Name of threat if detected by engine. Non ASCII characters removed.
def_time	Definition time of engine at the time of scan
scan_time	Elapsed scan time in mili-seconds per engine
rescan_available	This value will be set to true • if engine definition has changed. • any engine is missing scan results such as "failed to scan", "not scanned"
scan_all_result_i	See table below for descriptions
total_time	Elapsed scan time in milli-seconds from when scan is initiated with the first engine until the last engine has completed.
total_avs	Number of engines used for scanning
progress_percentage	Indicator if scan is in progress
in_queue	How many files are in queue before this scan request: 0 means it is being scanned or finished scanning.

upload_timestamp	First time file is uploaded to the server (even if file is rescanned this value will be preserved)
file_type_category	High level categorization of file type see reference for descriptions

Additional Information For An Archive File

Note: Only applied when scanning an archive file

original_file	Basic information and reference to the **data_id** of the original file (actual archive file, not the extracted contents)	**Object Structure** `"original_file": { "data_id": "96c7ce8170224bd29c427ec913e3d00b", "scan_result_i": 1, "progress_percentage": 100, "detected_by": 1 }`
extracted_files	Basic information about the included files in the archive can be found in the **files_in_archive** property. • Files_in_archive only contains maximum 50000 results. • Entries with a non-zero scan_result_i always appear first. • More information about individual entries can be queried with the data_id	**Object Structure** `"extracted_files": { "data_id": "e511f16da3cc422eaa4127ed88ee9446",`

	The **data_id** under extracted_files is the reference to the engines *summary* or consolidated result for the archive.	"scan_result_i": 0,
		"progress_percentage": 100,
		"detected_by": 0,
		"files_in_archive": [{
		"display_name": "\\U6367872802.exe",
		"data_id": "c421c7cfc5c34733ad037116cfe2b6f3",
		"scan_result_i": 0,
		"progress_percentage": 100,
		"file_size": -1,
		"file_type": "",
		"detected_by": 0
		}
		]
		}

parent_data_id	When the scan result for a file within an archive is queried, this refer to the containing archive file.	

Available Scan Results (scan_result_i, scan_all_result_i)

value	short description	long description
0	No Threats Found	No threat detection or the file is empty
1	Infected /Known	Threat is found
2	Suspicious	Classified as possible threat but not identified as specific threat.
3	Failed To Scan	Scanning is not fully performed (For example, invalid file or no read permission). If no engine is included and scan is enabled, this will be the final result.
4	Cleaned / Deleted	Threat is found and file is cleaned (repaired or deleted): repair is not supported yet
5	Unknown	Unknown signature. NOTE: this is only used in multiple hash lookup. For single hash lookup, scan_result_* are not returned as response. see Retrieve scan report by hash (duplicate) for more details.
6	Quarantined	File is quarantined
7	Skipped Clean	Scan is skipped because this file type is in white-list*
8	Skipped Infected	Scan is skipped because this file type is in black-list*
9	Exceeded Archive Depth	Threat is not found but there are more archive levels which were not extracted.
10		Scan is skipped by the engine either due to update or other engine specific reason. If scan is disabled, this will be the final result.

value	short description	long description
	Not Scanned / No scan results	
11	Aborted	All ongoing scans are purged
12	Encrypted	File/buffer is not scanned because the file type is detected as encrypted (password-protected). If the Internal Archive Library is ON encrypted return type is not going to be returned through Metascan scan progress callbacks since the engines do not perform any scan operations. If the Internal Archive Library is OFF Metascan will pass the encrypted files to the engines directly, bypassing the detection.
13	Exceeded Archive Size	The extracted archive is too large to scan
14	Exceeded Archive File Number	There are more files in the archive than configured on the server
15	Password Protected Document	Document that is protected by a password [e.g. Office documents or PDFs that require a password to view its contents]
16	Exceeded Archive Timeout	The archive process reached the given timeout value. This result is supported from Core version 4.4.0.

Scan is not carried out

If scanning is disabled or skipped, a scan_results object will be generated with the following values.

3.11.2 and newer:
scan_results.scan_all_result_i = 10
scan_results.scan_all_result_a = "Not Scanned"

3.11.0 & 3.11.1:
scan_results.scan_all_result_i = -1
scan_results.scan_all_result_a = ""

older than 3.11.0:
scan_results.scan_all_result_i = 1000
scan_results.scan_all_result_a = "No Scan Result"

Scan partially completed

Accumulated result at the time scan result was queried.

This is same as other Metascan interface such as COM interface and Java interface.

Available File Type Categories (file_type_category)

Mark	Note
"E"	Executable (EXE, DLL, …)
"D"	Document (MS Office word document, MS Office excel sheet)
"A"	Archive (Zip, Rar, Tar, …)
"G"	Graphical format (Jpeg, GIF, TIFF, BM P, …)
"T"	Text
"P"	PDF format
"M"	audio or video format
"Z"	mail messages (MSG, …)

Mark	Note
"I"	Disk image
"O"	Other (anything that is not recognize d as one of the above)

Response Example

```
Status: 200

{
  "extracted_files": {
        "data_id": "b3a0949e833840128a346fdd7ec42477",
        "scan_result_i": 0,
        "detected_by": 8,
        "files_in_archive": [
            {
                "display_name": "\\\\eicar.com",
                "data_id": "372d93b94be04ed89da2a7d938b72452",
                "file_type": "-",
                "scan_result_i": 1,
                "detected_by": 4,
                "progress_percentage": 100,
                "file_size": 68
            },
            {
                "display_name": "\\\\Printout.pdf",
                "data_id": "6acba97776694ac89b8e3c573439bfbe",
                "file_type": "PDF/AI",
                "scan_result_i": 0,
                "detected_by": 0,
                "progress_percentage": 100,
                "file_size": 246121
            }
        ]
    },
  "file_id": "",
  "scan_results": {
    "scan_details": {
      "Ahnlab": {
        "threat_found": "EICAR_Test_File",
        "scan_result_i": 1,
        "def_time": "2016-08-03T00:00:00Z",
        "scan_time": 1
      },
      "Avira": {
```

```
        "threat_found": "",
        "scan_result_i": 0,
        "def_time": "2016-08-03T00:00:00Z",
        "scan_time": 15
      },
      "ClamAV": {
        "threat_found": "Heuristics.PDF.ObfuscatedNameObject",
        "scan_result_i": 1,
        "def_time": "2015-07-03T00:00:00Z",
        "scan_time": 31
      },
      "ESET": {
        "threat_found": "",
        "scan_result_i": 0,
        "def_time": "2016-08-03T00:00:00Z",
        "scan_time": 1
      }
    },
    "rescan_available": false,
    "data_id": "c0983103f31f4e74889cc7188f6c0cdd",
    "scan_all_result_i": 1,
    "start_time": "2016-08-03T17:08:52.949Z",
    "total_time": 31,
    "total_avs": 4,
    "progress_percentage": 100,
    "in_queue": 0,
    "scan_all_result_a": "Infected"
  },
  "file_info": {
    "file_size": 2073,
    "upload_timestamp": "2016-08-03T17:08:52.949Z",
    "md5": "4D6B72DDC54514F82DDCDDA2E9923AAF",
    "sha1": "7B7B5487948E4EDC1AF62539ABF0DD72B456D03E",
    "sha256": "1A90EA551EE596F357794DFF314F042D84A9AFC3D56ED2C995A
4908395CBABBC",
    "file_type_category": "P",
    "file_type_description": "Adobe Portable Document Format",
    "file_type_extension": "pdf/ai",
    "display_name": "1470244133_dirty_315.pdf",
    "original_file_path": ""
  },
  "process_info": {
        "post_processing": {
            "actions_ran": "Sanitized",
            "actions_failed": "",
            "converted_to": "pdf",
            "copy_move_destination": "",
            "converted_destination": "toConvert_[sanitized].pdf"
        },
        "progress_percentage": 100,
        "user_agent": "cactus",
```

```
        "profile": "default",
        "result": "Blocked",
        "blocked_reason": "Dirty",
        "file_type_skipped_scan": false
    },
    "data_id": "c0983103f31f4e74889cc7188f6c0cdd",
    "rescan_count": 0,
    "source": "10.0.50.37",
    "scanned_on": "10.0.50.52"
}
```


Available File Type Categories (file_type_category)

Mark	Note
"E"	Executable (EXE, DLL, …)
"D"	Document (MS Office word document, MS Office excel sheet)
"A"	Archive (Zip, Rar, Tar, …)
"G"	Graphical format (Jpeg, GIF, TIFF, BM P, …)
"T"	Text
"P"	PDF format
"M"	audio or video format
"Z"	mail messages (MSG, …)
"I"	Disk image
"O"	Other (anything that is not recognize d as one of the above)

Available Scan Results (scan_result_i, scan_all_result_i)

value	short description	long description
0	No Threats Found	No threat detection or the file is empty
1	Infected /Known	Threat is found
2	Suspicious	Classified as possible threat but not identified as specific threat.
3	Failed To Scan	Scanning is not fully performed (For example, invalid file or no read permission). If no engine is included and scan is enabled, this will be the final result.
4	Cleaned / Deleted	Threat is found and file is cleaned (repaired or deleted): repair is not supported yet
5	Unknown	Unknown signature. NOTE: this is only used in multiple hash lookup. For single hash lookup, scan_result_* are not returned as response. see Retrieve scan report by hash (duplicate) for more details.
6	Quarantined	File is quarantined
7	Skipped Clean	Scan is skipped because this file type is in white-list*
8	Skipped Infected	Scan is skipped because this file type is in black-list*
9	Exceeded Archive Depth	Threat is not found but there are more archive levels which were not extracted.
10		Scan is skipped by the engine either due to update or other engine specific reason. If scan is disabled, this will be the final result.

value	short description	long description
	Not Scanned / No scan results	
11	Aborted	All ongoing scans are purged
12	Encrypted	File/buffer is not scanned because the file type is detected as encrypted (password-protected). If the Internal Archive Library is ON encrypted return type is not going to be returned through Metascan scan progress callbacks since the engines do not perform any scan operations. If the Internal Archive Library is OFF Metascan will pass the encrypted files to the engines directly, bypassing the detection.
13	Exceeded Archive Size	The extracted archive is too large to scan
14	Exceeded Archive File Number	There are more files in the archive than configured on the server
15	Password Protected Document	Document that is protected by a password [e.g. Office documents or PDFs that require a password to view its contents]
16	Exceeded Archive Timeout	The archive process reached the given timeout value. This result is supported from Core version 4.4.0.

Scan is not carried out

If scanning is disabled or skipped, a scan_results object will be generated with the following values.

3.11.2 and newer:
scan_results.scan_all_result_i = 10
scan_results.scan_all_result_a = "Not Scanned"

3.11.0 & 3.11.1:
scan_results.scan_all_result_i = -1
scan_results.scan_all_result_a = ""

older than 3.11.0:
scan_results.scan_all_result_i = 1000
scan_results.scan_all_result_a = "No Scan Result"

Scan partially completed

Accumulated result at the time scan result was queried.

This is same as other Metascan interface such as COM interface and Java interface.

4. Release Notes

3.12.3

New features

- Metadefender Core administrators can now configure which type of scan results will be labeled 'blocked', and which will be 'allowed'
- The Metadefender Mail Agent now supports Office365 and Google Apps email servers
- The Metadefender Mail Agent now supports installation on Microsoft Exchange servers

Other changes

- The Avira scan engine has been updated
- Display of scan results in the Management Console has been improved, including the source of the scan and the workflow applied
- The threat name is now included on the 'block' page when using the ICAP interface
- The number of files in the scan pending queue is now limited to 20,000. The /file API will now return 503 (Service unavailable) if the maximum of 20,000 files has been reached.
- A minimum of 3.5 GB of free disk space is now required when files are submitted for scanning. If the machine has less than 3.5GB free the /file API will return 503.
- Product documentation has been updated to a new format

General and Known Issues

General

- Metadefender Core 1, 4, and 8 come with a trial license good for 15 days after fresh installation (reinstalling does not reset trial).
- Metadefender Core 12, 16, and 20 do not come with a trial license. To obtain a trial license, please contact OPSWAT sales (sales@opswat.com).
- Metadefender Core should be restarted whenever a new license has been applied.
- Metadefender Core was previously called Metascan.
- Metadefender Core categorizes files as 'blocked' or 'allowed' based on the process results. Please refer to Callback for Processing a File (COM) for details.

Known issues

- Metadefender Core embedded engines should not be installed on a system which has an alternate product provided by the same vendor, for example, installation of the Norman Scan Engine with Norman Security Suite. This configuration is not supported by Metadefender Core.
- Metadefender Core license allows up to 10 Remote Desktop (RDP) sessions to a server running Metadefender Core . Please contact OPSWAT support if you need more than ten connections via RDP.
- Scanning boot sector is only supported by the Total Defense/Quick Heal scan engine. Other engines will fail to scan for boot sector scan request.
- REST Web Service installation may fail to install if UAC (User Access Control) is enabled. Turn off UAC and run the installation with Administrator privileges.
- If logging all scan history is enabled, the Metadefender Core Management console dashboard may cause high load on mongoDB.
- ICAP server does not block traffic if server is overloaded by default.
- Metadefender Core does not support installation to directory paths that include non-ASCII characters.
- Metadefender Core must have archive handling enabled in order to correctly scan GZIP content through the ICAP interface.
- When Metadefender Core is configured to use a local instance of MongoDB (the default configuration) the HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\OPSWAT\Metascan\db_instance (or HKEY_LOCAL_MACHINE\SOFTWARE\OPSWAT\Metascan\db_instance for 32 bit systems) key must be set to 'localhost' or '127.0.0.1'. The local system cannot be referred to by IP address.
- Metadefender Core does not support installation on systems where the TEMP environment variable contains more than one path.
- In Metadefender Core 20 packages, the Sophos engine must have its definitions updated before scans can successfully complete.
- Archives scanned through the COM interface do not have their scan results logged.
- The setting "Maximum File Size for files scanned through the web interface" found on the Configuration->Scan Configuration page in the Metadefender Core Management Console cannot currently be imported or exported through the Backup/Restore feature.
- Disabling the 'support_custom_engine' property will disable all engines in Metadefender Core.

- Threats found within archives will not be removed even if Metadefender Core is configured to delete or quarantine files that contain threats.
- Stopping the Metadefender Core service may not stop all engine handlers (omsAMEHandler.exe and omsCEHandler.exe processes) if there are engine updates in progress. Engine handlers will terminate after the update has completed or they can be terminated manually.

The following are applicable only if the Metadefender Mail functionality is being used:

- Metadefender Email agent will be installed as part of Metadefender Core. Before installing Metadefender Core, please uninstall mail agent before installing Metadefender Core.
- A separate SMTP server is required for notification.
- Exchange (malware agent configuration) may block release from quarantine.
- Email sanitization to HTML may be blocked and attachment may be replaced with text "This attachment was removed because it contains data that could pose a security risk."
- An attachment that has been sanitized may be larger than the original attachment. As a result, it may be necessary to increase the maximum attachment size setting on your mail server

Usable AVs (antivirus)

Usable AVs are the anti-malware engines that Metadefender Core uses to process scan requests. All embedded engines are usable anti-malware engines. For anti-malware applications that are installed by you, please check the list of products that Metascan supports.

This list can be retrieved from:

- Init API
- Any command line utility (omsCmdUtil.exe) command
- Metadefender Core Management Console Configuration Page

Current AVs

Current AVs are anti-malware engines that Metadefender Core automatically uses for scans and updates. This list is affected by licenses and properties such as included antiviruses, excluded antiviruses, and supported pre-installed antiviruses.

For example, if you installed Metadefender Core 8 and you have McAfee VirusScan installed on your machine, but you exclude the ESET scanning engine and disable customer-licensed products, your usable AVs and current AVs will be designated as below:

Usable AVs (Usable AMs)	Current AVs
• Avira scan engine • ESET scan engine • ClamAV scan engine • AhnLab scan engine • ThreatTrack scan engine • Bitdefender scan engine • Total Defense scan engine • Quick Heal scan engine • McAfee VirusScan	• Avira scan engine • ClamAV scan engine • AhnLabscan engine • ThreatTrack scan engine • Bitdefender scan engine • Total Defense scan engine • Quick Heal scan engine

Synchronous/asynchronous scan

Metadefender Core can be delegated to queue scan requests and return the results of queued scans via callbacks with extensive scan details. Even if your application is single threaded, you can process many scan requests without blocking. In other words, as soon as you have data to be scanned and you call the asynchronous scan API of Metadefender Core (for example, PutToScanQueue), you can continue to request scans without waiting for the results of the scan.

On the other hand, a synchronous scan is more suitable if your application creates scan requests in many threads. Refer to the ScanEx section on how to retrieve scan details when scanning synchronously.

Valid file name

A valid file name as a parameter for any API related to scanning files should meet the following requirements:

- Absolute path to file, folder, and drive (e.g., c:, C:\Windows, C:\boot.ini)
- File path should be limited to the local machine (e.g., “\\remote_machine\testfile.txt” would be recognized as an invalid path)

Legal

Copyright

DISCLAIMER OF WARRANTY

OPSWAT Inc. makes no representation or warranties, either express or implied by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect special or consequential damages.

COPYRIGHT NOTICE

Export Classification EAR99

EAR99 (Export Administration Regulation 99) is an export classification category regulated by the U.S. Department of Commerce that covers most commercial items exported out of the U.S.

OPSWAT's software is designated as EAR99, and there are no export restrictions other than embargoed countries and persons.