

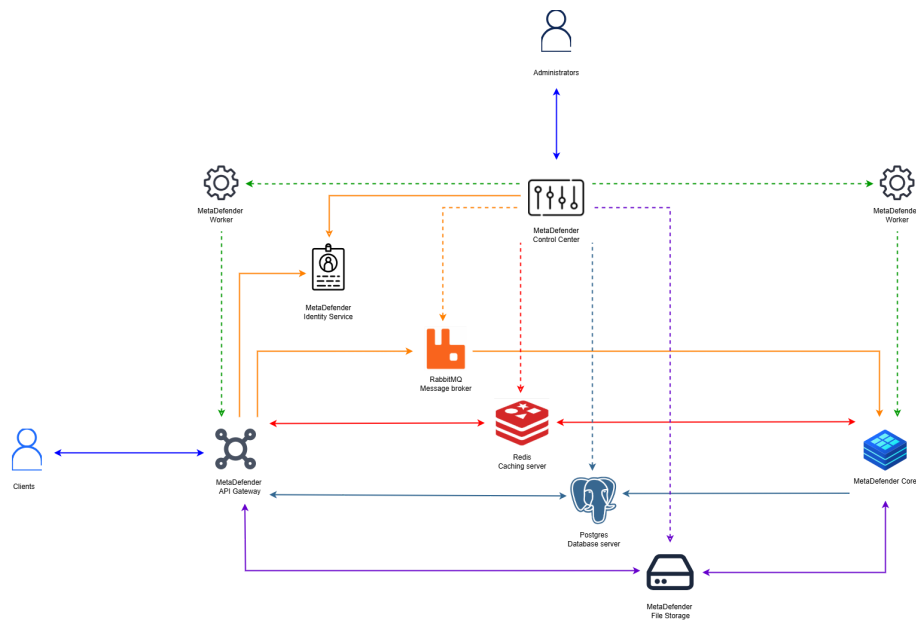
MetaDefender Distributed Cluster v2.4.0

Table of Contents

Installation	1
Overview	1
System requirements	4
Installation	8
Physical or Virtual Machine-Based Setup	9
MetaDefender Distributed Cluster File Storage	20
MetaDefender Distributed Cluster Identity Service	30
MetaDefender Distributed Cluster Worker	41
MetaDefender Distributed Cluster Control Center	51
Container-Based Setup	56
Recommended Setup	91
License activation	93
Online Activation	94
Offline Activation	97
Module update	102
Configurations	106
High Availability	106
High Availability support for File Storage	107
High Availability support for RabbitMQ	110
High Availability support for Redis	114
High Availability support for PostgreSQL Data lake	118
System settings	128
Data Retention	129
Remote Support Package Gathering	131
Security	134
File Storage	143
Upgrade	148

System Upgrade	148
Performance	153
Performance and Load Estimation	153
Troubleshooting	167
Log Gathering in MetaDefender Distributed Cluster	167
Release Notes	170
Release notes	170
API Gateway	173
Control Center	237

Overview



The MetaDefender Distributed Cluster (MDDC) is an approach to serve very large deployments while offering improved auto-scaling, high availability and fault tolerant capabilities for MetaDefender Core.

The MetaDefender Distributed Cluster consists of several components:

Component	Functionalities
MDDC Control Center	Assist administrators with user management, system health monitoring, and deploying or upgrading MetaDefender Core or API Gateway without any downtime.
MDDC Identity Service	Assist Control Center and API Gateway in client authentication, managing user activity sessions and authorization.
MDDC File Storage	Securely store and share files asynchronously across components in the cluster. The component manages the duration and duplication of files.
MDDC Worker	Deploy and monitor activities of MetaDefender Core and MDDC API Gateway.
MDDC API Gateway	Accept file scans, fetch scan statuses, and process download requests from clients.
MetaDefender Core	Scan the accepted files.
RabbitMQ - Message broker	Receives tasks from the API Gateway and forwards them to MetaDefender Core instances for processing.
Redis - Caching server	Store in-progress results in memory for rapid retrieval.
PostgreSQL - Database sever	Permanently store scan results, configuration and executive reports.

The Distributed Cluster offers users two distinct interfaces. The first is a RESTful interface provided by MDDC **API Gateway** for applications to upload files for scanning, retrieve scan status, download processed files, or abort file scanning. The other is a Web UI provided by MDDC **Control Center** for the system administrator to manage licenses and users, modify workflow configurations, monitor the overall system, and remotely deploy or upgrade MDDC **API Gateway** or **MetaDefender Core**.

When a file is submitted to the MDDC **API Gateway** for scanning, its body content is securely transmitted to MDDC **File Storage** for subsequent use. **API Gateway** submits a scan task in RabbitMQ queue, and responds to the application with `data_id`. The task is delivered to healthy **MetaDefender Core** instances and one of them will accept the task. The file corresponding to the task is transmitted from MDDC **File Storage** to the instance's local storage, and the processing of the file takes place. Scan results produced by the processing are continuously recorded in Redis for fast retrieval and are finally stored in the PostgreSQL database for long-term storage. If created, the sanitized or watermarked file is securely transmitted to MDDC **File Storage** for future download by MDDC **API Gateway**.

In certain rare situations, if one of the **MetaDefender Core** instances unexpectedly ceases operation, its 'broken' files are delivered to other **MetaDefender Core** instances for continued processing without the need for applications to resubmit the files. By leveraging MDDC **File Storage** and RabbitMQ, **MetaDefender Core** instances within MetaDefender Distributed Cluster can collaborate in distributing the workload of archive extraction, greatly decreasing the overall time required to process archive files while utilizing the resources much more efficiently.

Using the Web Console from by MDDC **Control Center**, the system administrator is able to adjust workflow settings centrally and, after which the updates are automatically synced across all **MetaDefender Core** instances. The administrator can scale out the number of MDDC **API Gateway** or **MetaDefender Core** instances if additional power is required. He or she can also upgrade the instances seamlessly while the file processing is occurring. All statistical data and health information for components, along with executive reports, can be accessed easily through the Web UI of MDDC **Control Center**.

System requirements

Windows

Component	Minimum version	Dependencies	Recommended System Specs
PostgreSQL Database Server	16.9		Vendor recommendation
RabbitMQ Messaging Broker	3.13.0	64 bit Erlang/OTP 25.0 or above.	Vendor recommendation
MetaDefender Distributed Cluster File Storage	2.3.0	Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above.	Minimum of 8 CPU cores and 8 GB of RAM required.
MetaDefender Distributed Cluster Control Center	2.3.0	Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above.	Minimum of 4 CPU cores and 4 GB of RAM required.
MetaDefender Distributed Cluster Identity Service	2.3.0	Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above.	Minimum of 4 CPU cores and 4 GB of RAM required.
MetaDefender Distributed Cluster Worker for MetaDefender Distributed Cluster API Gateway	2.3.0	Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above.	Minimum of 4 CPU cores and 8 GB of RAM required.
MetaDefender Distributed Cluster Worker for MetaDefender Core	2.3.0	Microsoft Visual C++ Redistributable 2019 version 14.29.30139.0 or above.	System Configuration

 Info

WMIC, by default, is disabled since Windows 11. To enable it, please run the following command as Administrator in Command Prompt:

```
DISM /Online /Add-Capability /CapabilityName:WMIC
```

Debian/Ubuntu or Red Hat/Rocky

Component	Minimum version	Dependencies	Recommended System Specs
PostgreSQL Database Server	16.9		Vendor recommendation
Redis Caching Server	7.0.5		Vendor recommendation
RabbitMQ Messaging Broker	3.13.0	64 bit Erlang/OTP 25.0 or above.	Vendor recommendation
MetaDefender Distributed Cluster File Storage	2.3.0	uuid package. tar tool. lsb_release tool.	Minimum of 8 CPU cores and 8 GB of RAM required.
MetaDefender Distributed Cluster Control Center	2.3.0	uuid package. tar tool. lsb_release tool.	Minimum of 4 CPU cores and 4 GB of RAM required.
MetaDefender Distributed Cluster Identity Service	2.3.0	uuid package. tar tool. lsb_release tool.	Minimum of 4 CPU cores and 4 GB of RAM required.
MetaDefender Distributed Cluster Worker for MetaDefender Distributed Cluster API Gateway	2.3.0	uuid package. tar tool. lsb_release tool.	Minimum of 4 CPU cores and 8 GB of RAM required.

Component	Minimum version	Dependencies	Recommended System Specs
MetaDefender Distributed Cluster Worker for MetaDefender Core	2.3.0	uuid package. tar tool. lsb_release tool.	System configuration

Info

`tar`, by default, is not included in some Linux distributions. Please run the following command in Terminal to install `tar`:

- Debian/Ubuntu: `sudo apt install tar`
- Red Hat/Rocky: `sudo dnf install tar`

Info

`lsb_release`, by default, is not included in **Rocky**. Please run the following command in Terminal to install `lsb_release`

```
sudo dnf install -y yum-utils
```

```
sudo dnf config-manager --set-enabled devel
```

```
sudo dnf update -y
```

```
sudo dnf install -y redhat-lsb-core
```

Installation

This section includes guidance for installing and setting up the MetaDefender Distributed Cluster on physical machines, virtual machines, or in containers.

Physical or Virtual Machine-Based Setup

Prerequisite

Before executing the setup, please ensure [System requirements](#) are met and that any necessary dependencies are installed.

Installation order

MetaDefender Distributed Cluster consists of the following components along with their corresponding default ports.

Step	Component	How to install [short name]	Default port
1	Redis Caching Server	Redis	6379
2	RabbitMQ Message Broker	RabbitMQ	5672
3	PostgreSQL Database Server	PostgreSQL	5432
4	MetaDefender Distributed Cluster File Storage	MDDC File Storage	8890
5	MetaDefender Distributed Cluster Identity Service	MDDC Identity Service	8891
6	MetaDefender Distributed Cluster Control Center	MDDC Control Center	8892
7	MetaDefender Distributed Cluster Worker	MDDC Worker	8893

The system administrator should adhere to the following service installation sequence to prevent conflicts:

1. Install Redis, RabbitMQ, Postgres, MDDC **File Storage**, MDDC **Identity Service**.

2. Install MDDC **Control Center**.
3. Install MDDC **Worker** on the targeted machines (for deploying MDDC **API Gateway** or **MetaDefender Core**).

i Warning

An exception rule for the firewall needs to be created to permit both incoming (inbound) and outgoing (outbound) connections to every component.

i Info

While many components can be set up on a single machine, they should be installed individually on different machines according to their features. Kindly consult [Best practices](#) for further information.

Installation

Install Redis Caching Server

i Info

Redis version 7.0 or higher is required.

Only Redis on Linux is officially recommended.

1. Follow steps to install Redis Caching server.
2. Access Redis configuration file `/etc/redis/redis.conf` for editing.
3. Comment out the `bind` setting and set `protected-mode` option to **no**.

redis.conf none

```
...
# The following line should be commented
# bind 127.0.0.1
...
# The following line should be uncommented and set to no
protected-mode no
...
```

4. Restart the service.

bash

```
# Red Hat/Rocky
$ sudo systemctl enable redis
$ sudo systemctl restart redis

# Debian/Ubuntu
$ sudo systemctl enable redis-server
$ sudo systemctl restart redis-server
```

Install RabbitMQ Message Broker

Info

RabbitMQ version 3.13.0 or higher is required.

Warning

RabbitMQ functions effectively only with specific supported versions of Erlang. Please refer to [the link](#) for the Erlang-RabbitMQ compatibility matrix.

Windows

1. Download Erlang and follow the instructions to install Erlang.
2. Download RabbitMQ for Windows.
3. Run the executable file as administrator, follow instructions to complete the RabbitMQ installation.
4. In Command Prompt, change working directory to `<RabbitMQ installation folder>/rabbitmq_server-<version>/sbin` and run the following command:

None bash

```
> rabbitmqctl.bat add_user <username> <password>
> rabbitmqctl.bat set_permissions -p / <username> "." "." "."
> rabbitmqctl.bat set_user_tags <username> administrator
```

Linux

1. Download Erlang and follow the instructions to install Erlang and its dependencies.
2. Download RabbitMQ for Red Hat/Rocky or Debian/Ubuntu .

3. In Terminal, run the following command:

bash

```
# Red Hat/Rocky
$ sudo rpm -Uvh --nodeps rabbitmq-server-<rabbitmq
version>.el8.noarch.rpm
$ sudo systemctl enable rabbitmq-server
$ sudo systemctl start rabbitmq-server

# Debian/Ubuntu
$ sudo dpkg -i rabbitmq-server-<rabbitmq version>_all.deb
```

4. In Terminal, run the following command:

None bash

```
$ sudo rabbitmqctl add_user <username> <password>
$ sudo rabbitmqctl set_permissions -p / <username> "." "." "."
$ sudo rabbitmqctl set_user_tags <username> administrator
```

Install PostgreSQL Database Server

Info

PostgreSQL version 16.9 or higher is required.

`pg_trgm` extension is required for PostgreSQL running on Linux.

1. Download PostgreSQL Database Server.
2. Follow steps to setup Postgres Database Server to allow connections from external applications.
3. Restart Postgres Database Server.

Install MDDC File Storage

1. Build Ignition file for MDDC **File Storage** service.
2. Start **Command Prompt as Administrator on Windows** or **Terminal on Linux** and run the following command:

bash

```
# Windows
> msixexec.exe /i <mddc_file_storage_package> /qn

# Debian or Ubuntu
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_file_storage_package> || sudo apt install
-f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_file_storage_package> -y
```

3. Check the service status.

Install MDDC Identity Service

1. Build Ignition file for MDDC **Identity Service**.
2. Start **Command Prompt as Administrator on Windows** or **Terminal on Linux** and run the following command:

bash

```
# Windows
> msixexec.exe /i <mddc_identity_service_package> /qn

# Ubuntu or Debian
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_identity_service_package> || sudo apt
install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_identity_service_package> -y
```

3. Check the service status.

Install MDDC Control Center

1. Build Ignition file for MDDC **Control Center** service.
2. Start **Command Prompt as Administrator on Windows** or **Terminal on Linux** and run the following command:

bash

```
# Windows
> msixexec.exe /i <mddc_control_center_package> /qn

# Ubuntu or Debian
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_control_center_package> || sudo apt
install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_control_center_package> -y
```

3. Check the service status.

Setup Data Lake and Data Warehouse

1. Go to `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Control Center` directory in **Windows Command Prompt** or `/usr/sbin` directory in **Linux Terminal**.
2. Run the following command:

bash

```
# Windows
> mddc-dbreedy.exe --host=<postgres-host> --port=<postgres-
port> --user=<postgres-user> --password=<postgres-password> --
target=lake,warehouse

# Linux (Ubuntu, Debian, Red Hat or Rocky)
$ mddc-dbreedy --host=<postgres-host> --port=<postgres-port> -
-user=<postgres-user> --password=<postgres-password> --
target=lake,warehouse
```

Info

Make sure the `postgres-user` possesses **superuser** rights to successfully create the database.

Install MDDC Worker

Info

You need to prepare at least **two** workers: one for **MetaDefender Core** and the other for **MDDC API Gateway**.

1. Build Ignition file for MDDC **Worker** service.
2. Start **Command Prompt as Administrator on Windows** or **Terminal on Linux** and run the following command:

bash

```
# Windows
> msixexec.exe /i <mddc_worker_package> /qn

# Ubuntu or Debian
$ sudo apt -y install uuid
$ sudo dpkg -i <mddc_worker_package> || sudo apt install -f

# Red Hat or Rocky
$ sudo dnf -y install uuid
$ sudo yum install <mddc_worker_package> -y
```

3. Check the service status.
4. Repeat the above steps for other MDDC **Workers**.

Configurations

When all MDDC **Worker** instances are installed successfully, it marks a completed installation. Now, heading to essential configuration steps.

Connect essential services

Info

Essential services for the Distributed Cluster includes Redis, Postgres, RabbitMQ, and MDDC **File Storage**.

The system is operational only when MDDC **Control Center** can effectively connect to all essential services.

1. Sign in to MDDC **Control Center** web console with the initial administrator user account that you created in Install MDDC **Identity Service**.

2. Go to **Inventory > Services**, open the relevant service category, and click on **Add service**.

OPSWAT
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Modules

Licenses

Certificates

Settings

Protecting the
World's Critical Infrastructure

Please connect all services below.

Type

Data Lake

Instance Count

0

Status

Unconnected

Service is not connected

No services have been added. Please add service to connect the system.

+ Add service

Type

Data Warehouse

Instance Count

0

Status

Unconnected

Service is not connected

No services have been added. Please add service to connect the system.

+ Add service

Type

File Storage

Instance Count

0

Status

Unconnected

Service is not connected

No services have been added. Please add service to connect the system.

+ Add service

4. Complete all necessary fields as specified by the selected services.

OPSWAT
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Modules

Licenses

Certificates

Settings

Protecting the
World's Critical Infrastructure

Type

Data Warehouse

Instance Count

0

Status

Unconnected

Service is not connected

No services have been added. Please add service to connect the system.

Name*

Host*

Port*

Data Warehouse Server

192.168.10.1

5432

Username*

Password*

postgres

✕

✓

+ Add service

Type

File Storage

Instance Count

0

Status

Unconnected

Service is not connected

No services have been added. Please add service to connect the system.

Name*

Host*

Port*

File Storage Server

192.168.10.11

8883

Connection Key*

5. Save result.

6. Check status of the service connections.

OPSWAT
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Modules

Licenses

Certificates

Settings

Protecting the
World's Critical Infrastructure

Inventory / Services

✓ All your services are connected. You can start deploying. [Go to Workers](#)

Refresh

Type

Data Lake

Instance Count

1/1

Status

Healthy

Type

Data Warehouse

Instance Count

1/1

Status

Healthy

Type

File Storage

Instance Count

1/1

Status

Healthy

Type

RabbitMQ

Instance Count

1/1

Status

Healthy

Type

Redis

Instance Count

1/1

Status

Healthy

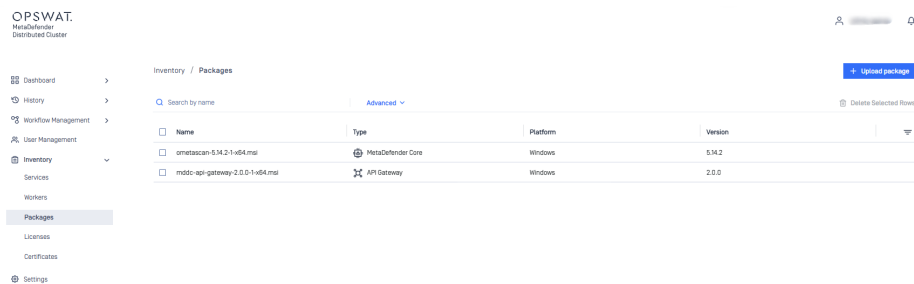
Submit MetaDefender Core and MDDC API Gateway packages

Info

The packages are installation files of **MetaDefender Core** and MDDC **API Gateway**, which will be subsequently deployed on MDDC **Worker** remotely by MDDC **Control Center**.

Various versions of installation files may be submitted to MDDC **Control Center**. The correct version to install will be chosen during Deployment phase.

1. Sign in to MDDC **Control Center** web console with the initial administrator user account that you created in Install MDDC **Identity Service**.
2. Go to **Inventory > Packages** and select **Upload package**.
3. Select **MetaDefender Core** or MDDC **API Gateway** installation files.
4. Click **Update**.

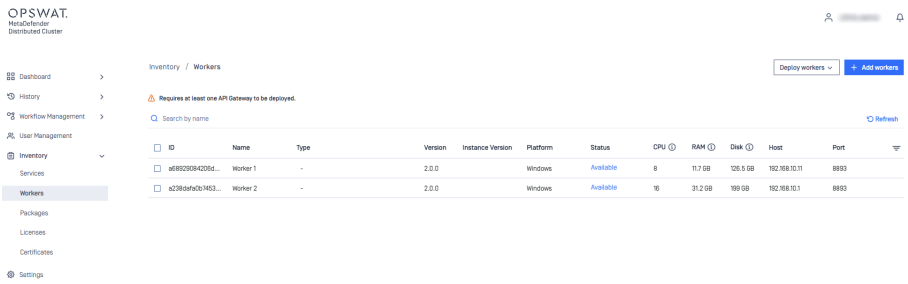


Connect to MDDC Workers

1. Sign in to MDDC **Control Center** web console with the initial administrator user account that you created in Install MDDC **Identity Service**.
2. Go to **Inventory > Workers** and select **Add workers**.
3. Complete the required fields to add new workers and select **Submit**.

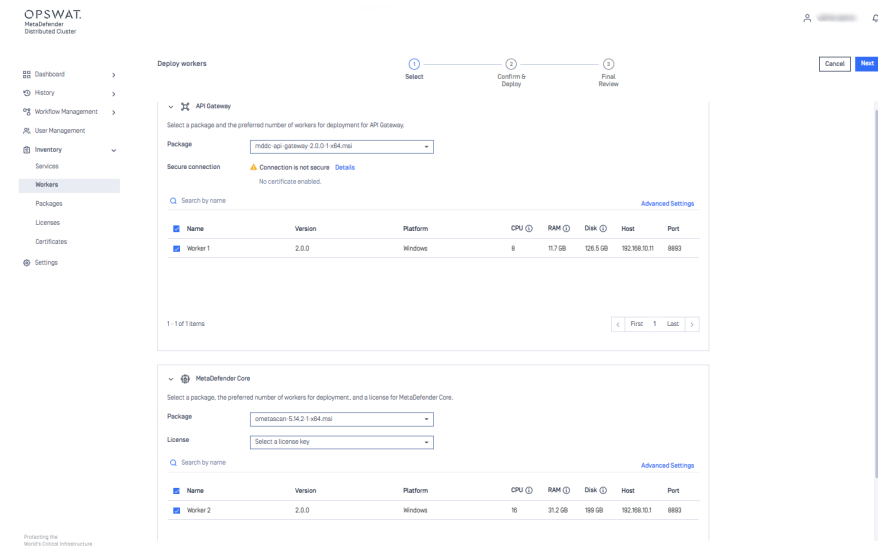


4. Check the status of MDDC **Worker** connections.



Deploy MetaDefender Core and MDDC API Gateway instances

1. Sign in to MDDC **Control Center** web console with the initial administrator user account that you created in MDDC **Identity Service**.
2. Go to **Inventory > Workers** and select **Deploy workers**.
3. Choose the workers for deployment and decide which package will be deployed on the workers.



Info

The license to activate **MetaDefender Core** instances can be selected in this phrase.

Advanced settings (enable HTTPS, select log level, define engine parallel count, etc.) are configurable.

4. Choose **Next**.
5. Confirm the deployment details, then click **Deploy** and **Finish**.
6. Hold off until the deployment is completed successfully.

OPSWAT
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Licenses

Certificates

Settings

Inventory / Workers

Deploy workers Add workers

Search by name

Refresh

ID	Name	Type	Version	Instance Version	Platform	Status	CPU	RAM	Disk	Host	Port
a68929594205d...	Worker 1	API Gateway	2.0.0	2.0.0	Windows	Running	8	11.7 GB	126.5 GB	192.168.10.11	8883
a238a9a5d67453...	Worker 2	MetaDefender Core	2.0.0	5.14.2	Windows	Running	16	31.2 GB	199 GB	192.168.10.1	8883

7. Once the system is up, MDDC **API Gateway** can efficiently accept scan requests.

MetaDefender Distributed Cluster File Storage

Ignition file

Info

The ignition file is required only for a clean installation.

The following fields are essential for the ignition file:

- `secure.connection_key`
- `secure.private_key`
- `secure.certificate`

To install MetaDefender Distributed Cluster (MDDC) **File Storage** server, ignition file in YAML format is required at the following location:

- Windows: `C:\opswat\mddc_file_storage.yml`
- Linux: `/etc/opswat/mddc_file_storage.yml`

The ignition file includes fields:

Key path	Value type	Accepted values	Required	Description
secure.connection_key	String	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	Required	An arbitrary string that enables clients to connect to the server. Use this value as input when adding MDDC File Storage in the UI of MDDC Control Center .
secure.private_key	String		Required	Content of private key in X509 format.
secure.certificate	String		Required	Content of certificate in X509 format.
storage.path	String		Optional	Path to an existing directory where the MDDC File Storage server stores its files. The server requires full permissions to access the path in Linux.
restrict.host	String		Optional	IP address (V4/V6) or host where the server resides on. Default value is '*' Notes: value '*' allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to 0.0.0.0
restrict.port	Number		Optional	The port where the server resides on. Default value is 8890

Key path	Value type	Accepted values	Required	Description
log. streams [@]. log_ type	String	- file - syslog	Optional	Type of log device.
log. streams [@]. log_ level	String	- dump - debug - info - warning - error	Optional	Level of log message.
log. streams [@]. log_ path	String	If log.streams[@].log_type is "file" then log.streams[@].log_path is the path to a file on file system where logs are written. If log.streams[@].log_type is "syslog" then - log.streams[@].log_path can be [tcp/udp]://host:port where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. - log.streams[@].log_path can be "local" to write log to local syslog server (Linux only).	Optional	Location where logs are written.

Info

If `storage.path` is not defined in the Ignition file, MetaDefender Distributed Cluster File Storage will save the submitted files in the default storage directory according to the platform:

- On Windows, `<install-directory>/data/storage`
- On Linux, `/var/lib/mddc-file-storage/storage`

Warning

The default storage directory will be deleted when MDDC **File Storage** is uninstalled.

Configuration file

After successfully installing, MDDC File Storage generates a configuration file with changeable settings at the following location:

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster File Storage\mddc_file_storage.yml`
- Linux: `/etc/mddc-file-storage/mddc_file_storage.yml`

Info

The service must be restarted to take the new configurations into effect.

Sample

Info

OpenSSL or a similar tool [e.g., `ssh-keygen`] can create a pair of public and private keys in X.509 format.

yaml

```
secure:
  connection_key: "1234abcd" # [0-9a-zA-Z]{4,64}
  private_key: |
    -----BEGIN PRIVATE KEY-----

MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0
tJ

PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAm
HK

nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5
ZZ

toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6D
SU

0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyM
BT

J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8
pS

Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2p
HN

uC53QVc/EF++GBLAxmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCFIBKIzv9cg
i9

fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqvV0nPN1IM
Sn

zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtgzoGMTvP/
Au

ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTM
aB

QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABAOICAFWe8MQZb37k2gdAV3Y6aq
8f

qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIo
H+

vXUpW0jnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0U0iSMMftp
k9

fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht
5A
```

Kqx9mld20v17NjgiC1Fx9r+fZw/i0abFFwQA4dr+R8mEMK/7bd4VxfQ1o/QGGB
MT

G+u1FrSiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBt0DU372a2CQK/u4
Q/

HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKT1
Q8

YZkpIcLNVLw0usoGYHFm2rvCyEV1fsE3Ub8cFyTFk50Se0cF2QL2xzKmbZEpX
gl

xBHR0hjgon0IKJDGfor4bH07Nt+1Ece8u2oTEKvpz5aIn440eC5mApRGy83/0b
vs

esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRI1G56ErxC3l0s11wrIpTmXXck
qw

zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BX1vp2GTBIeoxrWB4V3YArwu8QF80QBg
Vz

mgHl24nTg00UH10jZsABAOIBAQD0xftSdbSqGytcWqPYP3SZHAWDA004ACEM+e
Cw

au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN
77

40Qdyk1lTzTVR0qmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwf
s5

YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBSHnK8xrGaME7W0qTmb
FH

PgQz5FcXrjCCqwhilwNBELDTp/ZECEB7y34khVh531mBE2mNzSVIQCZP1I/Dv
Xj

W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFww
uR

naVJS6Wx0KjX8MCu9/cKPnwBv2mmI2jgXHTw5sr3ahmF5eTb8Zo19BowytN+t
r6

2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBL0mbSUHn3PG
2m

39I802u0fFNVQCJKhx1m1MFFL0u7lVcDS9JN+oYVPb6MDfBLm5j0iPuYkFZ4gH
79

J7gXI0/YKhaJ7yXthYVkdRsf6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+Rw

DC

R1thHrmf/IXp1xpNVkoMVxtzbrrbgnC25QmvRYc0r1S/kvM4yQhMH3eA7IycDZ
Mp

Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItN11IZZMW2TPUiY0rl6ya
Ch

BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxJzjaVtxN7ZXwZv0VrY7I7fPWYn
CE

fXCr4+IVpZI/ZHZWpGX6CGSgT6E0jCZ5IUufIvEpqVSmtF8MqfX09o9uIYLokr
WQ

x1dB15UnuTLDqw8bChq705y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEf
HI

U1GdNVomwXwYEzge/c19ruIowX7HU/NgxMWTMZhpaZlxgesXybel+YNcfDQ4e3
RM

OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+S0Yo26iBu3nw
3L

65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNg0wqu1gRuxcq7T0jt41UdqIK08vN7
/A

aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jd1/t08EXs79S5IKPcgAx87sTTi7KD
N5

SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiI1NS9foWeLyVkBgcQ
6S

me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRo0JafCaVfGI+175q/eWcqTX4q
+I

G4tKls4sL4mg0JLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVdQy2ttFdsq9
iK

TncCggEBAMmt/8yvPf1S+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03VmqsC9
nY

56s9w70U08perBX1JYmKZQh04293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIx
a2

gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCPjcQoaKvymAoCmAs8V2o
Mr

Ziw1YQ9u0UoWw0qm1wZqmVc0XvPIS2gWAs3fQ1WjH9hkcQTMsUaXQD0D0aqsY
3E

Nq0vbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F50TSEiGhhHesgR
rs
fBrpEY1IATtPq1taBZZogRqI3r0kkPk=
-----END PRIVATE KEY-----
certificate: |
-----BEGIN CERTIFICATE-----

MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIB3DQEBCwUAMIGGMQswCQ
YD

VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZD
Eb

MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDA1sb2NhbGhvc3
Qx

HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNj
A3

WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRX
Jl

d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2
V0

cy10ZXN0MRIwEAYDVQQDDA1sb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbm
VA

aW52YWxpZC5vcmcwggIiMA0GCSqGSIB3DQEBAQUAA4ICDwAwggIKAoICAQCjYt
uW

aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpo
R8

Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nT
ek

LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tIt
nH

KT/m6DSU0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+
m6

jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8bu
WQ

Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCnrpNjgJmIK3+mKk4kXX7ZTwefo
Az

TK412pHNuC53QVc/EF++GBLAXmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCfI

BK

Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfq
nV0

nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtg
zo

GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC
9p

sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFg
QU

9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAX
ar

jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAA0CAgEANjIBMr
ow

YNCbhAJdP7dh1hT2RUFRdeRUJD0IxrH/hkvb6myHHnK8n0YezFPjUlmRKUGNED
uA

xbnXZzPdCRNV9V2mShbXvCyidY7WCQE2Bn44z2600uWVw+7DNNLH9BnkWUtOnM
9P

wtmD9phWexm4q2GnTsiL6U16cy0Q1TJWKVLEUQ6yda582e23J1AXqtqFcpfoE
34

H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVc
jv

xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2c
Kk

ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqw0u/VdZv
2g

1WpTHlum1ClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148Q
YA

A01bU0JQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6
Gg

mnIZLRerQCu10ozr87r0QqQakPkyt8BUSNK3K42j2qcfhA0NdR18Hq8Qs5pupy
+s

8sdCGDlwR3JNCMv6u480K87F4mcIxhkSeFFJUFII25pCGN5WtE4p5l+9cn01Gr
IX

e2H1/7M0c/1bZ4FvXgARlex2rkgS0Ka06HE=
-----END CERTIFICATE-----

MetaDefender Distributed Cluster Identity Service

Ignition file

Info

The ignition file is required only for a clean installation.

The following fields are essential for the ignition file:

- `secure.connection_key`
- `secure.private_key`
- `secure.certificate`
- `database.host`
- `database.port`
- `database.user`
- `database.password`

To install MetaDefender Distributed Cluster (MDDC) Identity Service server, ignition file in YAML format is required at the following location:

- Windows: `C:\opswat\mddc_identity_service.yml`
- Linux: `/etc/opswat/mddc_identity_service.yml`

The ignition file includes fields:

Key path	Value type	Accepted values	Required	Description
secur e.con necti on_ke y	String	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	Required	An arbitrary string that enables clients to connect to the server. Use this value for the key <code>identity.connection_key</code> in configuration file of MetaDefender Distributed Cluster Control Center.
secur e.pri vate_ key	String		Required	Content of private key in X509 format.
secur e.cer tific ate	String		Required	Content of certificate in X509 format.
datab ase.h ost	String		Required	IP address / domain name of the server where PostgreSQL server locates.
datab ase.p ort	Number		Required	Port of PostgreSQL server is listening for connections from clients.
datab ase.u ser	String		Required	PostgreSQL server's user. SUPERUSER privilege is required to setup the server's database and extensions for the first time.
datab ase.p asswo rd	String		Required	PostgreSQL server's user credentials.

Key path	Value type	Accepted values	Required	Description
rest .host	String		Optional	IP address (V4/V6) or host where the server resides on. Default value is <code>''</code>. Notes: value <code>'*'</code> allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to <code>0.0.0.0</code>
rest .port	Number		Optional	The port where the server resides on. Default value is 8891
log.s tream s[@]. log_t ype	String	- file - syslog	Optional	Type of log device.
log.s tream s[@]. log_l evel	String	- dump - debug - info - warning - error	Optional	Level of log message.

Key path	Value type	Accepted values	Required	Description
<code>log.streams[@].log_path</code>	String	If <code>log.streams[@].log_type</code> is "file" then <code>log.streams[@].log_path</code> is the path to a file on file system where logs are written. If <code>log.streams[@].log_type</code> is "syslog" then <code>log.streams[@].log_path</code> can be <code>[tcp/udp]://host:port</code> where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. <code>log.streams[@].log_path</code> can be "local" to write log to local syslog server (Linux only).	Optional	Location where logs are written.
<code>user.name</code>	String		Optional	User name for the initial administrator user account.
<code>user.password</code>	String		Optional	Password for the initial administrator user account.
<code>user.email</code>	String	Basic email format, a string starts with non whitespace/non @ characters, contains one @ symbol, and ends with non whitespace/non @ characters.	Optional	E-mail address for the initial administrator user account.

Key path	Value type	Accepted values	Required	Description
user. apikey	String	string of exactly 36 characters composed of uppercase and lowercase letters [A-Z, a-z] and digits [0-9]	Optional	API key for the initial administrator user account.

Configuration file

After successfully installing, MDDC Identity Service generates a configuration file with changeable settings at the following location

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Identity Service\mddc_identity_service.yml`
- Linux: `/etc/mddc-identity-service/mddc_identity_service.yml`

Info

The service must be restarted to take the new configurations into effect.

Sample

Warning

`database.host`, `database.port`, `database.user`, and `database.password` should be updated with the appropriate values of your Postgres host/IP, port, username, and password.

Info

OpenSSL or a similar tool (e.g., `ssh-keygen`) can create a pair of public and private keys in X.509 format.

yaml

```

database:
  host: "your_postgres_host"
  port: 5432
  user: "your_postgres_username"
  password: "your_postgres_admin_password"
secure:
  connection_key: "1234abcd"      # [0-9a-zA-Z]{4,64}
  certificate: |
    -----BEGIN CERTIFICATE-----

MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIB3DQEBCwUAMIGGMQswCQ
YD

VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZD
Eb

MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDA1sb2NhbgHvc3
Qx

HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNj
A3

WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRX
Jl

d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2
V0

cy10ZXN0MRIwEAYDVQQDDA1sb2NhbgHvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbm
VA

aW52YWxpZC5vcmcwggIiMA0GCSqGSIB3DQEBAQUAA4ICDwAwggIKAoICAQCjYt
uW

aICCY0tJPubxpIgIL+WWmz/fmK8IQR11Wtee6/IUyUlo5I602mq1qcLhT/kmpo
R8

Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nT
ek

LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tIt
nH

KT/m6DSU0xLZ13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+
m6

jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8bu
WQ

```

Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefo
Az

TK412pHNuC53QVc/EF++GBLAXmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCFI
BK

Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf21on4fEZr3+vRuc9shfq
V0

nPN1IMSnzXCast7I2fiuRXdIz96Kj1GQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtg
zo

GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC
9p

sNcjTMaBQLN03T1Qe1bs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFg
QU

9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAX
ar

jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAA0CAgEANjIBMr
ow

YNCbhAJdP7dh1hT2RUFrdeRUJD0IxrH/hkvb6myHHnK8n0YezFPjUlmRKUgNED
uA

xbnXZzPdCRNV9V2mShbXvCyidY7WCQE2Bn44z2600uWVvk+7DNNLH9BnkWUtOnM
9P

wtmD9phWexm4q2GnTsiL6U16cy0Q1TJWKVLEUQQ6yda582e23J1AXqtqFcpfoE
34

H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTY1kpTQfLTZSsgPdEHYVc
jv

xQ2D+XyDR0aRLR1vxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2c
Kk

ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqw0u/VdZv
2g

1WpTHlum1ClZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148Q
YA

A01bU0JQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6
Gg

mnIZLRerQCu10ozr87r0QqQakPkyt8BUSNK3K42j2qcfa0NdR18Hq8Qs5pupy
+s

8sdCGDlwR3JNCMv6u480K87F4mcIxhkSefFJUFI25pCGN5WtE4p5l+9cn01Gr
IX

e2Hl/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=

-----END CERTIFICATE-----

private_key: |

-----BEGIN PRIVATE KEY-----

MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0
tJ

PubxpIgIL+WWmz/fmK8IQR11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAm
HK

nSWDPWtn1BtXLERLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5
ZZ

toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6D
SU

0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyM
BT

J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8
pS

Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2p
HN

uC53QVc/EF++GBLAxmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCFIBKIzv9cg
i9

fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqvV0nPN1IM
Sn

zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtgzoGMTvP/
Au

ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTM
aB

QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABaoICAFWe8MQZb37k2gdAV3Y6aq
8f

qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPio
H+

vXUpW0jnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0U0iSMMftp
k9

fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht5A

Kqx9mld20v17NjgiC1Fx9r+fZw/i0abFFwQA4dr+R8mEMK/7bd4VXfQ1o/QGGBMT

G+u1FrSiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBt0DU372a2CQK/u4Q/

HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKTlQ8

YZkpIcLNVLw0usoGYHFm2rvCyEVlfsE3Ub8cFyTFk50Se0cF2QL2xzKmmBZEpXgl

xBHR0hjgon0IKJDGfor4bH07Nt+1Ece8u2oTEKvpz5aIn440eC5mApRGy83/0bvs

esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRI1G56ErxC3l0s11wrIpTmXXckqw

zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BXlvp2GTBIeoxrWB4V3YArwu8QF80QBgVz

mgHl24nTg00UH10jZsABAOIBAQD0xftSdbSqGytcWqPYP3SZHAWDA004ACEM+eCw

au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN77

40Qdyk1lTzTVR0qmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwfS5

YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBSHnK8xrGaMErn7W0qTMbFH

PgQz5FcXrjCCqwHi1wNBELDTp/ZECEB7y34khVh531mBE2mNzSVIQCgZP1I/DvXj

W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFwwuR

naVJS6Wx0KjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+tr6

2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBL0mbSUhN3PG2m

39I802u0fFNVQCJKhx1m1MFFL0u7lVcDS9JN+oYVPb6MDfBLm5j0iPuYkFZ4gH

79

J7gXI0/YKhaJ7yXthYVkdRsf6Eooer4RZgma62Dd1VNzSq3JBo6rYjF7Lvd+Rw
DC

R1thHrmf/IXplxpNVkoMVxtzbrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZ
Mp

Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItN11IZZMW2TPUiY0rl6ya
Ch

BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxJzjaVtxN7ZXwZv0VrY7I7fPWYn
CE

fXCr4+IVpZI/ZHZWpGX6CGSgT6E0jCZ5IUufIvEpqVSmtF8MqfX09o9uIYLokr
WQ

x1dB15UnuTLDqw8bChq705y6yfuWa0WvL7nxI8NvSsfj4y635gIa/0dFeBYZEf
HI

U1GdNVomwXwYEzge/c19ruIowX7HU/NgxMWTMZhpaZlxgesXybel+YNcfDQ4e3
RM

OMz3ZFiaMaJsGGnf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+S0Yo26iBu3nw
3L

65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNg0wqu1gRuxcq7T0jt41UdqIK08vN7
/A

aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jd1/t08EXs79S5IKPcgAx87sTTi7KD
N5

SYt4tr2uPEe53NTXuSatilG5QCyExIEL0uzWAMKzg7CAiI1NS9foWeLyVkBgcQ
6S

me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRo0JafCaVfGI+175q/eWcqTX4q
+I

G4tK1s4sL4mg0JLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVdQy2ttFdsq9
iK

TncCggEBAMmt/8yvPf1S+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03Vmq5C9
nY

56s9w70U08perBX1JYmKZQh042931vxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIx
a2

gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVcPjcQoaKvymAoCmAs8V2o
Mr

Ziw1YQ9u0UoWw0qm1wZqmVc0XvPIS2gWAs3fQ1WjH9hkcQTMsUaXQD0D0aqkSY
3E

Nq0vbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F50TSEiGhhHesgR
rs

fBrpEY1IATtPq1taBZZogRqI3r0kkPk=

-----END PRIVATE KEY-----

MetaDefender Distributed Cluster Worker

Ignition file

Info

The ignition file is required only for a clean installation.

The following fields are essential for the ignition file:

- `secure.connection_key`
- `secure.private_key`
- `secure.certificate`

To install MetaDefender Distributed Cluster (MDDC) Worker server, ignition file in YML format is required at the following location:

- Windows: `C:\opswat\mddc_worker.yml`
- Linux: `/etc/opswat/mddc_worker.yml`

The ignition file includes fields:

Key path	Value type	Accepted values	Required	Description
security.connection_key	String	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	Required	An arbitrary string that enables clients to connect to the server. Use this value as input when adding a MDDC Worker in the UI of the MDDC Control Center.
security.private_key	String		Required	Content of private key in X509 format.
security.certificate	String		Required	Content of certificate in X509 format.
rest.host	String		Optional	IP address [V4/V6] or host where the server resides on. Default value is <code>*.*.*.*</code> Notes: value <code>*.*.*.*</code> allows the service to accept connections from all network interfaces. To bind the service to a specific interface, specify its IP address or domain name. For example, to listen on all IPv4 interfaces, set the host to <code>0.0.0.0</code>

Key path	Value type	Accepted values	Required	Description
res.t.port	Number	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	Optional	The port where the server resides on. Default value is 8893
log.streams[@].log_type	String	- file - syslog	Optional	Type of log device.
log.streams[@].log_level	String	- dump - debug - info - warning - error	Optional	Level of log message.
log.streams[@].log_path	String	If log.streams[@].log_type is "file" then log.streams[@].log_path is the path to a file on file system where logs are written. If log.streams[@].log_type is "syslog" then - log.streams[@].log_path can be [tcp/udp]://host:port where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. - log.streams[@].log_path can be "local" to write log to local syslog server (Linux only).	Optional	Location where logs are written.

Configuration file

After successfully installing, MDDC Worker generates a configuration file with changeable settings at the following location::

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Worker\mddc_worker.yml`
- Linux: `/etc/mddc-worker/mddc_worker.yml`

 Info

The service must be restarted to take the new configurations into effect.

Sample

 Info

OpenSSL or a similar tool (e.g., ssh-keygen) can create a pair of public and private keys in X509 format.

yaml

```
secure:
  connection_key: 1234abcd
  private_key: |
    -----BEGIN PRIVATE KEY-----

MIIJQwIBADANBgqhkiG9w0BAQEFAASCCS0wggkpAgEAAoICAQCjYtuWaICCY0
tJ

PubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAm
HK

nSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5
ZZ

toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItnHKT/m6D
SU

0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyM
BT

J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8
pS

Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2p
HN

uC53QVc/EF++GBLAxmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCFIBKIzv9cg
i9

fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqvV0nPN1IM
Sn

zXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtgzoGMTvP/
Au

ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTM
aB

QLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABaoICAFWe8MQZb37k2gdAV3Y6aq
8f

qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPIo
H+

vXUpW0jnCmM4muFA9/cX6mYMc8TmJsg0ewLdBCOZVw+wPABlaqz+0U0iSMMftp
k9

fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht
5A
```

Kqx9mld20v17NjgiC1Fx9r+fZw/i0abFFwQA4dr+R8mEMK/7bd4VxfQ1o/QGGB
MT

G+u1FrSiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBt0DU372a2CQK/u4
Q/

HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKT1
Q8

YZkpIcLNVLw0usoGYHFm2rvCyEV1fsE3Ub8cFyTFk50Se0cF2QL2xzKmbZEpX
gl

xBHR0hjgon0IKJDGfor4bH07Nt+1Ece8u2oTEKvpz5aIn440eC5mApRGy83/0b
vs

esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRI1G56ErxC3l0s11wrIpTmXXck
qw

zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BX1vp2GTBIeoxrWB4V3YArwu8QF80QBg
Vz

mgHl24nTg00UH10jZsABAOIBAQD0xftSdbSqGytcWqPYP3SZHAWDA004ACEM+e
Cw

au9ASutl0IDlNDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN
77

40Qdyk1lTzTVR0qmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwF
s5

YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBSHnK8xrGaME7W0qTMb
FH

PgQz5FcXrjCCqwhilwNBELDTp/ZECEB7y34khVh531mBE2mNzSVIQCZP1I/Dv
Xj

W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFww
uR

naVJS6Wx0KjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+t
r6

2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBL0mbSUHn3PG
2m

39I802u0fFNVQCJKhx1m1MFFL0u7lVcDS9JN+oYVPb6MDfBLm5j0iPuYkFZ4gH
79

J7gXI0/YKhaJ7yXthYVkdRsf6Eooer4RZgma62Dd1VNzSq3JB06rYjF7Lvd+Rw

DC

R1thHrmf/IXp1xpNVkoMVxtzbrrbgnC25QmvRYc0r1S/kvM4yQhMH3eA7IycDZ
Mp

Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItN11IZZMW2TPUiY0rl6ya
Ch

BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxJzjaVtxN7ZXwZv0VrY7I7fPWYn
CE

fXCr4+IVpZI/ZHZWpGX6CGSgT6E0jCZ5IUufIvEpqVSmtF8MqfX09o9uIYLokr
WQ

x1dB15UnuTLDqw8bChq705y6yfuWaOWvL7nxI8NvSsfj4y635gIa/0dFeBYZEf
HI

U1GdNVomwXwYEzge/c19ruIowX7HU/NgxMWTMZhpaZlxgesXybel+YNcfDQ4e3
RM

OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+S0Yo26iBu3nw
3L

65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNg0wqu1gRuxcq7T0jt41UdqIK08vN7
/A

aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jd1/t08EXs79S5IKPcgAx87sTTi7KD
N5

SYt4tr2uPEe53NTXuSatilG5QCyExIELOuzWAMKzg7CAiI1NS9foWeLyVkBgcQ
6S

me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRo0JafCaVfGI+175q/eWcqTX4q
+I

G4tKls4sL4mg0JLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVdQy2ttFdsq9
iK

TncCggEBAMmt/8yvPf1S+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03VmqsC9
nY

56s9w70U08perBX1JYmKZQh04293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIx
a2

gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCPjcQoaKvymAoCmAs8V2o
Mr

Ziw1YQ9u0UoWw0qm1wZqmVc0XvPIS2gWAs3fQ1WjH9hkcQTMsUaXQD0D0aqsY
3E

Nq0vbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F50TSEiGhhHesgR
rs
fBrpEY1IATtPq1taBZZogRqI3r0kkPk=
-----END PRIVATE KEY-----
certificate: |
-----BEGIN CERTIFICATE-----

MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQ
YD

VQQGEwJHQjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZD
Eb

MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDA1sb2NhbGhvc3
Qx

HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNj
A3

WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHQjEQMA4GA1UECAwHRX
Jl

d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2
V0

cy10ZXN0MRIwEAYDVQQDDA1sb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbm
VA

aW52YWxpZC5vcmcwggiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYt
uW

aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpo
R8

Di3DAmHKnSWdPWtn1BtXLErLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nT
ek

LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tIt
nH

KT/m6DSU0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+
m6

jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8bu
WQ

Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCnrpNjgJmIK3+mKk4kXX7ZTwefo
Az

TK412pHNuC53QVc/EF++GBLAXmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCfI

BK

Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfq
nV0

nPN1IMSnzXCast7I2fiuRXdIz96KjlGQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtg
zo

GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC
9p

sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFg
QU

9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAX
ar

jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAA0CAgEAnjIBMr
ow

YNCbhAJdP7dh1hT2RUFrdeRUJD0IxrH/hkvb6myHHnK8n0YezFPjUlmRKUgNED
uA

xbnXZzPdCRNV9V2mShbXvCyidY7WCQE2Bn44z2600uWVw+7DNNLH9BnkWUtOnM
9P

wtmD9phWexm4q2GnTsiL6U16cy0Q1TJWKVLEUQ6yda582e23J1AXqtqFcpfoE
34

H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTYlkpTQfLTZSsgPdEHYVc
jv

xQ2D+XyDR0aRLRlvxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2c
Kk

ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqw0u/VdZv
2g

1WpTHlum1C1ZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148Q
YA

A01bU0JQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6
Gg

mnIZLRerQCu10ozr87r0QqQakPkyt8BUSNK3K42j2qcfhA0NdR18Hq8Qs5pupy
+s

8sdCGDlwR3JNCMV6u480K87F4mcIxhkSeFFJUFII25pCGN5WtE4p5l+9cn01Gr
IX

e2H1/7M0c/1bZ4FvXgARlex2rkgS0Ka06HE=
-----END CERTIFICATE-----

MetaDefender Distributed Cluster Control Center

Ignition file

Info

The ignition file is required only for a clean installation.

The following fields are essential for the ignition file:

- `identity.host`
- `identity.port`
- `identity.connection_key`
- `database.host`
- `database.port`
- `database.user`
- `database.password`
- `secure.encryption_key`

To install MetaDefender Distributed Cluster Control (MDDC) Center server, ignition file in YAML format is required at the following location:

- Windows: `C:\opswat\mdc_control_center.yml`
- Linux: `/etc/opswat/mdc_control_center.yml`

The ignition file includes fields:

Key path	Value type	Accepted values	Required	Description
identity.host	String		Required	IP address of the server where MDDC Identity Service server locates.
identity.port	String		Required	Port of MDDC Identity Service server is listening for connections from clients.
identity.connection_key	String	A string from 4 to 64 character long containing digits from 0 to 9 and characters from a/A to z/Z	Required	The access key required to connect to the MDDC Identity Service server, ensuring it matches the value used by the server.
database.host	String		Required	IP address / domain name of the server where PostgreSQL server locates.
database.port	Number		Required	Port of PostgreSQL server is listening for connections from clients.
database.user	String		Required	PostgreSQL server's user. SUPERUSER privilege is required to setup the server's database and extensions for the first time.
database.password	String		Required	PostgreSQL server's user credentials.
secure.encryption_key	String	A 32-character plain text composed of characters 'a'-'z' and digits '0'-'9'.	Required	The encryption key is used to encrypt the sensitive data in the database.
rest.port	Number		Optional	The port where the server resides on. Default value is 8892
rest.logging_path	String		Optional	Location where logs are written.

Key path	Value type	Accepted values	Required	Description
rest.logs_level	String	- dump - debug - info - warning - error	Optional	Level of log message.
log.streams[@].log_type	String	- file - syslog	Optional	Type of log device.
log.streams[@].log_level	String	- dump - debug - info - warning - error	Optional	Level of log message.

Key path	Value type	Accepted values	Required	Description
<code>log.streams[@].log_path</code>	String	If <code>log.streams[@].log_type</code> is "file" then <code>log.streams[@].log_path</code> is the path to a file on file system where logs are written. If <code>log.streams[@].log_type</code> is "syslog" then <code>log.streams[@].log_path</code> can be <code>[tcp/udp]://host:port</code> where host:port is the host and port to a remote syslog server that supports TCP or UDP protocol. <code>log.streams[@].log_path</code> can be "local" to write log to local syslog server (Linux only).	Optional	Location where logs are written.

Warning

Avoid using the loopback IP address (such as `localhost` or `127.0.0.1`) for key `identity.host`.

It may prevent MDDC **API Gateway** from successfully establishing a connection to MDDC **Identity Service**.

Configuration file

After successfully installing, MDDC Control Center generates a configuration file with changeable settings at the following location:

- Windows: `C:\Program Files\OPSWAT\MetaDefender Distributed Cluster Control Center\mddc_control_center.yml`
- Linux: `/etc/mddc-control-center/mddc_control_center.yml`

i Info

The service must be restarted to take the new configurations into effect.

Sample

i Warning

`database.host`, `database.port`, `database.user`, and `database.password` should be updated with the appropriate values of your Postgres host/IP, port, username, and password.

`identity.host` should be updated with the appropriate host or IP of your MDDC Identity Service.

yaml

```
database:
  host: "your_postgres_host_ip"
  port: 5432
  user: "your_postgres_username"
  password: "your_postgres_admin_password"
identity:
  host: "your_mddc_identity_service_host_ip"
  port: 8891
  connection_key: "1234abcd"
secure:
  encryption_key: "12345678123456781234567812345678" # [a-z0-9]{32}
```

Container-Based Setup

Prerequisite

Before running the setup, please check ([System Requirements](#)) to install all required dependencies of MetaDefender Distributed Cluster (MDDC).

Setup order requirement

Please follow the installation order to complete the system setup properly.

Order	Service	Notes
1	Redis, RabbitMQ, PostgreSQL and MDDC Identity Service	• Could be setup in parallel in any order among them. • Make sure they are all fully functional and accessible before proceeding to the next setup order #2.
2	MDDC Control Center	• Ensure it's able to connect to those services in #1 • Make sure it is fully functional and accessible.
3	MDDC File Storage	• Ensure they're able to connect to MDDC Control Center • Make sure it is fully functional and accessible.
4	MDDC Worker for MDDC API Gateway and MDDC Worker for MetaDefender Core	• Could be setup in parallel in any order among them. • Ensure they're able to connect to MDDC Control Center • Make sure they are all fully functional and accessible.

Image name and version

All the images can be found at OPSWAT Docker Hub with the following information:

Info

version is the currently release version.

MDDC Identity Service

Docker image bash

```
opswat/metadefender-distributed-cluster:identity-service-  
<version>-debian-12
```

MDDC File Storage

Docker image bash

```
opswat/metadefender-distributed-cluster:file-storage-  
<version>-debian-12
```

MDDC Control Center

Docker image bash

```
opswat/metadefender-distributed-cluster:control-center-  
<version>-debian-12
```

MDDC Worker for MDDC API Gateway

Docker image bash

```
opswat/metadefender-distributed-cluster:worker-api-gateway-  
<version>-debian-12
```

MDDC Worker for MetaDefender Core

Docker image bash

```
opswat/metadefender-distributed-cluster:worker-core-<version>-  
debian-12
```

Environment variables

1. MDDC Identity Service

Environment Variable	Necessity	Description
MDDC_IDENTITY_SERVICE_DB_HOST	Required	Provide the database host for MDDC Identity Service
MDDC_IDENTITY_SERVICE_DB_PORT	Optional	Provide the database port for MDDC Identity Service Default: 5432
MDDC_IDENTITY_SERVICE_DB_USER	Required	Provide the database user for MDDC Identity Service
MDDC_IDENTITY_SERVICE_DB_PASSWORD	Required	Provide the database password for MDDC Identity Service

Environment Variable	Necessity	Description
MDDC_USER	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_PASSWORD	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_EMAIL	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_APIKEY	Optional	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.
MDDC_IDENTITY_SERVICE_CONNECTION_KEY	Required	Define the connection key in order to register to Control Center. Must be 4 to 64 characters long, using only letters and digits [0-9, a-z, A-Z].

Environment Variable	Necessity	Description
MDDC_IDENTITY_SERVICE_PORT	Optional	Define the expose port for MDDC Identity Service Default: 8891
LOG_LEVEL	Optional	Define the log level. Default value: info Accepted values: info / debug / error / warning

Start MDDC Identity Service container with docker run:

bash

```
docker run -d --name mddc-identity-service \
    -e MDDC_IDENTITY_SERVICE_DB_HOST=<your_postgres_host> \
    -e MDDC_IDENTITY_SERVICE_DB_USER=<your_postgres_user> \
    -e MDDC_IDENTITY_SERVICE_DB_PASSWORD=
<your_postgres_password> \
    -e MDDC_IDENTITY_SERVICE_CONNECTION_KEY=
<your_connection_key> \
    -e MDDC_USER=<your_mddc_admin_user> \
    -e MDDC_PASSWORD=<your_mddc_admin_password> \
    -e MDDC_EMAIL=<your_mddc_admin_email> \
    -p 8891:8891 opswat/metadefender-distributed-
cluster:identity-service-<version>-debian-12
```

2. MDDC File Storage

Environment Variable	Necessity	Description
MDDC_FILE_STORAGE_CONNECTION_KEY	Required	Define the connection key in order to register to MDDC Control Center. Must be 4 to 64 characters long, using only letters and digits [0-9, a-z, A-Z].
MDDC_FILE_STORAGE_PORT	Optional	Define the expose port for MDDC File Storage. Default is 8890.
MDDC_FILE_STORAGE_HOST	Optional	Define the MDDC File Storage's host address. If it's not specified, it will get the container's internal IP address.
LOG_LEVEL	Optional	Define the log level. Default value: <code>info</code> . Accepted values: <code>info / debug / error / warning</code> .
MDDC_CONTROL_CENTER_HOST	Required	Provide the MDDC Control Center's host address.
MDDC_CONTROL_CENTER_PORT	Optional	Provide the MDDC Control Center's port. Default is 8892.

Environment Variable	Necessity	Description
MDDC_USER	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_PASSWORD	Required	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_APIKEY	Optional	Define the information to initiate the administrator account. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Info

Persistent storage is located at `/opt/opswat/mddc-file-storage`. If end-users require data to be retained across container lifecycles, they must mount a volume to this path with `777` permissions to ensure full read/write access for all processes.

Start MDDC File Storage container with docker run.

bash

```
docker run -d --name mddc-file-storage \
    -e MDDC_FILE_STORAGE_CONNECTION_KEY=
    <your_connection_key> \
    -e MDDC_FILE_STORAGE_PORT=8890 \
    -e MDDC_CONTROL_CENTER_HOST=<control-center_host_address> \
    -e MDDC_USER=<your_mddc_admin_user> \
    -e MDDC_PASSWORD=<your_mddc_admin_password> \
    -p 8890:8890 opswat/metadefender-distributed-cluster:file-
    storage-<version>-debian-12
```

3. MDDC Control Center

Environment Variable	Necessity	Description
MDDC_LAKE_DB_HOST	Optional	Provide the database host for Data Lake. In case that the end-user does not have the Data Lake, it's required to provide this variable to automate the database preparation.
MDDC_LAKE_DB_PORT	Optional	Provide the database port for Data Lake. Default is 5432.
MDDC_LAKE_DB_USER	Optional	Provide the database user for Data Lake. In case that the end-user does not have the Data Lake, it's required to provide this variable to automate the database preparation.
MDDC_LAKE_DB_PASSWORD	Optional	Provide the database password for Data Lake. In case that the end-user does not have the Data Lake, it's required to provide this variable to automate the database preparation.
MDDC_WAREHOUSE_DB_HOST	Optional	Provide the database host for Data Warehouse. In case that the end-user does not have the Data Warehouse, it's required to provide this variable to automate the database preparation.
MDDC_WAREHOUSE_DB_PORT	Optional	Provide the database port for Data Warehouse. Default is 5432.
MDDC_WAREHOUSE_DB_USER	Optional	Provide the database user for Data Warehouse. In case that the end-user does not have the Data Warehouse, it's required to provide this variable to automate the database preparation.

Environment Variable	Necessity	Description
MDDC_WAREHOUSE_DB_PASSWORD	Optional	Provide the database password for Data Warehouse. In case that the end-user does not have the Data Warehouse, it's required to provide this variable to automate the database preparation.
MDDC_CACHE_HOST	Optional	Provide the caching host (Redis).
MDDC_CACHE_PORT	Optional	Provide the caching port (Redis).
MDDC_CACHE_USER	Optional	Provide the caching username (Redis). If the end-user does not provide it, Redis will be added without authentication.
MDDC_CACHE_PASSWORD	Optional	Provide the caching password (Redis). If the end-user does not provide it, Redis will be added without authentication. Do not support double quotes (") and backslash (\) in the password.
MDDC_BROKER_HOST	Optional	Provide the broker host (RabbitMQ).
MDDC_BROKER_PORT	Optional	Provide the broker port (RabbitMQ).
MDDC_BROKER_USER	Optional	Provide the broker username (RabbitMQ).
MDDC_BROKER_PASSWORD	Optional	Provide the broker password (RabbitMQ).
MDDC_CONTROL_CENTER_DB_HOST	Required	Provide the database host for MDDC Control Center.
MDDC_CONTROL_CENTER_DB_PORT	Optional	Provide the database port for MDDC Control Center. Default is 5432.

Environment Variable	Necessity	Description
MDDC_CONTROL_CENTER_DB_USER	Required	Provide the database username for MDDC Control Center.
MDDC_CONTROL_CENTER_DB_PASSWORD	Required	Provide the database password for MDDC Control Center.
MDDC_USER	Required	Provide the administrator account that is defined in MDDC Identity Service. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_PASSWORD	Required	Provide the administrator account that is defined in MDDC Identity Service. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_APIKEY	Optional	Provide the administrator account that is defined in MDDC Identity Service. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.
MDDC_IDENTITY_SERVICE_HOST	Required	Provide the MDDC Identity Service host in order to add it to MDDC Control Center.
MDDC_IDENTITY_SERVICE_PORT	Optional	Provide the IMDDC identity Service port in order to add it to MDDC Control Center. Default is 8891.

Environment Variable	Necessity	Description
MDDC_IDENTITY_SERVICE_CONNECTION_KEY	Required	Provide the MDDC Identity Service connection key in order to add it to MDDC Control Center. Must be 4 to 64 characters long, using only letters and digits (0–9, a–z, A–Z).
MDDC_CONTROL_CENTER_ENCRYPTION_KEY	Required	Define the encryption key for communication between MDDC Control Center and the services. Must be 32 characters long and contain only lowercase letters (a–z) and digits (0–9).
MDDC_CERT_PATH	Optional	Provide the directory path that contains the certificate and private key in order to enable https Note: when provide this variable, it's supposed to mount this path to <code>/certs/</code> as volume For example: <code>--volume /your-path:/certs</code> Note: In cases where SSL fails to enable due to the File Storage service not being ready, the end-user can either restart the MDDC Control Center or manually activate SSL as a workaround.
LOG_LEVEL	Optional	Define the log level. Default value: <code>info</code>. Accepted values: <code>info</code> / <code>debug</code> / <code>error</code> / <code>warning</code> .

Start MDDC Control Center container with Docker run.

bash

```

docker run -d --name mddc-control-center \
    -e MDDC_CONTROL_CENTER_DB_HOST=<your_postgre_host> \
    -e MDDC_CONTROL_CENTER_DB_USER=<your_postgre_user> \
    -e MDDC_CONTROL_CENTER_DB_PASSWORD=<your_postgre_password> \
    -e MDDC_IDENTITY_SERVICE_HOST=
<your_identity_service_host_address> \
    -e MDDC_USER=<your_mddc_admin_user> \
    -e MDDC_PASSWORD=<your_mddc_admin_password> \
    -e MDDC_IDENTITY_SERVICE_CONNECTION_KEY=
<your_connection_key> \
    -e MDDC_CONTROL_CENTER_ENCRYPTION_KEY=<your_encryption_key>
\
    -e MDDC_CERT_PATH=/certs \
    -v /new-certificates:/certs \
    -p 8892:8892 opswat/metadefender-distributed-
cluster:control-center-<version>-debian-12

```

4. MDDC Worker for API Gateway

Environment Variable	Necessity	Description
MDDC_USER	Required	Provide the administrator account that is defined in MDDC Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_PASSWORD	Required	Provide the administrator account that is defined in MDDC Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_APIKEY	Optional	Provide the administrator account that is defined in MDDC Identity Service. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.
MDDC_API_GATEWAY_PORT	Optional	Define the expose port to scan files via MDDC API Gateway. Default is 8899.
LOG_LEVEL	Optional	Define the log level. Default value: <code>info</code>. Accepted values: <code>info / debug / error / warning</code>.

Info

If multiple MDDC API Gateway containers are deployed on the same host, make sure their ports are configured to avoid conflicts.

Start MDDC Worker for MDDC API Gateway container with Docker run.

bash

```
docker run -d --name mddc-worker-api-gateway \
    -e MDDC_WORKER_CONNECTION_KEY=<your_connection_key> \
    -e MDDC_WORKER_HOST=<your_worker_host_address> \
    -e MDDC_CONTROL_CENTER_HOST=
<your_control_center_host_address> \
    -e MDDC_USER=<your_mddc_admin_user> \
    -e MDDC_PASSWORD=<your_mddc_admin_password> \
    -e MDDC_API_GATEWAY_PORT=8899 \
    -p 8893:8893 -p 8899:8899 opswat/metadefender-distributed-
cluster:worker-api-gateway-<version>-debian-12
```

5. MDDC Worker for Core

Environment Variable	Necessity	Description
MDDC_WORKER_CONNECTION_KEY	Required	Define the connection key in order to register to MDDC Control Center. Must be 4 to 64 characters long, using only letters and digits [0–9, a–z, A–Z].
MDDC_WORKER_PORT	Optional	Define the expose worker's port. Default is 8893.
MDDC_WORKER_HOST	Optional	Define the worker's host address. If it's not specified, it will get the container's internal IP address.
MDDC_CONTROL_CENTER_HOST	Required	Provide the MDDC Control Center's host address.
MDDC_CONTROL_CENTER_PORT	Optional	Provide the MDDC Control Center's port. Default is 8892.

Environment Variable	Necessity	Description
MDDC_USER	Required	Provide the administrator account that is defined in MDDC Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_PASSWORD	Required	Provide the administrator account that is defined in MDDC Identity Service. It can be optional if the end-user provides the MDDC_APIKEY. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.

Environment Variable	Necessity	Description
MDDC_APIKEY	Optional	Provide the administrator account that is defined in MDDC Identity Service. This account is to automatically do the following tasks: • Add Redis to MDDC Control Center if specified. • Add RabbitMQ to MDDC Control Center if specified. • Add Data Lake to MDDC Control Center if specified. • Add Data Warehouse to MDDC Control Center if specified. • Add MDDC File Storage to MDDC Control Center if specified. • Add MDDC Worker to MDDC Control Center, upload MDDC API Gateway installer to MDDC Control Center, and deploy MDDC API Gateway to MDDC Worker. • Add MDDC Worker to MDDC Control Center, upload MetaDefender Core installer to MDDC Control Center, and deploy MetaDefender Core to MDDC Worker.
LOG_LEVEL	Optional	Define the log level. Default value: <code>info</code>. Accepted values: <code>info / debug / error / warning</code>.
MDDC_LICENSE_KEY	Optional	Provide the license key to activate MetaDefender Core.
MDDC_LICENSE_DESCRIPTION	Optional	Define description of the license key.

Info

If multiple MetaDefender Core containers are deployed on the same host, make sure their ports and hosts are configured to avoid conflicts.

Start MDDC Worker for MetaDefender Core container with Docker run.

bash

```
docker run -d --name mddc-worker-core \
    -e MDDC_WORKER_CONNECTION_KEY=<your_connection_key> \
    -e MDDC_WORKER_HOST=<your_core_host_address> \
    -e MDDC_CONTROL_CENTER_HOST=
<your_control_center_host_address> \
    -e MDDC_USER=<your_mddc_admin_user> \
    -e MDDC_PASSWORD=>your_mddc_admin_password> \
    -p 8893:8893 opswat/metadefender-distributed-cluster:worker-
core-<version>-debian-12
```

Start MetaDefender Distributed Cluster with Docker Compose

1. Create a local file named `docker-compose.yaml` and copy the following content to this file:

yaml yaml

```

services:
  redis:
    image: redis:7.0.5
    container_name: redis
    ports:
      - "6379:6379"
    networks:
      - mddc
  rabbitmq:
    image: rabbitmq:3.13.0
    container_name: rabbitmq
    restart: always
    healthcheck:
      test: ["CMD", "rabbitmq-diagnostics", "-q", "ping"]
      interval: 10s
      timeout: 10s
      retries: 30
      start_period: 20s
    environment:
      - RABBITMQ_DEFAULT_USER=admin
      - RABBITMQ_DEFAULT_PASS=admin
    ports:
      - "5672:5672"
      - "15672:15672"
    networks:
      - mddc
  postgres:
    image: postgres:14.17
    container_name: postgres
    ports:
      - 5432:5432
    networks:
      - mddc
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=admin
    healthcheck:
      test: [ "CMD", "pg_isready", "-U", "admin", "-d",
"postgres" ]
      interval: 10s
      timeout: 10s
      retries: 30
  identity-service:
    env_file:
      - .env.example
    image: opswat/metadefender-distributed-cluster:identity-
service-2.0.0-debian-12
    container_name: identity-service
    ports:

```

```

    - 8891:8891
networks:
  - mddc
deploy:
  restart_policy:
    condition: on-failure
depends_on:
  postgres:
    condition: service_healthy
    restart: true
file-storage:
  env_file:
    - .env.example
  image: opswat/metadefender-distributed-cluster:file-
storage-2.0.0-debian-12
  container_name: file-storage
  ports:
    - 8890:8890
  networks:
    - mddc
  deploy:
    restart_policy:
      condition: on-failure
  depends_on:
    postgres:
      condition: service_healthy
      restart: true
control-center:
  env_file:
    - .env.example
  image: opswat/metadefender-distributed-cluster:control-
center-2.0.0-debian-12
  container_name: control-center
  ports:
    - 8892:8892
  networks:
    - mddc
  deploy:
    restart_policy:
      condition: on-failure
  healthcheck:
    test: ["CMD", "true"]
    interval: 60s
    start_period: 90s
    start_interval: 60s
  depends_on:
    - identity-service
    - redis
    - rabbitmq
    - file-storage

```

```

worker-api-gateway:
  env_file:
    - .env.example
  image: opswat/metadefender-distributed-cluster:worker-api-
gateway-2.0.0-debian-12
  container_name: worker-api-gateway
  ports:
    - "8893"
    - 7777:7777
  networks:
    - mddc
  deploy:
    restart_policy:
      condition: on-failure
  healthcheck:
    test: ["CMD", "true"]
    interval: 5s
    timeout: 2s
    start_period: 30s
  depends_on:
    control-center:
      condition: service_healthy
worker-core:
  env_file:
    - .env.example
  image: opswat/metadefender-distributed-cluster:worker-
core-2.0.0-debian-12
  container_name: worker-core
  ports:
    - "8008"
  networks:
    - mddc
  deploy:
    restart_policy:
      condition: on-failure
  depends_on:
    control-center:
      condition: service_healthy
    worker-api-gateway:
      condition: service_healthy

networks:
  mddc:
    driver: bridge
    ipam:
      config:
        - subnet: 10.0.0.0/24
          gateway: 10.0.0.1

```

```
##Ensure to replace with your specific image tag
```

2. Prepare an environment variable file named `.env.example` and provide with your own values
3. Run the application with the command:

yaml

```
docker compose up -d
```

Known limitation

- When the host experiences resource limitations or degraded performance, some containers may fail to start properly. In such cases, restarting the container is recommended to restore normal operation.

Recommended Setup

Although it is possible to install Redis Caching Server, RabbitMQ Message Broker and Postgres Database Server and MetaDefender Distributed Cluster (MDDC) File Storage on the same machine, they should be installed separately on various machines to optimize their performance.

Redis Caching Server

The caching server consumes a large amount of memory while operating; hence, a machine with ample and high-speed memory is best suited to this component.

RabbitMQ Message Broker

The broker is one of the keys that powers MetaDefender Distributed Cluster architecture, it ensures tasks are delivered to MetaDefender Core instances in an equitable manner, delivering a "broken" task to a healthy MetaDefender Core instance and spreading tasks to new instances if more MetaDefender Core instances are added to system. For that reason, the broker should be hosted on a separate machine.

Postgres Database Server

MetaDefender Distributed Cluster database is split into three main clusters.

Data Lake stores scan results and other details related to requests such as `data_id`, hashes, etc.

Since Data Lake is shared among MetaDefender **Core** and MDDC **API Gateway** instances, it should be hosted on a large-volume and high-speed disk. The network is also essential to Data Lake; a high-speed network is necessary.

Data Warehouse, which prepares materials for building executive reports, uses a single connection to Data Lake and collects data periodically.

Since executive reports may be stored for a long period of time for MDDC Control Center to access, Data warehouse should be hosted on a large-volume machine.

Control Center-related database, storing all user details, configurations and other settings. MDDC Control Center database can be hosted on the same machine as Control Center.

MetaDefender Distributed Cluster File Storage

MDDC **File Storage** is shared among **MetaDefender Core** and MDDC **API Gateway** instances. The server consumes a large amount of disk to store the submitted files from instances. Since all file-related traffic goes through MDDC **File Storage**, a high speed network is essential. Rocky 9.0 is recommended to host MDDC **File Storage**.

MetaDefender Distributed Cluster API Gateway

Due to differences in Operating System and Nginx support on Windows and Linux, MDDC **API Gateway** should be hosted on a Linux machine running Rocky 9.0 for high throughput of file scan submissions.

MetaDefender Core

One of strong aspects of MetaDefender Distributed Cluster is that it can support a hybrid architecture in which MDDC **API Gateway** and MDDC **File Storage** instances may be hosted on Linux while **MetaDefender Core** instances can be on Windows. Therefore, based on customer requirements, **MetaDefender Core** instances can be hosted on Windows or Linux machines.

Warning

It is recommended to setup all MetaDefender Core instances on Windows or on Linux.

Mixed OS run is unsupported.

License activation

MetaDefender Distributed Cluster supports two types of license activations:

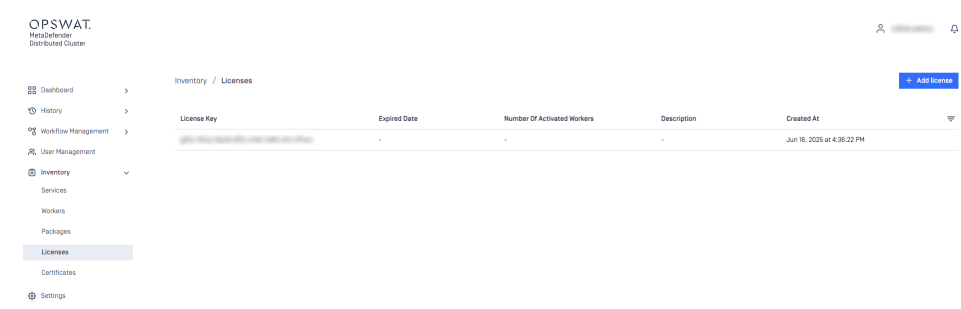
- Online Activation.
- Offline Activation.

Online Activation

MetaDefender Distributed Cluster (MDDC) supports seamless license activation for every deployed **MetaDefender Core** instance. The license key must be provided to the MDDC **Control Center** and will be activated on each individual **MetaDefender Core** instance either automatically during deployment or manually at a later time. If necessary, multiple license keys may also be supplied.

Adding License

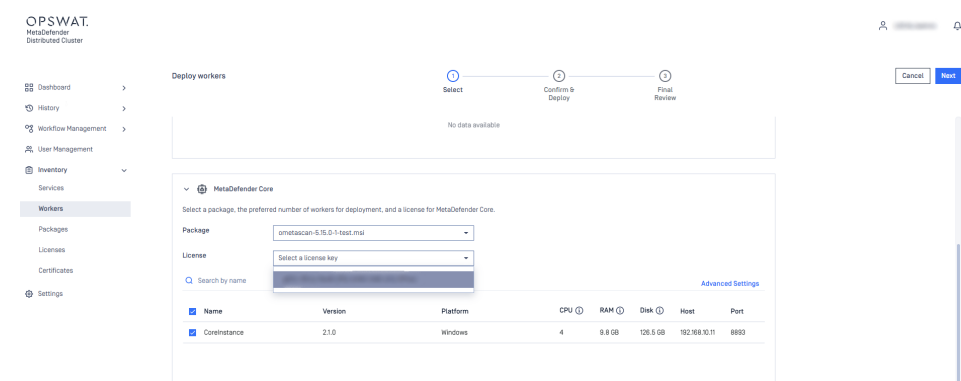
- 1. Sign in to MetaDefender Distributed Cluster **Control Center** console.
- 2. Go to **Inventory > Licenses** and select **Add license**.
- 3. Input your license key and click **Add**.



License Activation

Automatic activation during deployment

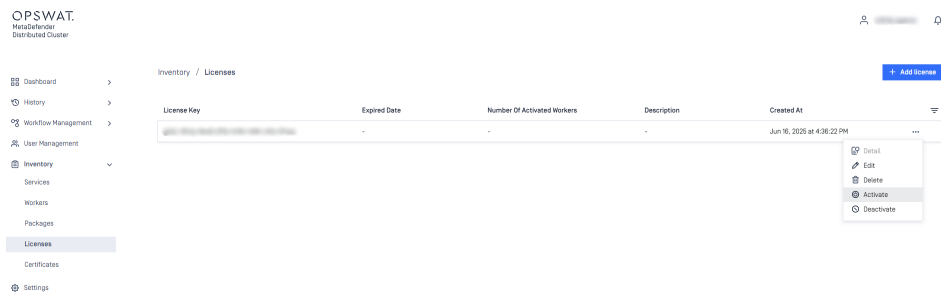
During the deployment process of **MetaDefender Core** instances, you can select a license key to automatically activate on the instances being deployed.



Activation after deployment

After deploying **MetaDefender Core** instances, follow these steps to activate your license:

1. Sign in to MDDC **Control Center** console.
2. From the left side bar, go to **Inventory** > **Licenses**.
3. From the list of available licenses, choose the key you wish to use for activation.
4. Click **Activate** to apply the license key to the appropriate instance(s).



Info

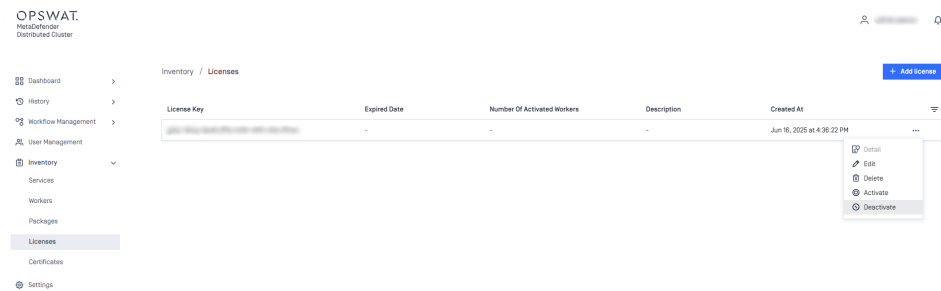
Once **Activate** is clicked, the license key will be applied to all unlicensed **MetaDefender Core** instances. Ensure that your license quota is sufficient to cover all unlicensed instances.

You can view the number of activated instances and available slots by selecting **Details** on the license key. At the moment, **Details** can only be viewed after activation is successful.

License Deactivation

Follow these steps to deactivate your license:

1. Sign in to MDDC **Control Center** console.
2. From the left side bar, select **Inventory** > **Licenses**.
3. From the list of available licenses, choose the key you wish to use for deactivation.
4. Click **Deactivate** to remove the license from all **MetaDefender Core** instances currently activated with the license key.



 Info

Once deactivated, the license slots will become available and can be reassigned when necessary.

When **MetaDefender Core** instances are undeployed through MDDC **Control Center**, their associated licenses are automatically deactivated.

Offline Activation

Collect Deployment IDs

1. Sign in to MetaDefender Distributed Cluster (MDDC) **Control Center** console.
2. Go to **Inventory > Licenses** and select **Offline License** tab.
3. Select Deployment IDs of MetaDefender Core instances you prefer to activate.
4. Press **Export** at the top right corner and save the exported file to your location of choice.

OPSWAT.
MetaDefender
Distributed Cluster

LOCAL/admin

Licenses

Export Refresh Activate

Online License	Offline License				
Deployment ID	Worker	Status	Expired Date		
☐	MSCLAUby0EMkrAssf84oFcXFip...	tuong-122	Inactivated	-	
☒	MSCLonRmFfy2zzK2M5N10XC6y...	tuong-142	Inactivated	-	
☐	MSCLJhb5L6Z44YUkpKhv5pndg...	R-88	Inactivated	-	
☒	MSCLyJBgs1RFgluahokn50xypZ...	D-202	Inactivated	-	
☐	MSCLT5lukfctzUS1Bmk0b9o5glU...	R-160	Inactivated	-	

Info

MetaDefender Distributed Cluster **Control Center** only displays the Deployment IDs of MetaDefender Core instances that have not been activated thus far.

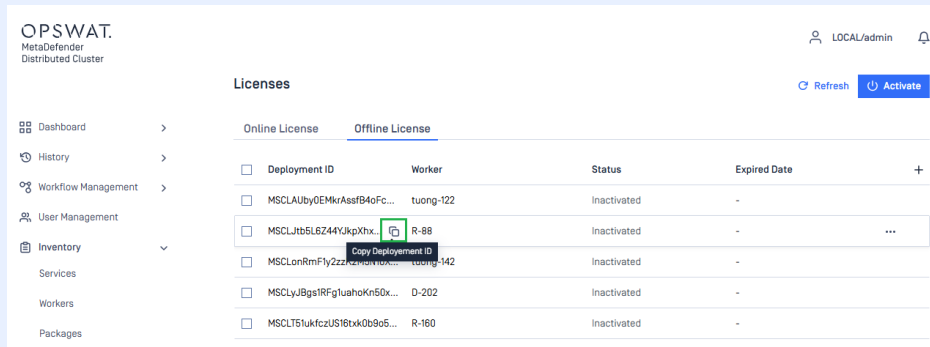
Info

The exported file includes a list of chosen Deployment IDs that will be used for activation in the subsequent stage.

Info

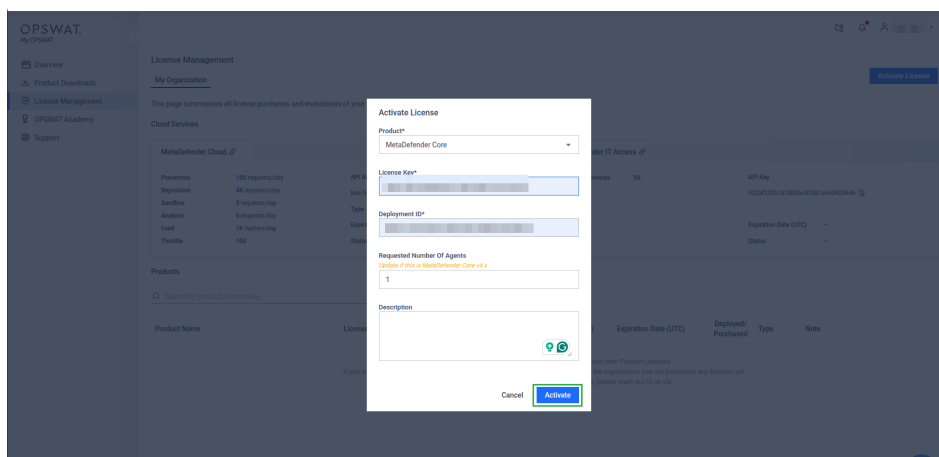
To collect the Deployment ID of a **single** MetaDefender Core instance, please:

1. Hover your mouse over the preferred Deployment ID to display the copy button.
2. Press the copy button.
3. Retain the copied Deployment ID and proceed to the next stage.

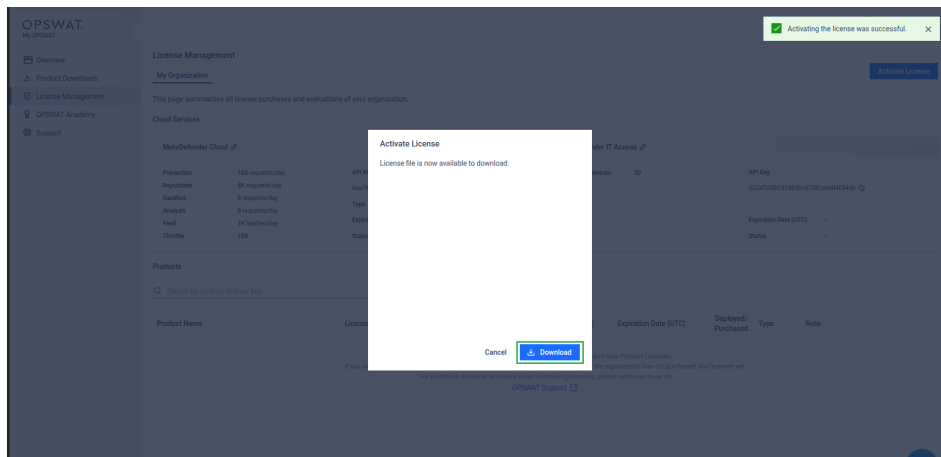


Activate license with Deployment ID

1. Sign in to MyOPSWAT with your account.
2. Navigate to **License Management** on the left side panel.
3. Click **Activate License**.
4. Fill out all necessary information, including your Activation Key, Deployment ID and selection of the Package you require.



5. Click **Activate**.
6. Click **Download** and store the license file to your secure location.



Info

The license file is associated with one unique Deployment ID. The users must carry out steps 3 to 6 for every deployment ID on their list.

Activate MetaDefender Core instances with license files

1. Sign in to MDDC **Control Center** console.
2. Go to **Inventory** > **Licenses** and select **Offline License** tab.
3. Click **Activate**.

OPSWAT.
MetaDefender
Distributed Cluster

LOCAL/admin

Refresh Activate

Licenses

Online License Offline License

Deployment ID	Worker	Status	Expired Date	
☐ MSCLAUby0EMkrAssfB4oFc...	tuong-122	Inactivated	-	
☐ MSCLHtb5L6Z44YkpxXhx...	R-88	Inactivated	-	...
☐ MSCLonRmFlyZzz...	tuong-142	Inactivated	-	
☐ MSCLyJBgs1Rfgluahokn50x...	D-202	Inactivated	-	
☐ MSCLT51ukfczUS16txk0b9o5...	R-160	Inactivated	-	

4. Drop the license files into the dash area for submission.

- Dashboard
- History
- Workflow Management
- User Management
- Inventory
 - Services
 - Workers
 - Packages
 - Modules
- Licenses

Licenses

Refresh Activate

Online License		Offline License		
☐	Deployment ID	Worker	Status	Expired Date
☐	MSCLdL2j8UjMScY28kmtZxwmdRZm...	tuong-142	Activated	01/03/2026
☐	MSCL2PG138TjeFta6oe4fa7LBEZSNW...	tuong-122	Inactivated	-
☐	MSCLyJ9ga1RFg\ushokK50ygzvehg...	D-202	Activated	01/03/2026
☐	MSCLuJAuBR5wuDyZ1mw9ktzRgAvZR...	R-160	Inactivated	-
☐	MSCLNroFSDyUuYVkkEaas54AYUm8c...	R-88	Inactivated	-

Module update

MetaDefender Distributed Cluster (MDDC) introduces two modes of Module Update. To switch between the modes, please sign in to MDDC **Control Center** console, navigate to **Settings** and select **Module Update** tab.

OPSWAT.
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Settings

Settings

Security

Module Update

Data Retention

Health Check

Export

About

Update mode

Online

Automatically update

Every 4 hours

Offline

Info

Online update mode is enabled by default.

Online module update

In online update mode, MDDC **Control Center** will base its checks on the activated licenses to find and download the latest engine packages from OPSWAT online update infrastructure, repeating this process every four hours.

OPSWAT.
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Modules

Licenses

Certificates

Settings

Modules

Schedule

Refresh

Upload Package

Update All

Engine Name	Status	Version	Database	Type	Platform	Last Update
ClamAV	Storing...	1.4.3-2545	1756339200	Anti-Malware	Linux	Aug 28, 2025 6:47:38 PM
Bitdefender	Storing...	3.0.1.297-2093	7.99399	Anti-Malware	Linux	Aug 28, 2025 6:47:38 PM
Varist	Ready	6.6.2-2219	202508281005	Anti-Malware	Linux	Aug 28, 2025 6:47:32 PM
Country of Origin	Ready	2.0.1-431	1.0.1	Country of Origin	Linux	Aug 28, 2025 6:47:30 PM
Archive Compression	Downloading...	7.5.2-7557	7.5.2-7557	Compression	Linux	Aug 28, 2025 6:47:25 PM
Avira	Downloading...	4.15.27-2314	1756377633	Anti-Malware	Linux	Aug 28, 2025 6:47:25 PM
AhnLab	Ready	3.27.2.4-2358	2025.08.28.03	Anti-Malware	Linux	Aug 28, 2025 4:08:23 PM
Archive Extraction	Ready	7.5.2-7557	7.5.2-7557	Archive	Linux	Aug 28, 2025 4:08:22 PM
Proactive DLP	Downloading...	3.0.0-4527	1755946854-fs2d	Proactive DLP	Linux	-
Deep CDR	Downloading...	7.5.2-23142	5.1.5	Deep CDR	Linux	-
ESET	Downloading...	1.4.2.0-2178	1756373084	Anti-Malware	Linux	-
FileType	Ready	7.5.3-9133	7.5.3-9133	Filetype Detection	Linux	Aug 28, 2025 4:08:24 PM
IKARUS	Downloading...	6.4.12-2193	108472	Anti-Malware	Linux	-
InSights Threat Intelli...	Downloading...	2.1.0-293	1756378926	InSights Threat Intelligence	Linux	-

All downloaded engine packages are verified and stored in MDDC **File Storage** for licensed MetaDefender Core instances to pull, install, or upgrade on their end. Through this mechanism, the instances cease to independently pull the engine packages from the update infrastructure, conserving network bandwidth while enhancing their readiness.

Offline module update

In offline update mode, administrators must download the licensed engine packages from **MetaDefender Update Downloader** and upload them manually to MDDC **Control Center**.

Info

Please reference [here](#) for more details about downloading engine packages from MetaDefender Update Downloader.

1. Sign in to MDDC **Control Center** console.
2. Go to **Inventory > Modules**.
3. Press **Upload Package** at the top right corner.

OPSWAT.
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Modules

Licenses

Certificates

Settings

LOCAL/admin

Schedule

Refresh

Upload Package

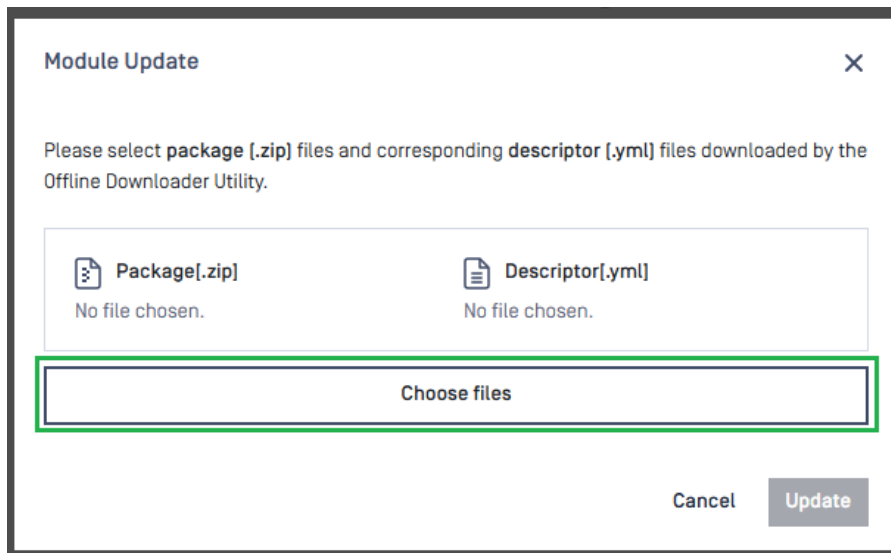
Update All

Engine Name	Status	Version	Database	Type	Platform	Last Update
Deep CDR	Ready	7.5.2-23142	5.1.5	Deep CDR	Linux	Aug 28, 2025 7:0...
Archive Extraction	Ready	7.5.2-7557	7.5.2-7557	Archive	Linux	Aug 28, 2025 7:0...
AhnLab	Ready	3.27.2.4-2358	2025.08.23.02	Anti-Malware	Linux	Aug 28, 2025 7:0...

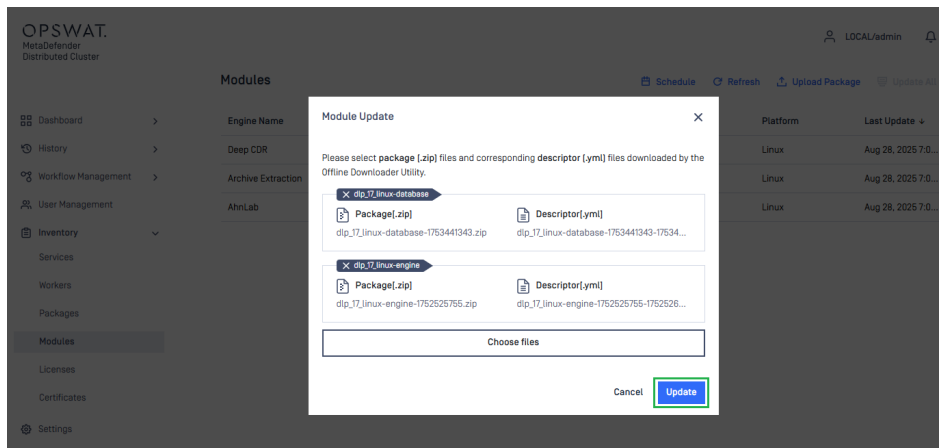
Info

Update All is always disabled if Offline Update mode is selected in **Settings > Module Update**.

4. Choose your engine package files.



5. Click **Update** to submit the package files.



Info

Package files from various engines can be selected simultaneously.

6. Wait until engine packages are ready.

OPSWAT.
MetaDefender
Distributed Cluster

Dashboard >

History >

Workflow Management >

User Management >

Inventory >

Services

Workers

Packages

Modules

Licenses

Certificates

Settings

Protecting the
World's Critical Infrastructure

Modules

Schedule Refresh Upload Package Update All

Engine Name	Status	Version	Database	Type	Platform	Last Update
Proactive DLP	Storing...	3.0.0-4627	1753441231-1...	Proactive DLP	Linux	Aug 28, 2025 7:2...
Deep CDR	Ready	7.5.2-23142	5.1.5	Deep CDR	Linux	Aug 28, 2025 7:0...
Archive Extraction	Ready	7.5.2-7557	7.5.2-7557	Archive	Linux	Aug 28, 2025 7:0...
AhnLab	Ready	3.27.2.4-2358	2025.08.23.02	Anti-Malware	Linux	Aug 28, 2025 7:0...

Uploading Files

dlp_17_linux-engine-1752525755.zip

dlp_17_linux-engine-1752525755-1752526319.yml

dlp_17_linux-database-1753441343.zip

dlp_17_linux-database-1753441343-1753441466.yml

Info

All uploaded engine packages are verified and stored in MDDC **File Storage** for licensed MetaDefender Core instances to pull, install, or upgrade on their end.

7. Engine update statuses on MetaDefender Core instances can be monitored in **Dashboard > System Health > Worker Health**.

OPSWAT.
MetaDefender
Distributed Cluster

Dashboard >

System Health

System Activity

Executive Report

History >

Workflow Management >

User Management >

Inventory >

Settings

Protecting the
World's Critical Infrastructure

RabbitMQ

Healthy

1/1

Instance count

Identity Service

Healthy

Workers Health

100%

Overall Health

0 Unhealthy Workers

0 Available Workers

0 Cores Windows 2 Cores Linux 0 API Gateway Windows 1 API Gateway Linux

Name	Status	Health												
R-88	Running	Healthy	1/1		-	-	-	-						
R-160	Running	Healthy	1/1		-	-	-	-						

High Availability

Overview

In MetaDefender Distributed Cluster (MDDC), critical components for its continuous operation include RabbitMQ, Redis, Postgres, and MDDC File Storage. Any disruption of these components will lead to an interruption in the scanning processes and result in a failed verdict for the processed files. To prevent the interruption, high-availability solutions must be implemented on the components.

A strategy for achieving high availability is the replication and redundancy of essential components. The key concept is that if a single component fails, the redundant system takes over seamlessly, avoiding any interruption in service. Following are guidelines to set up the high availability solution on individual components and apply them in MetaDefender Distributed Cluster.

- High availability support for MDDC File Storage.
- High availability support for RabbitMQ.
- High availability support for Redis.
- High availability support for Data lake.

High Availability support for File Storage

Key concept

The High Availability solution for MetaDefender Distributed Cluster (MDDC) **File Storage** is implemented in this manner:

- A file is stored across multiple MDDC **File Storages** by MDDC **API Gateway** or **MetaDefender Core**.
- MDDC **API Gateway** or **MetaDefender Core** must request all MDDC **File Storage** instances for a file existence or a file download.

Info

A minimum of **three** MDDC File Storage instances must be installed on separate hosts for High Availability solution to function properly.

All MDDC File Storage instances must be of an identical version.

Setup Instructions

1. Setup MDDC **File Storage** instances on individual servers.
2. Sign to MDDC **Control Center** console with your Administrator account.
3. Navigate to **Inventory > Services**.
4. Expand the **File Storage Service** group.
5. Click **Add service**.
6. Enter the values for **Name**, **Host**, **Port** and **Connection Key** fields of individual MDDC File Storage instances set up in Step 1.
7. Click the **Check** icon in the bottom right to complete.

OPSWAT
MetaDefender
Distributed Cluster

LOCAL/admin

Dashboard >
History >
Workflow Management >
User Management >
Inventory >
Services >
Workers
Packages
Licenses
Certificates
Settings

Type

File Storage Service

Instance Count

3/3

Status

Healthy

Name	Status	Version	Platform	Last Healthy	Last Update	Added By	
File-Storage-1	Healthy	2.1.0	Windows	Jul 24, 2025 at 4:55:06 PM	Jul 24, 2025 at 4:55:38 PM	LOCAL\admin	+
File-Storage-2	Healthy	2.1.0	Windows	Jul 24, 2025 at 4:55:06 PM	Jul 24, 2025 at 4:55:43 PM	LOCAL\admin	
File-Storage-3	Healthy	2.1.0	Windows	Jul 24, 2025 at 4:55:06 PM	Jul 24, 2025 at 4:55:46 PM	LOCAL\admin	

Name*

Enter name

Host*

Enter host name

Port*

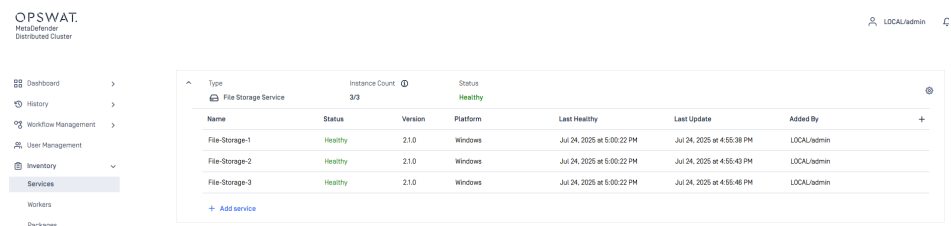
Port

Connection Key*

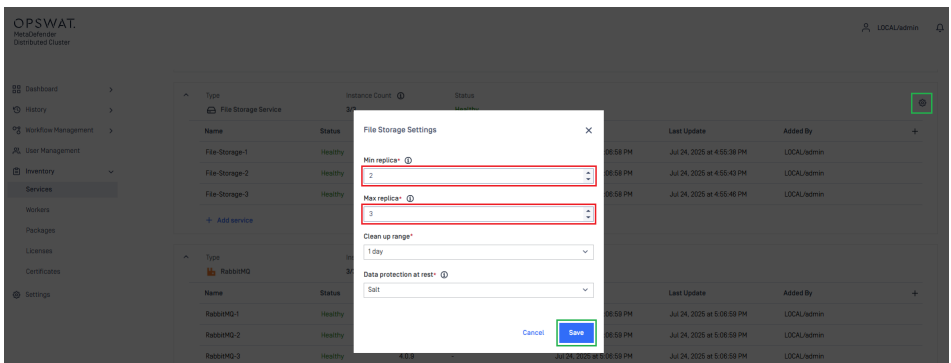
Existing key preserved if blank

+ Add service

8. Ensure all MDDC File Storage instances are healthy and reachable by MDDC Control Center .



9. Click on the gear icon in the top left of File Storage Service group to configure the minimum and maximum replicas.



- Minimum replica: The minimum number of data copies that must be written for the operation to succeed.
- Maximum replica: The maximum of data copies stored across the system.

Info

To balance performance and high availability efficiency, the minimum and maximum replicas should be set to the following values:

- Min replica = 2
- Max replica = 3

10. Click Save to complete.

High Availability support for RabbitMQ

Info

A minimum of **three** RabbitMQ nodes must be installed on separate hosts for High Availability solution to function properly.

An **odd number** of RabbitMQ nodes is required.

All RabbitMQ nodes must be of an identical version.

RabbitMQ cluster

1. Install RabbitMQ nodes on servers.
2. Ensure each node can resolve its own hostname and those of the others.
 - Start **Command Prompt on Windows** or **Terminal on Linux**, and run the following command to get hostname.

bash

```
# Windows
> hostname

# Debian/Ubuntu or Red Hat/Rocky
$ hostname
```

- In **Command Prompt on Windows** or **Terminal on Linux** of any RabbitMQ node, and run the following command to ping to the other using its hostname.

bash

```
# Windows
> ping <other_node_hostname>

# Debian/Ubuntu or Red Hat/Rocky
$ ping <other_node_hostname>
```

3. On each RabbitMQ nodes, open the following ports.

Default port	Process
5672	Used by MDDC Control Center , MDDC API Gateway and MetaDefender Core .
4369	Used by discovery daemon on each RabbitMQ nodes and <code>rabbitmqctl</code> tool.
25672	Used by each RabbitMQ nodes and <code>rabbitmqctl</code> tool to communicate to the other nodes.
15672	Used by <code>rabbitmq-management</code> plugin.

4. Verify that the Erlang cookies of all RabbitMQ nodes are identical.

Info

RabbitMQ nodes and `rabbitmqctl` tool use a cookie to determine whether they are allowed to communicate with each other. For two nodes to be able to communicate they must have the same shared secret called the Erlang cookie. The cookie is a string of alphanumeric characters up to 255 characters in size.

- In **Windows**, access the specified locations to check the cookie contents.

Type	Location
Server cookie	<code>C:\Windows\system32\config\systemprofile.erlang.cookie</code>
Command line cookie	<code>C:\Users%USERNAME%.erlang.cookie</code>

- In **Linux**, access the specified locations to check the cookie contents.

Type	Location
Server cookie	<code>/var/lib/rabbitmq/.erlang.cookie</code>
Command line cookie	<code>\$HOME/.erlang.cookie</code>

5. Select one node to be the leader of RabbitMQ cluster.

6. In **Command Prompt on Windows** or **Terminal on Linux** of the server hosting the leader, run the following command to obtain its node name.

bash

```
# Windows
> rabbitmqctl status

# Debian/Ubuntu or Red Hat/Rocky
$ rabbitmqctl status
```

7. In **Command Prompt on Windows** or **Terminal on Linux** of each member node server, run the following command to join the node to the same cluster as the leader.

bash

```
# Windows
> rabbitmqctl stop_app
> rabbitmqctl reset
> rabbitmqctl join_cluster <leader_node_name>
> rabbitmqctl start_app

# Debian/Ubuntu or Red Hat/Rocky
$ rabbitmqctl stop_app
$ rabbitmqctl reset
$ rabbitmqctl join_cluster <leader_node_name>
$ rabbitmqctl start_app
```

8. In **Command Prompt on Windows** or **Terminal on Linux** of all nodes, ensure they are in the same cluster.

bash

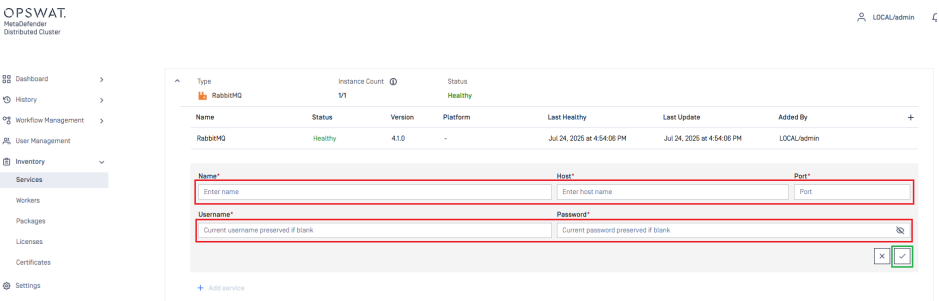
```
# Windows
> rabbitmqctl cluster_status

# Debian/Ubuntu or Red Hat/Rocky
$ rabbitmqctl cluster_status
```

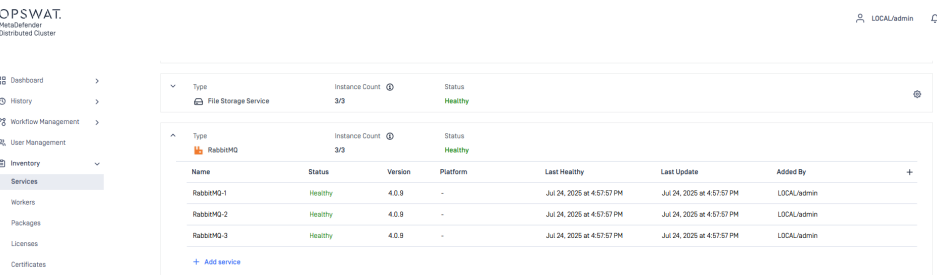
Setup Instructions

1. Sign to MDDC **Control Center** console with your Administrator account.
2. Navigate to **Inventory** > **Services**.

- 3. Expand the RabbitMQ group.
- 4. Click Add service.
- 5. Enter the values for Name, Host, Port, Username and Password fields of individual RabbitMQ nodes set up in Build RabbitMQ cluster.



- 6. Click the Check icon in the bottom right to complete.
- 7. Ensure all RabbitMQ nodes are reachable by the MDCC **Control Center**.



High Availability support for Redis

Info

A minimum of **two** Redis instances must be installed on separate hosts for High Availability solution to function properly.

An **odd number** of **Redis Sentinels** should be installed.

Redis Sentinel

1. Install Redis instances on servers.
2. Select one instance as primary. In **Linux Terminal** of the other instances (replicas), run the following command:

bash

```
# Debian/Ubuntu or Red Hat/Rocky
$ redis-cli replicaof <primary_host> <primary_port>
```

3. Build configuration file for **Redis Sentinel**.

bash

```
# The port on which the Sentinel should run
port <SENTINEL_PORT>

# By default Redis does not run as a daemon. Use 'yes' if you
need it.
# Note that Redis will write a pid file in /var/run/redis.pid
when daemonized.
daemonize yes

sentinel monitor myprimary <PRIMARY_IP> <PRIMARY_PORT> 2
# sentinel monitor <master-name> <ip> <port> <quorum>
# quorum is the number of Sentinels that need to agree about
the
# fact the master is not reachable, in order to really mark
the master as
# failing, and eventually start a failover procedure if
possible.

sentinel down-after-milliseconds myprimary 2000
# means sentinel will consider master down after 2 seconds

sentinel failover-timeout myprimary 4000
# means the chosen sentinel has 4 seconds to perform failover

sentinel parallel-syncs myprimary 2
# sets the number of replicas that can be reconfigured to use
the new master
# after a failover at the same time. The lower the number, the
more time it
# will take for the failover process to complete, however if
the replicas are
# configured to serve old data, you may not want all the
replicas to
# re-synchronize with the master at the same time. While the
replication process is
# mostly non blocking for a replica, there is a moment when it
stops to
# load the bulk data from the master. You may want to make
sure only one
# replica at a time is not reachable by setting this option to
the value of 1.
```

 Info

Duplicate the configuration file and modify `SENTINEL_PORT` to the appropriate port that the Redis Sentinel instance listens on.

4. Install Redis Sentinel instances on servers with the corresponding configuration files.

bash

```
# Debian/Ubuntu or Red Hat/Rocky
$ sudo redis-server </path/to/sentinel-config-file> --sentinel
```

5. Verify the Redis primary and its replicas. In **Linux Terminal** of any machine, run the following command:

bash

```
# Debian/Ubuntu or Red Hat/Rocky
$ redis-cli -h <sentinel_host> -p <sentinel_port>

# Provides information about the Primary
> sentinel master myprimary

# Gives you information about the replicas connected to the
Primary
> sentinel replicas myprimary

# Provides information on the other Sentinels
> sentinel sentinels myprimary

# Provides the IP address of the current Primary
> sentinel get-master-addr-by-name myprimary
```

Setup instructions

1. Sign to MDDC **Control Center** console with your Administrator account.
2. Navigate to `Inventory > Services`.
3. Expand the `Redis` group.
4. Click `Add service`.
5. Enter the values for `Name`, `Host`, `Port`, `Username` and `Password` fields of individual Redis instance.

- Dashboard
- History
- Workflow Management
- User Management
- Inventory
- Services
- Workers
- Packages
- Licenses
- Certificates
- Settings

Protecting the
World's Critical Infrastructure

RabbitMQ

3/3

Healthy

Type

Redis

Instance Count

3/3

Status

Healthy

Name	Host	Port	Status	Role	Version	Platform	Last Healthy	Last Update	Added By	+
Redis-3	10.40.170.87	6379	Healthy	Replica	8.0.3	Linux	Jul 30, 2025 at 4:1...	Jul 30, 2025 at 4:...	LOCAL/admin	
Redis-2	10.40.170.75	6379	Healthy	Primary	8.0.3	Linux	Jul 30, 2025 at 4:1...	Jul 30, 2025 at 4:...	LOCAL/admin	
Redis-1	10.40.170.74	6379	Healthy	Replica	8.0.3	Linux	Jul 30, 2025 at 4:1...	Jul 30, 2025 at 4:...	LOCAL/admin	

Name*

Host*

Port*

Enter name

Enter host name

Port

Username

Password

Enter username

Enter password

X

✓

+ Add service

Warning

MDDC Control Center **only accepts Redis** and **not Redis Sentinel**.

- 6. Click the Check icon in the bottom right to complete.
- 7. Ensure all RabbitMQ nodes are reachable by the MDDC **Control Center**.

- Dashboard
- History
- Workflow Management
- User Management
- Inventory
- Services
- Workers
- Packages
- Licenses
- Certificates
- Settings

Protecting the
World's Critical Infrastructure

Type

Data Warehouse

Instance Count

1/1

Status

Healthy

Type

File Storage

Instance Count

3/3

Status

Healthy

Type

RabbitMQ

Instance Count

3/3

Status

Healthy

Type

Redis

Instance Count

3/3

Status

Healthy

Name	Host	Port	Status	Role	Version	Platform	Last Healthy	Last Update	Added By	+
Redis-3	10.40.170.87	6379	Healthy	Replica	8.0.3	Linux	Jul 30, 2025 at 4:1...	Jul 30, 2025 at 4:...	LOCAL/admin	
Redis-2	10.40.170.75	6379	Healthy	Primary	8.0.3	Linux	Jul 30, 2025 at 4:1...	Jul 30, 2025 at 4:...	LOCAL/admin	
Redis-1	10.40.170.74	6379	Healthy	Replica	8.0.3	Linux	Jul 30, 2025 at 4:1...	Jul 30, 2025 at 4:...	LOCAL/admin	

+ Add service

High Availability support for PostgreSQL Data lake

Installation

Info

- repmgr is compatible solely with Linux-based operating systems.
- The repmgr version in use must be compatible with the major version of the installed PostgreSQL.
- All PostgreSQL servers must be of the same version and run on the same Operating System.

High availability solution for PostgreSQL data lake requires a single primary server along with a minimum of two standby servers. Both PostgreSQL and repmgr must be installed on every server.

Warning

On the servers that target to run as standby:

- Do not create a PostgreSQL instance [i.e., do not execute `initdb` or any database creation scripts provided by packages].
- Ensure the destination data directory exists and is owned by the `postgres` system user.

1. Select your Linux distribution here and follow the steps to install PostgreSQL accordingly.
2. Follow the steps to install repmgr.

Primary configuration

1. Choose one of the installed servers to be the primary one.
2. Navigate to the folder containing `postgresql.conf` file and create a replication config file named `postgresql.replication.conf`.

i Info

By default, `postgresql.conf` file is placed at

- `/var/lib/pgsql/<version>/data/` on Red Hat/Rocky.
- `/var/lib/postgresql/<version>/main` on Debian/Ubuntu.

bash

```

# Enable replication connections; set this value to at least
one more
# than the number of standbys which will connect to this
server
# (note that repmgr will execute "pg_basebackup" in WAL
streaming mode,
# which requires two free WAL senders).
#
# See: https://www.postgresql.org/docs/current/runtime-config-replication.html#GUC-MAX-WAL-SENDERS

max_wal_senders = 10

# If using replication slots, set this value to at least one
more
# than the number of standbys which will connect to this
server.
# Note that repmgr will only make use of replication slots if
# "use_replication_slots" is set to "true" in "repmgr.conf".
# (If you are not intending to use replication slots, this
value
# can be set to "0").
#
# See: https://www.postgresql.org/docs/current/runtime-config-replication.html#GUC-MAX-REPLICATION-SLOTS

max_replication_slots = 10

# Ensure WAL files contain enough information to enable read-
only queries
# on the standby.
#
# See: https://www.postgresql.org/docs/current/runtime-config-wal.html#GUC-WAL-LEVEL

wal_level = 'hot_standby'

# Enable read-only queries on a standby
#
# See: https://www.postgresql.org/docs/current/runtime-config-replication.html#GUC-HOT-STANDBY

hot_standby = on

# Enable WAL file archiving
#
# See: https://www.postgresql.org/docs/current/runtime-config-wal.html#GUC-ARCHIVE-MODE

```

```

archive_mode = on

# Set archive command to a dummy command; this can later be
# changed without
# needing to restart the PostgreSQL instance.
#
# See: https://www.postgresql.org/docs/current/runtime-config-wal.html#GUC-ARCHIVE-COMMAND

archive_command = '/bin/true'

# This config should be added if you plan to use repmgrd for
# automatic failover
# See: https://www.repmgr.org/docs/current/repmgrd-basic-configuration.html
shared_preload_libraries = 'repmgr'

wal_log_hints = on # for pg_rewind when rejoin

```

3. Add the replication configuration file name to the end of `postgresql.conf` file and save the modifications.

bash

```

...
include 'postgresql.replication.conf'

```

4. In Terminal, run the following commands to create `repmgr` user and database.

bash

```

$ createuser -s repmgr
$ createdb repmgr -O repmgr

```

Info

In this guideline, although the term `repmgr` is used for both user and database, any names can be used.

5. Edit `pg_hba.conf` file to configure the authentication.

bash

```
# Ensure the repmgr user has appropriate permissions in
pg_hba.conf
# and can connect in replication mode
# pg_hba.conf should contain entries similar to the following:
# Uncomment this if you want to access Postgresql database via
pgadmin with user "postgres":
#host      all                postgres          0.0.0.0/0
scram-sha-256

local      replication  repmgr
trust
host      replication  repmgr          127.0.0.1/32
trust
#or
host      replication  repmgr          0.0.0.0/0
trust

local      repmgr        repmgr
trust
host      repmgr        repmgr          127.0.0.1/32
trust
#or
host      repmgr        repmgr          0.0.0.0/0
trust
```

6. Restart PostgreSQL server.

bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> restart
```

7. Create `repmgr.conf` file, fill out information in brackets and store it in a location of your choice.

Warning

`repmgr.conf` file should not be placed inside PostgreSQL data folder as it may be overwritten.

bash

```

node_id=<any_node_id>
node_name=<any_node_name>
# connection info of the current node
conninfo='host=<host_address_of_node> user=repmgr
dbname=repmgr connect_timeout=2'
data_directory='<postgres_data_dir>'
failover='automatic' # for repmgrd (automatic failover)
promote_command='<postgres_dir>/repmgr standby promote -f "
<your_dir>/repmgr.conf" --log-level INFO'
follow_command='<postgres_dir>/repmgr standby follow -f "
<your_dir>/repmgr.conf" -W --log-level INFO'
reconnect_attempts='5'
reconnect_interval='1'
monitor_interval_secs='1'
pg_bindir='<postgres_dir>'
# enable this so that repmgr only vote new primary
# when none of the standbys can connect to current primary
primary_visibility_consensus=true

```

Key	Red Hat/Rocky	Debian/Ubuntu
postgres_data_dir	/var/lib/pgsql/<version>/data/	/var/lib/postgresql/<version>/main/
postgres_dir	/usr/pgsql-<version>/bin/	/usr/lib/postgresql/<version>/bin/
your_dir	Directory to repmgr.conf file.	Directory to repmgr.conf file.

8. In Terminal, run the following commands to register the primary server.

bash

```

$ cd path/to/repmgr
$ repmgr -f <repmgr_config_file_path> primary register

INFO: connecting to primary database...
NOTICE: attempting to install extension "repmgr"
NOTICE: "repmgr" extension successfully installed
NOTICE: primary node record (id: 1) registered

```

Standby configuration

1. Create `repmgr.conf` file and modify values of `node`, `node_name`, `conninfo` accordingly.

2. Store the file in your reference location.
3. Stop PostgreSQL server.

bash

```
$ cd /path/to/pg_ctl  
$ pg_ctl -D <postgresql_data_dir> stop
```

4. In Terminal, run the following commands to clone data from the primary server.

bash

```
$ cd path/to/repmgr  
$ repmgr -h <primary_server_host> \  
    -U repmgr -d repmgr \ # primary repmgr <user> and  
<database>  
    -f <standby_repmgr_config_file_path> \  
    -c \ # fast checkpoint to speed up process  
    standby clone \  
    --dry-run # dry run to check if the primary can be  
cloned  
  
$ repmgr -h <primary_server_host> \  
    -U repmgr -d repmgr \ # primary repmgr <user> and  
<database>  
    -f <standby_repmgr_config_file_path> \  
    -c \ # fast checkpoint to speed up process  
    standby clone
```

5. Start PostgreSQL server.

bash

```
$ cd /path/to/pg_ctl  
$ pg_ctl -D <postgresql_data_dir> start
```

6. In Terminal, run the following commands to register the standby server.

bash

```
$ cd /path/to/repmgr
$ repmgr -f <standby_repmgr_config_file_path> \
    standby register
```

7. Check if the node was registered successfully.

bash

```
$ cd /path/to/repmgr
$ repmgr -f /etc/repmgr.conf cluster show
```

Automatic failover

In Terminal, run the following command to start Replication manager daemon on all PostgreSQL servers (including primary and standbys)

bash

```
$ cd /path/to/repmgr
$ repmgrd -f <repmgr_config_file_path>
```

Rejoin after a failure

Info

Replication manager daemon `repmgrd` does not automatically join a failed PostgreSQL server node to the cluster. Consequently, the cluster contains at least two primary nodes at one time, and the system administrator has to join the node to the cluster manually.

1. Do not restart the failed PostgreSQL server, run the following command.

bash

```
$ cd /path/to/repmgr
$ repmgr -f <repmgr_config_file> node rejoin \
    --force-rewind \ # use pg_rewind to help with diverge
    timeline
    -d 'host=<current_primary> dbname=repmgr
    user=repmgr'
```

2. If a node rejoin fails, do register the failed node as a standby. In Terminal, run the following command.

bash

```
$ cd /path/to/repmgr
$ repmgr -h <current_primary_server_host> \
        -U repmgr -d repmgr \ # primary repmgr <user> and
<database>
        -f <standby_repmgr_config_file_path> \
        -c \ # fast checkpoint to speed up process
        -F \ # this overwriten the the data folder if it
was created
        standby clone \
```

3. Start PostgreSQL server.

bash

```
$ cd /path/to/pg_ctl
$ pg_ctl -D <postgresql_data_dir> start
```

4. Force register the node as a standby.

bash

```
$ cd /path/to/repmgr
$ repmgr -f <standby_repmgr_config_file_path> \
        -F \ # forcefully overwrite an existing node record
or user --force
        standby register
```

Setup instructions

1. Sign to MDDC **Control Center** console with your Administrator account.
2. Navigate to **Inventory > Services**.
3. Expand the **Data Lake** group.
4. Click **Add service**.
5. Enter the values for **Name**, **Host**, **Port**, **Username** and **Password** fields of individual PostgreSQL instance.
6. Click the Check icon in the bottom right to complete.

Services

✓ All your services are connected. You can start deploying. [Go to Workers](#)

Refresh

Type

Data Lake

Instance Count

2/2

Status

Healthy

Name	Host	Port	Status	Role	Version	Platform	Last Healthy	Last Update	Added By	
10.40.170.54	10.40.170.54	5432	Healthy	Primary	16.9	Linux	Aug 28, 2025 a...	Aug 28, 2025 a...	LOCAL/admin	+
10.40.170.110	10.40.170.110	5432	Healthy	Standby	16.10	Linux	Aug 28, 2025 a...	Aug 28, 2025 a...	LOCAL/admin	...

Name*

10.40.170.109

Host*

10.40.170.109

Port*

5432

Username*

Enter username

Password*

Enter password

✕

✓

+ Add service

>

Type

Data Warehouse

Instance Count

3/3

Status

Healthy

>

Type

File Storage

Instance Count

3/3

Status

Healthy

⌵

7. Ensure all PostgreSQL instances are reachable by the MDDC **Control Center**.

System settings

This section shows MetaDefender Distributed Cluster settings.

Data Retention

You can find this feature under: **Settings > Data Retention**.

This setting enables users to define the retention period for specific data types, helping optimize system storage and maintain efficiency.

Available Data Categories

- 1. **Processing History:** History of scan results.
- 2. **Executive Report:** Statistics data.
- 3. **Audit Log:** Detailed logs of user actions and system events.

In case you do not want to enable automatic clean up, set the value to off. This will prevent automatic removal.

 Warning

Disabling automatic clean-up may lead to data accumulation, which can affect system performance and increase storage costs.

Settings


Security


Module Update


Data Retention


Health Check

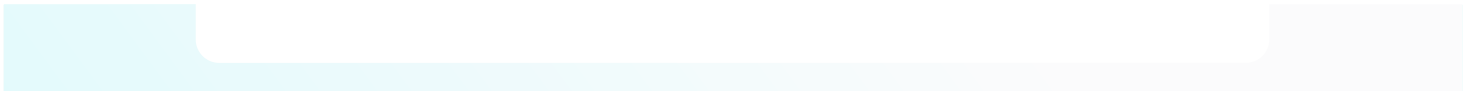

Export


About

Data Retention

Automatically delete data after a certain time

Processing history	Older than	Off
Executive report	Older than	6 months
Audit log	Older than	Off



Remote Support Package Gathering

The support package contains log files and is essential for OPSWAT to troubleshoot issues. Since version **2.2.0**, it is now possible to gather a support package remotely via the web console of the MetaDefender Distributed Cluster Control Center.

i Info

Ensure that all MDDC services are upgraded to version 2.2.0 or higher to fully support this feature.

Remote support package gathering steps:

- 1. Go to **Settings > Export**.
- 2. Select which MetaDefender Distributed Cluster services need to generate support package, then select **Generate**

OPSWAT.

MetaDefender Distributed Cluster

Dashboard >

History >

Workflow Management >

User Management

Inventory >

Settings

Settings

Security

Module Update

Data Retention

Export

About

Export Support Package

Collect log files, system information and other diagnostic data.

1. Select Services

Q Search...

All

Generate

☐ All

☒ File Storage

☒ Workers

☐ Control Center

☒ Identity Service

☒ Identity Service

☒ File Storage: Fs

☒ Worker: Worker-Core

☒ Worker: Worker-Apigw

192.168.10.11

192.168.10.11

192.168.10.12

8890

8893

8893

2. Support Package Details

ID

Start Time

Duration

Status

Action

No data available

131

Info

Select the MetaDefender Distributed Cluster Worker service to generate a support package for its deployed instance as well.

3. Wait for the generation process to complete successfully. Once it is done, the download button will appear, and the support packages will be ready for download.

2. Support Package Details

☐	ID	Start Time	Duration	Status	Action
☐	1753872495977	7/30/25, 5:48 PM	15s 790ms	 Success	Download
Service Details					
	Service Name	Host	Start time	Duration	Status
	Identity Service	-	7/30/25, 5:48 PM	10s 39ms	 Success
	fs	192.168.10.11	7/30/25, 5:48 PM	10s 39ms	 Success
	worker-core	192.168.10.11	7/30/25, 5:48 PM	5s 10ms	 Success
	worker-apigw	192.168.10.12	7/30/25, 5:48 PM	10s 20ms	 Success

Info

The size of the support package may vary depending on log size and the number of days for collection. If disk space is insufficient, certain log files may be excluded from the support package.

Warning

All support package files will be downloaded to the MetaDefender Distributed Cluster Control Center. Please monitor the disk space on the host running this service when using this functionality, as the size of log files can be very large.

4. Click **Download**

Info

Some services may fail due to connection issues or insufficient disk space. In such cases, only the successfully generated support packages will be available for download. Users can view detailed error information if failures occur.

Security

Setup HTTPS

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

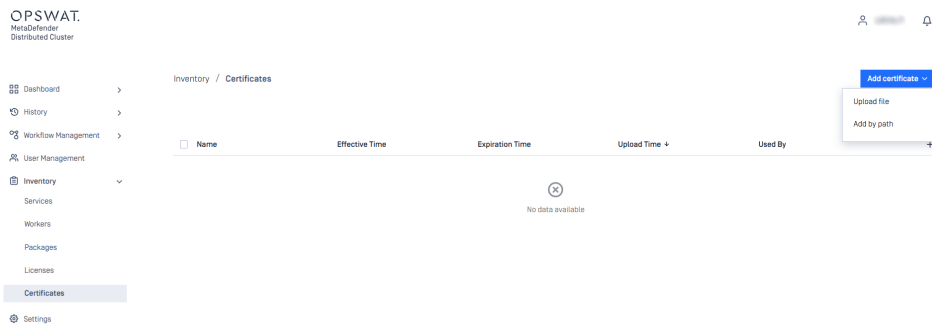
The TLS protocol aims primarily to provide confidentiality (privacy) and data integrity between two communicating computer applications.

Info

HTTPS is not enabled by default. As a consequence sessions between the wizard's backend and the browser may be insecure.

Steps to setup this feature:

1. Go to **Inventory > Certificates**
2. Click **Add certificate**
 - a. To add a certificate using a file path, choose **Add by path** and enter the location of both the certificate and its corresponding private key file.
 - b. To upload certificate file, select **Upload file**.



Certificate YML sample file:

yaml

```
---
private_key: |
  -----BEGIN PRIVATE KEY-----

MIIJQwIBADANBgkqhkiG9w0BAQEFAASCCS0wgGkpAgEAAoICAQCjYtuWaICCY0
tJ

PubxpIgIL+WWmz/fmK8IQR11Wtee6/IUyUlo5I602mq1qcLhT/kmpoR8Di3DAm
HK

nSWdPWtn1BtXLERLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nTekLWcfI5
ZZ

toGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tItNHT/m6D
SU

0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+m6jzhNyM
BT

J1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8buWQUjy5N8
pS

Np7szdYsnLJpvAd0sibrNPjC0FQCNrpNjgJmIK3+mKk4kXX7ZTwefoAzTK4l2p
HN

uC53QVc/EF++GBLXmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCFIBKIzv9cg
i9

fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfqNv0nPN1IM
Sn

zXCast7I2fiuRXdIz96Kj1GQpP4XfNVA+RGL7aMnW0FIaVrKWLzAtgzoGMTvP/
Au

ehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC9psNcjTM
aB

QLN03T1Qe1bs7Y27sxdEnNUth4kI+wIDAQABaoICAFWe8MQZb37k2gdAV3Y6aq
8f

qokKQqbCNLd3giGFwYkezHXoJfg6Di7oZxNcKyw35LFEghkgtQqErQqo35VPio
H+

vXUpW0jnCmM4muFA9/cX6mYMc8TmJsg0ewLdBC0ZVw+wPABlaqz+0U0iSMMftp
k9

fz9JwGd8ERyBsT+tk3Qi6D0vPZVsC1KqxxL/cwIFd3Hf2ZBtJXe0KBn1pktWht
5A
```

Kqx9mld20v17NjgiC1Fx9r+fZw/i0abFFwQA4dr+R8mEMK/7bd4VXFQ1o/QGGB
MT

G+u1FrSiDyP+rBIAaGC0i7gDjLAIBQeDhP409ZhswIEc/GBt0DU372a2CQK/u4
Q/

HBQvuBtKFNkGUooLgCCbFxzgNUGc83GB/6IwbEM7R5uXqsFiE71LpmroDyjKT1
Q8

YZkpIcLNVLw0usoGYHFm2rvCyEV1fsE3Ub8cFyTFk50Se0cF2QL2xzKmmBZEpX
gl

xBHR0hjgon0IKJDGfor4bH07Nt+1Ece8u2oTEKvpz5aIn440eC5mApRGy83/0b
vs

esnWjDE/bGpoT8qFuy+0urDEPNId44XcJm1IRI1G56ErxC3l0s11wrIpTmXXck
qw

zFR9s2z7f0zjeyxqZg4NTPI7wkM3M8BX1vp2GTBIeoxrWB4V3YArwu8QF80QBg
Vz

mgHl24nTg00UH10jZsABAOIBAQD0xftSDbSqGytcWqPYP3SZHAWDA004ACEM+e
Cw

au9ASut10ID1NDMJ8nC2ph25BMe5hHDWp2cGQJog7pZ/3qQogQho2gUniKDifN
77

40Qdyk11TzTVR0qmP8+efreIvqlzHmuqaGfGs5oTkZaWj5su+B+bT+9rIwZcwF
s5

YRINhQRx17qa++xh5mfE25c+M9fiIBTiNSo4lTxWMBSHnK8xrGaME7W0qTMb
FH

PgQz5FcXrJCCqWHilwNBeLDtp/ZECEB7y34khVh531mBE2mNzSVIQCgZP1I/Dv
Xj

W7UUNdgFwii/GW+6M0uUDy23UVQpbFzcV8o1C2nZc4Fb4zwBAoIBAQDKSJkFww
uR

naVJS6Wx0KjX8MCu9/cKPnwBv2mmI2jgGxHTw5sr3ahmF5eTb8Zo19BowytN+t
r6

2ZFoIBA9Ubc9esEAU8l3fggdfM82cuR9sGcfQVoCh8tMg6BP8IBL0mbSUhN3PG
2m

39I802u0fFNVQCJKhx1m1MFFLOu7lVcDS9JN+oYVPb6MDfBLm5j0iPuYkFZ4gH
79

J7gXI0/YKhaJ7yXthYVkdRsf6Eooer4RZgma62Dd1VNzSq3JB06rYjF7Lvd+Rw
DC

R1thHrmf/IXplxpNVkoMVxtzbrrbgnC25QmvRYc0rlS/kvM4yQhMH3eA7IycDZ
Mp

Y+0xm7I7jTT7AoIBAGKzKIMDXdCxBWKhNYJ8z7hiItN11IZZMW2TPUiY0rl6ya
Ch

BVXjM9W0r07QPnHZsUiByqb743adkbTUjmxJzjaVtxN7ZXwZv0VrY7I7fPWYn
CE

fXCr4+IVpZI/ZHZWpGX6CGSgT6E0jCZ5IUufIvEpqVSmtF8MqfX09o9uIYLokr
WQ

x1dB15UnuTLDqw8bChq705y6yfuWa0WvL7nxI8NvSsfj4y635gIa/0dFeBYZEf
HI

U1GdNVomwXwYEzge/c19ruIowX7HU/NgxMWTMZhpazlxgesXybel+YNcfDQ4e3
RM

OMz3ZFiaMaJsGGNf4++d9TmMgk4Ns6oDs6Tb9AECggEBAJYzd+S0Yo26iBu3nw
3L

65uEeh6xou8pXH0Tu4gQrPQTRZZ/nT3iNg0wqu1gRuxcq7T0jt41UdqIK08vN7
/A

aJavCpaKoIMowy/aGCbvAvjNPpU3unU8jd1/t08EXs79S5IKPcgAx87sTTi7KD
N5

SYt4tr2uPEe53NTXuSatilG5QCyExIEL0uzWAMKzg7CAiI1NS9foWeLyVkBgCQ
6S

me/L8ta+mUDy37K6vC34jh9vK9yrwF6X44ItRo0JafCaVfGI+175q/eWcqTX4q
+I

G4tKls4sL4mg0JLq+ra50aYMxbcuommctPMXU6CrrYyQpPTHMNVdQy2ttFdsq9
iK

TncCggEBAMmt/8yvPf1S+xv3kg/ZBvR9JB1In2n3rUCYYD47ReKFqJ03VmqsC9
nY

56s9w70U08perBX1JYmKZQh04293lvxZD2Iq4NcZbVSCMoHAUzhzY3brdgtSIx
a2

gGveGAezZ38qKIU26dkz7deECY4vrsRkwhpTW0LGVCpjCQoaKvymAoCmAs8V2o
Mr

Ziw1YQ9u0UoWw0qm1wZqmVc0XvPIS2gWAs3fQ1WjH9hkcQTMsUaXQD0D0aqsY
3E

NqOvbCV1/oUpRi3076khCoAXI1bKSn/AvR3KDP14B5toHI/F50TSEiGhhHesgR

```
rs
  fBrpEY1IATtPq1taBZZogRqI3r0kkPk=
  -----END PRIVATE KEY-----
certificate: |
  -----BEGIN CERTIFICATE-----

MIIF5jCCA86gAwIBAgIJANq50IuwPFKgMA0GCSqGSIb3DQEBCwUAMIGGMQswCQ
YD

VQQGEwJHqjEQMA4GA1UECAwHRXJld2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZD
Eb

MBkGA1UECgwSbGlid2Vic29ja2V0cy10ZXN0MRIwEAYDVQQDDA1sb2NhbGhvc3
Qx

HzAdBgkqhkiG9w0BCQEWEG5vbmVAaW52YWxpZC5vcmcwIBcNMTgwMzIwMDQxNj
A3

WhgPMjExODAyMjQwNDE2MDdaMIGGMQswCQYDVQQGEwJHqjEQMA4GA1UECAwHRX
Jl

d2hvbjETMBEGA1UEBwwKQWxsIGFyb3VuZDEbMBkGA1UECgwSbGlid2Vic29ja2
V0

cy10ZXN0MRIwEAYDVQQDDA1sb2NhbGhvc3QxHzAdBgkqhkiG9w0BCQEWEG5vbm
VA

aW52YWxpZC5vcmcwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQCjYt
uW

aICCY0tJPubxpIgIL+WWmz/fmK8IQr11Wtee6/IUyUlo5I602mq1qcLhT/kmpo
R8

Di3DAmHKnSWdPWtn1BtXLERlLlUiHgZDrZWInmEBjKM1DZf+CvNGZ+EzPgBv5nT
ek

LWcfI5ZZtoGuIP1Dl/IkNDw8zFz4cpiMe/BFGemyxdHhLrKHSm8Eo+nT734tIt
nH

KT/m6DSU0x1Z13d6ehLRm7/+Nx47M3XMTRH5qKP/7TTE2s0U6+M0tsGI2zpRi+
m6

jzhNyMBTJ1u58qAe3ZW5/+YAiuZYAB6n5bhUp4oFuB5wYbcBywVR8ujInpF8bu
WQ

Ujy5N8pSNp7szdYsnLJpvAd0sibrNPjC0FQCnrpNjgJmIK3+mKk4kXX7ZTwefo
Az

TK412pHNuC53QVc/EF++GBLAXmvCDq9ZpMIYi70mzkkAKKC9Ue6Ef217LFQCFI
BK
```

Izv9cgi9fwPMLhrKleoVRNsecBsCP569WgJXhUnwf2lon4fEZr3+vRuc9shfq
nV0

nPN1IMSnzXCast7I2fiuRXdIz96KjlgQpP4XfNVA+RGL7aMnWOFIaVrKWLzAtg
zo

GMTvP/AuehKXncBJhYtW0ltTioVx+5yTYSAZWl+IssmXjefxJqYi2/7QWmv1QC
9p

sNcjTMaBQLN03T1Qelbs7Y27sxdEnNUth4kI+wIDAQABo1MwUTAdBgNVHQ4EFg
QU

9mYU23tW2zsomkKTAXarjr2vjuswHwYDVR0jBBgwFoAU9mYU23tW2zsomkKTAX
ar

jr2vjuswDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAGeANjIBMr
ow

YNCbhAJdP7dh1hT2RUFrdeRUJD0IxrH/hkvb6myHHnK8n0YezFPjUlmRKUGNED
uA

xbnXZzPdCRNV9V2mShbXvCyidY7WCQE2Bn44z2600uWVvk+7DNNLH9BnkWUtOnM
9P

wtmD9phWexm4q2GnTsiL6U16cy0Q1TJWKVLEUQQ6yda582e23J1AXqtqFcpfoE
34

H3afEiGy882b+ZBiwkeV+oq6XVF8sFyr9zYrv9CvWTY1kpTQfLTZSsgPdEHYVc
jv

xQ2D+XyDR0aRLR1vxUa9dHGFHLICG34Juq5Ai6lM1EsoD8HSsJpMcmrH7MWw2c
Kk

ujC3rMdFTtte83wF1uuF4FjUC72+SmcQN7A386BC/nk2TTsJawTDzqwOu/VdZv
2g

1WpTHlum1C1ZeP+G/jkSyDwqNnTu1aodDmUa4xZodfhP1HWPwUKFcq8oQr148Q
YA

A01bU0JQU7QwRWd1VbnwhDtQWXC92A2w1n/xkZSR1BM/NUSDhkBSUU1WjMbWg6
Gg

mnIZLRerQCu10ozr87r0QqQakPkyt8BUSNK3K42j2qcfhAONdR18Hq8Qs5pupy
+s

8sdCGDlwR3JNCMv6u480K87F4mcIxhkSefFJUFI25pCGN5WtE4p5l+9cn01Gr
IX

e2H1/7M0c/lbZ4FvXgARlex2rkgS0Ka06HE=
-----END CERTIFICATE-----

3. Go to **Settings > Security**
4. On the **Secure Connection** section, click **Details**
5. Select **Enable Certificate** , then select your certificate added in step 2.

Secure Connection



Information

It may take up to 30 seconds for certificate to be applied.
MetaDefender Distributed Cluster will not be accessible during the process.



Enable certificate

[* indicates required]

Select certificate*

My Cert



[Add Certificate](#)

[Cancel](#)

[Save changes](#)



Warning

Applying HTTPS settings may take some time. During this process, the MetaDefender Distributed Cluster Control Center web console will be temporarily unavailable.

Password policies

Password Policy settings are accessible under **Settings > Security** tab.



Info

These password policies changes only apply to new user creations and future password changes. Existing users' passwords are unaffected.

Local users' password can be enforced to meet requirements set by administrators, which includes following constraints:

- **Enforce password policy:**
 - Determines the number of unique new passwords that must be associated with a user account before an old password can be reused
 - Range: [0-24]
 - Default: 0 [to disable enforcement]
- **Minimum password length:**
 - The least number of characters that can make up a password for a user account
 - Range: [0-30]
 - Default: 0 [to disable enforcement]
- **Password must meet complexity requirements:**
 - Determines whether passwords must meet a series of guidelines that are considered important for a strong password.
 - Default: unchecked

Password policies

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

Passwords remembered [0-24]

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

☒ **Enforce min password length**
 Min password length [0-30]

Session policies

Administrators can enforce session policies for local users to ensure compliance with organizational requirements, using the following settings:

- **Enable idle session timeout:**
 - Idle timeout automatically terminates a user's session based on how long since their last recorded activity.
 - Default: 300 seconds.
- **Enable session timeout**
 - Absolute timeout terminates an individual user's session after a fixed duration, regardless of any user activity.
 - Default: 0 [to disable enforcement]
- **Allow Duplicate Sessions**
 - Permit the same user to log in and operate multiple sessions at once.
 - Default: Enabled.

- **Allow Cross IP Sessions**

- Permit requests from sources other than the authenticated origin.
- Default: Disabled.

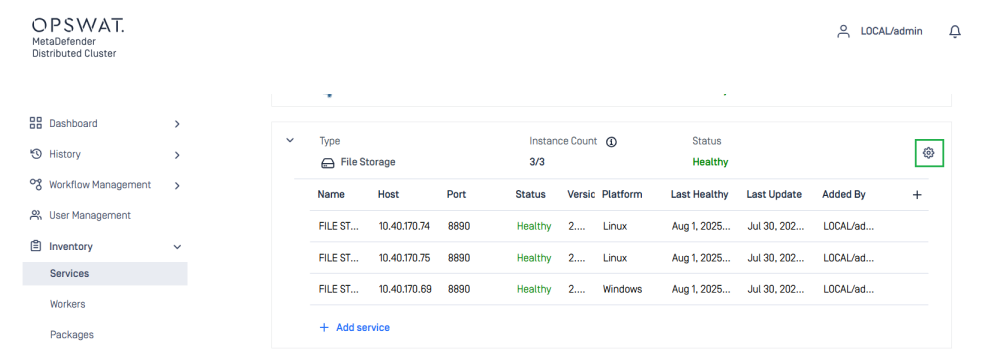
Session policies

☒	Enable idle session timeout Idle timeout to invalidate individual user's session based on that user last activity.	300 sec
☐	Enable session timeout Absolute timeout to invalidate individual user's session regardless of that user activities.	0 sec
☒	Allow Duplicate Sessions Allow same user to have multiple active sessions.	
☐	Allow Cross IP Sessions Allow requests coming from sources different from the authenticated origin.	

File Storage

MetaDefender Distributed Cluster (MDDC) introduces a built-in file storage server known as MDDC **File Storage**. The server stores and manages the live time of files and their duplications.

The administrator can set up MDDC to work with a single instance of MDDC **File Storage** or build a group of MDDC **File Storage** instances.



From `Inventory > Services` of the MDCC **Control Center** web console, the administrator can click on the gear icon in the top left corner of the `File Storage` group to access MDCC **File Storage** settings.

File Storage Settings

Min replica* ⓘ

1

Max replica* ⓘ

1

Clean up range*

Off

Data protection at rest* ⓘ

Salt

Cancel

Save

Multiple instances

When several instances of MDDC **File Storage** are added to the File Storage group, `Min Replica` and `Max Replica` enable the administrator to configure the operation of the storage group, as shown in the table below.

Setting	Behavior
Min replica = 1 Max replica = 1	Every file is stored without a backup across all File Storage servers. File Storage servers in the group implement a Sharding solution for file storage. Since there is no backup for any file, if one server in the group goes down, files managed by that server will be lost to the clients. This setup provides the best performance but also poses a high risk of data loss.
Min replica > 1 Max replica > Min replica	Every file is stored on at least Min replica number of File Storage servers and at most Max-replica number of servers. The setting provides High Availability support for File Storage. In most cases, Min replica and Max replica are configured to 2 and 3, creating a balance between performance and efficiency in High Availability.
Min replica > 1 Max replica = Min replica	Every file is fully stored on Max replica number of file storage servers and will not succeed if it can not be. This setting is the strictest among three options and should be considered carefully due to its impact on system performance.

Warning

Replication of a file across several MDDC File Storage servers significantly impacts the overall system performance. Hence, the number of replications must be evaluated thoughtfully.

Data retention

The administrator can configure data retention for files stored in the File Storage group with the Clean up range option. By default, this option is disabled. The administrator has the option to retain files for 12 hours, 1 day, 1 week, etc., starting from the present time.

12 hours

1 day

1 week

2 weeks

3 weeks

4 weeks

3 months

6 months

12 months

pff

Data protection at rest* ⓘ

Salt

CancelSave

i Info

Files eligible for retention include those produced by CDR, DLP, SBOM, Quarantine engines.

Package [.msi, .deb or .rpm] and module files are marked for cleanup manually and will never be affected by data retention.

Data protection

By default, all files stored on the MDDC File Storage server are XOR bitwise with a randomly generated binary string. This option is enabled by default to prevent the file from being executed successfully due to unexpected factors. The administrator can disable the option to optimize MDDC File Storage performance, though it may expose the system to security risks.

Data protection at rest* ⓘ

Salt



None

Salt



System Upgrade

A primary focus of MetaDefender Distributed Cluster (MDDC) is to reduce the disruption of file processing during system upgrades. The levels of impact on file processing during component updates are outlined as follows:

Component	Impact level	Upgrade method
MetaDefender Core	0	Deferral by MDDC Control Center
MetaDefender Distributed Cluster Worker	1	Manual by Installer file
MetaDefender Distributed Cluster API Gateway	1	Deferral by MDDC Control Center
MetaDefender Distributed Cluster Control Center	1	Manual by Installer file
MetaDefender Distributed Cluster Identity Service	1	Manual by Installer file
MetaDefender Distributed Cluster File Storage	2	Manual by Installer file

The potential impacts of each level are detailed below.

Level	Impact
0	The upgrade does not impact file processing.
1	The upgrade does not impact the processing of existing files but may affect the submission of new files, fetching scan result, downloading processed files, monitoring or management.
2	The upgrade needs the entire system to go down.

MetaDefender Distributed Cluster hosts a vast majority of **MetaDefender Core** instances. During the **MetaDefender Core** upgrade, each instance is upgraded sequentially to prevent interference with the system's file processing. Intrinsically, the **MetaDefender Core** upgrade is controlled and

managed strictly by MDDC **Worker** and MDDC **Control Center**. These services guarantee that the **MetaDefender Core** instance is safely isolated from new file submissions and continues processing until all files are finished on its end before the upgrade procedure takes place.

While not impacting file processing, the upgrade of MDDC **Worker** may cause a hiccup in reporting the status and resources consumed by the **MetaDefender Core** or MDDC **API Gateway** instance to MDDC **Control Center**. Consequently, users may slightly notice that one of their workers is occasionally missing from their dashboards.

During the MDDC **API Gateway** upgrade, clients may not be able to submit files, fetch scan statuses, or download processed files from the service. The deployment of multiple instances of the MDDC **API Gateway** should be considered to reduce service interruptions.

MDDC **Control Center** is designed for system administrators to manage and monitor operational services (MetaDefender Core, MDDC API Gateway, etc.), thus its upgrade solely affects the administrators and does not file processing.

Although file processing remains uninterrupted, the upgrade of MDDC **Identity Service** may affect the authentication of users accessing MDDC **Control Center**. It may also cause temporary failures in validating requests that contain API key header in MDDC **API Gateway**.

Most of services within the system establish connections to MDDC **File Storage**, thus its upgrade results in system downtime. Consequently, the upgrade of MDDC **File Storage** requires the system administrator to place the entire system in scheduled maintenance mode and this should be executed during a period when no files are sent for scanning.

Upgrade Methods

Manual by Installer file

Info

Applicable for MDDC **Control Center**, **Worker**, **Identity Service** and **File Storage**.

1. Download installer package from My OPSWAT.
2. Access the machine that hosts component service pending for upgrade.
3. Start **Command Prompt as Administrator on Windows** or **Terminal as Super user on Linux** and run one of the following commands:

```
bash
```

```
# Windows
> msixexec.exe /i <new_installer> /qn

# Debian or Ubuntu
$ sudo dpkg -i <new_installer> || sudo apt install -f

# Red Hat or Rocky
$ sudo yum install <new_installer> -y
```

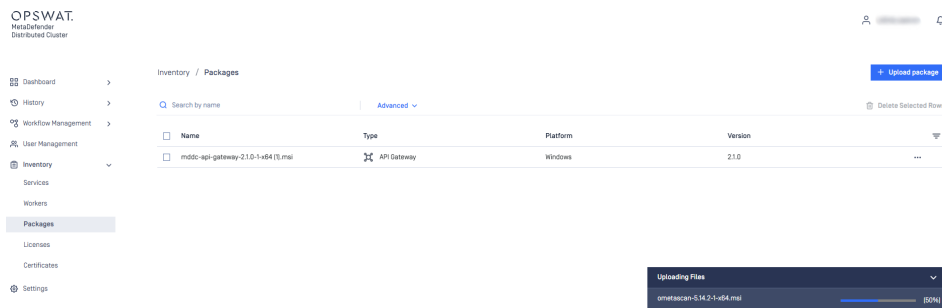
4. Confirm the service starts successfully.

Deferral by MetaDefender Distributed Cluster Control Center

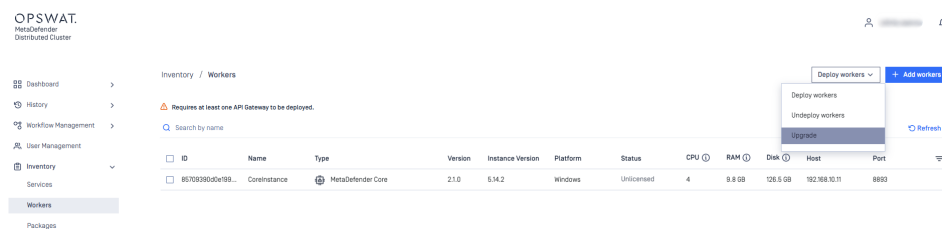
Info

Applicable for MDDC **API Gateway** and **MetaDefender Core**.

1. Download installer package from My OPSWAT.
2. Sign in to MDDC **Control Center** console with your administrator account.
3. Navigate to **Inventory** > **Packages**, click **Upload packages**.



4. Go to **Inventory** > **Worker**, click **Deploy workers** and select **Upgrade**.



5. Select the correct installer version and click **Upgrade**.

Upgrade

Warning!

If your system includes multiple instances running on different operating systems, ensure you have uploaded the appropriate installers for all those OS versions before proceeding.

API Gateway

Select a version for the upgrade

MetaDefender Core

5.15.0

Cancel

Upgrade

- Wait until the upgrade finishes and verify that the components have been upgraded to the correct versions.

OPSWAT.
MetaDefender
Distributed Cluster

Dashboard

History

Workflow Management

User Management

Inventory

Services

Workers

Packages

Licenses

Inventory / Workers

Deploy workers

Add workers

Requires at least one API Gateway to be deployed.

Search by name

Refresh

ID	Name	Type	Version	Instance Version	Platform	Status	CPU	RAM	Disk	Host	Port	
☐	85709390da799...	Containerstance	MetaDefender Core	2.1.0	5.14.2	Windows	Upgrading	4	9.8 GB	126.5 GB	192.168.10.11	8883

- Verify system health and status.

Info

During the upgrade process, modifications to the settings outlined below are prohibited:

- Workflow
- Module Update
- Health check

Upgrade procedures

Follow the steps to upgrade to MetaDefender Distributed Cluster v2.4.0

Performance and Load Estimation

Disclaimer

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

Factors that affect performance

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

Performance metrics

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

Performance metrics	Description
Number of processed objects per hour vs. Number of processed files per hour	On MetaDefender Core, meaning of “files” and “objects” are not the same. “files”: exclusively refers to original files submitted to MetaDefender Core. These could be either archive or non-archive file formats. For archives, depending on archive handling settings, MetaDefender Core may need to extract them and process all nested files inside as well. For example, one archive file could contain millions of nested files inside. “objects”: refers to any individual files that MetaDefender Core must process. These could be separate original files submitted to MetaDefender Core, or extracted files coming from an archive. The number of processed objects is considered to be a more accurate throughput metric to measure MetaDefender Core performance. The primary metric used to measure average vs peak throughput of a MetaDefender Core system is “processed objects per hour.”
Submission load (number of successful requests per second)	This performance metric measures the load generated by a test client application that simulates loads submitted to MetaDefender Core. A submission is considered successful when the client app submits a file to MetaDefender Core and receives a dataID, which indicates that the file has successfully been added to the Queue. Submission load should measure both average and peak loads.
Average processing time per object	The primary metric used to measure processing time of a MetaDefender Core system is “avg processing time (seconds/object).”

Performance metrics	Description
Total processing time [against certain data set]	Total processing time is a typical performance metric to measure the time it takes to complete the processing of a whole dataset.

How test results are calculated

Performance [mainly scanning speed] is measured by throughput rather than unit speed. For example, if it takes 10 seconds to process 1 object, and it also takes 10 seconds to process 10 objects, then performance is quantified as 1 second per object, rather than 10 seconds.

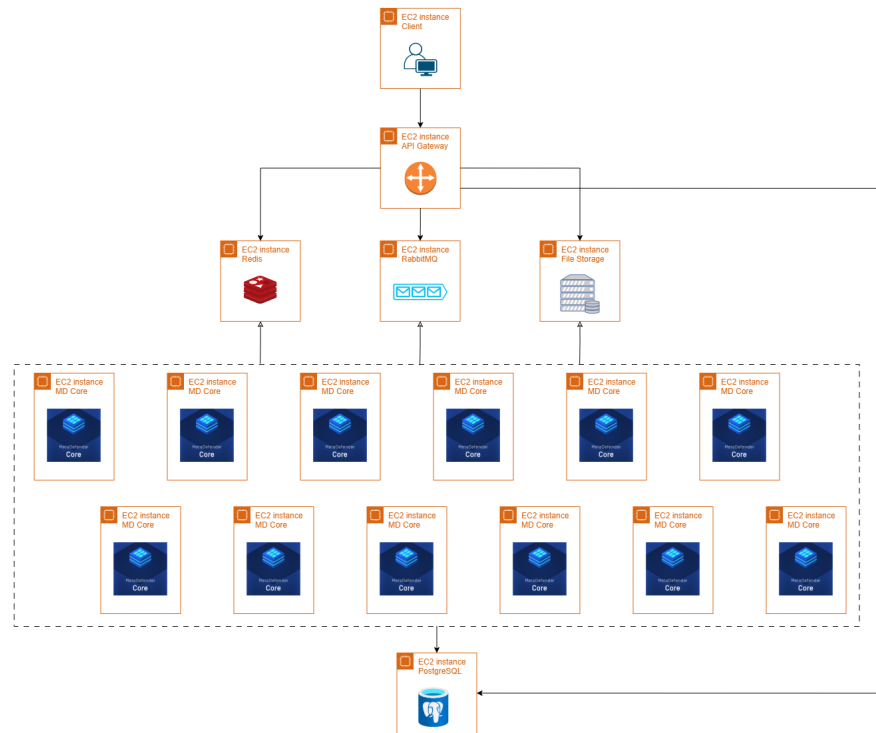
- total time / total number of objects processed: 10 seconds / 10 objects = 1 second / object.

Dataset

File category	File type	Number of files	Total size	Average file size
Document	DOC	3,820	534 MB	0.14 MB
Medium archive files	RPM CAB EXE	50	Compressed size: 2.8 GB Extracted size: 12.09 GB	Compressed size: 56.02 MB Extracted size: 0.036 MB
Big archive files	CAB	4	Compressed size: 2.9 GB Extracted size: 124 GB	Compressed size: 715 MB

Environment

Topology



Using AWS environment with the specification below:

MDDC system

	MD Core	File Storage	API Gateway	PostgreSQL	RabbitMQ	Redis
OS	Windows Server 2022	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9	Rocky Linux 9	Rocky Linux
AWS instance type	c5.2xlarge	c5n.4xlarge	c5n.2xlarge	c5.xlarge	c5.xlarge	c5.xlar
vCPU	8	16	4	4	4	4
Memory	16GB	32GB	8GB	8GB	8GB	32GB
Disk Type	gp3	gp3	gp3	gp3	gp3	gp3
IOPS	3000	12000	3000	10000	3000	3000
Throughput	125MB/s	1000MB/s	256MB/s	550MB/s	125MB/s	125MB/s
Size	100GB	150GB	100GB	100GB	80GB	80GB
Network bandwidth (baseline & burst)	2.5 Gbps 10 Gbps	15 Gbps 25 Gbps	5 Gbps 25 Gbps	1.25 Gbps 10 Gbps	1.25 Gbps 10 Gbps	1.25 Gbps 10 Gbps
Benchmark (Geekbench)	EC2 c5.2xlarge	EC2 c5n.4xlarge	EC2 c5n.2xlarge	EC2 c5.xlarge	EC2 c5.xlarge	EC2 c5.xlar

Client tool

Detail	
OS	Rocky Linux 9
AWS instance type	c5n.xlarge
vCPU	4
Memory	10GB
Disk	Type: gp3 IOPS: 3000 Throughput: 125MB/s Size: 80GB
Network bandwidth	Baseline: 5 Gbps Burst: 10 Gbps

Product information

- MetaDefender Core v5.14.2
- Engines:
 - Metascan 8: Ahnlab, Avira, ClamAV, ESET, Bitdefender, K7, Quick Heal, ViriT Explorer
 - Archive v7.4.0
 - File type analysis v7.4.0
- MDDC Control Center v2.0.0
- MDDC API Gateway v2.0.0
- MDDC File Storage v2.0.0
- PostgreSQL v14.17
- RabbitMQ v3.12.6
- Redis v7.2.1

MetaDefender Core settings

General settings

- Turn off data retention
- Turn off engine update
- Scan queue: 1000 [for Load Balancer deployment]

Archive Extraction settings

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

Metascan settings

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

Advanced settings

RabbitMQ

- RABBITMQ_SERVER_ADDITIONAL_ERL_ARGS=-rabbit consumer_timeout unlimited
default_consumer_prefetch {false,525}

Redis

- redis-cli flushall
- redis-cli config set save ""
- redis-cli config set maxmemory 25gb
- redis-cli config set maxmemory-policy volatile-ttl

Performance results

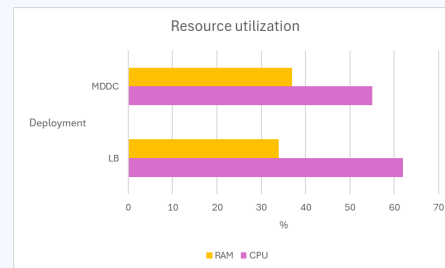
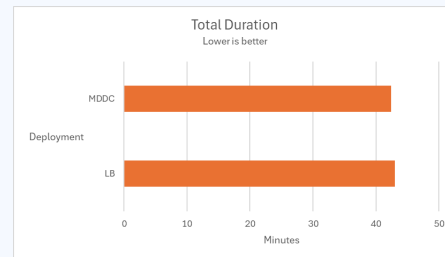
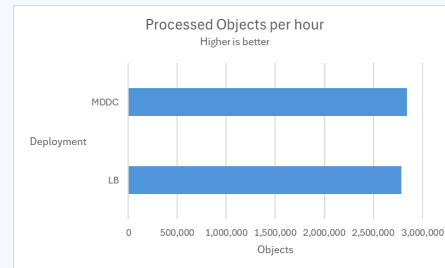
Load-balance deployment vs MDDC deployment

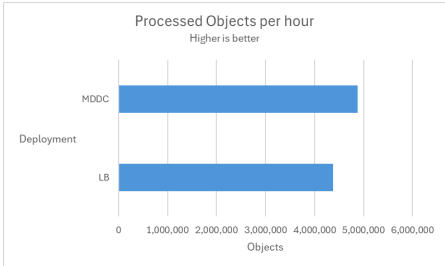
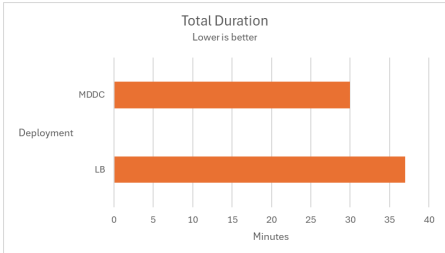
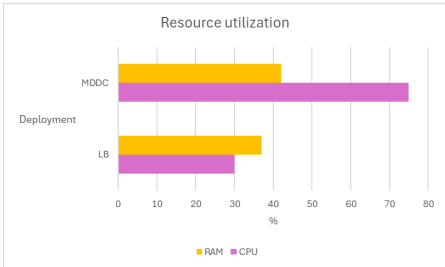
Multiple tests are conducted using 12 MetaDefender Core instances across two deployment types, MetaDefender Distributed Cluster (MDDC) and Load Balancer, to determine the superiority of the MDDC in 4 different datasets.

Scenario

Aggressively submitted 2M non-archive files
at a rate of 800 files per second.

Result

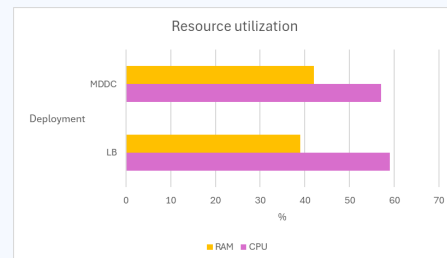
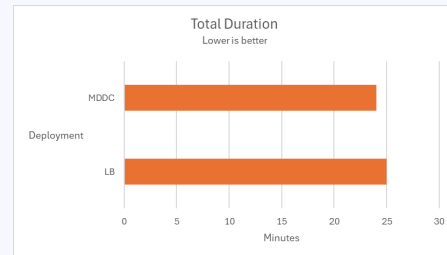
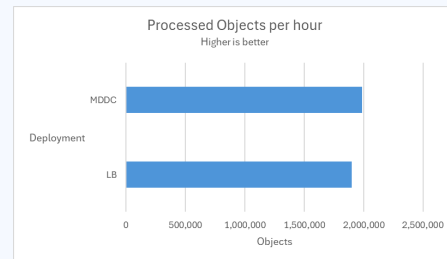


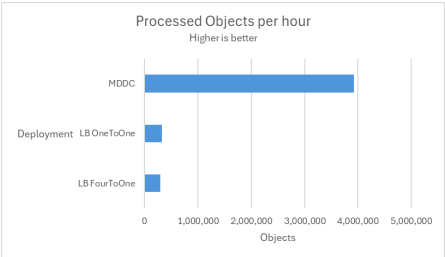
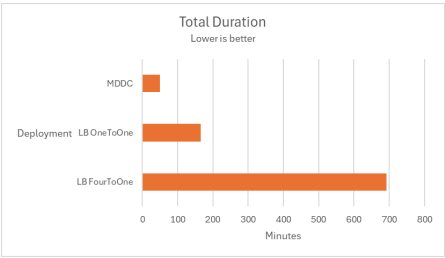
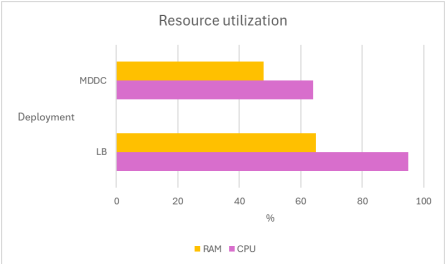
Scenario	Result																					
Submitted 400 medium archive files at a rate of 1 files per second.	Processed Objects per hour Higher is better <table><tr><th>Deployment</th><th>Objects</th></tr><tr><td>MDDC</td><td>4,800,000</td></tr><tr><td>LB</td><td>4,500,000</td></tr></table> Total Duration Lower is better <table><tr><th>Deployment</th><th>Minutes</th></tr><tr><td>MDDC</td><td>30</td></tr><tr><td>LB</td><td>35</td></tr></table> Resource utilization <table><tr><th>Deployment</th><th>RAM (%)</th><th>CPU (%)</th></tr><tr><td>MDDC</td><td>40</td><td>75</td></tr><tr><td>LB</td><td>35</td><td>30</td></tr></table>	Deployment	Objects	MDDC	4,800,000	LB	4,500,000	Deployment	Minutes	MDDC	30	LB	35	Deployment	RAM (%)	CPU (%)	MDDC	40	75	LB	35	30
Deployment	Objects																					
MDDC	4,800,000																					
LB	4,500,000																					
Deployment	Minutes																					
MDDC	30																					
LB	35																					
Deployment	RAM (%)	CPU (%)																				
MDDC	40	75																				
LB	35	30																				

Scenario

Submitted a mix of 189K non-archive and medium archive files at a rate of 180 files per second.

Result



Scenario	Result																												
Submitted 4 large CAB files. The scenarios replicate 2 different routing cases of a common Load Balancer. LB OneToOne: An ideal routing ensures that one CAB file is routed to a single MD Core. LB FourToOne: The worst routing that delivered four CAB files to a single MD Core. # Archive distribution In workflow, setting "Load shared among MetaDefender Core instances for archive processing" is enabled. ☒ Load shared among MetaDefender Core instances for archive processing Applicable to Distributed Cluster deployment, nested files in archive could be processed in multiple MetaDefender Core instances, recommended when processing mostly big archive files.	Processed Objects per hour Higher is better <table border="1"><thead><tr><th>Deployment</th><th>Processed Objects per hour (Objects)</th></tr></thead><tbody><tr><td>MDDC</td><td>~4,000,000</td></tr><tr><td>Deployment LB OneToOne</td><td>~500,000</td></tr><tr><td>LB FourToOne</td><td>~500,000</td></tr></tbody></table> Total Duration Lower is better <table border="1"><thead><tr><th>Deployment</th><th>Total Duration (Minutes)</th></tr></thead><tbody><tr><td>MDDC</td><td>~50</td></tr><tr><td>Deployment LB OneToOne</td><td>~150</td></tr><tr><td>LB FourToOne</td><td>~650</td></tr></tbody></table> Resource utilization <table border="1"><thead><tr><th>Deployment</th><th>RAM (%)</th><th>CPU (%)</th></tr></thead><tbody><tr><td>MDDC</td><td>~48</td><td>~65</td></tr><tr><td>Deployment</td><td>~65</td><td>~95</td></tr><tr><td>LB</td><td>~65</td><td>~95</td></tr></tbody></table>	Deployment	Processed Objects per hour (Objects)	MDDC	~4,000,000	Deployment LB OneToOne	~500,000	LB FourToOne	~500,000	Deployment	Total Duration (Minutes)	MDDC	~50	Deployment LB OneToOne	~150	LB FourToOne	~650	Deployment	RAM (%)	CPU (%)	MDDC	~48	~65	Deployment	~65	~95	LB	~65	~95
Deployment	Processed Objects per hour (Objects)																												
MDDC	~4,000,000																												
Deployment LB OneToOne	~500,000																												
LB FourToOne	~500,000																												
Deployment	Total Duration (Minutes)																												
MDDC	~50																												
Deployment LB OneToOne	~150																												
LB FourToOne	~650																												
Deployment	RAM (%)	CPU (%)																											
MDDC	~48	~65																											
Deployment	~65	~95																											
LB	~65	~95																											

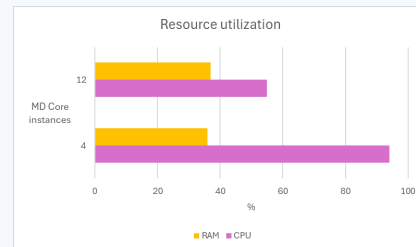
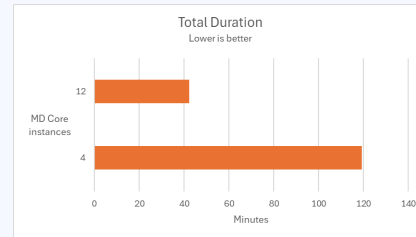
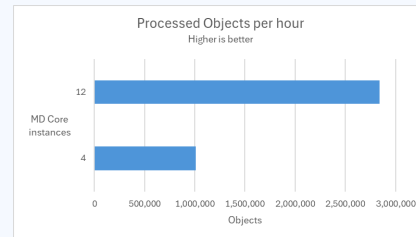
Scaling out

In the following test scenarios, we conducted experiments on four datasets using 4 and 12 of MD Core instances in MetaDefender Distributed Cluster (MDDC), demonstrating the benefits of increased instance counts.

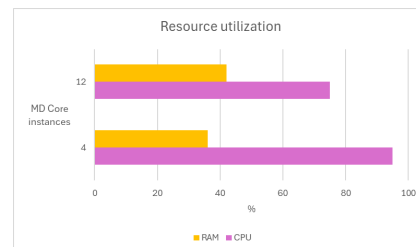
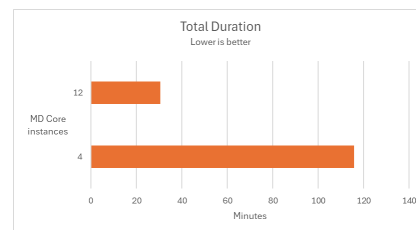
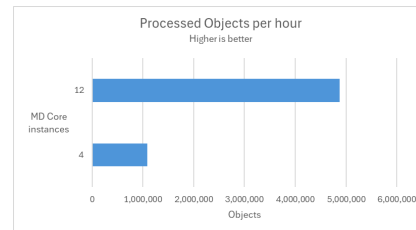
Scenario

Result

Aggressively submitted 2M non-archive files at a rate of 800 files per second.



Submitted 400 medium archive files at a rate of 1 files per second.



Scenario	Result																					
Submitted a mix of 189K non-archive and medium archive files at a rate of 60 files per second.	Processed Objects per hour Higher is better <table><tr><th>MD Core instances</th><th>Objects</th></tr><tr><td>12</td><td>~900,000</td></tr><tr><td>4</td><td>~500,000</td></tr></table> Total Duration Lower is better <table><tr><th>MD Core instances</th><th>Minutes</th></tr><tr><td>12</td><td>~55</td></tr><tr><td>4</td><td>~85</td></tr></table> Resource utilization <table><tr><th>MD Core instances</th><th>RAM (%)</th><th>CPU (%)</th></tr><tr><td>12</td><td>~45</td><td>~30</td></tr><tr><td>4</td><td>~40</td><td>~85</td></tr></table>	MD Core instances	Objects	12	~900,000	4	~500,000	MD Core instances	Minutes	12	~55	4	~85	MD Core instances	RAM (%)	CPU (%)	12	~45	~30	4	~40	~85
MD Core instances	Objects																					
12	~900,000																					
4	~500,000																					
MD Core instances	Minutes																					
12	~55																					
4	~85																					
MD Core instances	RAM (%)	CPU (%)																				
12	~45	~30																				
4	~40	~85																				
Submitted 4 large CAB files. Archive distribution In workflow, setting "Load shared among MetaDefender Core instances for archive processing" is enabled. ☒ Load shared among MetaDefender Core instances for archive processing Applicable to Distributed Cluster deployment, nested files in archive could be processed in multiple MetaDefender Core instances, recommended when processing mostly big archive files.	Processed Objects per hour Higher is better <table><tr><th>MD Core instances</th><th>Objects</th></tr><tr><td>12</td><td>~4,000,000</td></tr><tr><td>4</td><td>~1,500,000</td></tr></table> Total Duration Lower is better <table><tr><th>MD Core instances</th><th>Minutes</th></tr><tr><td>12</td><td>~50</td></tr><tr><td>4</td><td>~180</td></tr></table> Resource utilization <table><tr><th>MD Core instances</th><th>RAM (%)</th><th>CPU (%)</th></tr><tr><td>12</td><td>~55</td><td>~65</td></tr><tr><td>4</td><td>~55</td><td>~85</td></tr></table>	MD Core instances	Objects	12	~4,000,000	4	~1,500,000	MD Core instances	Minutes	12	~50	4	~180	MD Core instances	RAM (%)	CPU (%)	12	~55	~65	4	~55	~85
MD Core instances	Objects																					
12	~4,000,000																					
4	~1,500,000																					
MD Core instances	Minutes																					
12	~50																					
4	~180																					
MD Core instances	RAM (%)	CPU (%)																				
12	~55	~65																				
4	~55	~85																				

Log Gathering in MetaDefender Distributed Cluster

Download support packages

From the web console of MetaDefender Distributed Cluster (MDDC), the administrator can easily download the support packages of the following services:

- MDDC **Control Center**
- MDDC **Identity Service**
- MDDC **File Storage**
- MDDC **Worker** including MDDC **API Gateway** or **MetaDefender Core** deployed by the worker.

Please refer to Remote Support Package Gathering for more information.

Collect service logs

Logs from the services Redis, RabbitMQ, and PostgreSQL need to be collected manually.

Redis - Caching Server

Info

Redis caching server is officially supported on Linux.

1. Run Terminal as root privilege [`sudo`].
2. Open Redis config file `/etc/redis/redis.conf` in edit mode e.g.:

bash

```
$ vi /etc/redis/redis.conf
```

3. Find and replace `logfile` directive with your desired location.

bash

```
logfile "<path/to/your/redis/log>.log"
```

4. Save the file, and restart Redis daemon.

bash

```
$ sudo systemctl restart redis
```

5. Find and collect Redis log `<path/to/your/redis/log>.log`

RabbitMQ - Message Broker Server

Windows

1. Locate and collect RabbitMQ log files that match the pattern
`%APPDATA%\RabbitMQ\log\rabbit@<computer name>.log`.
2. Locate and collect RabbitMQ upgrade log files that match the pattern
`%APPDATA%\RabbitMQ\log\rabbit@<computer name>_upgrade.log`.

Linux

1. Run terminal as root privilege [sudo].
2. Run following command to retrieve RabbitMQ log location:

bash

```
$ rabbitmq-diagnostics -q log_location
```

3. Access RabbitMQ log folder and find log files:
 - `rabbit@<computer name>.log`
 - `rabbit@<computer name>_upgrade.log`

PostgreSQL - Database Server

Windows

1. Locate and collect log files that match the pattern `C:\Program Files\PostgreSQL\12\data\log` with names `postgresql-<yyyy-mm-dd>_<HHMMSS>.log`

Linux

1. Run terminal as root privilege [sudo].
2. Open the PostgreSQL config file `/etc/postgresql/12/main/postgresql.conf` in edit mode e.g.:

bash

```
$ vi /etc/postgresql/12/main/postgresql.conf
```

3. Find and turn `logging_collector` directive on :

bash

```
logging_collector = on
```

4. Save the file and restart PostgreSQL daemon, e.g.:

bash

```
$ sudo systemctl restart postgresql
```

5. Locate and collect log files that match the pattern

```
/var/lib/postgresql/12/main/log/postgresql-<yyyymmdd>_<HHMMSS>.log.
```

Release notes

Version	2.4.0
Release date	30 September 2025
Scope	Support <code>/readyz</code> API endpoint, export scan result in JSON format, export processing history in STIX or CSV format, support grouping and removing abandoned engine packages, fix eventual crashes of services

New Features, Improvements and Enhancements

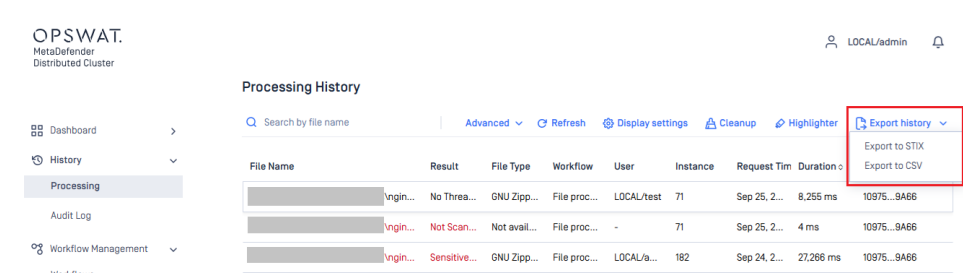
Export scan result in JSON format

From MetaDefender Distributed Cluster (MDDC) Control Center, users can export scan result in JSON format.



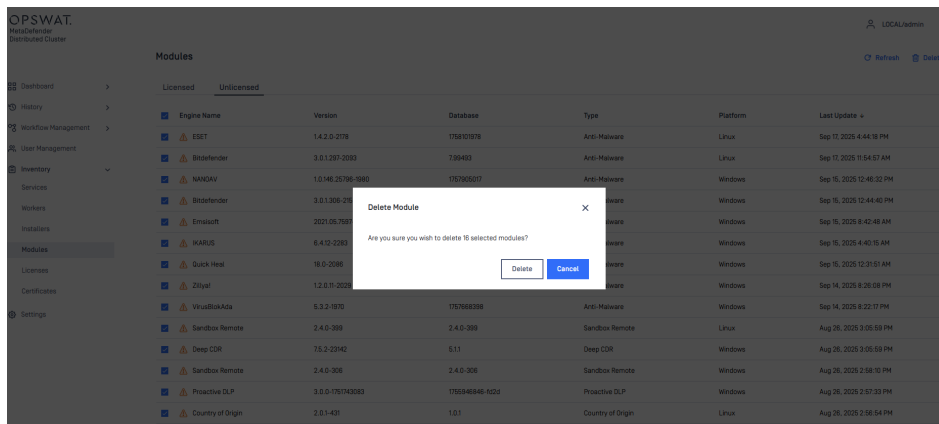
Export processing history in STIX or CSV format

Processing history can be exported in STIX or CSV format from MDDC Control Center.



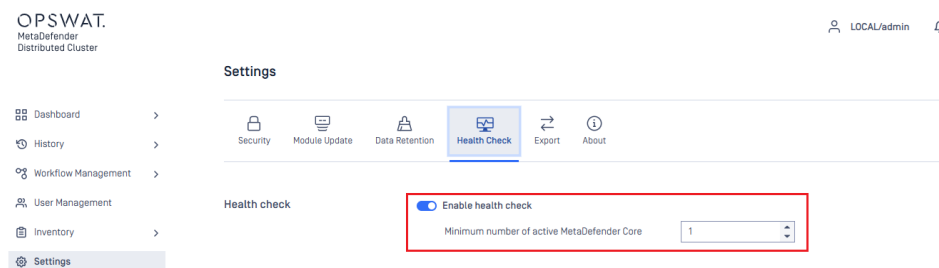
Remove abandoned module packages

Abandoned module packages can be selected and removed on web console of Control Center.



Customize the system health check

System administrators can enable the health check option and set the minimum number of required MetaDefender Core instances in the Health Check settings of the MDDC Control Center.



RESTful API

- Introduce a new API endpoint in MetaDefender Distributed Cluster API Gateway to verify if the system is ready for new scan requests `GET /readyz`.
- Introduce a new field, `dlp_wait_time`, in the response of `GET /file/{data_id}` API requested from MetaDefender Distributed Cluster API Gateway.
- Include `username` field in the response of `GET /file/{data_id}`, `GET /file/batch/{batch_id}` and `GET /hash/{md5|sha1|sha256|sha512}`.

Further Enhancements

- Verify the minimum version requirement when adding a new instance of Redis, RabbitMQ, and PostgreSQL to MetaDefender Distributed Cluster Control Center.
- Improve storing scan results from AV engines to MetaDefender Distributed Cluster Data Lake.

Security Enhancements

Upgraded library for vulnerability fixes:

- OpenSSL 3.5.2

Bug Fixes

- Fixed the issue that caused occasional service crashes when halted.
- Fixed the issue that made it impossible to close a batch if its name contained special characters.
- Fixed the issue that led to the batch name not appearing in the UI of MDDC Control Center.
- Fixed the issue that caused the COO engine to fail or time out during installation.
- Fixed the issue that caused the executive report to eventually miss data.

Known Limitations

Details	
MetaDefender Core becomes unlicensed following the MDDC Worker upgrade	This issue will be resolved in MDDC version 2.5.0. In version 2.4.0, MetaDefender Core instance that has already deployed and activated successfully with a valid license becomes unlicensed after its MDDC Worker is upgraded. Workaround: • To online activation: follow steps to activate a MetaDefender Core instance after deployment. • To offline activation: follow steps to activate a MetaDefender Core instance with with license file.

API Reference

API Gateway

API Version: v2.4.0

Developer Guide

This is the API documentation for *MetaDefender Distributed Cluster API Gateway Public API*. If you would like to evaluate or have any questions about this documentation, please contact us via our [Contact Us](#) form.

How to Interact with MetaDefender Distributed Cluster API Gateway using REST API

MetaDefender Distributed Cluster API Gateway is used to submit files for analysis, retrieve scan results, manage file processing, download processed files, and manage file batches. OPSWAT recommends using the JSON-based REST API. The available methods are documented below.

Note: MetaDefender Distributed Cluster API doesn't support chunk upload, however is recommended to stream the files to MetaDefender Distributed Cluster API Gateway as part of the upload process.

File Analysis Process

MetaDefender Distributed Cluster is a system with multiple components that work together to utilize the power of multiple MetaDefender Core instances. The system is designed to handle large volumes of files and provide high throughput for file analysis. The system can be deployed in a distributed manner, allowing for horizontal scaling and load balancing across multiple MetaDefender Core instances.

Below is a brief description of the API integration flow:

1. Upload a file for analysis to MetaDefender Distributed Cluster API Gateway (POST /file), which returns the data_id: [File Analysis](#).
2. The following method can be used to retrieve the analysis report:
 - **Polling:** Fetch the result with previously received data_id (GET /file/{data_id} resource) until scan result belonging to data_id doesn't reach the 100 percent progress_percentage: ([Fetch analysis result](#))

Note: Too many data_id requests can reduce performance. It is enough to just check every few hundred milliseconds.

3. Retrieve the analysis results anytime after the analysis is completed with hash for files

(md5, sha1, sha256, sha512) by calling [Fetch analysis result by hash](#).

- The hash can be found in the scan results

4. Retrieve processed file (sanitized, redacted, watermarked, etc.) after the analysis is complete.

Note: Based on the configured retention policy, the files might be available for retrieval at a later time.

OPSWAT provides some sample codes on [GitHub](#) to make it easier to understand how the MetaDefender REST API works.

CONTACT

NAME: API Support

EMAIL: feedback@opswat.com

URL: <https://github.com/OPSWAT/metadefender-core-openapi3>

Terms of service: <https://onlinehelp.opswat.com/policies/>

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
apikey	apiKey	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

API

1. ANALYSIS

File analysis APIs

Submit each file to MetaDefender Distributed Cluster API Gateway individually or group them in batches. Each file submission will return a `data_id` which will be the unique identifier used to retrieve the analysis results.

Note: MetaDefender API doesn't support chunk upload. You shouldn't load the file in memory, is recommended to stream the files to MetaDefender Distributed Cluster API Gateway as part of the upload process.

1.1 POST /file

Analyze File (Asynchronous mode)

Scanning a file using a specified workflow. Scan is done asynchronously and each scan request is tracked by data id of which result can be retrieved by API Fetch Scan Result.

Note: Chunked transfer encoding (applying header Transfer-Encoding: Chunked) is **not supported** on /file API.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
------	------	---------	-------------

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
filename	string		The name of the submitted file
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
batch	string		Batch id to scan with, coming from `Initiate Batch` (If it is not given, it will be a single file scan.)
archivepwd	string		Password for archive (URL encoded UTF-8 string) Multiple passwords is also supported, format: archivepwdX * X: Could be empty * When having value, X must be a number >= 1 For example: * archivepwd1: "fox" * archivepwd2: "cow" * archivepwd3: "bear"
content-encoding	string		Content encoding of the file. This header is used to specify the encoding of the file content. The value should be a valid content encoding type, such as "base64", "gzip". This header is optional and can be omitted if the encoding is not applicable.
metadata	json		Could be utilized for: * Additional parameter for pre-defined post actions and external scanners (as a part of STDIN input). * Customized macro variable for watermarking text (Proactive DLP engine feature). * Additional context / verbose information for each file submission (appended into JSON response scan result). It is strongly recommended to apply URL encoding before sending `metadata` to Metadefender Core to prevent unexpected issues related to encoding errors or unsafe characters.

NAME	TYPE	EXAMPLE	DESCRIPTION
engines-metadata	json		Since MetaDefender Core 5.0.0, preferred context / verbose information can be sent to the engines. Please see the below pages for the details: * [File Type engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata) (supported since Core 5.2.1) * [Archive engine](https://docs.opswat.com/mdcore/utilities-engines/supported-engines-metadata-header) (supported since Core 5.4.1) * [Deep CDR](https://docs.opswat.com/mdcore/deep-cdr/supported-engines-metadata-json) * [Proactive DLP](https://docs.opswat.com/mdcore/proactive-dlp/supported-engines-metadata-json)
global-timeout	integer		This custom global timeout (in seconds) will override the global timeout predefined in corresponding workflow rule.

RESPONSE

STATUS CODE - 200: Successful file submission

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id*	string	Unique submission identifier. Use this value to reference the submission.

EXAMPLE:

```
{
  "data_id": "61dffea728844adbf49eb090e4ece0e"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 411: Content-Length header is missing from the request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
  "err": "Missing Content-Length header."
}
```

STATUS CODE - 422: Body input is empty.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
  "err": "File is empty."
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

STATUS CODE - 503: Server is too busy. Try again later.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

EXAMPLE:

```
{
  "err": "Server is too busy. Try again later."
}
```

1.2 GET /file/{data_id}

Fetch Analysis Result

Retrieve scan results.

Scan is done asynchronously and each scan request is tracked by a data ID.

Initiating file scans and retrieving the results need to be done using two separate API calls. This request needs to be made multiple times until the scan is complete. Scan completion can be traced using `scan_results.progress_percentage` value from the response.

Note: The REST API also supports pagination for archive file result. A completed response description with archive detection:

- `extracted_files`: information about extracted files
 - `files_extracted_count`: the number of extracted files
 - `files_in_archive`: array of files in archive
 - `detected_by`: number of engines reported threat
 - `scanned_with`: number of engines used for scanning the file
- `first_index`: it tells that from which file (index of the file, 0 is the first) the result JSON contains information about extracted files. (default=0)
- `page_size`: it tells how many files the result JSON contains information about (default=50). So by default, the result JSON contains information about the first 50 extracted files.
- `worst_data_id`: data id of the file that has the worst result in the archive
- `scan_results`
 - `last_file_scanned` (stored only in memory, not in database): If available, the name of the most recent processed file

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer		The first item order in the list child files of archive file
size	integer		The number of items to be fetched next, counting from the item order indicated in first header

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.

RESPONSE

STATUS CODE - 200: Entire analysis report generated by MetaDefender Core

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	data identifier of the requested file
dlp_info	object	
certainty	enum	ALLOWED: Very Low, Low, Medium, High, Very High Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's workflow rule)

NAME	TYPE	DESCRIPTION
hits	object	
ccn	object	
display_name	string	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	array	
after	string	The context after the matched data.
before	string	The context before the matched data.
certainty	enum	ALLOWED: Very Low, Low, Medium, High, Very High The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	ALLOWED: 0, 1 (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
metadata_removal	object	
result	enum	ALLOWED: removed, not removed, failed to remove Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
redact	object	
result	enum	ALLOWED: redacted, not redacted, failed to redact Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
severity	enum	ALLOWED: 0, 1 (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	enum	ALLOWED: 0, 1, 2, 3, 4 The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned

NAME	TYPE	DESCRIPTION
watermark	object	
result	enum	ALLOWED: added, not added, failed to add Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	object	
error_detail	string	Revealed detailed reason why the download failed.
progress	number	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [1, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`) if the download stopped unexpectedly.

NAME	TYPE	DESCRIPTION
status	string	Indicates download status, which could be either - `Downloading` - Check `progress` key value for actual download percentage <pre> '''json "download_info": { "progress": 7, "status": "Downloading", "url": "http://192.168.200.97:8080/5gb.zip" } ''' </pre> - `Download Success` - Check `error_detail` key value for an error explanation <pre> '''json "download_info": { "status": "Download Success", "url": "https://secure.eicar.org/eicar.com" } ''' </pre> - `Download Failed` - Check `error_detail` key value for an error explanation <pre> '''json "download_info": { "error_detail": "Connection error", "status": "Download Failed", "url": "http://192.168.200.97:8080/2gb.zip" } ''' </pre> - `Download Timeout` - Expecting to occur when the download progress takes longer than what time window allowed in MetaDefender Core's pre-configured setting under workflow rule (under "SCAN" tab) <pre> '''json "download_info": { "status": "Download Timeout", "url": "http://192.168.200.97:8080/2gb.zip" } ''' </pre> - `Download Cancelled` - Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to <pre> '''json "download_info": { "status": "Download Cancelled", "url": "http://192.168.200.97:8080/5gb.zip" } ''' </pre>
url	string	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	object	
decrypted_status	enum	ALLOWED: Success, Failed Indicate that decryption phase is successful or not.
err_category	string	Error category
err_code	integer	Error code
err_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.

NAME	TYPE	DESCRIPTION
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	

NAME	TYPE	DESCRIPTION
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	ALLOWED: allowed, blocked Final verdict of the file type analysis.
verdict_explanation*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
opswatfilescan_info	object	
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
av_scan_time	integer	AV engines' processing time.
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product

NAME	TYPE	DESCRIPTION
parse_dgsg_time	integer	Digital signature analyzing time.
vul_time	integer	Vulnerability engine's lookup time.
yara_time	integer	YARA engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by " ".
actions_ran	string	List of successful actions, separated by " ". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	ALLOWED: sanitized, removed The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
details	object	
action	enum	ALLOWED: sanitized, removed The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.

NAME	TYPE	DESCRIPTION
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	ALLOWED: Sanitized, Sanitized failed, Sanitized skipped The result of the CDR process. - **Sanitized** : the file was successfully sanitized. - **Sanitized failed** : the file could not be sanitized due to an error during the process. - **Sanitized skipped** : the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MetaDefender Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.
verdict	enum	ALLOWED: blocked, allowed The verdict of the CDR process. - **blocked** : the file is recommended for blocking by Deep CDR. - **allowed** : the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
scan_results	object	
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by C00, Blocked Verdict by C00, Unknown Verdict by C00, In Progress, Skip Processing Fast SymLink The overall scan result as string

NAME	TYPE	DESCRIPTION
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: 50700 The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained

NAME	TYPE	DESCRIPTION
severity	enum	ALLOWED: LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor
availability-impact	string	A CVSS availability impact descriptor
confidentiality-impact	string	A CVSS confidentiality impact descriptor
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time
integrity-impact	string	A CVSS integrity impact descriptor
score	string	A CVSS 10-point severity score
source	string	A CVSS source descriptor
cwe	string	A CWE group identification string
last_modified_epoch	string	An epoch timestamp indicating source last update time

NAME	TYPE	DESCRIPTION																					
published-epoch	string	An epoch timestamp indicating source publishing time																					
references	array																						
severity	enum	ALLOWED: LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`																					
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown																					
static_id	integer	An OPSWAT identifier for the vulnerability																					
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out																					
yara	object																						
hits	object																						
verdict	enum	ALLOWED: 0, 1, 2, 3, 4 The overall result for the analyzed file. Value will be one of the following: <table><tr><td> index</td><td> status</td><td> </td></tr><tr><td> ----- </td><td>----- </td><td> </td></tr><tr><td> 0</td><td> Clean</td><td> </td></tr><tr><td> 1</td><td> Found matched data</td><td> </td></tr><tr><td> 2</td><td> Suspicious</td><td> </td></tr><tr><td> 3</td><td> Failed</td><td> </td></tr><tr><td> 4</td><td> Not scanned</td><td> </td></tr></table>	index	status		-----	-----		0	Clean		1	Found matched data		2	Suspicious		3	Failed		4	Not scanned	
index	status																						
-----	-----																						
0	Clean																						
1	Found matched data																						
2	Suspicious																						
3	Failed																						
4	Not scanned																						

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.3 GET /hash/{md5|sha1|sha256|sha512}

Fetch Analysis Result By Hash

Retrieve analysis result by hash

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*md5 sha1 sha256 sha512	string		Hash value to search. This can be md5, sha1, sha256, sha512

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer		The first item order in the list child files of archive file
size	integer		The number of items to be fetched next, counting from the item order indicated in first header

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)

NAME	TYPE	EXAMPLE	DESCRIPTION
selfonly	boolean		Useful to archive hash lookup. Allow specifying to only perform hash lookup against the original archive file self only, and skip searching all child files result within the original archive. Default value is false.
timerange	integer		Scoping down the recent number of hours that hash lookup task should start from till now, instead of searching the entire scan history in MetaDefender Core database. Default value is 0. That means no time scope.
include-inprogress	boolean		False (default): API will return "Not Found" if the verdict is in progress. True: If the queried hash has a completed processing result before, API will return the completed processing result. If this hash doesn't have any completed processing result, API will return this In-progress result.

RESPONSE

STATUS CODE - 200: Get information of file

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
data_id	string	data identifier of the requested file
dlp_info	object	
certainty	enum	ALLOWED: Very Low, Low, Medium, High, Very High Describes how certain the hit is, possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
errors	object	
filename	string	Output processed file name (pre-configured on engine settings under Core's workflow rule)
hits	object	
ccn	object	

NAME	TYPE	DESCRIPTION
display_name	string	Credit Card Number, Social Security Number, or in case of RegEx, the name of the rule that has been given by the user
hits	array	
after	string	The context after the matched data.
before	string	The context before the matched data.
certainty	enum	ALLOWED: Very Low, Low, Medium, High, Very High The text version of "certainty_score", possible values: * `Very Low` * `Low` * `Medium` * `High` * `Very High`
certainty_score	integer	Is defined by the relevance of the given hit in its context. It is calculated based on multiple factors such as the number of digits, possible values: [0-100]
hit	string	The matched data.
location	string	The location of the hit that is found in a file.
severity	enum	ALLOWED: 0, 1 (NOTE: this field is deprecated): can be 0 (detected) or 1 (suspicious).
tryRedact	boolean	If file was redacted or not.
metadata_removal	object	
result	enum	ALLOWED: removed, not removed, failed to remove Result of the metadata removal process, possible values: * `removed` * `not removed` * `failed to remove`
redact	object	
result	enum	ALLOWED: redacted, not redacted, failed to redact Result of the redaction process, possible values: * `redacted` * `not redacted` * `failed to redact`
severity	enum	ALLOWED: 0, 1 (NOTE: this field is deprecated): represents the severity of the data loss, possible values: * `0` - Certainly is data loss * `1` - Might be data loss
verdict	enum	ALLOWED: 0, 1, 2, 3, 4 The overall result for the scanned file. Possible values: * `0` - Clean * `1` - Found matched data * `2` - Suspicious * `3` - Failed * `4` - Not scanned
watermark	object	

NAME	TYPE	DESCRIPTION
result	enum	ALLOWED: added, not added, failed to add Result of the watermarking process, possible values: * `added` * `not added` * `failed to add`
download_info	object	
error_detail	string	Revealed detailed reason why the download failed.
progress	number	Only applicable when "status" is `Downloading`, indicates download finished percentage, in a range of [1, 99]. * Once hitting 100, the status will be changed to `Download Success`. * or other problematic status (`Download Cancelled`, `Download Failed`) if the download stopped unexpectedly.
status	string	Indicates download status, which could be either - `Downloading` - Check `progress` key value for actual download percentage <pre> '''json "download_info": { "progress": 7, "status": "Downloading", "url": "http://192.168.200.97:8080/5gb.zip" } ''' </pre> - `Download Success` <pre> '''json "download_info": { "status": "Download Success", "url": "https://secure.eicar.org/eicar.com" } ''' </pre> - `Download Failed` - Check `error_detail` key value for an error explanation <pre> '''json "download_info": { "error_detail": "Connection error", "status": "Download Failed", "url": "http://192.168.200.97:8080/2gb.zip" } ''' </pre> - `Download Timeout` - Expecting to occur when the download progress takes longer than what time window allowed in MetaDefender Core's pre-configured setting under workflow rule (under "SCAN" tab) <pre> '''json "download_info": { "status": "Download Timeout", "url": "http://192.168.200.97:8080/2gb.zip" } ''' </pre> - `Download Cancelled` - Expecting to occur when user explicitly cancelled that file scan request, or batch request that the scan belongs to <pre> '''json "download_info": { "status": "Download Cancelled", "url": "http://192.168.200.97:8080/5gb.zip" } ''' </pre>

NAME	TYPE	DESCRIPTION
url	string	Original download link which was specified in HTTP(S) request's `downloadfrom` header
extraction_info	object	
decrypted_status	enum	ALLOWED: Success, Failed Indicate that decryption phase is successful or not.
err_category	string	Error category
err_code	integer	Error code
err_details	string	Error message
is_encrypted_file	boolean	Indicate if file is password-protected or not.
file_info	object	
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
md5	string	File's MD5 hash.
sha1	string	File's SHA1 hash.
sha256	string	File's SHA256 Hash.
sha512	string	File's SHA512 Hash.
signer_infos	array	
digest_algorithm	string	Digest algorithm.
digest_encryption_algorithm	string	Encryption algorithm.
issuer	string	Entity that develops and registers the certificate.
serial_number	string	Serial number of the certificate.
vendor_name	string	Entity that is issued a certificate and utilize it for creating a digital signature.
version	string	Version of X.509 that is used in the certificate. This version field is zero-based. * 0: v1 * 1: v2 * 2: v3
type_category	array	
receive_data_timestamp	string	The timestamp when upload progress started (first byte received) (in milliseconds)
upload_time	integer	Total time elapsed for upload process (in milliseconds).
upload_timestamp	string	The timestamp when upload progress finished (all bytes received) (in milliseconds)
filetype_info	object	
file_info*	object	

NAME	TYPE	DESCRIPTION
description*	string	File type description
detected_by	string	Analyzer that detected the file type
encrypted*	boolean	File is password-protected or not
extensions*	string	File type extension
groupID*	string	File type category
groupIDs*	array	
group_description	string	File type category description
likely_type_ids	array	
score*	integer	Likelihood score of the file type
typeID*	string	File type ID
type*	string	MIME type
typeID*	string	File type ID
type_ids*	array	
final_verdict	object	
verdict*	enum	ALLOWED: allowed, blocked Final verdict of the file type analysis.
verdict_explanation*	string	Explanation of the final verdict.
is_file_type_mismatch	boolean	Indicates if the file type does not match the expected type.
other_detections	array	Other file type detections.
result_template_hash	string	SHA256 Hash of user-interface template. For web console only.
spoofing_info	object	
detection_result	string	Result of the spoofing detection.
result_explanation	string	Explanation of the spoofing detection result.
result_overview	string	Overview of the spoofing detection result.
opswatfilescan_info	object	
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
blocked_reasons	array	
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
hash_time	integer	Total time elapsed for computing hashes (in milliseconds).
outdated_data	array	
processing_time	integer	Total time elapsed during processing file (in milliseconds).
processing_time_details	object	
av_scan_time	integer	AV engines' processing time.

NAME	TYPE	DESCRIPTION
cdr_time	integer	Deep CDR engine's sanitization time.
dlp_time	integer	Proactive DLP engine's processing time.
extraction_time	integer	Archive extraction engine's processing time.
filetype_time	integer	FileType engine's processing time.
opswatfilescan_time	integer	OPSWAT Filescan engine's processing time.
others_time	integer	Total time elapsed for following processing tasks in the product (in milliseconds): * Decryption time (if receiving an encrypted file) * External scanner (if configured) * Post action (if configured) * Other internal processing time among components in the product
parse_dgsg_time	integer	Digital signature analyzing time.
vul_time	integer	Vulnerability engine's lookup time.
yara_time	integer	YARA engine's processing time.
filetype_wait_time	integer	FileType engine's wait time.
profile	string	The used rule name.
progress_percentage	integer	Percentage of processing completed (from 1-100).
queue_time	integer	Total time elapsed for file processing task was waiting in MetaDefender Core's queue until being picked up (queue_time = start_time - upload_timestamp) (in milliseconds).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
verdicts	array	
post_processing	object	
actions_failed	string	Empty string if no action failed or list of failed actions, separated by " ".
actions_ran	string	List of successful actions, separated by " ". Empty string if otherwise.
converted_destination	string	Contains the name of the sanitized file.
converted_to	string	Contains target type name of sanitization.
copy_move_destination	string	Contains target type name of sanitization.
sanitization_details	object	
cdr_wait_time	integer	The time in milliseconds that the CDR process took to complete.
description	string	Action was successful or not.
details	array	
action*	enum	ALLOWED: sanitized, removed The type of action that was performed
count	integer	The number of objects that were sanitized/removed.

NAME	TYPE	DESCRIPTION
details	object	
action	enum	ALLOWED: sanitized, removed The type of action that was performed
count	integer	The number of objects that were sanitized/removed.
object_details	array	
object_name	string	The object type that was sanitized/removed.
description	string	Action was successful or not.
file_name	string	If an embedded file was sanitized.
object_details	array	
object_name*	string	The object type that was sanitized/removed.
failure_category	string	Deep CDR errors are classified into different categories. For more details, please find [Troubleshooting sanitization failures](https://docs.opswat.com/mdcore/deep-cdr/troubleshooting-sanitization-failures)
result	enum	ALLOWED: Sanitized, Sanitized failed, Sanitized skipped The result of the CDR process. - **Sanitized** : the file was successfully sanitized. - **Sanitized failed** : the file could not be sanitized due to an error during the process. - **Sanitized skipped** : the file was skipped from sanitization. Common reasons include the file being digitally signed or other policy-based exclusions.
result_template_hash	string	The hash value of the result template, which is used for displaying results on the Core UI and for internal communication between MetaDefender Core and the Deep CDR engine. This value is intended for system use only and is not required for external integration.
sanitized_file_info	object	
file_size	integer	Size of sanitized file in bytes.
sha256	string	SHA256 hash of sanitized file.
verdict	enum	ALLOWED: blocked, allowed The verdict of the CDR process. - **blocked** : the file is recommended for blocking by Deep CDR. - **allowed** : the file is recommended for allowing by Deep CDR as it found no reason to recommend blocking it.
verdict_explanations	array	
scan_results	object	
data_id	string	Data ID of the requested file
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by C00, Blocked Verdict by C00, Unknown Verdict by C00, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis.
total_time	integer	Total time elapsed during scan (in milliseconds).
scan_details	object	
ClamAV	object	
def_time	string	The database definition time for this engine
eng_id	string	The unique identification string for the engine
location	string	Where this engine is deployed (local/remote).
scan_result_i	integer	Scan result as index in the Processing Results table
scan_time	integer	The time elapsed during scan with this engine (in milliseconds).
threat_found	string	The threat name, IF scan result is Infected or Suspicious. Otherwise empty string or error message from the engine.
wait_time	integer	Time elapsed between sending file to Core and receiving the result from the engine (in milliseconds).
vulnerability_info	object	
result	object	
code	integer	The result code for vulnerability check, 0 means a successful check
hash	string	The file's SHA1 hash value
method	enum	ALLOWED: 50700 The method used by OESIS Framework, it should be 50700 every time.
timestamp	string	Timestamp of the request issued

NAME	TYPE	DESCRIPTION
timing	integer	The vulnerability check's duration in milliseconds
detected_product	object	
has_kb	boolean	Indicates whether any KBs or MSBs exist for this hash
has_vulnerability	boolean	Indicates whether any vulnerabilities have been associated with the particular product
is_current	boolean	True if this product's patch level is current, defaults to true
product	object	
id	integer	The OPSWAT product id
name	string	The product name
remediation_link	string	A link where product updates or patches can be obtained
severity	enum	ALLOWED: LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`
sig_name	string	Product signature descriptor
signature	integer	OPSWAT signature id
vendor	object	
id	integer	The OPSWAT vendor id
name	string	The vendor name
version	string	The installed product version
version_data	object	
count_behind	integer	The number of patches behind of the installed product
feed_id	integer	The remote feed ID used to determine patch level
version	string	The current version of the product in the remote feed
vulnerabilites	array	
description	string	A text description of the specific vulnerability
details	object	
cpe	string	A CPE product reference
cve	string	A CVE identification string
cvss	object	
access-complexity	string	A CVSS access-complexity descriptor
access-vector	string	A CVSS access-vector descriptor
authentication	string	A CVSS authentication descriptor

NAME	TYPE	DESCRIPTION												
availability-impact	string	A CVSS availability impact descriptor												
confidentiality-impact	string	A CVSS confidentiality impact descriptor												
generated-on-epoch	string	An epoch timestamp indicating CVSS generation time												
integrity-impact	string	A CVSS integrity impact descriptor												
score	string	A CVSS 10-point severity score												
source	string	A CVSS source descriptor												
cwe	string	A CWE group identification string												
last_modified_epoch	string	An epoch timestamp indicating source last update time												
published-epoch	string	An epoch timestamp indicating source publishing time												
references	array													
severity	enum	ALLOWED: LOW, MODERATE, IMPORTANT, CRITICAL, NOT_AVAILABLE, UNKNOWN String description of Severity level: * `LOW` * `MODERATE` * `IMPORTANT` * `CRITICAL` * `NOT_AVAILABLE` * `UNKNOWN`												
severity_index	integer	A 5 point scale numerical description of Severity level with 5 being greatest and 0 being unknown												
static_id	integer	An OPSWAT identifier for the vulnerability												
verdict	integer	The vulnerability check's duration in milliseconds * `0` - No Vulnerability Found * `1` - Vulnerability Found * `3` - Failed * `16` - Processing Timed Out												
yara	object													
hits	object													
verdict	enum	ALLOWED: 0, 1, 2, 3, 4 The overall result for the analyzed file. Value will be one of the following: <table><tr><th>index</th><th>status</th></tr><tr><td>0</td><td>Clean</td></tr><tr><td>1</td><td>Found matched data</td></tr><tr><td>2</td><td>Suspicious</td></tr><tr><td>3</td><td>Failed</td></tr><tr><td>4</td><td>Not scanned</td></tr></table>	index	status	0	Clean	1	Found matched data	2	Suspicious	3	Failed	4	Not scanned
index	status													
0	Clean													
1	Found matched data													
2	Suspicious													
3	Failed													
4	Not scanned													

STATUS CODE - 404: Invalid hash format

1.4 GET /file/rules

Fetching Available Analysis Rules

Retrieve all available rules with their custom configurations. Fetching available processing rules.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication. Only those rules are returned, that: * Match the apikey's role sent using the apikey header, or * Are not restricted to a specific role.
user_agent	string		The user agent string value sent in the header (specified by the client). Only those rules are returned, that: * Match the client's user agent sent using the user_agent header, or * Are not restricted to a specific user agent. For details see KB article [What are Security Policies and how do I use them?](https://onlinehelp.opswat.com/corev4/What_are_Security_Policies_and_how_do_I_use_them_.html).

RESPONSE

STATUS CODE - 200: Returns the list of available rules.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
max_file_size	integer	The maximum allowed file size (in bytes) for this rule.
name	string	A unique identifier for identify in the used rule for a scan..
global_timeout	object	
value	integer	The timeout value in seconds.
enabled	boolean	Indicates whether the global timeout is enabled.

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.5 GET /file/converted/{data_id}

Download Sanitized Files

Retrieve sanitized file based on the data_id

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the sanitized content.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.6 GET /file/download/{data_id}

Download either sanitized files or DLP processed files

Retrieve sanitized file based on the data_id. In case there's no sanitized file, and DLP processed file is available, user will retrieve DLP processed file.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string	8101abae27be4d63859c55d9e0ed0135	The data_id comes from the result of `Analyze a file`. In case of sanitizing the content of an archive, the data_id of contained file can be found in `Fetch analysis result`.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Returns the sanitized or DLP processed content.

RESPONSE MODEL - application/octet-stream

STATUS CODE - 404: File could not be found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "File could not be found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.7 POST /file/{data_id}/cancel

Cancel File Analysis

When cancelling a file analysis, the connected analysis (e.g. files in an archive) that are still in progress will be cancelled also.

The cancelled analysis will be automatically closed.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*data_id	string		Unique submission identifier. Use this value to reference the submission.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Analysis was successfully cancelled.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
  "<<data_id>>": "cancelled"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Data ID not found (invalid id) or Requests resource was not found

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

2. AUTH

Authentication APIs

User authentication is done via username & password.

2.1 POST /login

Login

Initiate a new session. Required for using protected REST APIs.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
user*	string	Username
password*	string	User's password

EXAMPLE:

```
{
  "user": "admin",
  "password": "admin"
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
oms-csrf-token*	string	The randomly generated token used to prevent CSRF attacks
session_id*	string	The apikey used to make API calls which requires authentication

STATUS CODE - 403: Invalid credentials

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	<error message> will describe the incident. More details would be logged in MetaDefender Distributed Cluster services logs

EXAMPLE:

```
{
  "err": "Failed to login"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

2.2 POST /logout

Logout

Destroy session for not using protected REST APIs.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
response*	string	

STATUS CODE - 400: Bad Request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3. BATCH

Group the analysis requests in batches. Supported with endpoints: MetaDefender Distributed Cluster API Gateway.

3.1 POST /file/batch

Initiate Batch

Create a new batch and retrieve the batch_id

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
rule	string		Select rule for the analysis, if no header given the default rule will be selected (URL encoded UTF-8 string of rule name)
user_agent	string		user_agent header used to identify (and limit) access to a particular rule. For rule selection, `rule` header should be used.
user-data	string		Name of the batch (max 1024 bytes, URL encoded UTF-8 string).

RESPONSE

STATUS CODE - 200: Batch created successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_id*	string	The batch identifier used to submit files in the batch and to close the batch.

EXAMPLE:

```
{
  "batch_id": "74c85f475147439bac4d33b181853923"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3.2 POST /file/batch/{batchId}/close

Close Batch

The batch will be closed and files can no longer be added to the current batch.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch successfully closed.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	object	
batch_count	integer	The total number of files/entries in the batch.
files_in_batch	array	
data_id	string	Unique identifier for the file.
detected_by	integer	Total number of engines that detected this file.
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
progress_percentage	integer	Percentage of processing completed (from 1-100).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
verdicts	array	

NAME	TYPE	DESCRIPTION
progress_percentage	integer	Track analysis progress until reaches 100.
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by COO, Blocked Verdict by COO, Unknown Verdict by COO, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 The overall scan result as index in the Processing Results table.
scanned_with	integer	The total number of engines used to analyze this file.
first_index	integer	The starting index in the batch. Used for pagination.
page_size	integer	The number of entries per page.
batch_id	string	The batch unique identifier
is_closed	boolean	The batch status (open/close).
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
profile	string	The used rule name.
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
scan_results	object	
batch_id	string	The batch unique identifier

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by C00, Blocked Verdict by C00, Unknown Verdict by C00, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.
total_time	integer	Total time elapsed during scan (in milliseconds).
user_data	string	Metadata submitted at batch creation.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3.3 GET /file/batch/{batchId}

Status of Batch Analysis

Retrieve status report for the entire batch

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
first	integer		The first item order in the list of files in this batch
size	integer		The number of items to be fetched next, counting from the item order indicated in first header

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch progress paginated report (50 entries/page).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
batch_files	object	
batch_count	integer	The total number of files/entries in the batch.
files_in_batch	array	
data_id	string	Unique identifier for the file.
detected_by	integer	Total number of engines that detected this file.
display_name	string	The filename reported via `filename` header.
file_size	integer	Total file size in bytes.
file_type	string	The filetype using mimetype.
file_type_description	string	The filetype in human readable format.
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
progress_percentage	integer	Percentage of processing completed (from 1-100).
result	string	The final result of processing the file (Allowed / Blocked / Processing).
verdicts	array	
progress_percentage	integer	Track analysis progress until reaches 100.

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by C00, Blocked Verdict by C00, Unknown Verdict by C00, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 The overall scan result as index in the Processing Results table.
scanned_with	integer	The total number of engines used to analyze this file.
first_index	integer	The starting index in the batch. Used for pagination.
page_size	integer	The number of entries per page.
batch_id	string	The batch unique identifier
is_closed	boolean	The batch status (open/close).
process_info	object	
blocked_reason	string	Provides the reason why the file is blocked (if so).
file_type_skipped_scan	boolean	Indicates if the input file's detected type was configured to skip scanning.
profile	string	The used rule name.
result	string	The final result of processing the file (Allowed / Blocked / Processing).
user_agent	string	Identifier for the REST Client that calls the API.
username	string	User identifier who submitted scan request earlier.
scan_results	object	
batch_id	string	The batch unique identifier

NAME	TYPE	DESCRIPTION
scan_all_result_a	enum	ALLOWED: No Threat Detected, Infected, Suspicious, Failed, Whitelisted, Blacklisted, Exceeded Archive Depth, Not Scanned, Encrypted Archive, Exceeded Archive Size, Exceeded Archive File Number, Password Protected Document, Exceeded Archive Timeout, Mismatch, Potentially Vulnerable File, Cancelled, Sensitive Data Found, Yara Rule Matched, Potentially Unwanted, Unsupported File Type, Extraction Failed, Scan Failed, Suspicious Verdict by Sandbox, Likely Malicious Verdict by Sandbox, Malicious Verdict by Sandbox, Blocked Verdict by Sandbox, Blocked Verdict by Deep CDR, Global Timeout Exceeded, Vulnerable Verdict by SBOM, Non-vulnerable Verdict by SBOM, Blocked Verdict by SBOM, Blocked by Post Action, Known Bad, Known Good, Unknown, Allowed Verdict by C00, Blocked Verdict by C00, Unknown Verdict by C00, In Progress, Skip Processing Fast Symlink The overall scan result as string
scan_all_result_i	enum	ALLOWED: 0, 1, 2, 3, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 38, 39, 40, 41, 42, 43, 255, 1014 The overall scan result as index in the Processing Results table.
start_time	string	Timestamp when the scanning process starts.
total_avs	integer	Total number of scanning engines used as part of this analysis. Not like files, batch is not processed by engine, so this value is always 0.
total_time	integer	Total time elapsed during scan (in milliseconds).
user_data	string	Metadata submitted at batch creation.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3.4 GET /file/batch/{batchId}/certificate

Download Signed Batch Result

Download digitally signed status report for the entire batch

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Signed batch result and certificate are sent back in response body (YAML format).

RESPONSE MODEL - application/x-yaml

EXAMPLE:

```
--- batch_id: 092876200fb54cfb80b6e3332c410ae9 user_data: the user data from the header from
batch creation cert_sha1_fingerprint: <some cert serial value> batch_files:\n batch_count:
1\n files_in_batch:\n - data_id: 9112b225f0634f189a2bb46ec1a7826f\n display_name:
New%20Text%20Document.txt\n file_size: 5\n scan_all_result_i: 0\n process_info:\n
blocked_reason:\n result: Allowed\n md5: 42b130c3ce46e058f30712838cebf420\n sha1:
ed94baf55ca851055fb76045f6949bca2f865605\n sha256:
f4191b3ec6ce93aaf712919a38e52815c5da9c91d2b141df920bc8bcb5cbb8e3\n sha512: \"\n
vulnerabilities:\n - cve: CVE-2021-45463\n cvss:\n score: 6.8\n
cvss_3_0:\n base_score: 7.8\n - cve: CVE-2018-12713\n cvss:\n
score: 6.4\n cvss_3_0:\n base_score: 9.1\nprocess_info:\n blocked_reason:\n
file_type_skipped_scan: false\n profile: File scan\n result: Allowed\n user_agent:
webscan\nscan_results:\n scan_all_result_a: No Threat Detected\n scan_all_result_i: 0\n
start_time: 2017-05-23T11:22:03.010Z\n total_avs: 14\n total_time: 995\n...\n--- signature:
881d22220c4ca0557d7c7d5c5794d53a8a2780997cd65b27b6e7f1c099a15de03dbcb5edbeaea7aafa6099fab37be
07017b39e3e3a7d66c550f44eb59a096c54d5b9555cb28198546fbec57c33b717751d333a09733d95dd876e2798d0
44c8caef828f4352b91f9a6d057253bb1a9461e0e0ebf4313a80895998d645bebc81841ff3499589c80ffc4e8a19
0d1ec9b3e4126d86659d303b0e1f22d9289c9c4671d35532b55ad4620e048a78bb405b573897da63efdd5f036692c
934a82d9bdc9b9862e7fea5e8abeeb1444be0689d50373c5c0632484950c0fe0337ed5f91bdf26986f7cfff8aa3431
bf4bc948fc127c16ba13ec679fe9f67e7586075c1f467454fa8cf40e9cd501291c95d862eb16f4477c17d1711294f
0ff2b3a1140bd53dbd1fbb0846af6062e9e4e2e1a09af3448503ed11e342164e535fc268bf7d8fbc28ed946cd2bb8
ea075f2295d2fa8392076d41608c3b5decf8fab3a5ec7de190f07583331e0517e5f361735cd59326622dc8b07b10a
464028de781a063e408f918c1d5534329140f4e4dc1a717d808d6784410410b00d36cb9a345f5bbc11fa1c58ee28f
8e7b863f3ea2c923ec5fb2ac29eaa4ddc0d6d9dfd3f16a97f207dc2858410a577c7f4a92ff01bad3229f5fcd08e2
1df9869a113272aa9d96bfdfe8bfb3a50414c174e16a3504e5780c2718779b0757298546f287ef7ea86e67510d48a
8 certificate: |\n -----BEGIN CERTIFICATE-----\n
MIIGJzCCBA+gAwIBAgIBATANBgkqhkiG9w0BAQUFADCBsjELMAkGA1UEBhMCRlIx\n
DzANBgNVBAGMBKfSc2FjZTETMBEGA1UEBwwKU3RyYXNib3VyZzEYMBYGA1UECgwP\n
d3d3LmZyZWVsYW4ub3JnMRAdDgYDVQQLDAdmcmVlbGFMUS0wKwYDVQDDCRGcmVl\n
bGFuIFNhbXBsZSBdZXJ0aWZpY2F0ZSBBDXR0b3JpdHkxIjAgBgkqhkiG9w0BCQEW\n
E2NvbRhY3RAZnJlZWxhbi5vcmcwHhcNMTIwNDI3MTAzMTE4WWhcNMjIwNDI3MTAz\n
MTE4WjB+MQswCQYDVQQGEWJGUjEPMA0GA1UECwwYd3d3LmZyZWVsYW4ub3JnMRAdDgYDVQQLDAdmcmVlbGFMUS0wKwYDVQDDCRGcmVl\n
d3d3LmZyZWVsYW4ub3JnMRAdDgYDVQQLDAdmcmVlbGFMUS0wKwYDVQDDCRGcmVl\n
MSIwIAYJKoZIhvcNAQkBFHJjb250YWN0QGZyZWVsYW4ub3JnMRAdDgYDVQQLDAdmcmVlbGFMUS0wKwYDVQDDCRGcmVl\n
9w0BAQEFAAOCAg8AMIICGKCAgEA3W29+ID6194bH6ejLrIC4hb2Ugo8v6ZC+Mrc\n
k2dNYMNPjckABvxxEtBamnSaeU/IY7FC/giN622LEtV/3oDcrua0+yWuVafyxmZ\n
eiX9urWurtIK7XgNGFNUjYPq4dSJQPPhwCHE/LKAYkWNZBX\n
RrX0Dq4XyApNku0IppjIjEXH+8ixE12wH8wt7DEvd07T3N3CfUbaIT11qBX+Nm2Z6\n
q4Ag/
```

```

u5r18NjfXg71ZmXA3X0j7zFvpyapRIZcPmkvZYn7SMCP8dXyXHPdpSiIWL2\n  uB3Ki04JrUYvt2GzLBUThp+1NSZaZ/
Q3y0aAAUk0x+1h08285Pi+P810+H2Xic4S\n
vMq1xtLg2bNoPC5KnBRfuFPuUD2/3dSiiragJ6uYDLoyWJDivKGt/720VTEPAL9o\n
6T2pGZrwbQuiFGrGTMZ0vWMSpQtN1+tCCX1T4mWqJDRwuMGrI4DnnGzt3IKqNwS4\n
Qyo9KqjMIPwnXZAmWPm3FOKe4sFwc5fpawK001JZewDsYTDxVj+cwXwFxbE2yBiF\n
z2FAHwfoPwaH35p3C61kcgP2k/zgAlnB1uzACUI+MKJ/G0gv/uAhj10HJQ3L6kn1\n  SpvQ41/
ueBj1unExqQSYD7GtZ1Kg8u0cq2r+WISE3Qc9MpQFFkUV11mgWGwYDuN3\n
Zsez95kCAwEAAa7MHkwCQYDVR0TBAlwADAABglghkgBhvhCAQ0EHxYdT3B1b1NT\n
TCBHZW51cmF0ZWQgQ2VydG1maWNhdGUwHQYDVIR00BBYEFFlfyR06G8y5qEFKik15\n
ajb2fT7XMB8GA1UdIwQYMBaAFCNsLT0+KV14uGw+quK7Lh5sh/JTMA0GCSqGSib3\n
DQEBBQUAA4ICAQAT5wJFPqervbja5+90iKxi1d0QVtVGB+z6aoAMuWK+qgi0vgvr\n
mu9ot21vTSCSnRhjeiP0SIdqFMORmBtOCfk/kYDp9M/91b+vS+S9eAlxrNCB5V0f\n  PqxEPp/wv1rBcE4GB0/
c6HcFon3F+oBYCsUqbZDKSSZxhDm3mj7pb67FNbZbJIzJ\n
70HDsRe2004oiTx+h6g6pW3c0QMgIAvFgKN5Ex727K4230B0NIdGkzu4KSML0NM\n
s1SAcXZ410oSKNjy44BVEZv0ZdxTDrm4EwJtNygGFzmtTuV02nkUj1bYYYC5f0L\n
Adr6s0XMyaK8twlWY1YDZ5uKDPVRVBfiGcq0uJizIvemhuTrofh8pBQqNkPRDFT\n  Rq1iTo1Ihh13/
F11kXk1WR3jTjNb4jHX71IoXwpwp767HAPKGhjQ9cFbnHMEtkro\n
RIJYdtRq5mccDtwT0GFyoJLLBZdHHMHJz0F9H7FNk2tTQMHK5MVYwg+LIae586\n
CQVqfbscP7evlgjLW98H+5zylRHAg0H2G79aHljNKMP9B0uq6SnEg1EsiWGVtu21\n  hnx8SB3sVJZHeer8f/
UQQwqbAO+Kdy70NmbSaqavtp8jOxLiidWkwSyRTsuU6D8i\n
DiH5uEqBXEjRj0Fs1xcVKdVj5glVcSmkLwZKbEU10KwleT/iXfHvooWhQ==\n  -----END CERTIFICATE-----
\n...\n"

```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```

{
  "err": "Invalid header"
}

```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```

{
  "err": "Access denied"
}

```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3.5 POST /file/batch/{batchId}/cancel

Cancel Batch

When cancelling a batch, the connected analysis that are still in progress will be cancelled also.

The cancelled batch will be closed.

REQUEST

PATH PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*batchId	string		The batch identifier used to submit files in the batch and to close the batch.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Batch cancelled.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
  "<<batch_id>>": "cancelled"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Batch not found (invalid id)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
------	------	-------------

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4. LICENSE

Retrieve the current license information.

4.1 GET /admin/license

Get current license information

Fetch details about the longest expiry active license among all activated licenses.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Information about the licensed product (product type, number of activations, deploymentId, expiration date and days left)

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
days_left	integer	Number of days left before expiration
expiration	string	Expiration date in MM/DD/YYYY format.
licensed_engines	array	
licensed_to	string	Name of the entity to which the license is issued.
max_agent_count	string	Total number of deployed MetaDefender Agents attached to this MetaDefender Core instance.
online_activated	boolean	Track online/offline activation mode
product_id	string	Official MetaDefender base SKU licensed.

NAME	TYPE	DESCRIPTION
product_name	string	Official MetaDefender base product name licensed.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

5. STATS

Health check and statistics about MetaDefender Core instance usage.

5.1 GET /stat/engines

Engine Status

Return the status of the latest engines between the MetaDefender Core instances.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: An array with all the engines and their details.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
abandoned	boolean	Indicates if this engine is abandoned.
active	boolean	If used by at least one engine
def_time	string	The database definition time for this engine
download_progress	integer	The percentage progress of download
download_time	string	When this engine downloaded from the update server.
eng_id	string	Engine internal ID
eng_name	string	Engine name
eng_type	string	Engine type in human readable form

NAME	TYPE	DESCRIPTION
eng_ver	string	Engine's version (format differs from one engine to another).
engine_type	enum	ALLOWED: av, archive, filetype Engine's type: * av * archive * filetype
notified_messages	array	A list of messages from engine.
pinned	boolean	Indicate if this engine is pinned.
state	enum	ALLOWED: downloading, downloaded, staging, production, removed, temporary failed, permanently failed, content invalid, download failed Status of the engine: * downloading * downloaded * staging * production * removed * temporary failed * permanently failed * content invalid * download failed
type	string	The type of information, whether it is engine or engine's database.

5.2 GET /stat/nodes

Instance Status Overview

Retrieve status details of all available MetaDefender Core instances.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Status details of MetaDefender Core instances.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
external_nodes_allowed	boolean	Indicates whether external nodes can connect; always true.
max_node_count	integer	Total number of available MetaDefender Core instances.
statuses	array	
address	string	Location of the Core instance; currently always return empty string.
available_mem	integer	The number of available RAM in this system.
cpu_cores	integer	The number of CPU Cores allocated to this Core instance.
current_processing_files	integer	Number of objects currently being processed by the Core instance.
engines	array	
active	boolean	If used by at least one engine
db_ver	string	The database version for this engine
def_time	string	The database definition time for this engine
download_time	string	The database download time for this engine
eng_name	string	Engine name
eng_ver	string	Engine's version (format differs from one engine to another).
engine_type	enum	ALLOWED: av, archive, filetype Engine's type: * av * archive * filetype
id	string	Engine internal ID
issues	array	A list of all potential problems on this engine.
free_disk_space	integer	Reported available disk on Core instance (in bytes).
id	string	Identifier of the worker that deployed this Core instance.
info_disk_space	array	
dirs	array	list of directories used by MetaDefender Core.
free	integer	Free space on the disk (in bytes).
location	string	Disk location.
total	integer	Total space on the disk (in bytes).
issues	array	
description	string	Text detailing the issue.
severity	string	How important is the reported issue.
load	integer	Current CPU utilization on Core instance (in percentage).
os	string	Current used OS
scan_queue	integer	Number of objects currently being processed by the Core instance.

NAME	TYPE	DESCRIPTION
scan_queue_details	object	
archive_scan_queue_ratio	number	Ratio of archive scan queue, always -1 for Core in Cluster mode.
available_slots	integer	The number of slots is available, always -1 for Core in Cluster mode.
extracted_file_slots	integer	Number of child files being processing
file_slots	integer	Number of files taken from REST by the current Core instance
total_scan_queue	integer	Total scan queue, always -1 for Core in Cluster mode.
total_disk_space	integer	The amount of disk space is allocated on Core instance (in Byte).
total_mem	integer	How much memory is allocated on Core instance (in MB).
total_scan_queue	integer	The maximum queue size is allowed, always -1 for Core in Cluster mode.
uptime	integer	How long this Core is in operation (in second).
version	string	Product version

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

5.3 GET /readyz

Get health check status

Fetch current status of system health.

REQUEST

QUERY PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
verbose	boolean	true	Optional. Show detailed result of system health.

RESPONSE

STATUS CODE - 200: System is currently healthy.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
status*	boolean	System-wide status, indicate if all components are healthy.
scan_queue	object	
number_in_queue*	integer	Number of objects being processed by the system.
status*	boolean	The operational status of the scan process; true if the system contains the required minimum of healthy MetaDefender Core instances.
license*	object	
status*	enum	ALLOWED: expired, invalid, ok License status.
components*	object	
status*	boolean	Aggregate component status.
datalake	object	
status	boolean	DataLake overall status.
detail	string	Status detail message
caching	object	
status	boolean	Caching overall status.
detail	string	Status detail message.
broker	object	
status	boolean	Broker overall status.
detail	string	Status detail message.
filestorage	object	

NAME	TYPE	DESCRIPTION
status	boolean	File storage overall status.
detail	string	Status detail message.
identity	object	
status	boolean	Identity service overall status.
detail	string	Status detail message.
ometascan	object	
status*	boolean	MetaDefender Core overall status.
detail	string	Detail message.
instance	object	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

STATUS CODE - 503: System is currently unhealthy.

API Reference

Control Center

API Version: v2.4.0

Developer Guide

This is the API documentation for *MetaDefender Distributed Cluster Control Center Public API*. If you would like to evaluate or have any questions about this documentation, please contact us via our [Contact Us](#) form.

How to Interact with MetaDefender Distributed Cluster Control Center using REST API

The MetaDefender Distributed Cluster Control Center empowers administrators and system engineers to efficiently manage system operations, including:

1. Establishing and maintaining essential service connections.

2. Deploying and managing MetaDefender Core, MetaDefender Distributed Cluster API Gateway instances.
3. Managing licenses.
4. Administering user accounts and access controls.
5. Configuring and enforcing security protocols.
6. Monitoring the overall system health and system performance.

OPSWAT recommends using the JSON-based REST API. The available methods are documented below.

OPSWAT provides some sample codes on [GitHub](#) to make it easier to understand how the MetaDefender REST API works.

CONTACT

NAME: API Support

EMAIL: feedback@opswat.com

URL: <https://github.com/OPSWAT/metadefender-core-openapi3>

Terms of service: <https://onlinehelp.opswat.com/policies/>

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
apikey	apiKey	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

API

1. USER MANAGEMENT

User management APIs

The APIs for manage users and user directories.

1.1 GET /admin/user

List all users

Returns a list of all users in the server.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of users retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
api_key	string	Associated apikey with this user
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address
id	integer	User's unique identifier

NAME	TYPE	DESCRIPTION
name	string	User's full name
description	string	User's description, 256 characters maximum
roles	array	
ui_settings	object	

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.2 POST /admin/user

Create user

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
api_key	string	Associated apikey with this user
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address
name	string	User's full name
description	string	User's description, 256 characters maximum
roles	array	
ui_settings	object	
password	string	The user's password

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
api_key	string	Associated apikey with this user
directory_id	integer	To which User Directories belongs to (LOCAL/System/etc.)
display_name	string	Which is the name showed in the Management Console
email	string	User's email address
name	string	User's full name
description	string	User's description, 256 characters maximum
roles	array	
ui_settings	object	

STATUS CODE - 400: Bad Request (e.g. invalid header, invalid request body).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.3 DELETE /admin/user/{user_id}

Delete a user

Delete a user by id from the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
  "result": "Success"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Item does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not allowed"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

1.4 POST /user/changetpassword

Change Password for local user

Modify the current password for the user identified by apikey

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
old_password	string	The current password in plain text
new_password	string	The new password in plain text

EXAMPLE:

```
{
  "old_password": "admin",
  "new_password": "123456"
}
```

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, invalid request body).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

2. ADMIN

Admin specific API requests.

2.1 GET /admin/userdirectory

List all user directories

Retrieve a list of all user directories configured in the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: List of user directories retrieved successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
id	integer	Internal used identifier
name	string	Name of the user directory
type	string	Type of the user directory (e.g., LDAP, Local, etc.)
enabled	boolean	If the user directory is enabled or not
lockout_attempts	integer	Number of failed login attempts before the user is locked out
lockout_timeout	integer	Time in seconds before the user can try to log in again after being locked out

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

2.2 POST /admin/role

Create new role

Add a new user role to the system.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
cert	array	
configlog	array	
engines	array	
license	array	
retention	array	
rule	array	
scanlog	array	
update	array	
updatelog	array	
users	array	
workflow	array	
zone	array	
healthcheck	array	
fetch	array	
download	array	
deployment	array	
service	array	
packageupload	array	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
name	string	The name identifier of the role
display_name	string	The extended name showed in the Management Console.
rights	object	
cert	array	
configlog	array	
engines	array	
license	array	
retention	array	
rule	array	
scanlog	array	
update	array	
updatelog	array	
users	array	
workflow	array	
zone	array	
healthcheck	array	
fetch	array	
download	array	
deployment	array	
service	array	
packageupload	array	
editable*	boolean	If the role can be altered or not.
id*	integer	Internal used identifier

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

2.3 DELETE /admin/role/{role_id}

Delete a role

Delete a role by id from the system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

EXAMPLE:

```
{
  "result": "Success"
}
```

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Item does not exist"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not allowed"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3. AUTH

Authentication APIs

User authentication is done via username & password.

3.1 POST /login

Login

Initiate a new session. Required for using protected REST APIs.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
user*	string	Username
password*	string	User's password

EXAMPLE:

```
{
  "user": "admin",
  "password": "admin"
}
```

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
oms-csrf-token*	string	The randomly generated token used to prevent CSRF attacks
session_id*	string	The apikey used to make API calls which requires authentication

STATUS CODE - 403: Invalid credentials

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	<error message> will describe the incident. More details would be logged in MetaDefender Distributed Cluster services logs

EXAMPLE:

```
{
  "err": "Failed to login"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

3.2 POST /logout

Logout

Destroy session for not using protected REST APIs.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
response*	string	

STATUS CODE - 400: Bad Request.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 403: Invalid user information.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err*	string	

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4. CONFIG

Configure the product through APIs (especially the Settings). Will require admin apikey..

4.1 PUT /admin/config/auditlog/cleanup

Audit clean up

Setting audit record cleanup time (cleanup records older than).

Note: The cleanup range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. _**Note**_: If `cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4.2 PUT /admin/config/filestorage/cleanup

File storage clean up

Setting file storage clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. _**Note**_: If `cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4.3 PUT /admin/config/warehouse/cleanup

Executive report clean up

Setting executive report clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. **Note** : If

NAME	TYPE	DESCRIPTION
		`cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4.4 PUT /admin/config/scanhistory/cleanup

Processing history clean up

Setting processing history clean up time (clean up records older than).

Note:The clean up range is defined in hours.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
cleanuprange	integer	The number of hours of data retention. Anything older than this number will be deleted. **Note** : If `cleanuprange` is `0`, the cleanup functionality will be disabled.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
cleanup_range	integer	The number of hours of data retention. Anything older than this number will be deleted.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4.5 PUT /admin/config/session

Session settings

Configure settings for session generated upon a successful login See more at [Login](#)

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
absoluteSessionTimeout	integer	The interval (in milliseconds) for overall session length timeout (regardless of activity).
allowCrossIpSessions	boolean	Allow requests from the same user to come from different IPs.
allowDuplicateSession	boolean	Allow same user to have multiple active sessions.
sessionTimeout	integer	The interval (in milliseconds) for the user's session timeout, based on last activity. Timer starts after the last activity for the apikey.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
absoluteSessionTimeout	integer	The interval (in milliseconds) for overall session length timeout (regardless of activity).
allowCrossIpSessions	boolean	Allow requests from the same user to come from different IPs.
allowDuplicateSession	boolean	Allow same user to have multiple active sessions.
sessionTimeout	integer	The interval (in milliseconds) for the user's session timeout, based on last activity. Timer starts after the last activity for the apikey.

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4.6 GET /admin/config/sessioncookie

Get session cookie attributes

Getting session cookie attributes

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
------	------	---------	-------------

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
samesite	number	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

4.7 PUT /admin/config/sessioncookie

Update session cookie attributes

Modifying session cookie attributes

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
samesite	number	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
samesite	number	SameSite attribute accepts three values: * `Strict` - cookies will only be sent in a first-party context, not be sent along with requests initiated by third party websites. * `Lax` - cookies are not sent on normal cross-site subrequests, but are sent when a user is navigating to the origin site. * `None` - cookies will be sent in all contexts. Default value: `Lax`

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{  
  "err": "Access denied"  
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{  
  "err": "Not found"  
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{  
  "err": "<error message>"  
}
```

}

5. INSTALLERS

Upload and manage installers for the MetaDefender Core and MetaDefender Distributed Cluster API Gateway.

5.1 GET /admin/installer

Get uploaded installers

Retrieve information about an uploaded installer.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

5.2 POST /admin/installer

Upload installer

Upload installers for the MetaDefender Core and MetaDefender Distributed Cluster API Gateway.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.
*filename	string		The name of the installer file to upload. Note : Ensure the filename remains same with the original MY OPSWAT download (e.g: ometascan-5.15.0-1-x64.msi)

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
installer_id	string	Unique identifier of the uploaded installer.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

```
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

5.3 DELETE /admin/installer/{installer_id}

Delete an uploaded installer

Delete an uploaded installer.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request processed successfully

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	Success message.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

6. SERVICES

Add essential services and view connection status.

6.1 GET /admin/service

Get the status of all essential services.

Retrieve the status of all added services within the MetaDefender Distributed Cluster system.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey y	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Details of all added services and their status.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
service_type	object	
healthy_instances	number	Number of healthy instances for the service.
overall_status	string	Aggregated status across all instances of the service.
overall_status_description	string	Description of the overall status.
instances	array	
service_id	string	Unique service identifier.
message	string	Optional status/message.
display_name	string	Human friendly name. Defaults to "host:port" if absent.

NAME	TYPE	DESCRIPTION
status_description	string	Human readable status explanation.
host	string	Hostname or IP.
port	number	Service's port.
version	string	Service version (semantic or other).
added_by	string	User or system that registered the service.
last_update	number	Unix epoch milliseconds of last update.
last_healthy	number	Unix epoch milliseconds of last confirmed healthy state.
detail	object	
cpu_usage	number	CPU usage (implementation specific units).
platform	string	Operating system/platform the service is running on.
role	string	Service role (e.g. primary, secondary).
db_size	number	Database size in bytes.
ram	object	
total_bytes	number	Total RAM available.
used_bytes	number	RAM currently in use.
disk	object	
total_bytes	number	Total disk space available.
used_bytes	number	Disk space currently in use.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

6.2 POST /admin/service

Connect and check essential services status.

Establish connections and retrieve the status of essential services within the MetaDefender Distributed Cluster system.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
ONE:OF	object	
OPTION:1	object	

NAME	TYPE	DESCRIPTION
host*	string	the host address of the service.
port*	integer	the port number of the service.
connection_key*	string	the connection key for the service.
OPTION:2	object	
host*	string	the host address of the service
port*	integer	the port number of the service
user*	string	the user name for the service.
password*	string	the password for the service.
OPTION:3	object	
host*	string	the host address of the service.
port*	integer	the port number of the service.
user	string	the user name for the service.
password	string	the password for the service.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
result	string	the result of the service addition, can be either "ok" or "error"
service_id	string	The unique identifier of the service if result is "ok"
detail	string	The error details if result is "error"

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

6.3 PUT /admin/service/{service_id}

Edit service details.

Update the display name and/or configuration details for a specific service. **Note:** Service configuration cannot be modified after instances have been deployed.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
display_name	string	Display name for the service.
config	object	
host	string	the host address of the service
port	integer	the port number of the service
user	string	the user name for the service. Applicable for type `caching`, `broker`, `datalake`, and `warehouse`
password	string	the password for the service. Applicable for type `caching`, `broker`, `datalake`, and `warehouse`
connection_key	string	the connection key for the service. Applicable for type `filestorage`

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
service_id	string	Unique service identifier.
message	string	Optional status/message.
display_name	string	Human friendly name. Defaults to "host:port" if absent.
status_description	string	Human readable status explanation.
host	string	Hostname or IP.
port	number	Service's port.
version	string	Service version (semantic or other).
added_by	string	User or system that registered the service.

NAME	TYPE	DESCRIPTION
last_update	number	Unix epoch milliseconds of last update.
last_healthy	number	Unix epoch milliseconds of last confirmed healthy state.
detail	object	
cpu_usage	number	CPU usage (implementation specific units).
platform	string	Operating system/platform the service is running on.
role	string	Service role (e.g. primary, secondary).
db_size	number	Database size in bytes.
ram	object	
total_bytes	number	Total RAM available.
used_bytes	number	RAM currently in use.
disk	object	
total_bytes	number	Total disk space available.
used_bytes	number	Disk space currently in use.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

6.4 DELETE /admin/service/{service_id}

Disconnect to service and remove their configurations.

Remove the connection and configuration details for a specific service. **Note:** Service configuration cannot be deleted after instances have been deployed.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to remove service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
service_id	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

6.5 GET /admin/service/{service_type}

Get status for a specific service.

Retrieve the current status of a specific service, including all instance details. **Note:** The `service_type` must be one of: datalake, warehouse, caching, broker, filestorage.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to get service type was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
overall_status	string	Aggregated status for the service type (e.g. healthy, degraded, down)
overall_status_description	string	Human readable description of the aggregated status
healthy_instances	number	Count of instances currently considered healthy
instances	array	
service_id	string	Unique service identifier.
message	string	Optional status/message.
display_name	string	Human friendly name. Defaults to "host:port" if absent.
status_description	string	Human readable status explanation.
host	string	Hostname or IP.
port	number	Service's port.
version	string	Service version (semantic or other).
added_by	string	User or system that registered the service.
last_update	number	Unix epoch milliseconds of last update.
last_healthy	number	Unix epoch milliseconds of last confirmed healthy state.
detail	object	
cpu_usage	number	CPU usage (implementation specific units).
platform	string	Operating system/platform the service is running on.
role	string	Service role (e.g. primary, secondary).
db_size	number	Database size in bytes.
ram	object	
total_bytes	number	Total RAM available.
used_bytes	number	RAM currently in use.

NAME	TYPE	DESCRIPTION
disk	object	
total_bytes	number	Total disk space available.
used_bytes	number	Disk space currently in use.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

6.6 GET /admin/service/{service_type}/setting

Get service settings

Retrieve the current configuration settings for a specific service. **Note:** Supported only for the filestorage service type.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to retrieve service settings was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
max_replica	number	Maximum number of replicas, default is 1
min_replica	number	Minimum number of replicas, default is 1
cleanuprange	number	Cleanup interval in hours, default is 0
storage	object	
type	string	Storage backend type, can be `salt` or `none`
config	object	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
```

```

    "err": "Not found"
  }

```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```

{
  "err": "Access denied"
}

```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```

{
  "err": "<error message>"
}

```

6.7 PUT /admin/service/{service_type}/setting

Edit setting of service

Update the configuration settings for a specific service. **Note:** Supported only for the filestorage service type.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
max_replica	number	Maximum number of replicas, default is 1
min_replica	number	Minimum number of replicas, default is 1

NAME	TYPE	DESCRIPTION
cleanuprange	number	Cleanup interval in hours, default is 0
storage	object	
type	string	Storage backend type, can be `salt` or `none`
config	object	

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	Success message

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7. WORKERS

Connect, deploy, undeploy and manage workers.

7.1 GET /admin/worker

List connected workers

Retrieve a list of currently connected MetaDefender Distributed Cluster Worker services.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of connected workers.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
worker_id	string	Unique identifier of the worker.
display_name	string	Display name for the worker.
platform	string	Operating system / platform of the worker.
os	string	Operating system details of the worker.
package_type	string	The deployment package type.
hardware	object	
cpu	object	
count	integer	Number of CPU cores.

NAME	TYPE	DESCRIPTION
model	string	CPU model name.
usage	number	CPU usage percentage.
disk	object	
available_bytes	integer	Available disk space in bytes.
total_bytes	integer	Total disk space in bytes.
memory	object	
available_bytes	integer	Available memory in bytes.
total_bytes	integer	Total memory in bytes.
user_name	string	Name of the user who added the worker.
host	string	The address (IP or hostname) of the worker.
port	integer	Port on which the worker is listening.
status	string	Current status of worker.
status_description	string	The description of worker's status
version	string	The version of the worker.
deployment_info	object	
type	string	Deployment type, can be `ometascan` or `api-gateway`.
installer_id	string	Identifier of the installer.
version	string	The instance version.
user_name	string	Name of the user who deployed the instance.
custom_config	object	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7.2 POST /admin/worker

Connect to workers

Connect to MetaDefender Distributed Cluster Worker services.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
display_name*	string	Display name for the worker.
host*	string	The address of the worker.
port*	number	The port on which the worker is listening.
result*	enum	ALLOWED: ok, failed Connection attempt result.
worker_id	string	Present only when result = ok. Unique identifier of the worker.
error	string	Present only when result = failed. Error message.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7.3 DELETE /admin/worker

Disconnect from workers

Disconnect from specified MetaDefender Distributed Cluster Worker services.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
------	------	---------	-------------

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey y	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to disconnect workers was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
worker_id	enum	ALLOWED: Deleted Disconnection status of the worker.

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
------	------	-------------

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7.4 GET /admin/worker/available/{installer_id}

Get available workers by installer_id.

Retrieve the list of available workers eligible for deployment for the specified installer ID.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to get available workers was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
ARRAY OF OBJECT WITH BELOW STRUCTURE		
worker_id	string	Unique identifier of the worker.
display_name	string	Display name for the worker.
platform	string	Operating system / platform of the worker.
os	string	Operating system details of the worker.
package_type	string	The deployment package type.
hardware	object	
cpu	object	
count	integer	Number of CPU cores.
model	string	CPU model name.
usage	number	CPU usage percentage.
disk	object	
available_bytes	integer	Available disk space in bytes.
total_bytes	integer	Total disk space in bytes.
memory	object	
available_bytes	integer	Available memory in bytes.
total_bytes	integer	Total memory in bytes.
user_name	string	Name of the user who added the worker.
host	string	The address (IP or hostname) of the worker.
port	integer	Port on which the worker is listening.
status	string	Current status of worker.
status_description	string	The description of worker's status
version	string	The version of the worker.
deployment_info	object	
type	string	Deployment type, can be `ometascan` or `api-gateway`.
installer_id	string	Identifier of the installer.
version	string	The instance version.
user_name	string	Name of the user who deployed the instance.
custom_config	object	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 404: Requests resource was not found.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Not found"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7.5 POST /admin/worker/deploy

Deploy workers

Deploy the selected installer on one or more selected workers.

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
ONE:OF	object	
OPTION:1	object	
type*	enum	ALLOWED: ometascan
installer_id*	string	Identifier of the installer.
worker*	array	
license_id	string	Identifier of the license.
config	object	
log_level	enum	DEFAULT: info ALLOWED: debug, info, warning, error
connection_per_file_service	integer	>=1 DEFAULT: 4
OPTION:2	object	
type*	enum	ALLOWED: api-gateway
installer_id*	string	Identifier of the installer.
worker*	array	
cert	string	Certificate name (default empty). Only for api-gateway.
config	object	

NAME	TYPE	DESCRIPTION
port	integer	between 1 and 65535 DEFAULT:8899
log_level	enum	DEFAULT:info ALLOWED: debug, info, warning, error

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to add service was successful

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7.6 DELETE /admin/worker/deploy

Undeploy workers

Undeploy the specified workers.

REQUEST

REQUEST BODY - application/json

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to undeploy workers was successful

RESPONSE MODEL - application/json

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		

NAME	TYPE	DESCRIPTION
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

7.7 POST /admin/worker/upgrade

Upgrade deployed instances

Upgrade the deployed instances managed by the worker to a newer version

REQUEST

REQUEST BODY - application/json

NAME	TYPE	DESCRIPTION
version*	string	Target version to upgrade to.
type*	enum	ALLOWED: ometascan, api-gateway Worker deployment type.

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey	string	y	Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: Request to upgrade workers was successful

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
result	string	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Invalid header"
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```

```
}
```

7.8 GET /admin/worker/upgrade/version

Get upgradable instance version

Retrieve a list of available versions of MetaDefender Core and MetaDefender Distributed Cluster API Gateway for upgrading.

REQUEST

HEADER PARAMETERS

NAME	TYPE	EXAMPLE	DESCRIPTION
*apikey y	string		Generated `session_id` from [Login](/docs/mdcore/metadefender-distributed-cluster/ref#userlogin) call can be used as an `apikey` for API calls that require authentication.

RESPONSE

STATUS CODE - 200: A list of available versions for upgrading.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
ometascan*	array	
api-gateway*	array	

STATUS CODE - 400: Bad Request (e.g. invalid header, apikey is missing or invalid).

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{  
  "err": "Invalid header"  
}
```

STATUS CODE - 403: Invalid user information or Not Allowed

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 405: The user has no rights for this operation.

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "Access denied"
}
```

STATUS CODE - 500: Unexpected event on server

RESPONSE MODEL - application/json

NAME	TYPE	DESCRIPTION
OBJECT WITH BELOW STRUCTURE		
err	string	Error reason

EXAMPLE:

```
{
  "err": "<error message>"
}
```
