Connectors Guide

FortiSOAR 7.3.1

FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO GUIDE
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/training-certification

NSE INSTITUTE
https://training.fortinet.com

FORTIGUARD CENTER
https://www.fortiguard.com

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdoc@fortinet.com

January, 2023
FortiSOAR 7.3.1 Connectors Guide
00-400-000000-20201228

TABLE OF CONTENTS

Change Log

Date	Change Description
2023-01-24	Initial release of 7.3.1

Introduction to connectors

Connectors are used to send and retrieve data from various third-party sources. Using connectors, you can connect to external cyber security tools and perform various automated interactions using FortiSOAR™ playbooks.

FortiSOAR has already developed several connectors that can be used to integrate with many external cyber security tools like SIEMs, such as Splunk, and Ticketing systems such as Jira. Connector-specific documentation that is included with each connector covers the process of installing, configuring, and using these connectors. You can see a list of published connectors on the Fortinet Support Site.

FortiSOAR also provides you with several pre-installed connectors or built-ins that you can use within FortiSOAR playbooks, as a connector step, and perform automated operations. For more information on FortiSOAR Built-in connectors, see the "FortiSOAR Built-in connectors" article.

From FortiSOAR release 7.3.1 onwards, integrations are run using the `fsr-integrations` user instead of the `nginx` user. Therefore, code snippets that try to write on a file system that is outside `/opt/cyops-integrations` or `/tmp` might be impacted and you also need to ensure appropriate permissions have been given to the `fsr-integrations` user.
Writing on file systems using code snippets outside `/tmp` is highly discouraged.

You can write a custom connector that can retrieve data from custom sources. You can then use the connector either standalone or within FortiSOAR playbooks and perform automated operations. You can write a custom connector manually, or you can use the FortiSOAR Connector SDK to develop your custom connector. For more information on writing custom connectors, see the Building your own connector chapter.

Use the Connector Store (Content Hub) in the FortiSOAR UI to easily view, search, install, update, and uninstall connectors that are part of the FortiSOAR repository. You can also specify connector configuration using a jinja variable that contains the connector configuration name.

In Release 7.2.0, the Connector navigation take you to the Content Hub with the relevant 'Connectors' filter selected for browsing ease.

Connector Store

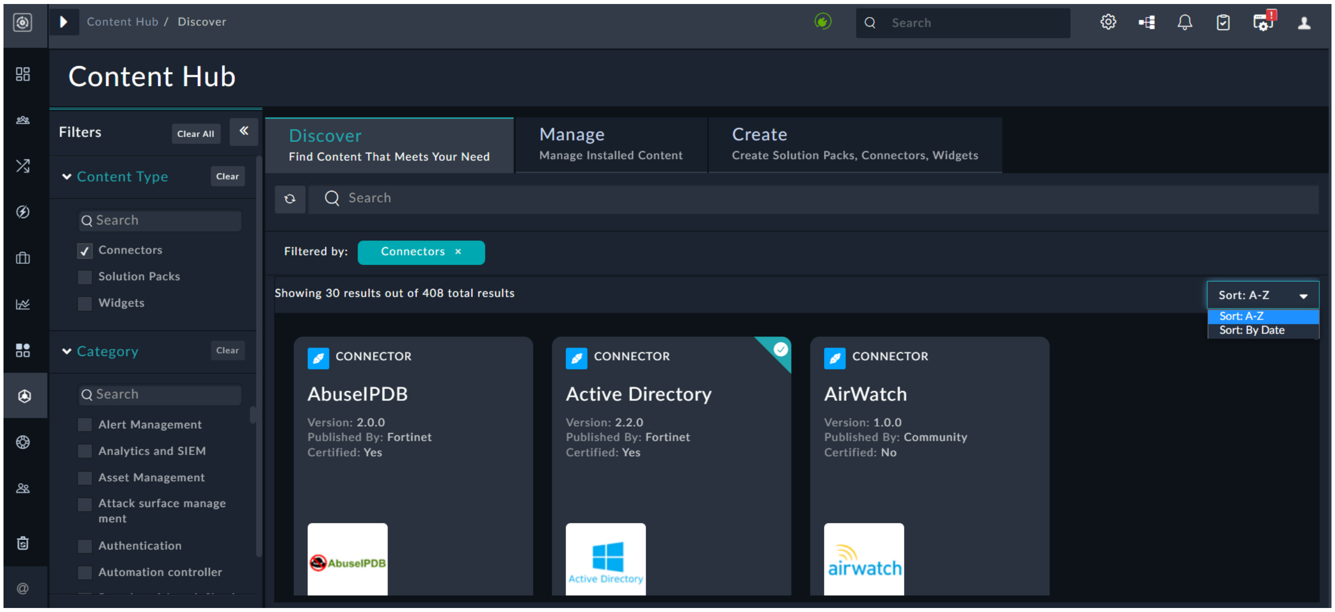
Use the Connector Store to easily view, search, install, upgrade, and uninstall connectors that are part of the FortiSOAR repository. Therefore, you can now perform these operations using the FortiSOAR UI instead of the required CLI access. The Content Hub contains all out-of-the-box reference material and product add-ons such as connectors, widgets, and solution packs; whereas the Connector Store has been filtered to display only Connectors.

You must ensure that repo.fortisoar.fortinet.com is reachable from your FortiSOAR instance. Otherwise, you will see a blank page when you click **Content Hub** or **Automation** > **Connectors** in the left navigation.

Following are the permissions that you must be assigned to perform operations on connectors:

- To work with connectors, i.e., view the connectors listed in the Content Hub and changes made to the Content Hub, you must be assigned a role that has a minimum of `Read` access on the `Application`, `Connectors`, `Playbooks`, and `Content Hub` modules, a minimum of `Create`, `Read`, `Update` permissions on the `Solution Packs` module.
- To install a connector from the Content Hub or to upload a connector to the Content Hub you must be assigned a role that has a minimum of `Read` permission on the `Security` module.
- Apart from the above, to work with connectors, you need appropriate permissions on the `Connectors` and `Playbooks` modules. For example, to install a connector or create a new custom connector, you must be assigned a role that has a minimum of `Create` and `Read` access on the `Connectors` module and `Read` access on the `Playbooks` modules, or to upgrade or configure a connector, you must be assigned a role that has a minimum of `Update` and `Read` access on the `Connectors` module and `Read` access on the `Playbooks` module.

To open the **Connector Store**, go to **Automation** > **Connectors**, or to go to **Content Hub**, click **Content Hub** in the left menu. The **Connector Store** is filtered to display only connectors, whereas the **Content Hub** displays all the add-ons. In this chapter the screenshots included are from the **Content Hub** page; similar screens are displayed on the **Connector Store** page.
The **Discover** tab displays all the add-ons, i.e., Connectors, Widgets, and Solution Packs, available in the Content Hub. Use the Filter panel to filter the connectors by clicking the **>** arrow in the **Content Type** list and then selecting **Connectors**:



Filtered by: Connectors ×

Showing 30 results out of 408 total results

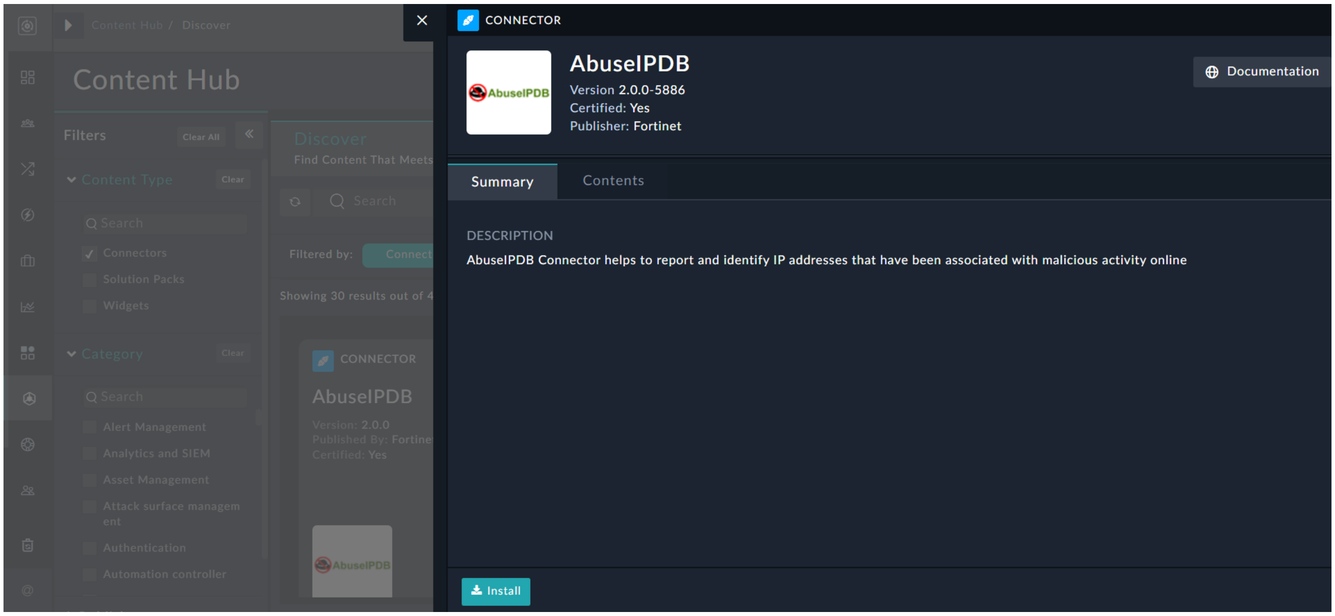
AbuseIPDB — Version: 2.0.0, Published By: Fortinet, Certified: Yes

Active Directory — Version: 2.2.0, Published By: Fortinet, Certified: Yes

AirWatch — Version: 1.0.0, Published By: Community, Certified: No

You can search for a connector by its name in the **Search** field and sort the content alphabetically (A-Z) or by date. Using the **Filters** panel, you can filter the connectors displayed in all the tabs based on varied criteria. For more information on Content Hub, see the *Content Hub* chapter in the "User Guide.' Connectors that are installed appear with a tick on their card. For example, the Active Directory connector in the above image.

To install a connector, click that connector's card, for example, AbuseIPDB, which opens a popup with the connector name:



AbuseIPDB

Version 2.0.0-5886

Certified: Yes

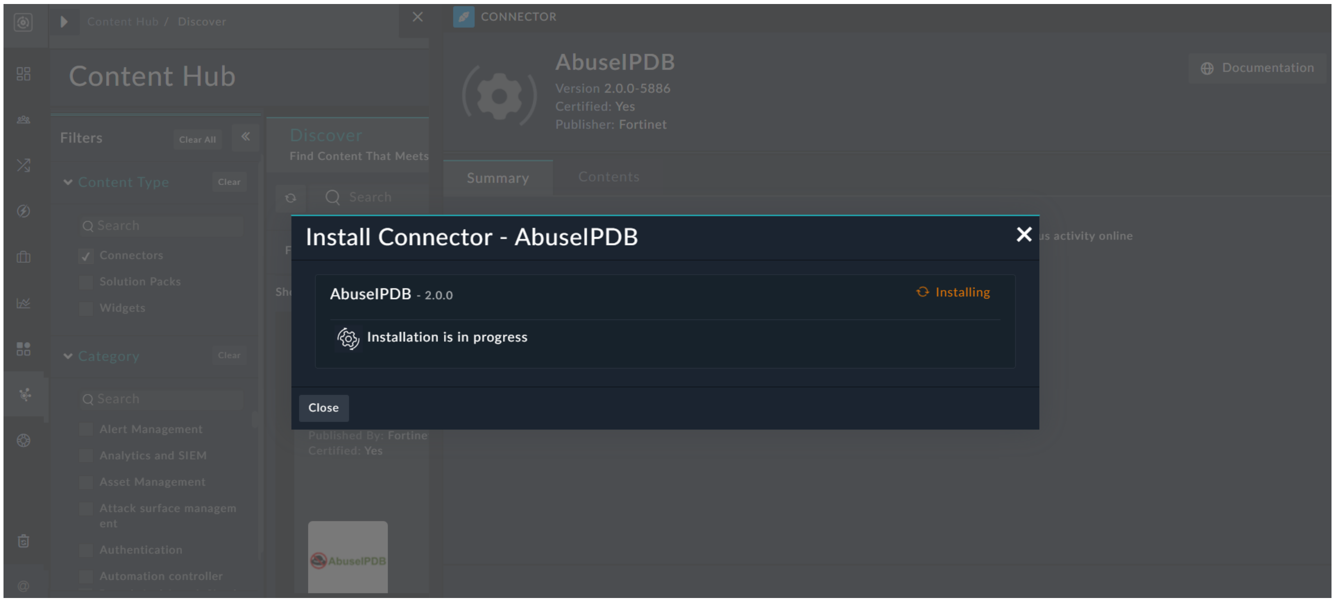
Publisher: Fortinet

DESCRIPTION

AbuseIPDB Connector helps to report and identify IP addresses that have been associated with malicious activity online

The connector popup contains details such as the name and version of the connector, whether the connector is certified or not, who is the publisher of the connector, and the link to the connector documentation. The **Summary** tab contains a brief description of the connector, and the **Contents** tab contains a list of actions the connector can perform.

Click **Install** in the connector name popup to begin the installation of the connector after confirmation, as shown in the following image:



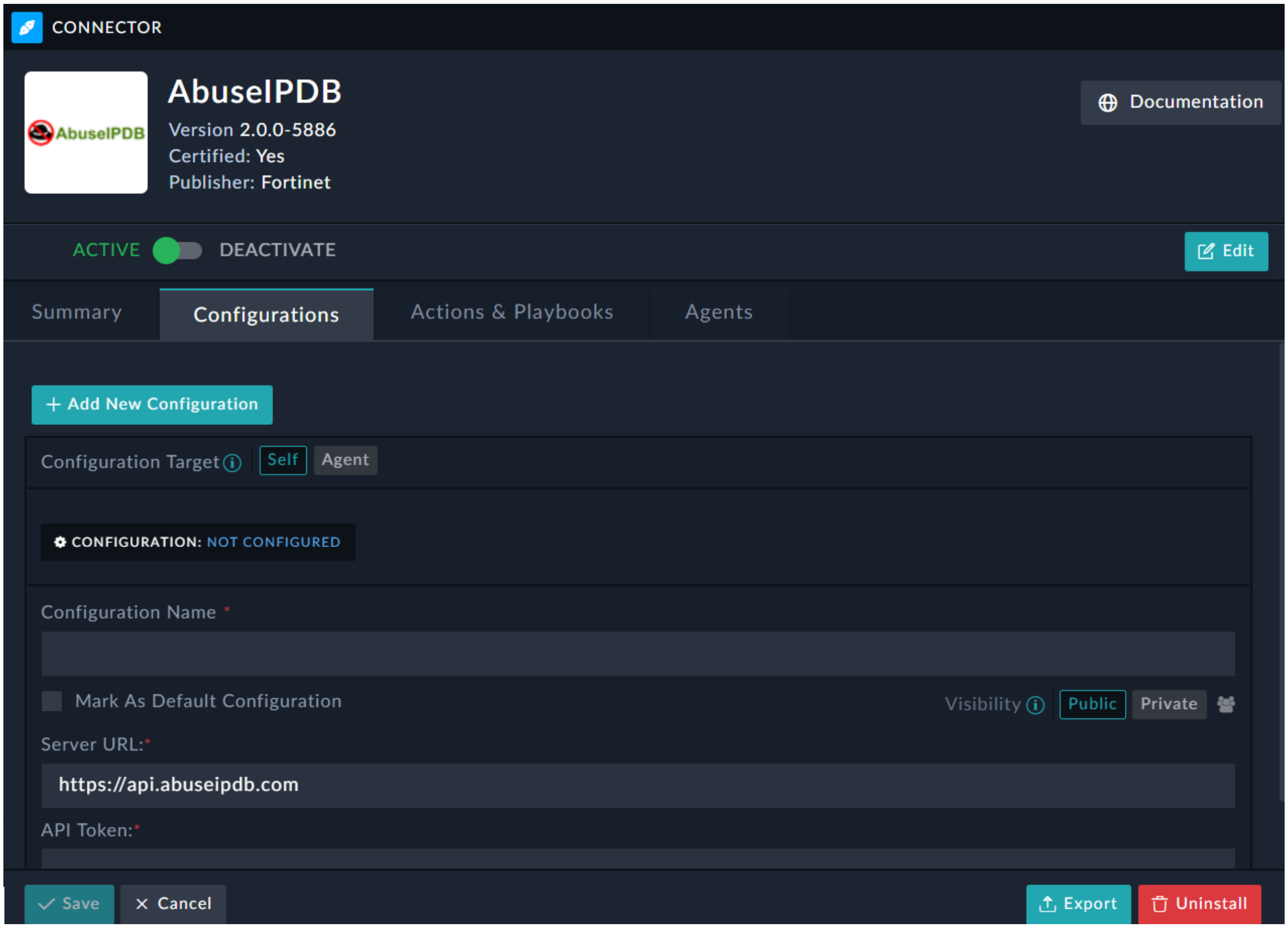
Install Connector - AbuseIPDB

AbuseIPDB - 2.0.0

Installing

Installation is in progress

Once installation is complete, FortiSOAR displays the "Connector Installed successfully" message, and the connector gets added to the **Manage** page. After installing the connector, you must configure the connector by entering the required configuration details in the connector popup:



AbuseIPDB
Version 2.0.0-5886
Certified: Yes
Publisher: Fortinet

Configuration Target ⓘ Self Agent

CONFIGURATION: NOT CONFIGURED

Configuration Name *

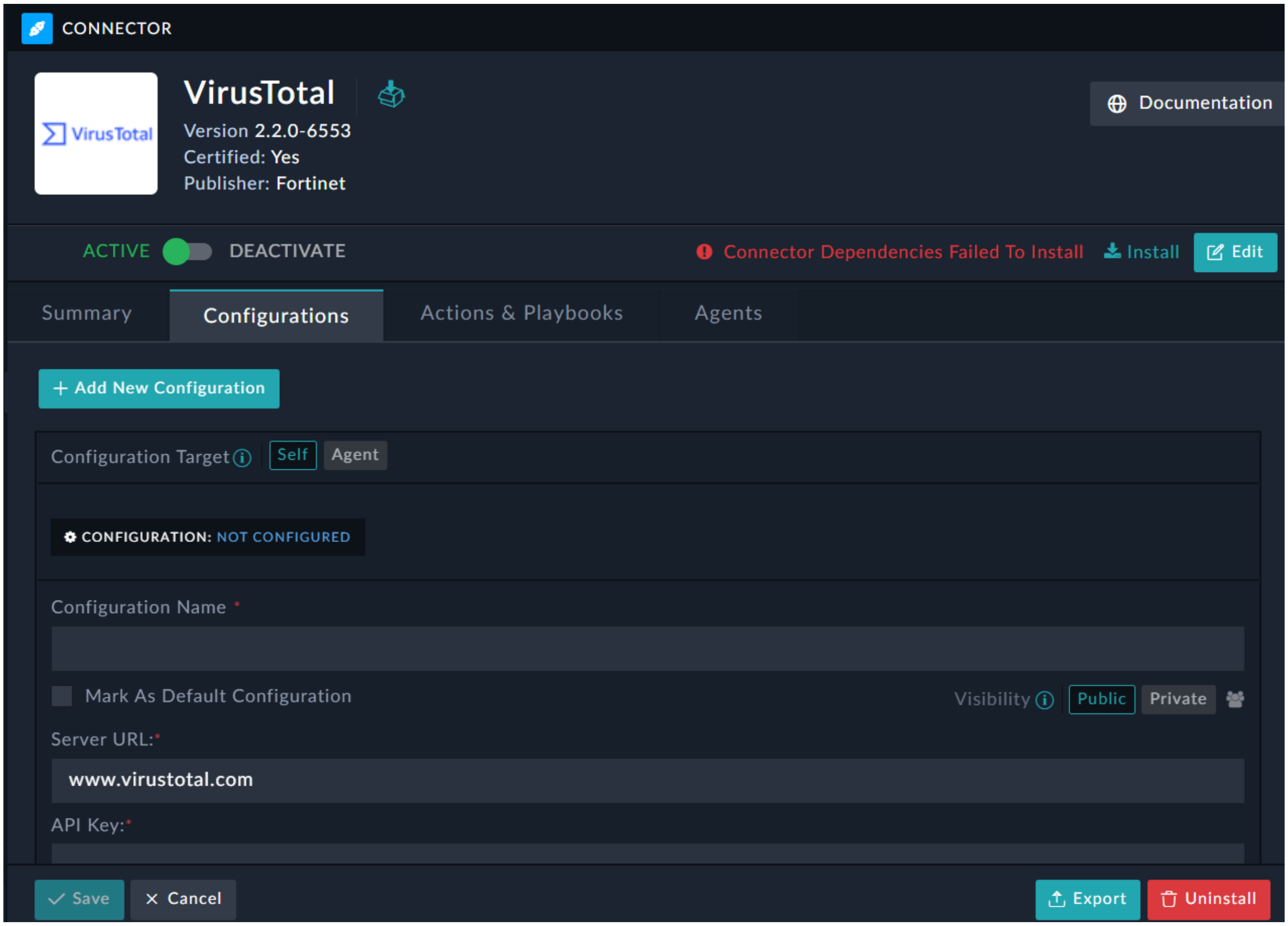
□ Mark As Default Configuration Visibility ⓘ Public Private

Server URL:*
https://api.abuseipdb.com

API Token:*

You can also configure the connector on the **Manage** page by clicking the connector name card, which will also display the same connector popup. For more information, see the Installing or importing and configuring a connector in FortiSOAR section.

Some connectors have dependencies on additional python packages, which if not installed would hamper the functioning of the connector. By default, the dependencies are resolved from pypi.org and repo.fortisoar.fortinet.com. Dependencies could fail to install due to reasons such as the blocking of users' pypi or not having the latest deps repository on repo.fortisoar.fortinet.com. In such cases where the connectors get installed but the dependencies fail to install, FortiSOAR displays a **Connector Dependencies Failed To Install** message on the connector configuration popup as shown in the following image:



VirusTotal
Version 2.2.0-6553
Certified: Yes
Publisher: Fortinet

CONFIGURATION: NOT CONFIGURED

Server URL:* www.virustotal.com

You can try to reinstall the dependencies by clicking the **Install** link.

Dependencies of connectors that are not installed from FortiSOAR Content Hub might fail to install even after a retry since the dependencies for such connectors are not present in the FortiSOAR repository. In such cases, you can download the dependencies from pypi.org and install the dependencies using the information present in https://pip.pypa.io/en/latest/user_guide/#installing-from-local-packages.
To install a package in FortiSOAR, use the following command:

```
sudo -u fsr-integrations /opt/cyops-integrations/.env/bin/pip install <package>
```

Installing or importing and configuring a connector in FortiSOAR

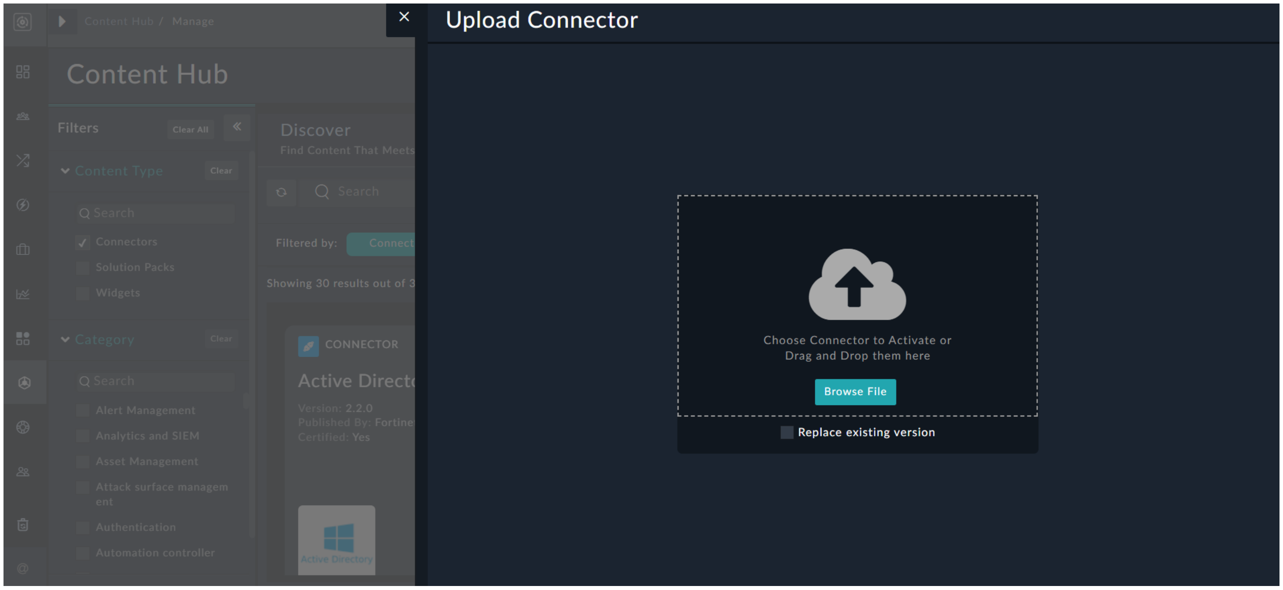
Use the Connector Store to install and configure connectors in FortiSOAR.

To install a connector, you must be assigned a role that has a minimum of `Create` access to the `Connectors` module. To configure connectors into FortiSOAR, you must be assigned a role that has a minimum of `Update` access to the `Connectors` module.

The following procedure describes how to install and configure connector using Connector Store:

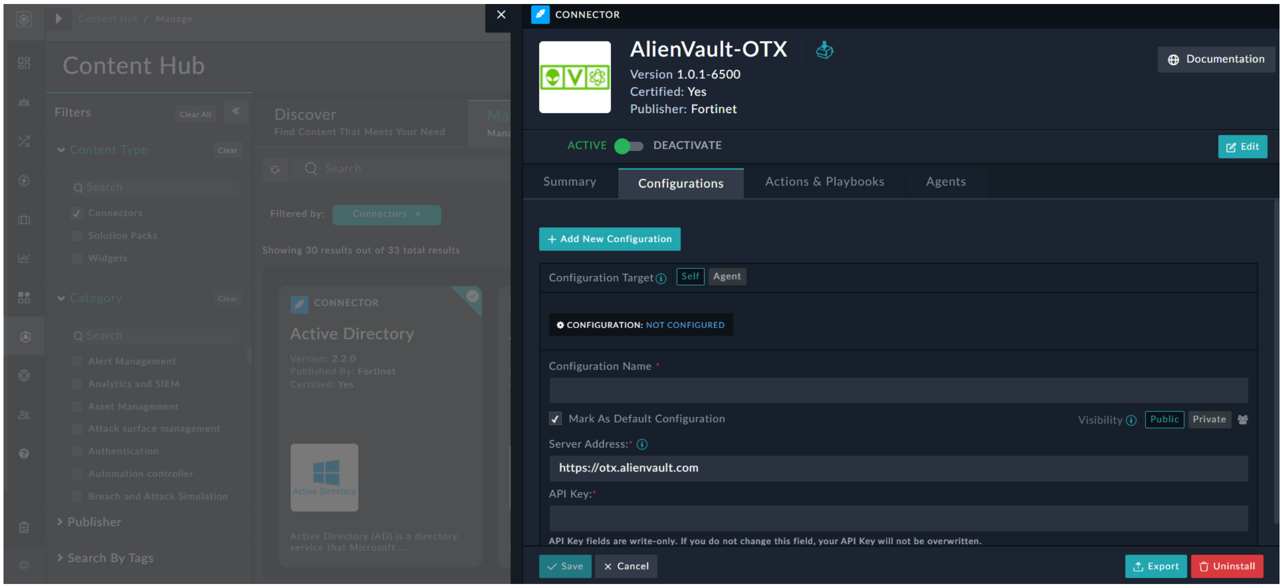
1. Log on to FortiSOAR.
2. On the left navigation pane, click **Connector Store** > **Discover**.
   On the `Discover` page, you will see all the connectors, both which are installed and which are not installed.

Installed connectors appear with a with a tick on their card. You can search for a connector by its name in the **Search** box and sort the connectors either alphabetically or by date.
To install a connector, click that connector's card, which opens a popup with the connector name on which click **Install**. This process is described in the Connector Store section.
Once installation is complete, FortiSOAR displays the "Connector Installed successfully" message, and the connector gets added to the **Manage** page in the 'Active' state; however, you can change the state of the connector from active to inactive by toggling the **ACTIVE** button.
To import a connector (`.tgz`) file into FortiSOAR, click the **Manage** tab and on the `Manage` page, click **Upload** > **Upload Connector** to display the **Upload Connector** popup as shown in the following image:



You can drag and drop the connector .tgz file onto the popup or browse to the .tgz file to install the connector in FortiSOAR. If you have an existing version of the connector on your system, then you can click the **Replace existing version** checkbox, to replace that version of the connector.
You must check that all the connector dependencies are available or install the additional dependencies to ensure that the connector functions as expected.

3. To configure a connector, on the `Manage` page, click the connector card to open the **Connector Configuration** popup. Enter the required configuration details in the connector configuration popup, as shown in the following image:



AlienVault-OTX
Version 1.0.1-6500
Certified: Yes
Publisher: Fortinet

Configuration Target: Self

CONFIGURATION: NOT CONFIGURED

Configuration Name *

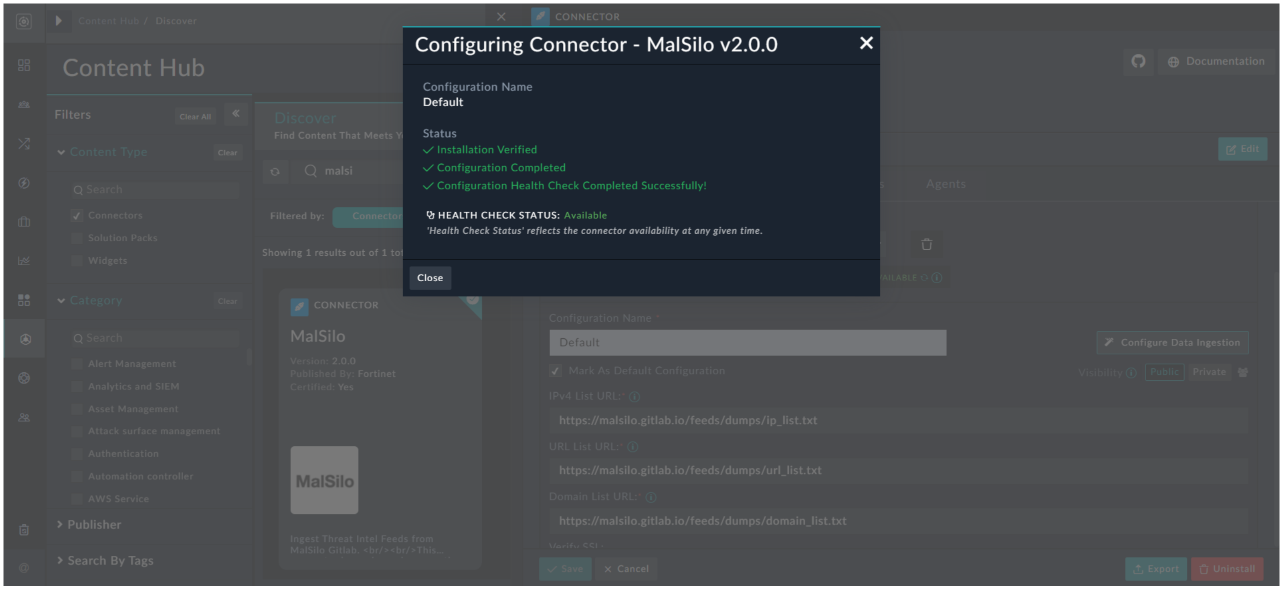
☑ Mark As Default Configuration

Server Address: * https://otx.alienvault.com

API Key: *

API Key fields are write-only. If you do not change this field, your API Key will not be overwritten.

Once you add the configuration details in the connector popup and click **Save**, FortiSOAR opens a modal window that displays the progress of the configuration and the status of the health check while saving the connector configuration:



Configuring Connector - MalSilo v2.0.0

Configuration Name
Default

Status
✓ Installation Verified
✓ Configuration Completed
✓ Configuration Health Check Completed Successfully!

HEALTH CHECK STATUS: Available
'Health Check Status' reflects the connector availability at any given time.

Configuration Name: Default

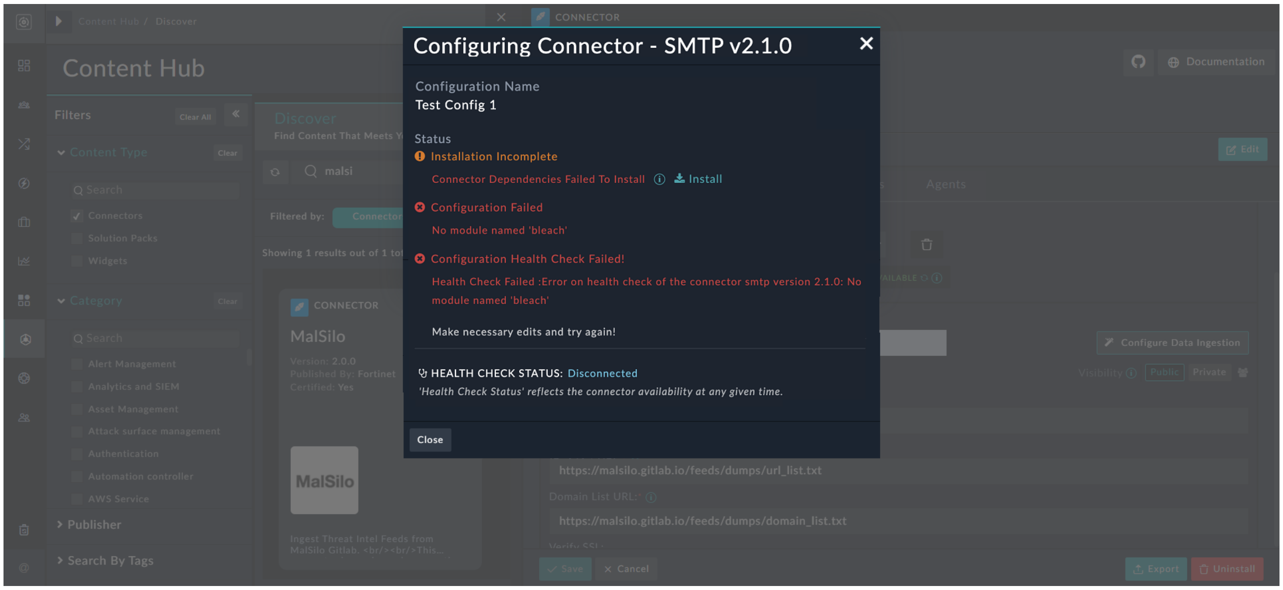
☑ Mark As Default Configuration

IPv4 List URL: https://malsilo.gitlab.io/feeds/dumps/ip_list.txt

URL List URL: https://malsilo.gitlab.io/feeds/dumps/url_list.txt

Domain List URL: https://malsilo.gitlab.io/feeds/dumps/domain_list.txt

All errors or warnings related to connector configuration, including missing dependencies, invalid configurations, and health check status, are displayed in the modal window:



Configuring Connector - SMTP v2.1.0

Configuration Name
Test Config 1

Status
Installation Incomplete
Connector Dependencies Failed To Install Install
Configuration Failed
No module named 'bleach'
Configuration Health Check Failed!
Health Check Failed :Error on health check of the connector smtp version 2.1.0: No module named 'bleach'

Make necessary edits and try again!

HEALTH CHECK STATUS: Disconnected
'Health Check Status' reflects the connector availability at any given time.

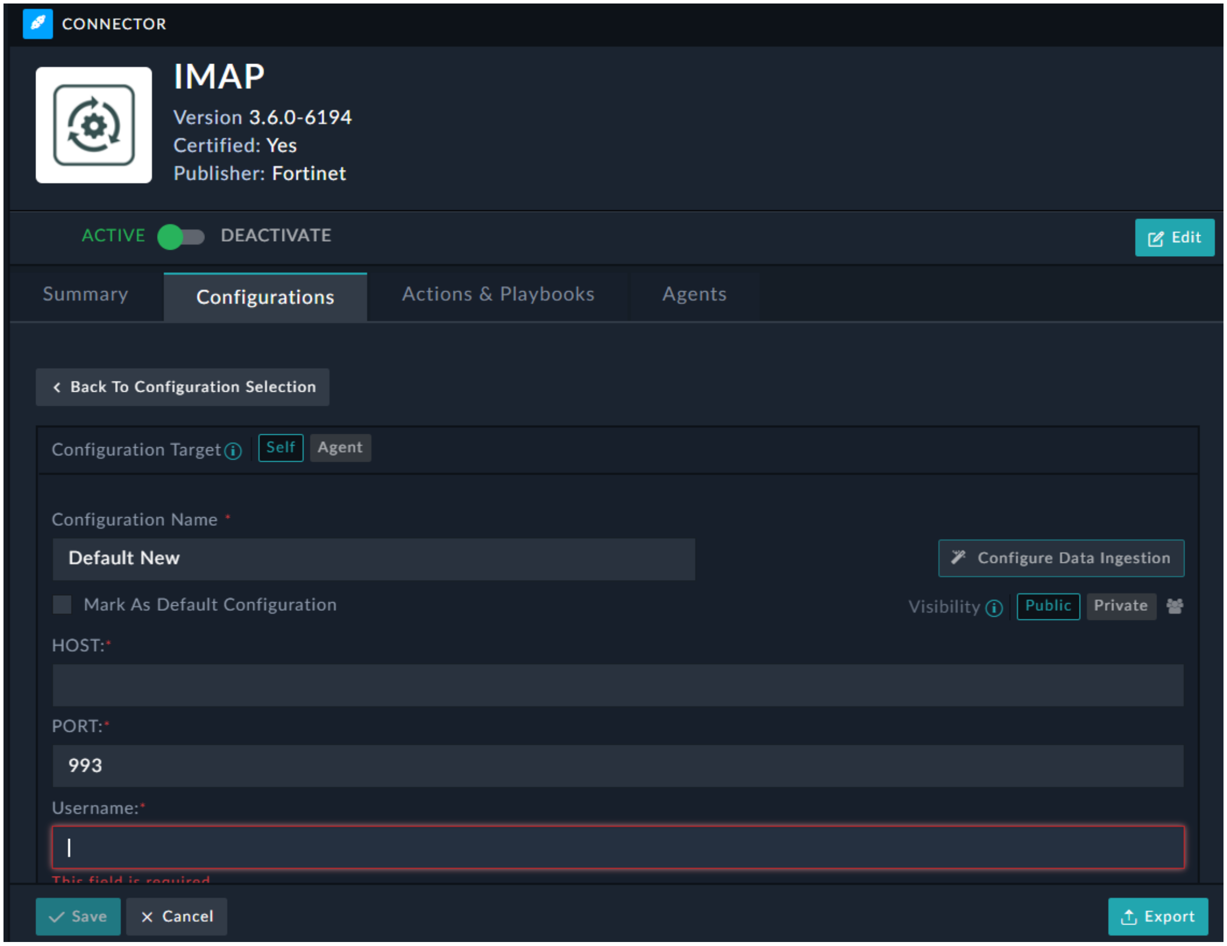
The health check status and missing dependencies are also displayed on the connector configuration popup. You can choose to try to reinstall the dependencies by clicking the **Install** link that appears alongside the **Connector Dependencies Failed To Install** message on the modal window or the connector configuration popup as described in the Connector Store section.

Note: Once you have installed a connector dependency on a FortiSOAR node using the **Install** link on the connector's Configurations dialog, then that dependency is installed seamlessly on the other nodes of the HA cluster.

Notes on Configuration:

- If you are configuring the connector for the first time, then in the Configuration Name field add a name of the configuration.
  Note: You can add multiple configurations for your connector, therefore, you must add a unique `Name` for each configuration in the **Configuration Name** field.
- If you have previous versions of a connector and you are configuring a newer version of that connector, with the same configuration parameters, then FortiSOAR fetches the configuration and input parameters of the latest available version of that connector. For example, if you have 1.0.0 and 2.0.0 versions of the Database connector and you are configuring the 2.0.0 version of the Database connector, then while configuring the 2.0.0 version, FortiSOAR will fetch the configuration and input parameters from the 1.0.0 version of the Database connector. You can review the configuration and input parameters, and then decide to change them or leave them unchanged.
- To add a new configuration, click the **+Add New Configuration** button, and then add the name of the configuration and specify the configuration parameters. You can click the **< Back to Configuration Selection**

button to go back to the Configuration page:



IMAP

Version 3.6.0-6194

Certified: Yes

Publisher: Fortinet

Configuration Target: Self

Configuration Name *: Default New

□ Mark As Default Configuration

Visibility: Public

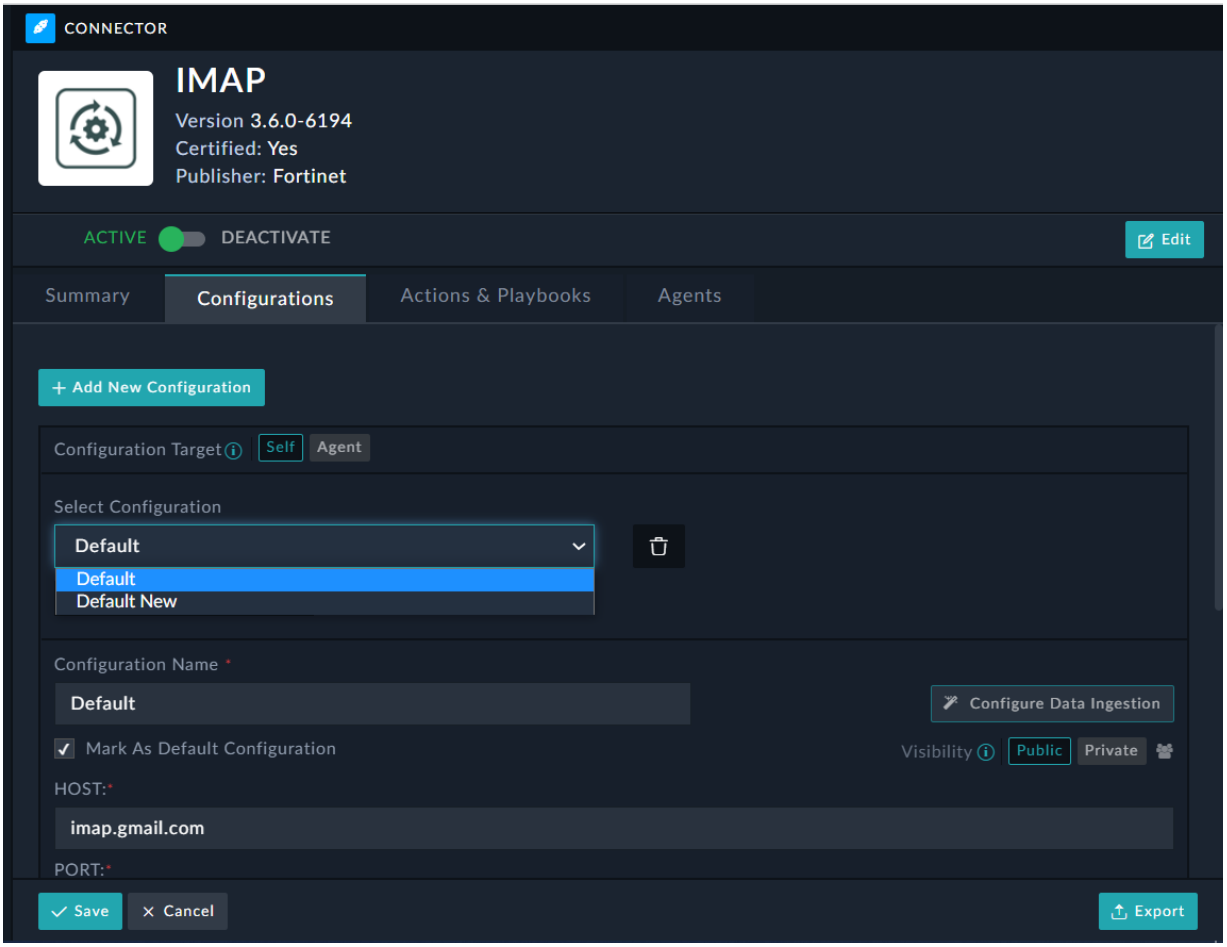
HOST:*

PORT:* 993

Username:*

This field is required

- If you want to update an existing configuration, then select the configuration from the **Select Configuration** drop-down list and update the configuration parameters. If there is only one configuration, then that configuration will be selected automatically.



IMAP
Version 3.6.0-6194
Certified: Yes
Publisher: Fortinet

Select Configuration: Default
- Default
- Default New

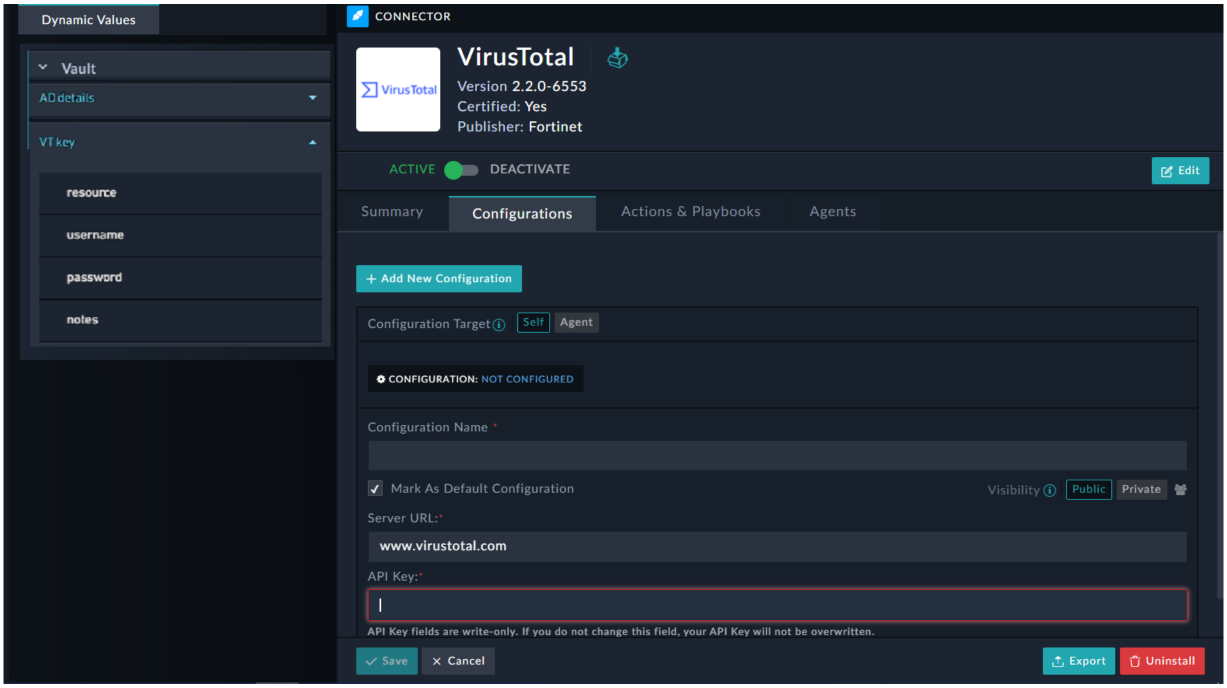
Configuration Name *: Default

☑ Mark As Default Configuration

HOST:*: imap.gmail.com

PORT:*

- Check the **Mark As Default Configuration** option to make the selected configuration, the default configuration of this connector, on the particular FortiSOAR instance. This connector will point to this configuration by default, and it is important that you mark at least one configuration as default, so that the connector uses that configuration for performing actions.
  Important: In the case of the SMTP connector, you must ensure that this option is selected for the configuration that is to be used for sending system notifications.
- The `password` type fields in FortiSOAR include encryption and decryption. Passwords are encrypted before saving them into the database and decrypted when they are used in actions. In case of an upgrade, connectors that are already installed will work with stored passwords. If your administrator has configured an external vault to securely store your organization's sensitive data and credentials, then you can use the Dynamic Values dialog to enter the credentials for your connector as shown in the following image:



VirusTotal

Version 2.2.0-6553

Certified: Yes

Publisher: Fortinet

Configuration Target: Self

CONFIGURATION: NOT CONFIGURED

Configuration Name *

☑ Mark As Default Configuration

Visibility: Public

Server URL:* www.virustotal.com

API Key:*

API Key fields are write-only. If you do not change this field, your API Key will not be overwritten.

For information on Dynamic Values, see the *Dynamic Values* chapter in the "Playbooks Guide." For information on Password Vault, see the *Security Management* chapter in the "Administration Guide."

- You can also define the ownership of connector configurations by setting the visibility of a connector's configuration to 'Private'. For more information, see the Defining the ownership of a connector's configuration section.
- You can configure connectors for the current FortiSOAR node, an agent, which is used to remotely run actions, or both. For more information on agents and how to run remote actions using agents, see the *Segmented Network support in FortiSOAR* chapter in the "Administration Guide." You can configure on the current node (Self) or on a remote Agent node (Agent) by clicking the **Self**, which is the default, or **Agent** buttons besides `Target`. You can configure multiple configurations for a connector on both the current node and the agent node. To configure and execute connector actions on the "Agent" node, click **Agent**, and from the **Select Agent** drop-down list select the agent on which you want to run the connector actions. If there is only one agent installed, then that agent will be selected automatically.
- Connectors also include a **Verify SSL** field, that specifies whether the SSL certificate for the server is to be verified or not. By default, this option is set as `True`. For more information, see How the connector framework verifies the server certificate when it's self-signed.
- To view the documentation associated with a connector, click the **Documentation** link on the top-right corner of the connector configuration pane.

Points to be considered for connector configurations while upgrading to a newer version of the connector

If you are upgrading a connector to a newer version, you must be assigned a role that has a minimum of `Upgrade` access to the `Connectors` module. For example, if you are upgrading the Symantec Security Analytics connector version from v1.0.0 to v2.0.0, then keep a note of the following points:

- Existing (older) connector configuration fields retain their value, i.e., the value from the older configuration will be displayed in the configuration pane of the newer connector version. New connector configuration field(s), if any, will be added to the connector configuration pane.
- If the newly added configuration field is mandatory, and FortiSOAR has specified its default value (in the `info.json` file of the connector), then the configuration pane of the newer version of the connector will contain the default value for this configuration field. For more information on the connector framework and the `info.json` file, see the Building your own connector chapter.

For information on common connector framework issues, see the *Common connector framework errors* section in the Debugging common playbook and connector issues article present in the Fortinet Knowledge Base.

- If the newly added configuration field is mandatory, and FortiSOAR has not specified its default value (in the `info.json` file of the connector), then the configuration pane of the newer version of the connector will contain a `blank` value for this configuration field. If you also do not specify a value for this mandatory configuration field, then the connector configuration pane will display **Partially Configured**, and an error will also be displayed in the Playbook Execution Log. For more information on the Playbook Execution Log, see the *Debugging and Optimizing Playbooks* chapter in the "Playbooks Guide."
- If the field type of a mandatory configuration field is changed from the older version to the newer version, for example from a text field to a drop-down list, then the value of that field will ***not*** be retrieved from the older version. However, if FortiSOAR has specified its default value (in the `info.json` file of the connector), then that value will be displayed for this configuration field in the configuration pane of the newer version of the connector. If however FortiSOAR has not defined the default value and you also do not specify a value for this mandatory configuration field, then the configuration pane of the newer version of the connector will contain a `blank` value for this configuration field, and the connector configuration pane will display **Partially Configured**. An error will also be displayed in the Playbook Execution Log. For more information on the Playbook Execution Log, see the *Debugging and Optimizing Playbooks* chapter in the "Playbooks Guide."
- If the newly added configuration field is optional, and FortiSOAR has specified its default value (in the `info.json` file of the connector), then the configuration pane of the newer version of the connector will contain the default value for this configuration field. If there is no default value is set, then its value is set as `blank`.

Installing or Updating a FortiSOAR connector using the CLI

All connectors published by FortiSOAR are delivered using a FortiSOAR repository. Use the Connector Store to easily install, update and install connectors that are part of the FortiSOAR repository.

The recommended way to install a connector is by using the Connector Store.

You could also set up your FortiSOAR repository and use the `yum` command to install connectors as a *root* user:

```
# yum install cyops-connector-<connectorname>
```

After you install a connector using the `yum` command the new connector is not reflected until you refresh the `Connectors` page.

To update a FortiSOAR-published connector use the following command:

```
# yum update cyops-connector-<connectorname>
```

To remove a FortiSOAR-published connector use the following command:

```
# yum remove cyops-connector-<connectorname>
```

If you delete a FortiSOAR-published connector using the Connector page in FortiSOAR then you cannot reinstall the RPM of the same connector because the RPM of the connector does not get deleted. Therefore, to remove a FortiSOAR-published connector, you must use the `yum remove cyops-connector-<connectorName>` command.

Some of the connectors have dependencies on additional python packages. During connector installation, these dependencies are also installed using `pip`. The default pip settings point to the pypi.python.org repository for downloading these packages. We have also added an alternate repository on repo.fortisoar.fortinet.com to host the connector dependencies. If your instance restricts access to pypi.python.org, the installer fallbacks to this alternate repository for installing the dependencies. However, the connector installation might take longer if there are multiple dependencies since it first tries to fetch each of those from pypi.python.org. In such a case, it is recommended to switch to repo.fortisoar.fortinet.com as the main repository for the python packages also. You can switch the main repository to repo.fortisoar.fortinet.com so by editing the `pip.conf` file located at: `/opt/cyops-integrations/.env/pip.conf` as follows: `index-url=`https://repo.fortisoar.fortinet.com/connectors/deps/simple/. If some dependencies fail while installing the connector, you can use the connector configuration popup to try and reinstall the failed dependencies as described in the Connector Store section.

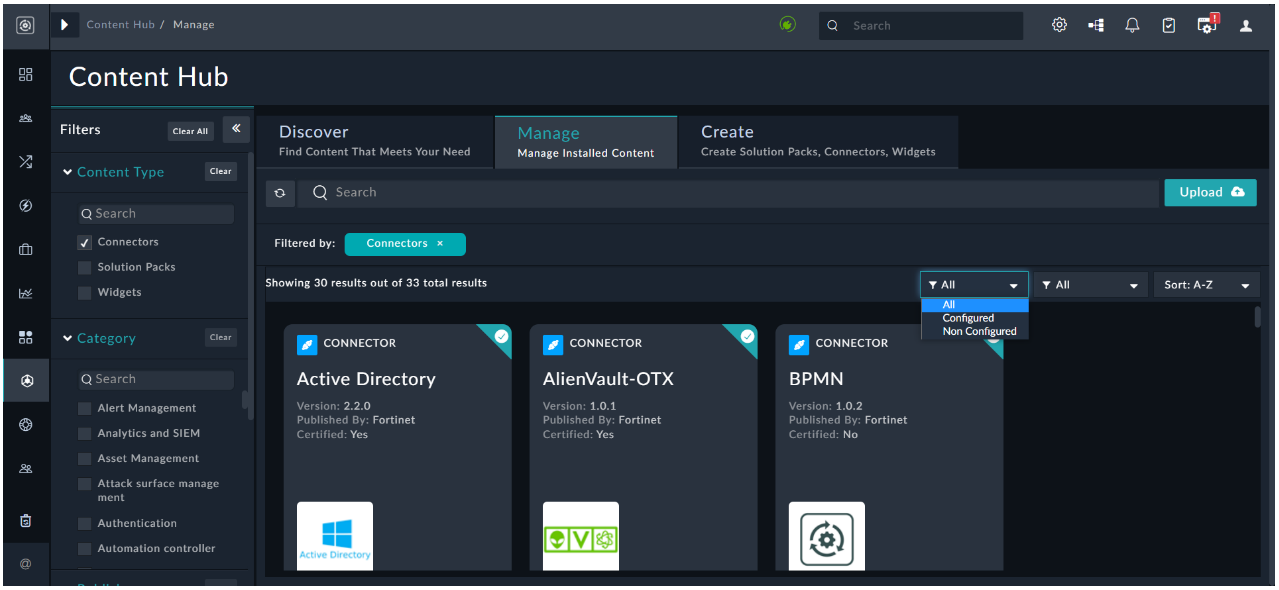
You can also change other relevant pip settings such as altering the timeout or retry count setting. See https://pip.pypa.io/en/stable/user_guide/#config-file for a listing of the relevant pip settings. For example, to increase the timeout value add the following in the `pip.conf` file: `timeout = 60`.

For information on installing connectors on an FSR agent, see the *Segmented Network Support* chapter in the "Administration Guide."

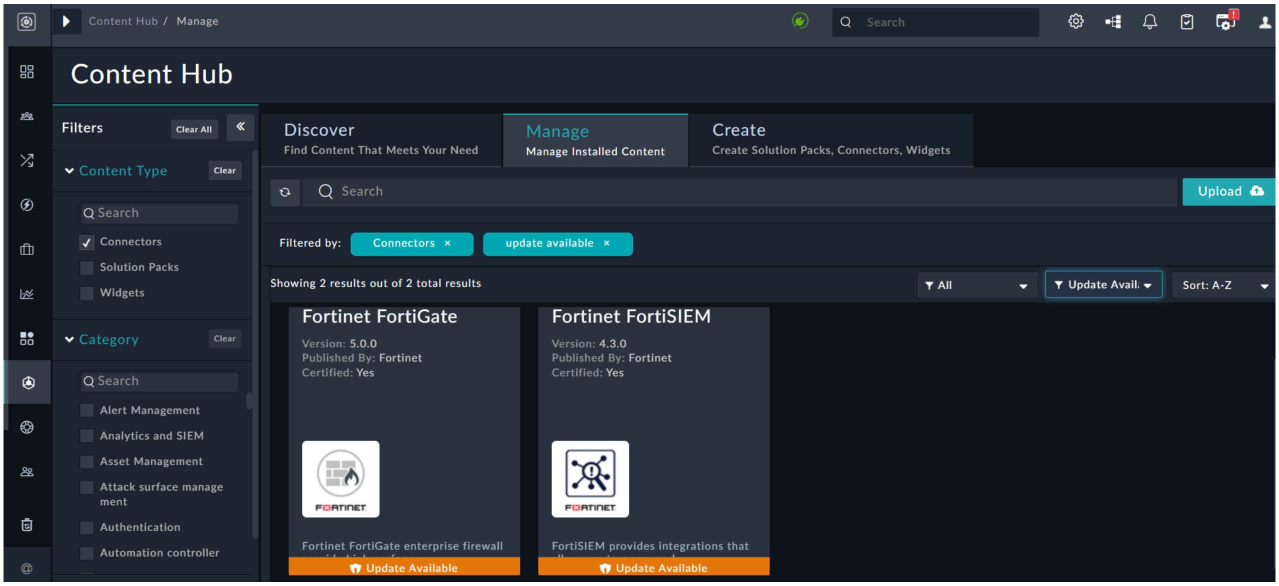
Working with connectors

On the `Manage` page, you will see the list of installed connectors in the card view. You can perform the following operations on this page:

- You can search for a connector by its name in the **Search** box and sort the connectors either alphabetically or by date. Also, when you select **Connectors** as the **Content Type** by default, **Non Configured** connectors are displayed in the **Manage** tab. You can select **Configured** from the drop-down to view the configured connectors.



- Similarly, you can filter the installed connectors that have an upgraded version, by selecting **Upgrade Available** from the second drop-down list. For example, version 5.0.0 of the Fortinet FortiGate connector is installed on your system and Fortinet has released a newer version of this connector, then you will see the Fortinet FortiGate connector filtered by the Upgrade Available filter, and you can also see **Update Available** link on the connector's card:

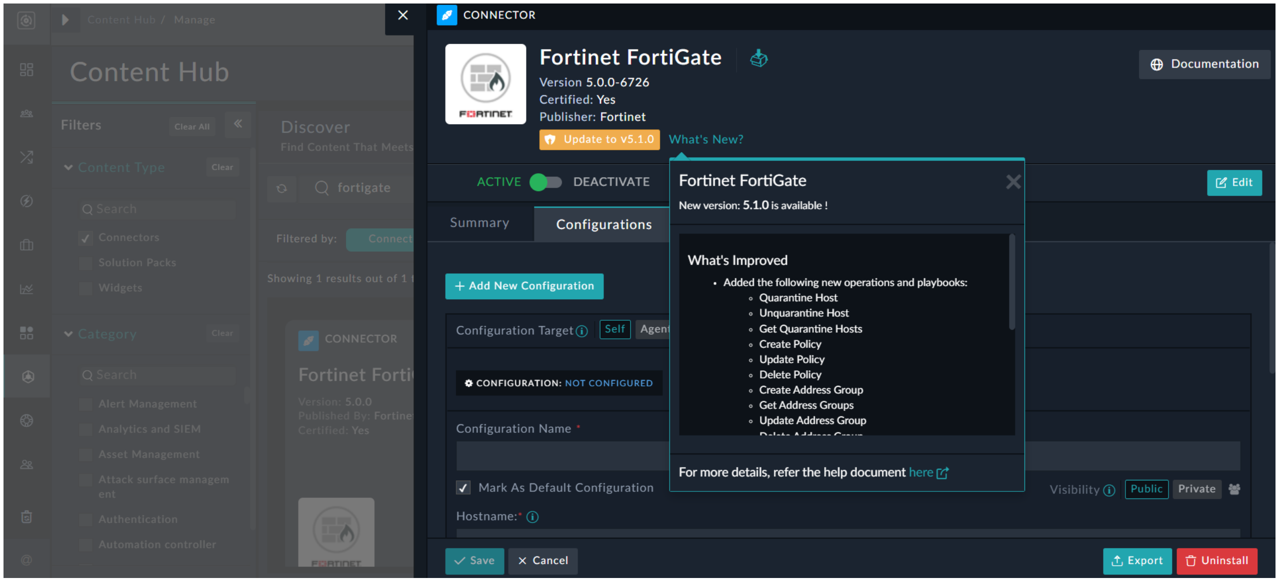


Showing 2 results out of 2 total results

Fortinet FortiGate
Version: 5.0.0
Published By: Fortinet
Certified: Yes
Fortinet FortiGate enterprise firewall

Fortinet FortiSIEM
Version: 4.3.0
Published By: Fortinet
Certified: Yes
FortiSIEM provides integrations that

To update a connector, click the **Update Available** link on the connector card. When you click on a connector that has an updated version, you will see a '**What's new?**' section on the configuration page, besides the **Update to <version number>** button. Clicking the **What's New?** link displays the highlights of the new version, which you can view to understand what you can expect when you upgrade your connector version. Click the **Update to <version number>** button to upgrade your connector.



Fortinet FortiGate
Version 5.0.0-6726
Certified: Yes
Publisher: Fortinet

Fortinet FortiGate
New version: 5.1.0 is available !

What's Improved

- Added the following new operations and playbooks:
  - Quarantine Host
  - Unquarantine Host
  - Get Quarantine Hosts
  - Create Policy
  - Update Policy
  - Delete Policy
  - Create Address Group
  - Get Address Groups
  - Update Address Group

For more details, refer the help document here

Configuration Name *

☑ Mark As Default Configuration

Hostname: *

You can perform the following operations on the connector configuration popup (click the connector card in the `Manage` page to open the connector configuration popup) of a configured connector:

- To perform a Health Check by clicking the **Refresh** icon that is present in the `Health Check` bar. The Health Check checks if the configuration parameters you have specified are correct and if connectivity can be established to the specified server, endpoint, or API. If all the details are correct and the connectivity to the server can be established, then on the Connectors page, **Available** is displayed in the health check dialog. If any or all the details are incorrect or if the connectivity to the server cannot be established then on the Connectors page, **Disconnected** is displayed in the health check dialog.

- To modify a connector according to your requirements, click the **Edit** on the connector configuration popup. However, to edit a connector, you must make a local copy of the connector and then make the changes in the same, and then you can save and publish the connector as defined in the Building your own connector chapter.
- To uninstall a connector, click the **Uninstall** button, in the connector configuration popup. FortiSOAR displays a Confirmation dialog, click Confirm to uninstall the connector. FortiSOAR displays the "Connector uninstalled successfully" message and the connector is removed from the Manage page. To uninstall a connector, you must be assigned a role that has a minimum of `Delete` access on the `Connectors` module.
- To export the .tgz file of the connector, click the **Export** button.
- To view the list of actions that can be performed by the connector and the playbook file that is bundled with the connector, click the **Actions & Playbooks** tab. Clicking the 'Actions' option on this tab, you can also restrict specific connector actions to specific roles, i.e., users with only those roles would be able to perform that action. For more information, see the Restricting specific connector actions to specific roles section,
- To view the list of agents on which the connector is installed, click the **Agents** tab. For more information on agents and how to run remote actions using agents, see the *Segmented Network support in FortiSOAR* chapter in the "Administration Guide."

How the connector framework verifies the server certificate when it is self-signed

The connector framework is explained in the Building your own connector chapter.

All connector calls are made by the python requests library reading the certificate from `/opt/cyops-integrations/.env/lib/python3.9/site-packages/certifi/cacert.pem`. Therefore, for any connector, when you set `verify_ssl` to `true`, and it's a self-signed cert, then the `cacert` must be appended to this file. If it's a chain of trust, then you must add the entire chain in the `pem` format. You must also ensure that the server address added in the connector configuration matches the CN in the certificate.

A `.key` file has the path to a PEM encoded file containing the private key. A `.pem` file has the path to a PEM encoded file containing the certificate (or certificate chain) that will be presented when requested.

If you are using the HTTPS proxy for external connections, then you must ensure that the proxy certificate is added here also, if the `Verify SSL` is set to true in the connector configuration.

Some commands that you can use to get the `pem` certificate chain:

```
# openssl s_client -connect {HOSTNAME}:{PORT} -showcerts
```

OR

If you have the certificate already in a `.crt`, `.cer`, `.der` format, then you need to convert to the `pem` format: `# openssl x509 -inform der -in certificate.cer -out certificate.pem`

Role-based Access Control for connector actions

From version 7.0.0 onwards, FortiSOAR introduces Role-based Access Control (RBAC) for connectors, i.e., 'Connector Administrators', i.e., users with Connector Read and Update permissions and Security Read permission can allow access to only certain teams or users, based on roles, to perform certain connector actions. For example, the administrator might want to allow a "Block IP" action to be performed by only certain teams or users in the organization.

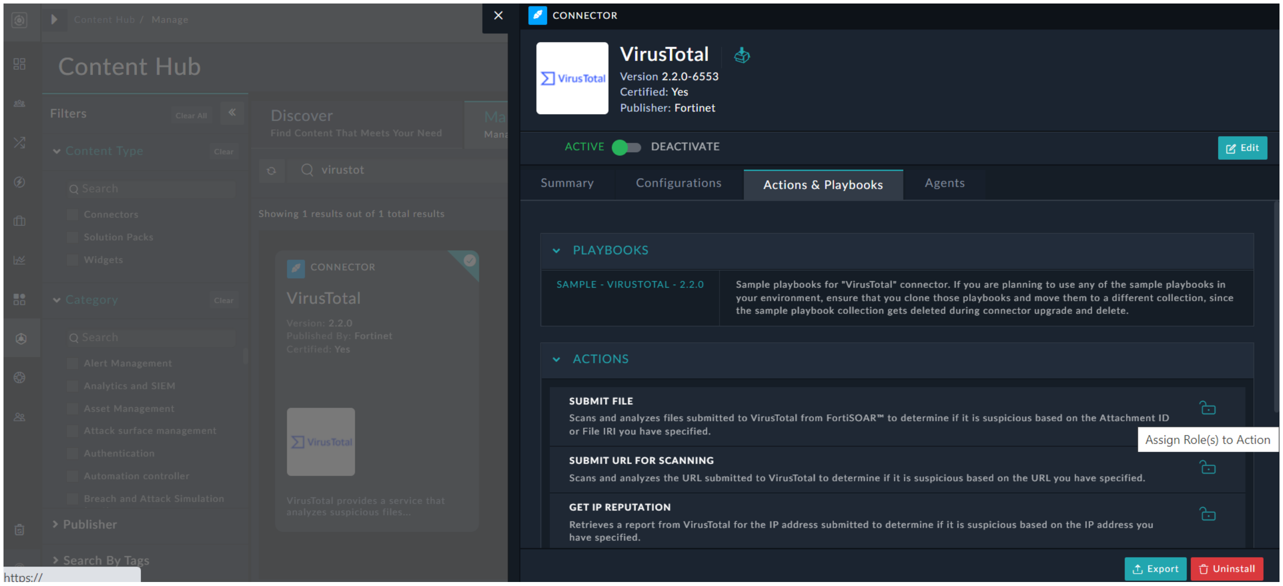
The ownership of connector configurations can also be defined, by marking the connector configuration as 'Private'; thereby, controlling who can view and execute that particular connector configuration.

You must configure access control for connector configurations and actions separately for the tenant nodes. Access control for connector configurations and actions cannot be configured from the master.

Restricting specific connector actions to specific roles

By default, all users and roles with appropriate permissions, as defined in the Connector Store section, can view and execute all the connector actions; however, connector administrators can restrict specific actions to specific roles as follows:

1. Log on to FortiSOAR.
2. On the left navigation pane, click **Connector Store** > **Manage**.
3. Select the connector whose connector actions you want to restrict to certain users, based on the roles or specific set of roles that have been assigned to open the `Connector Configurations` page.
4. Click the **Actions & Playbooks** tab:



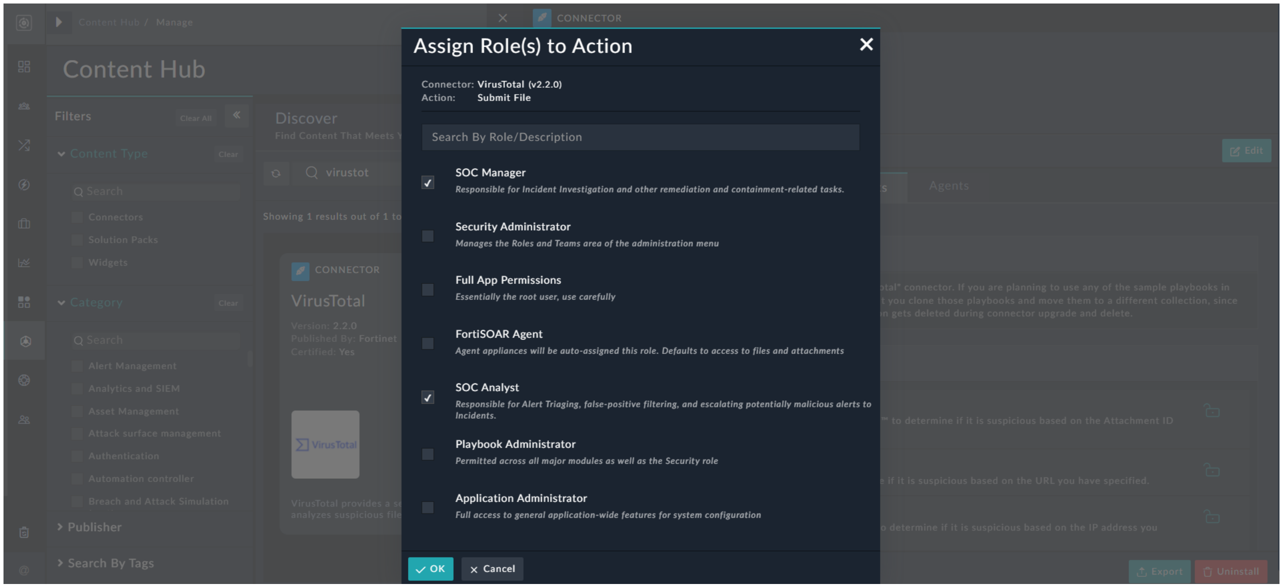
SAMPLE - VIRUSTOTAL - 2.2.0: Sample playbooks for "VirusTotal" connector. If you are planning to use any of the sample playbooks in your environment, ensure that you clone those playbooks and move them to a different collection, since the sample playbook collection gets deleted during connector upgrade and delete.

SUBMIT FILE
Scans and analyzes files submitted to VirusTotal from FortiSOAR™ to determine if it is suspicious based on the Attachment ID or File IRI you have specified.

SUBMIT URL FOR SCANNING
Scans and analyzes the URL submitted to VirusTotal to determine if it is suspicious based on the URL you have specified.

GET IP REPUTATION
Retrieves a report from VirusTotal for the IP address submitted to determine if it is suspicious based on the IP address you have specified.

5. From the **Actions** tab, in the row that contains the action that you want to restrict, click the **Assign Role(s) to Action** icon to display the `Assign Role(s) to Action` dialog:



Assign Role(s) to Action

Connector: VirusTotal (v2.2.0)
Action: Submit File

Search By Role/Description

☑ SOC Manager
Responsible for Incident Investigation and other remediation and containment-related tasks.

□ Security Administrator
Manages the Roles and Teams area of the administration menu

□ Full App Permissions
Essentially the root user, use carefully

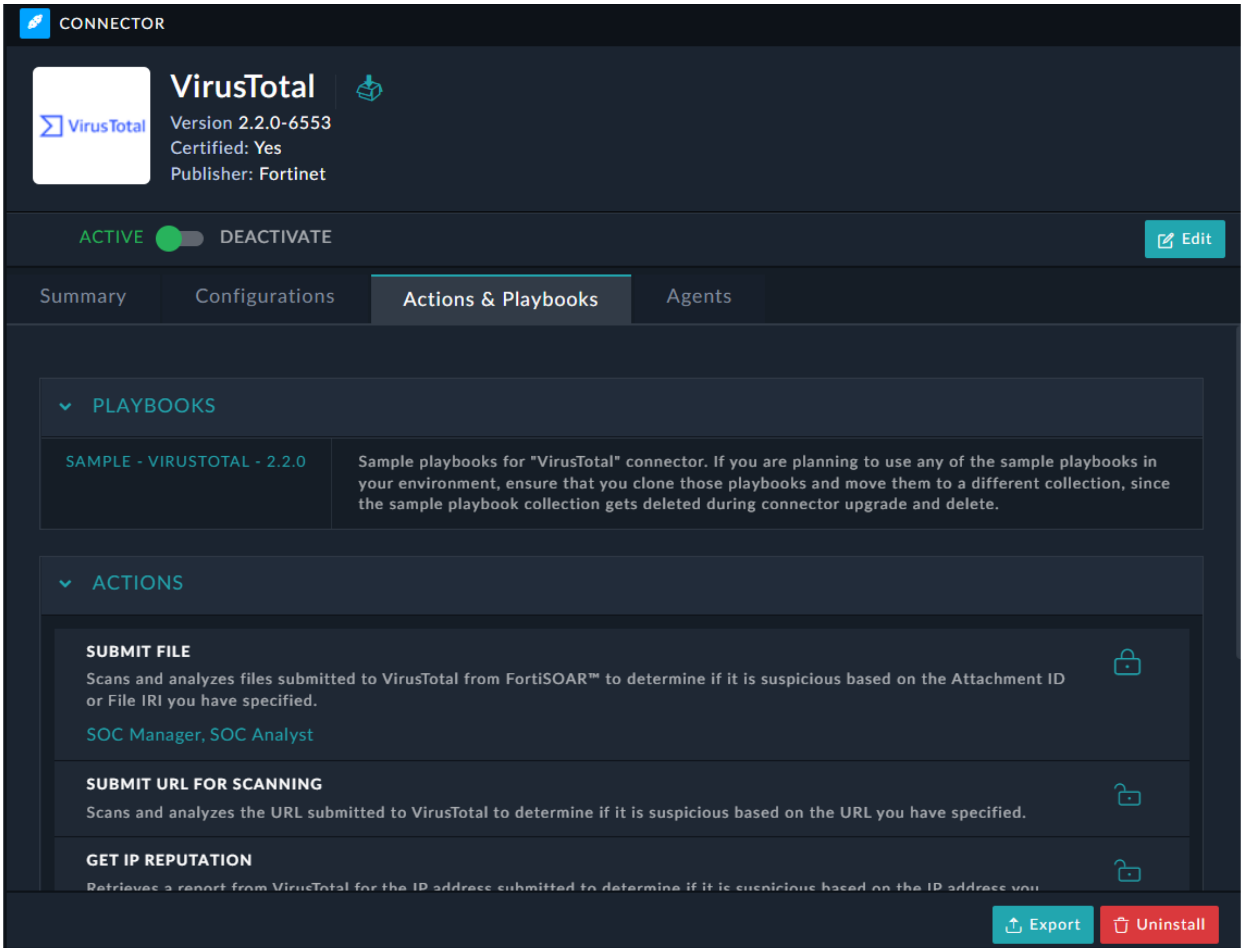
□ FortiSOAR Agent
Agent appliances will be auto-assigned this role. Defaults to access to files and attachments

☑ SOC Analyst
Responsible for Alert Triaging, false-positive filtering, and escalating potentially malicious alerts to Incidents.

□ Playbook Administrator
Permitted across all major modules as well as the Security role

□ Application Administrator
Full access to general application-wide features for system configuration

6. Select the roles that would be able to perform this action and then click **OK**.
Once a particular action is assigned to specific roles, you can see a **Roles Assigned** bar in that action's role, which specifies the roles that the users should be assigned to perform that connector action:



VirusTotal

Version 2.2.0-6553

Certified: Yes

Publisher: Fortinet

SAMPLE - VIRUSTOTAL - 2.2.0

Sample playbooks for "VirusTotal" connector. If you are planning to use any of the sample playbooks in your environment, ensure that you clone those playbooks and move them to a different collection, since the sample playbook collection gets deleted during connector upgrade and delete.

SUBMIT FILE

Scans and analyzes files submitted to VirusTotal from FortiSOAR™ to determine if it is suspicious based on the Attachment ID or File IRI you have specified.

SOC Manager, SOC Analyst

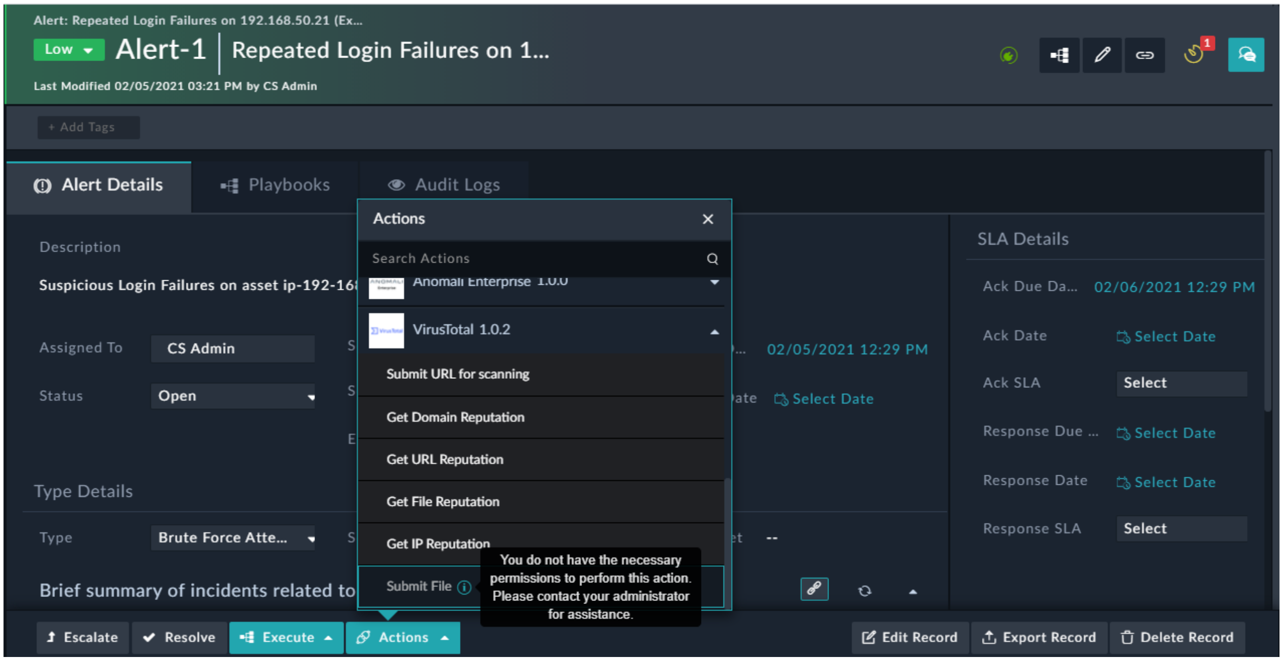
SUBMIT URL FOR SCANNING

Scans and analyzes the URL submitted to VirusTotal to determine if it is suspicious based on the URL you have specified.

GET IP REPUTATION

Retrieves a report from VirusTotal for the IP address submitted to determine if it is suspicious based on the IP address you

Therefore, in this case, if there are users who are not assigned the 'Full App Permissions', 'Playbook Administrator' or 'SOC Analyst' role, and if they try to invoke this action directly on a record, they will be unable to run that action as the action will be disabled:



Description: Suspicious Login Failures on asset ip-192-16

Assigned To: CS Admin

Status: Open

Type Details

Type: Brute Force Atte...

Brief summary of incidents related to

Ack Due Da... 02/06/2021 12:29 PM

Ack Date: Select Date

Ack SLA: Select

Response Due ...: Select Date

Response Date: Select Date

Response SLA: Select

You do not have the necessary permissions to perform this action. Please contact your administrator for assistance.

Notes:

- When a connector is upgraded to a higher version, if there are any roles defined for any connector action in the previous version of the connector, those roles are carried forwarded to the upgraded version.
- The Playbook designer displays all the connector actions irrespective of the roles of the user creating the playbook. Also, while executing a connector action from the playbook, FortiSOAR does not perform any role check. You can create a playbook as a 'Private' playbook to restrict access. For more information, see the "Playbooks Guide."
- When a connector action is being executed directly on a record, the roles of the logged-in user will be matched against the roles permitted for the action.

Defining the ownership of a connector's configuration

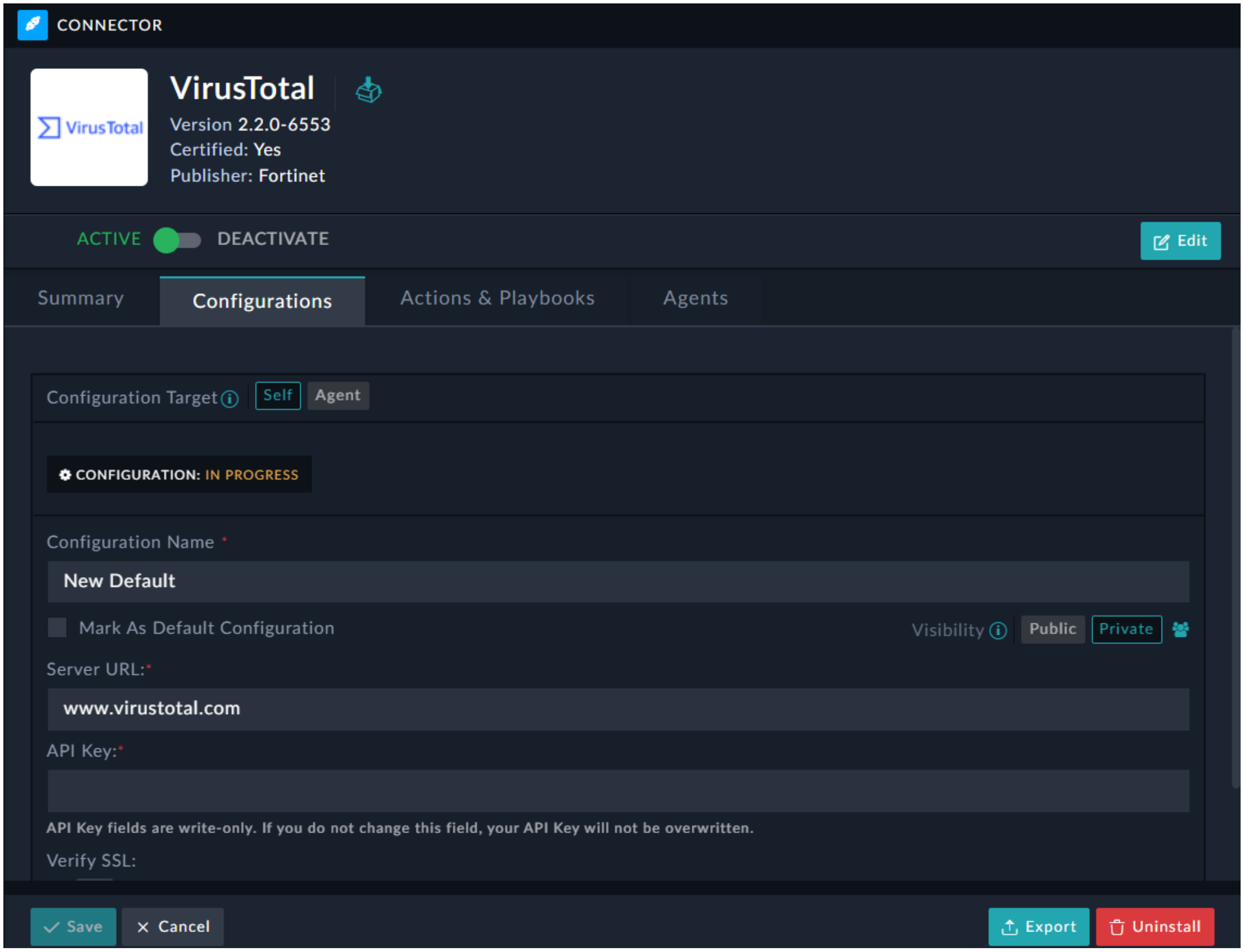
The ownership of connector configurations can also be defined by setting the visibility of a connector's configuration to 'Private'. By default, the connector configuration is set to 'Public', i.e., it can be viewed by anyone who has appropriate permissions, as defined in the Connector Store section. By setting a connector's configuration to 'Private', you are assigning ownership of that connector configuration to particular teams and thereby only allowing the assigned team owners the right to view and execute that particular connector configuration.

You must configure access control for connector configurations and actions separately for the tenant nodes. Access control for connector configurations and actions cannot be configured from the master.

To set the visibility of connector configurations to 'Private' to restrict the teams who can view and execute that particular connector configuration, do the following:

1. Log on to FortiSOAR.
2. On the left navigation pane, click **Automation** > **Connectors** > **Installed**.
3. Select the connector whose configuration you want to set to 'Private', i.e., whose configuration you want to restrict to certain teams. By default, connector configurations are set to 'Public'.

4. To mark a connector configuration as 'Private', click the **Private** button beside the **Visibility** parameter:



CONNECTOR

VirusTotal

Version 2.2.0-6553

Certified: Yes

Publisher: Fortinet

Configuration Target: Self | Agent

CONFIGURATION: IN PROGRESS

Configuration Name *: New Default

☐ Mark As Default Configuration

Visibility: Public | Private

Server URL:*: www.virustotal.com

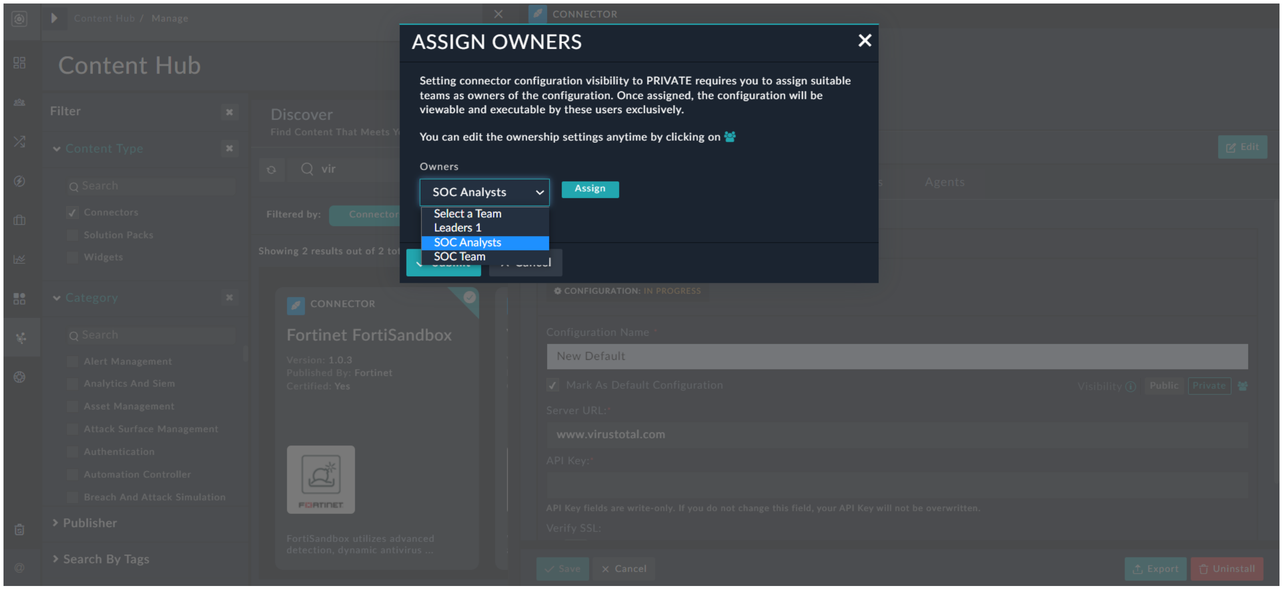
API Key:*

API Key fields are write-only. If you do not change this field, your API Key will not be overwritten.

Verify SSL:

5. To assign the connector configuration to a particular team, once you have clicked **Private**, click the **Teams** (👥) icon to display the `Assign Owners` dialog.
6. In the `Assign Owners` dialog, you will see that the logged-in user's teams are assigned ownership, by default. Click the **Red Cross** beside the teams whose ownership you want to remove, and select the teams that you want to

configure as owners for this connector configuration from the **Owners** drop-down list and then click **Assign**:



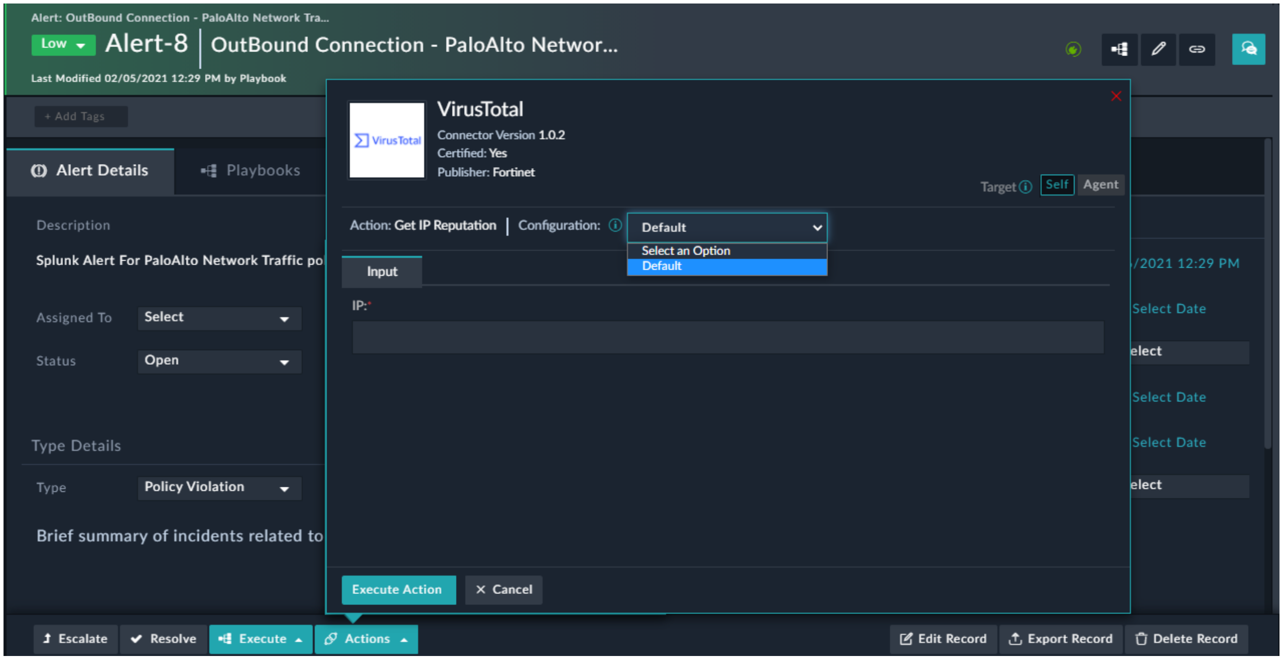
ASSIGN OWNERS

Setting connector configuration visibility to PRIVATE requires you to assign suitable teams as owners of the configuration. Once assigned, the configuration will be viewable and executable by these users exclusively.

You can edit the ownership settings anytime by clicking on

Owners: SOC Analysts

7. Click **Submit** to assign ownership of this connector configuration.
   Now, in this case, if there are users who are not part of the 'SOC Analysts' team, and if they want to run the actions of the VirusTotal connector on a record, for example, they will observe that the "New Default" configuration will not be visible and only the "Default" configuration is visible:



VirusTotal
Connector Version 1.0.2
Certified: Yes
Publisher: Fortinet

Action: Get IP Reputation | Configuration: Default

Therefore, users can see only those connector configurations to which they have access, as is the case with the above example, the 'New Default' configuration is visible only to users that belong to the 'SOC Analysts' team. However, users belonging to teams other than 'SOC Analysts' can execute playbooks containing a connector step with the 'New Default' configuration that has been created by 'SOC Analysts' users.

Troubleshooting

From FortiSOAR release 7.3.0 onwards, PIP dependencies packages for connectors are located at `/opt/cyops/configs/integrations/packages/lib/python3.9/site-packages/`

Troubleshooting connector installation issues

Issues uninstalling integrations pip dependencies

This topic uses an example of the `tzlocal` package and demonstrates uninstalling this package. You can replace `tzlocal` with the name of the package that you want to uninstall.

Run the following command to uninstall integrations pip dependencies (the `tzlocal` package):

```
# sudo -u fsr-integrations /opt/cyops-integrations/.env/bin/pip uninstall tzlocal
```

The following message get returned:

```
Found existing installation: tzlocal 4.2
Not uninstalling tzlocal at
/opt/cyops/configs/integrations/packages/lib/python3.9/site-packages, outside
environment /opt/cyops-integrations/.env
Can't uninstall 'tzlocal'. No files were found to uninstall.
#
```

Resolution:

If you want to uninstall the pip (`tzlocal`) package, do the following:

1. Run the following command to find the files that are part of the pip package:
   ```
   find /opt/cyops/configs/integrations/packages/lib/python3.9/site-packages -name
   tzlocal*
   ```
2. Manually remove files that are listed as the output of previous '`find`' command.

Issues downgrading integrations pip dependencies

Usually, you should not be required to downgrade the integrations pip dependencies; however, if after debugging you require to downgrade, then use the following steps.

Resolution:

1. Remove the pip dependencies files that are already installed.
   For example if you want to downgrade `tzlocal`, first find the files that are already installed for `tzlocal` and then remove those files:
   a. Run the following command to find the files that are part of the pip package (`tzlocal`):
      ```
      find /opt/cyops/configs/integrations/packages/lib/python3.9/site-packages -name
      tzlocal*
      ```
   b. Manually remove files that are listed as the output of previous '`find`' command.

2. Install the required version using the following command:
   ```
   sudo -u fsr-integrations /opt/cyops-integrations/.env/bin/pip install
   tzlocal==<required-version-here>
   ```

Troubleshooting connector usage issues

Handling rounding of large integer values

You might find that fields with large integer values get round-off causing issues while working with connectors. For example, when you submit a URL or file to FortiSandBox, a SID (Session ID) gets created. This session is further used to get job id "JID", using which indicator information can be retrieved. However, due to the large integer values getting rounded-off, FortiSOAR replaces the last four digits of the "JID" to 0000. Due to this you would be unable to retrieve the information for submitted indicator from FortiSandBox. This issue occurs as JavaScript lacks an explicit integer type in JavaScript; and instead represents all numbers in double-precision 64-bit floating-point format as defined by the IEEE 754-2008 standard. Under this standard, very large integers that cannot be exactly represented are automatically rounded. To be precise, the Number type in JavaScript can only safely represent integers between -9007199254740991 (-(253-1)) and 9007199254740991 (253-1). Any integer value that falls out of this range might lose precision.

Resolution

As a workaround to these limitations, large integers can be represented using the 'String' type, i.e., instead of passing integers directly such as, 123456, pass integers in a single quote such as, '123456'. The Twitter API, for example, adds a string version of IDs to objects when responding with JSON. For now, such a large number must be returned as a 'String' in the response to the connector action.

Building your own connector

You can write a custom connector that can retrieve data from custom sources. You can then use the connector either standalone or within FortiSOAR playbooks and perform automated operations. From version 7.0.0, you can create your own connector using the Connector Wizard. You can also write a custom connector manually, or you can use the FortiSOAR Connector SDK to develop your custom connector.

The process of installing, configuring, and using FortiSOAR-provided connectors is defined in the respective connector-specific documentation.

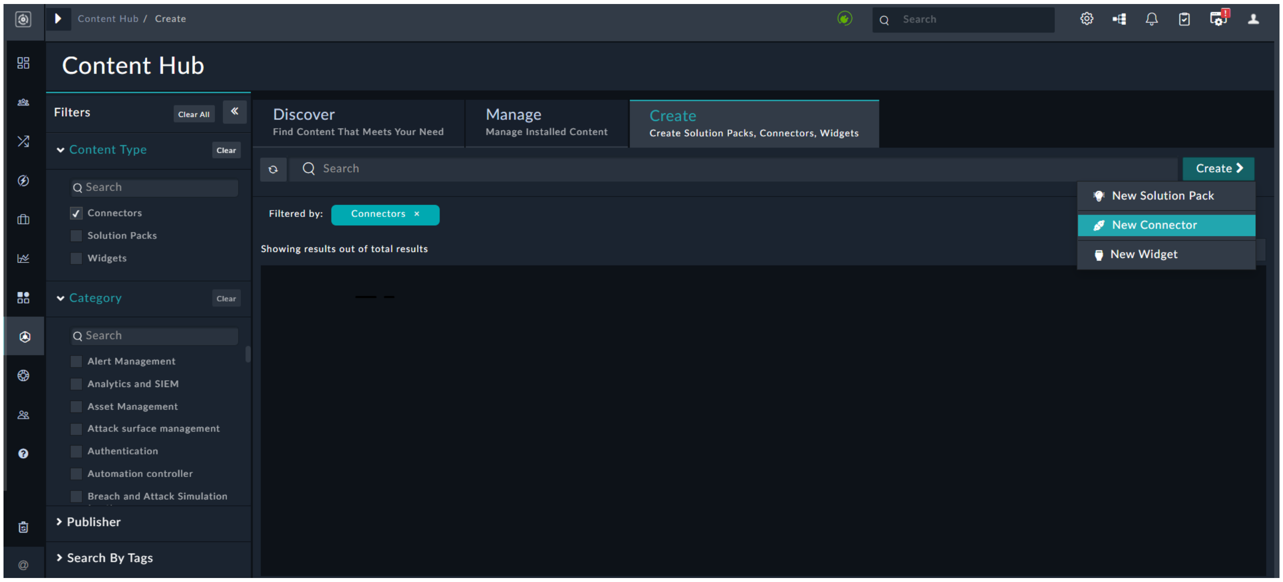
For permissions required to build a connector, see the Introduction to Connectors chapter.

Building a connector using the Connector Wizard

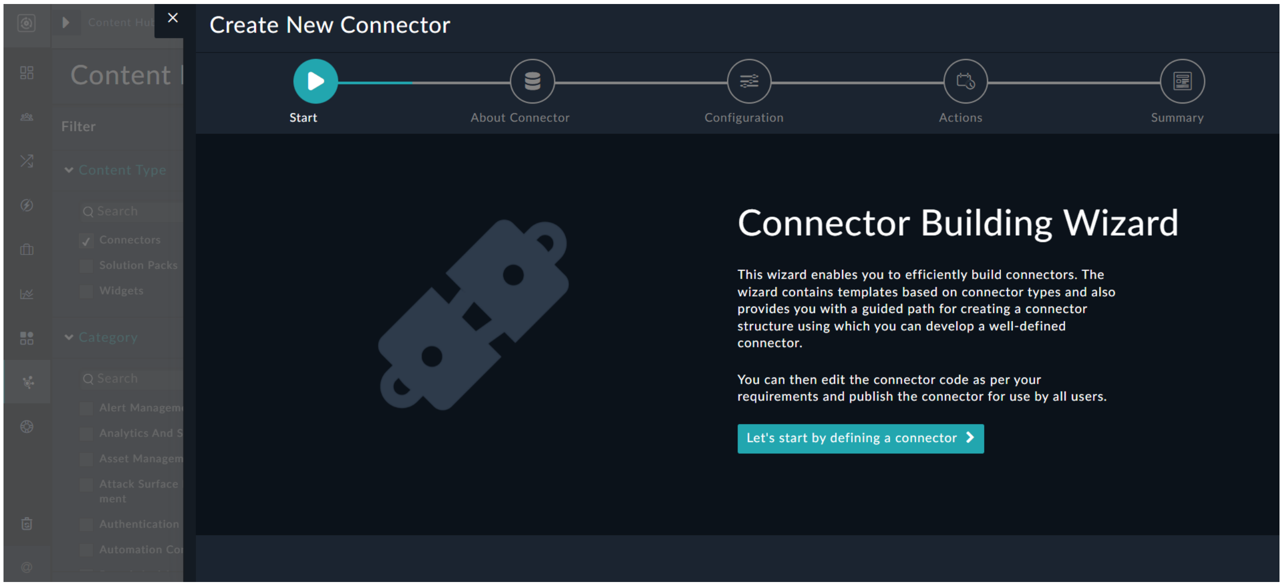
Use the Connector wizard to create a new connector. You can also edit an existing connector according to your requirements by clicking **Edit** on a connector card in the **Manage** tab.

To create a new connector, do the following:

1. Log on to FortiSOAR.
2. On the left navigation pane, click **Content Hub** > **Create** or **Automation** > **Connectors** > **Create**.
3. To create a new connector, click **Create** > **New Connector:**



This opens the "Connector Wizard":



Connector Building Wizard

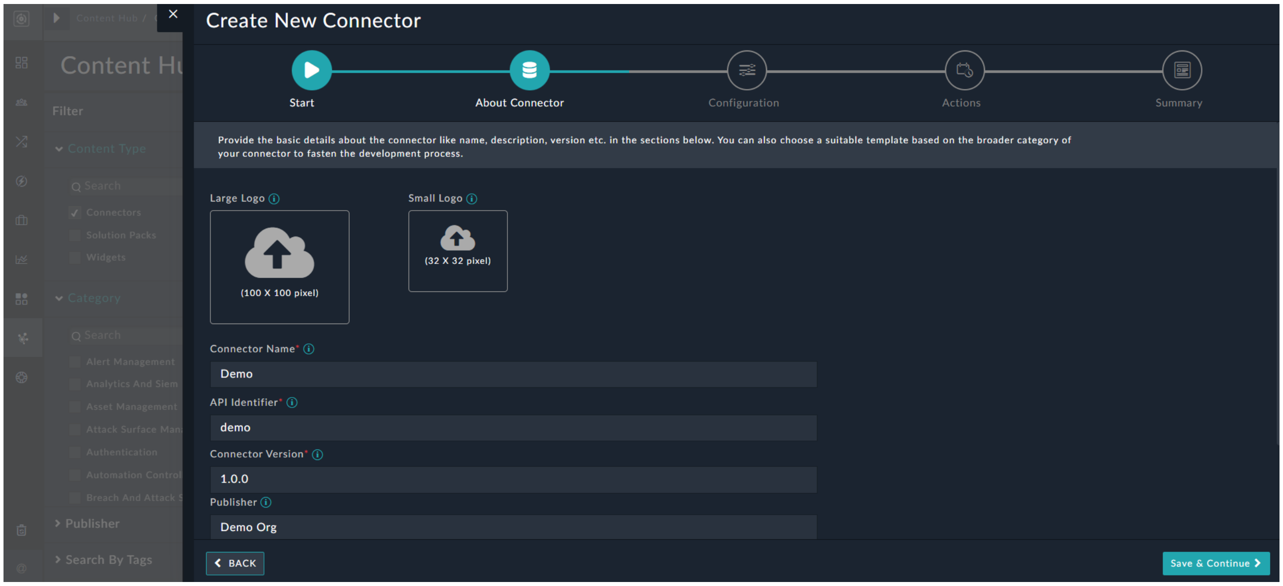
This wizard enables you to efficiently build connectors. The wizard contains templates based on connector types and also provides you with a guided path for creating a connector structure using which you can develop a well-defined connector.

You can then edit the connector code as per your requirements and publish the connector for use by all users.

Let's start by defining a connector

Click the **Lets's start by defining a connector** button to start building your connector.

4. In the Connector Wizard, in the **About Connector** screen, you need to provide the basic details such as name, description, etc. of the connector that you want to create, as well as, upload a logo for the connector, and choose a suitable template for the connector:



Provide the basic details about the connector like name, description, version etc. in the sections below. You can also choose a suitable template based on the broader category of your connector to fasten the development process.

Large Logo (100 X 100 pixel)

Small Logo (32 X 32 pixel)

Connector Name*: Demo

API Identifier*: demo

Connector Version*: 1.0.0

Publisher: Demo Org

You need to add the following details on this screen:

a. Click the **Upload** icons buttons to upload large, medium, or small logos for your connector. Clicking the upload button opens the **Upload Connector Logo** dialog, where you can drag-and-drop your logo, or browse to the logo on your disk and import the logo.

b. In the **Connector Name** field, add the name of the connector that you want to create.
It is a good practice to include both the name of the organization and the name of the product in the connector name, for example, Fortinet FortiSOAR.

c. In the **API Identifier** field, enter a name that would be used as a variable in the connector code to reference this connector. The variable that you use here can be alphanumeric; however, it should not contain any special characters and it must not start with a number.
Also, note that the value that you enter in this field must not match the name of any other connector that is available in the Connector Store or Content Hub. For example, you cannot enter '`virustotal`' in this field,

since the VirusTotal connector is available in Content Hub.
Note: You cannot change the 'name' and 'version' of a connector once it is set.

d. In the **Connector Version** field, enter the version of the connector in the x.y.z format. For example, 1.0.0.
Note: It is recommended that you increase the version number if you are making any changes to an installed connector.

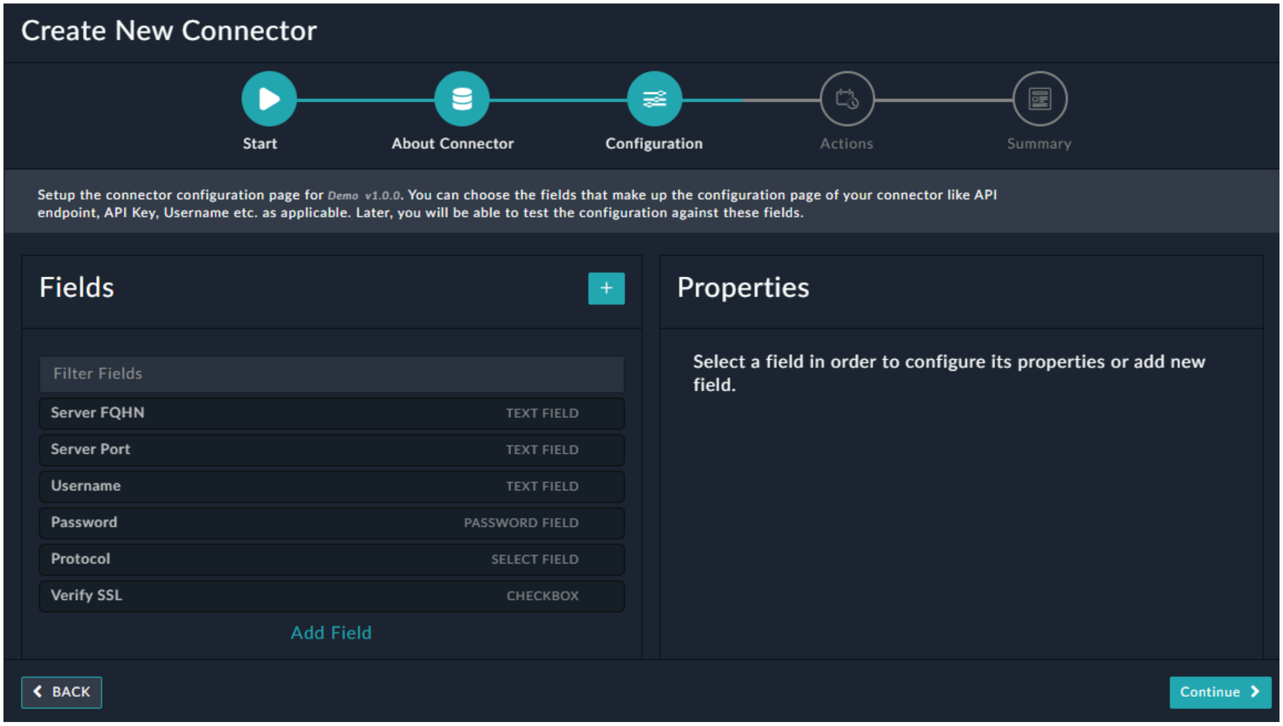
e. (Optional) In the **Publisher** field, enter the name of your organization as the publisher of this connector. The publisher of the connector is responsible for maintaining and supporting the connector.
If you want to keep the connector anonymous, then you can add the "Community" keyword. If this field is left blank, again the connector's publisher is automatically set to "Community".
Note: Do not enter "Fortinet" in this field.

f. In the **Description** field, you can optionally enter information for the connector that you are creating. The Description is displayed on the Connector card on the Connector Store or Content Hub's listing page and enables users to understand more about the Connector.

g. From the **Select Template** drop-down list, select the template that you want to apply to the connector that you want to create.
Templates are based on the broader category of connectors, for example, Threat Intelligence connectors, Ticketing connectors, etc. Templates automatically add suggestive actions, configuration details, and other connector details based on the type of connector you want to create, which helps to speed up the development process.

h. Once you have completed entering all the details, click **Save & Continue**, to go to the `Configuration` screen.

5. Use the **Configuration** screen to set up the connector configuration page. If you have specified a template in the **About Connector** screen, then you will see that some suggestive configuration fields are added based on the selected template, else you will not see any configuration fields and you require to add all the required configuration fields.
If, for example, in the **About Connector** screen, the **Threat Intel** is selected from the **Select Template** drop-down list, then you will see the following `Configuration` screen:



Create New Connector

Setup the connector configuration page for *Demo v1.0.0*. You can choose the fields that make up the configuration page of your connector like API endpoint, API Key, Username etc. as applicable. Later, you will be able to test the configuration against these fields.

Fields

Filter Fields

Server FQHN	TEXT FIELD
Server Port	TEXT FIELD
Username	TEXT FIELD
Password	PASSWORD FIELD
Protocol	SELECT FIELD
Verify SSL	CHECKBOX

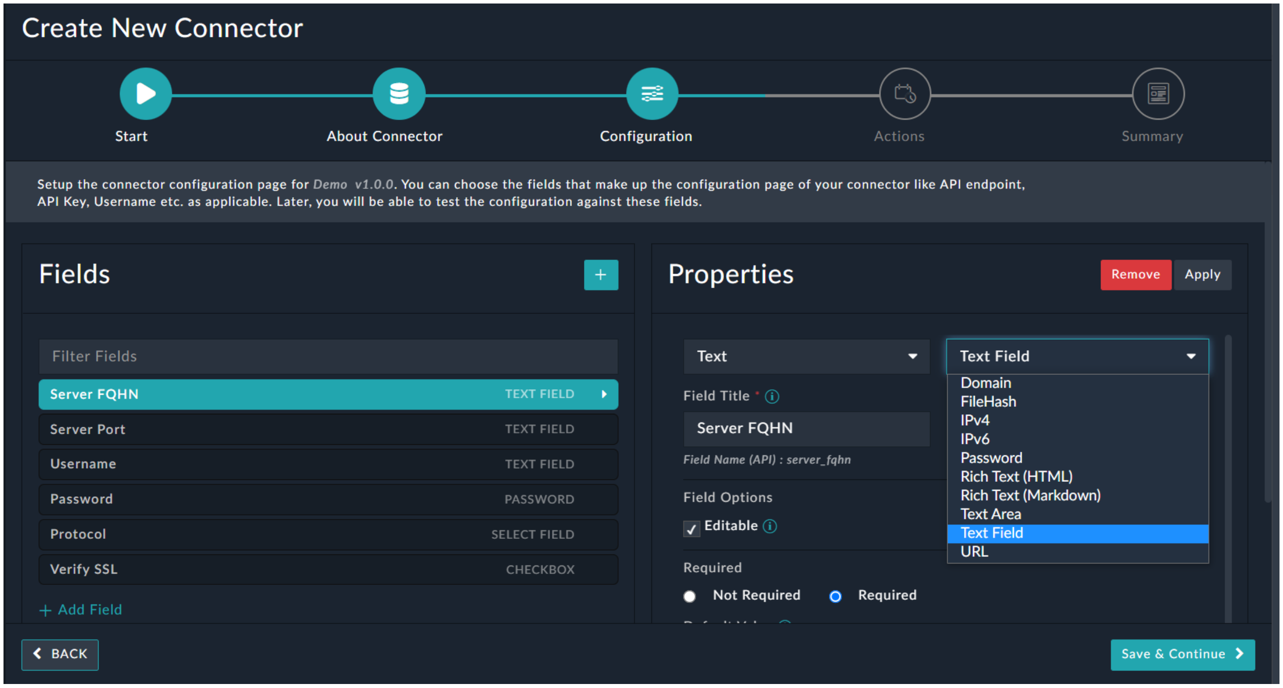
Properties

Select a field in order to configure its properties or add new field.

To add fields to the connector configuration, click **+** button alongside **Fields** or click the **Add Field** link and can easily add properties to that field, like selecting the type of field you want to add such as Text, Integer, Email, etc., and even a sub-type of `Text` fields such as URL, Filehash, etc., all of which help you in ensuring that the user enters appropriate values in the field. You can also set other properties such as visibility and a default value of the field, whether it is required or not, etc.

Important: You cannot add fields that are named 'name' and 'default' to the connector configuration since 'name' and 'default' are reserved keywords.

To edit existing fields and change their properties, select the field to change the properties and click **Apply**.



Create New Connector

Setup the connector configuration page for *Demo v1.0.0*. You can choose the fields that make up the configuration page of your connector like API endpoint, API Key, Username etc. as applicable. Later, you will be able to test the configuration against these fields.

Fields

Server FQHN	TEXT FIELD
Server Port	TEXT FIELD
Username	TEXT FIELD
Password	PASSWORD
Protocol	SELECT FIELD
Verify SSL	CHECKBOX

Properties

Text

Text Field

Field Title *: Server FQHN

Field Name (API) : server_fqhn

Field Options

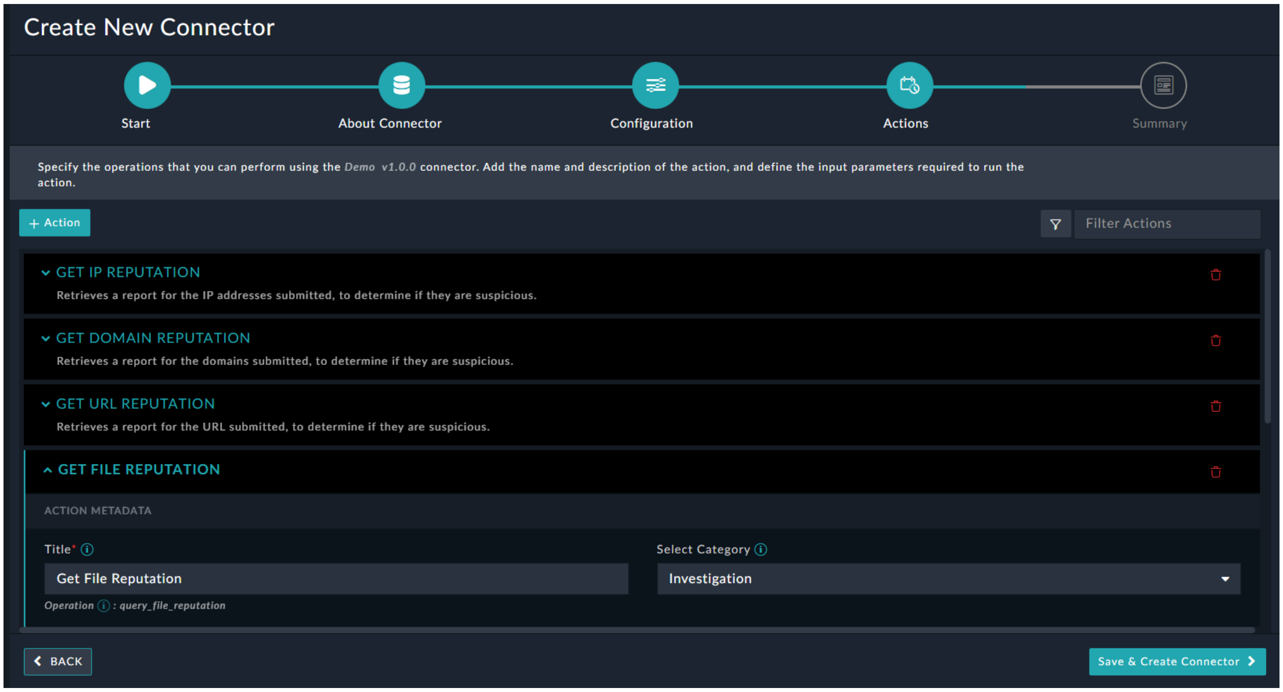
☑ Editable

Required

○ Not Required ● Required

Once you have completed entering all the details, click **Save & Continue**, to go to the `Actions` screen.

6. Use the **Actions** screen to set up the connector actions. From version 7.0.2 onwards, you can define the actions that the connector should perform in the form view, instead of having to edit a raw JSON file. If you have specified a template in the **About Connector** screen, then you will see that some suggestive actions are added based on the selected template, else you will not see any actions and you require to add all the required actions to the connector. If, for example, in the **About Connector** screen, the **Threat Intel** is selected from the **Select Template** drop-down list, then you will see the following `Actions` screen:



Specify the operations that you can perform using the *Demo v1.0.0* connector. Add the name and description of the action, and define the input parameters required to run the action.

GET IP REPUTATION
Retrieves a report for the IP addresses submitted, to determine if they are suspicious.

GET DOMAIN REPUTATION
Retrieves a report for the domains submitted, to determine if they are suspicious.

GET URL REPUTATION
Retrieves a report for the URL submitted, to determine if they are suspicious.

GET FILE REPUTATION

ACTION METADATA

Title*: Get File Reputation

Select Category: Investigation

Operation : query_file_reputation

To add actions, click the **+ Action** button, to edit an action click the down (**v**) arrow to display the metadata and parameters for the action, and click the **Delete** icon to remove the action. You can also type a keyword in the **Filter Actions** field to filter the actions of the connector.

To add a new operation to the connector, click **+ Action**, and enter the following details for the action:

a. In the `Action Metadata` section, enter the following metadata details for the action:

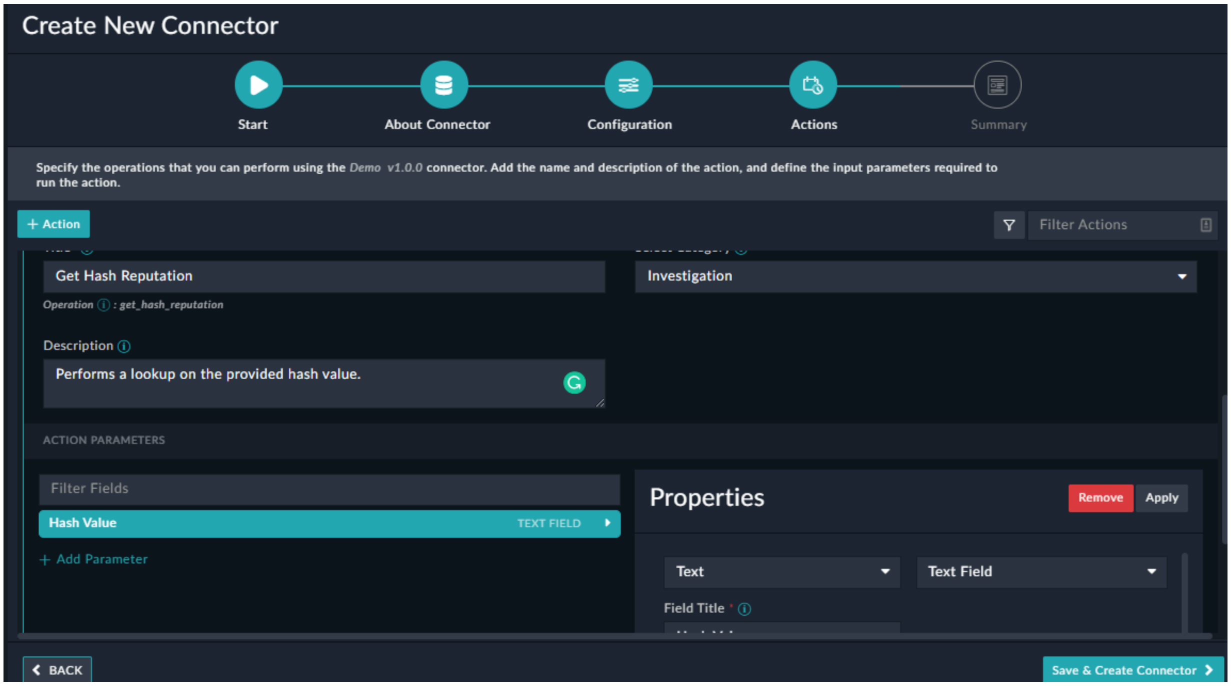
  i. In the **Title** field, enter a suitable name for the action that would explain the purpose of the action. For example, if you want to perform a lookup for a specific domain, then you can enter *Lookup Domain* in the Title field.

  ii. From the **Select Category** drop-down list, select a suitable category for the action from options such as Investigation, Remediation, Utilities, etc. For example, select *Investigation*.

  iii. In the **Description** field, enter a description that explains the action and the utility of the action.

b. In the `Action Parameters` section, define the parameters associated with the action. If your action does not need any input parameters to work, you do not need to add any parameters in this section. To add parameters, click the **Add Field** link and define its properties.

You need to specify the type of field, its field name and title, and other properties such as visibility and required field or not, a default value of the field, etc.



Create New Connector

Specify the operations that you can perform using the *Demo v1.0.0* connector. Add the name and description of the action, and define the input parameters required to run the action.

Get Hash Reputation

Investigation

Operation ⓘ : get_hash_reputation

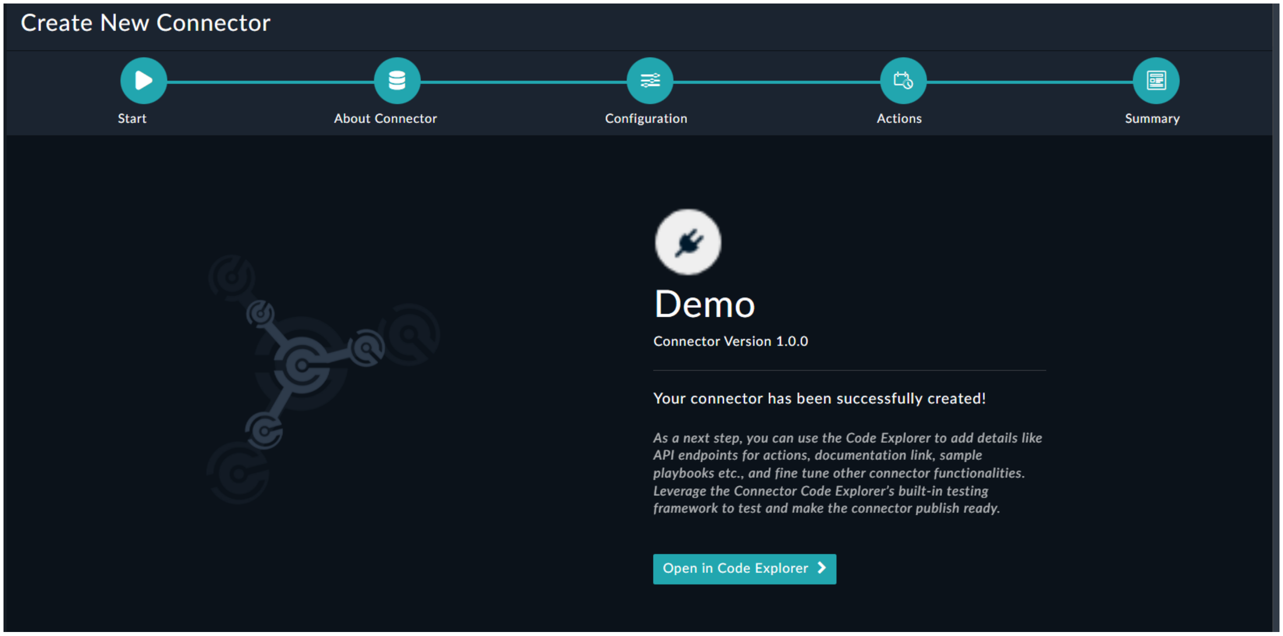
Description ⓘ

Performs a lookup on the provided hash value.

ACTION PARAMETERS

Once you have completed editing the connector configuration and adding the actions to the connector, click **Save & Create Connector**, which displays the **Summary** screen.

7. On the **Summary** screen and you can click **Open in Code Explorer** to edit and test the connector.



Create New Connector

Demo

Connector Version 1.0.0

Your connector has been successfully created!

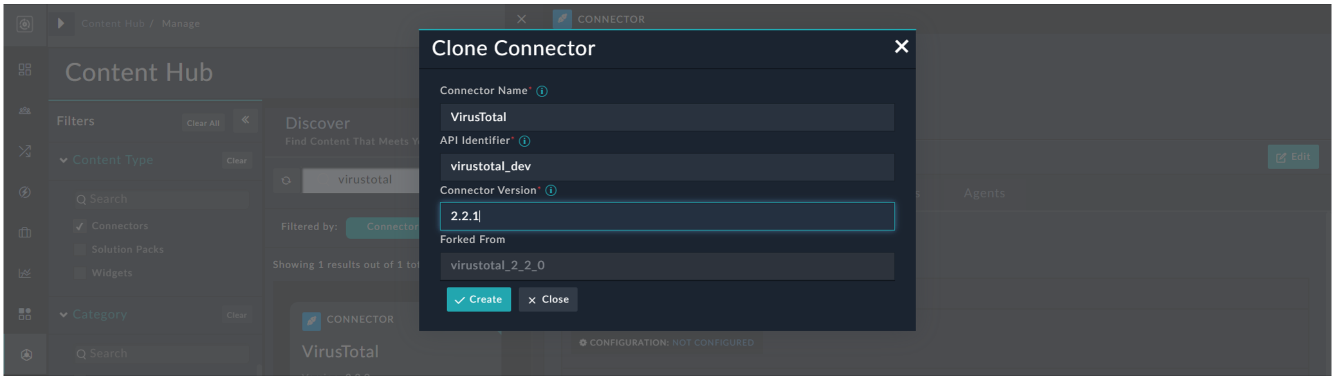
As a next step, you can use the Code Explorer to add details like API endpoints for actions, documentation link, sample playbooks etc., and fine tune other connector functionalities. Leverage the Connector Code Explorer's built-in testing framework to test and make the connector publish ready.

The created connector also gets saved in the **Create** tab, from where you also edit the connector. For more information, see the Editing a connector section.

Editing a connector

You can use the connector's code editor interface to edit existing connectors and build new connectors for custom use cases.

To edit an existing connector that is installed on your system, open the **Manage** tab and click the connector card to open the connector popup, and click **Edit.** However, before you can make any changes to an installed connector, you are required to create a local copy of that connector. For example, clicking **Edit** on the *VirusTotal* connector. This displays a `Confirm` dialog that requires you to clone the connector before making any changes, click **Confirm** to open the `Clone` dialog, where you should update the connector name, version, and also ensure that you have specified a unique API Identifier and then click **Create**:



Clone Connector

Connector Name* VirusTotal

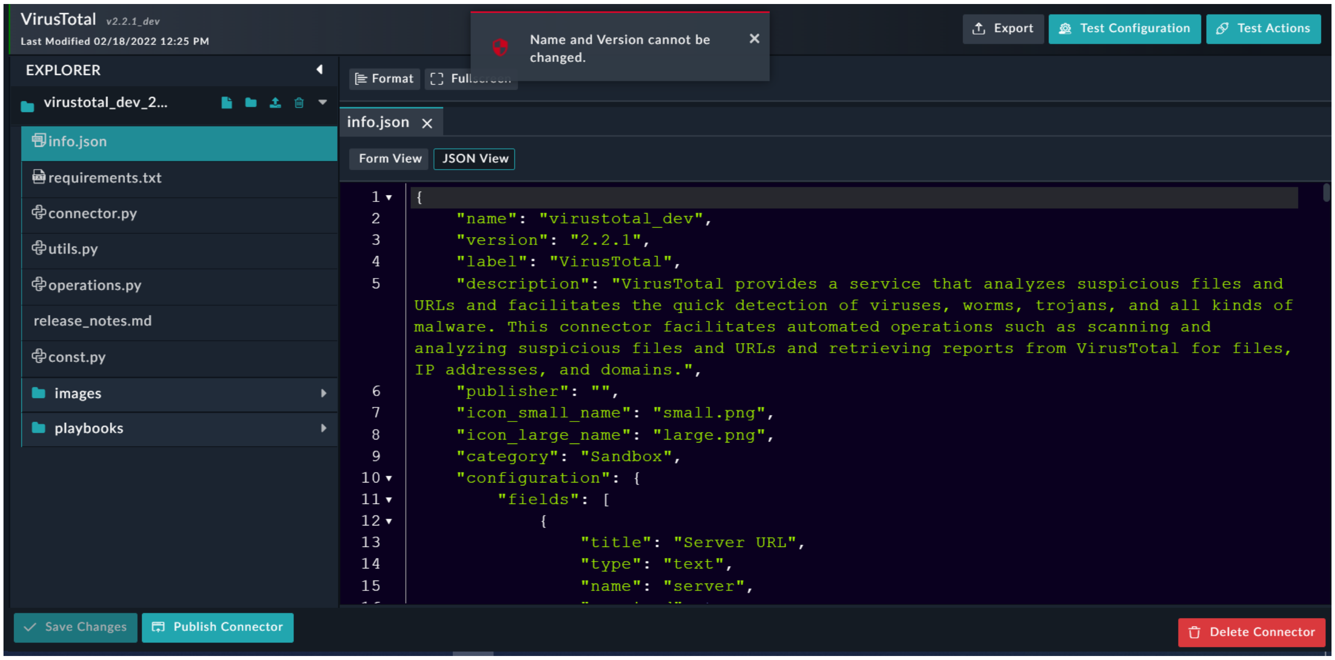
API Identifier* virustotal_dev

Connector Version* 2.2.1

Forked From virustotal_2_2_0

✓ Create ✕ Close

Clicking **Create** opens the code editor interface where you can make the required changes to the connector.



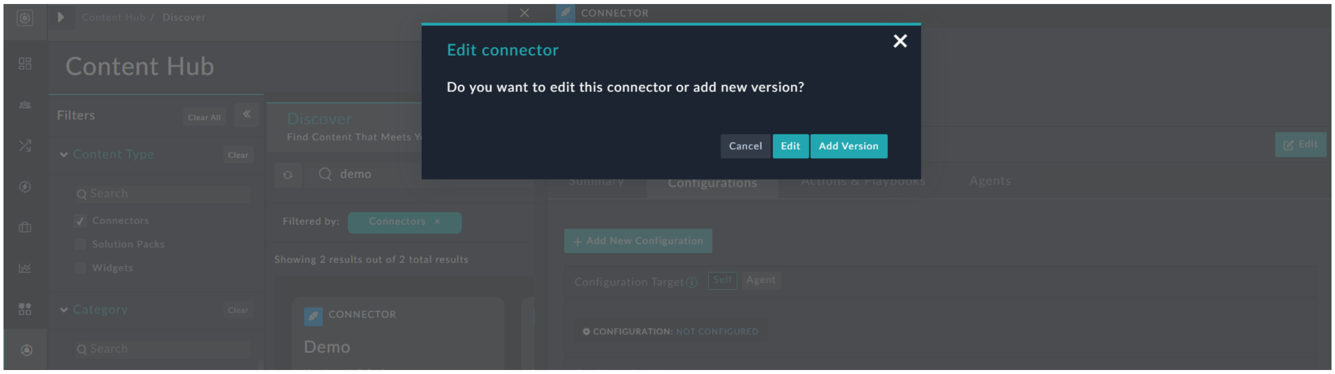
Name and Version cannot be changed.

```
{
    "name": "virustotal_dev",
    "version": "2.2.1",
    "label": "VirusTotal",
    "description": "VirusTotal provides a service that analyzes suspicious files and URLs and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware. This connector facilitates automated operations such as scanning and analyzing suspicious files and URLs and retrieving reports from VirusTotal for files, IP addresses, and domains.",
    "publisher": "",
    "icon_small_name": "small.png",
    "icon_large_name": "large.png",
    "category": "Sandbox",
    "configuration": {
        "fields": [
            {
                "title": "Server URL",
                "type": "text",
                "name": "server",
```

It also saves the cloned connector in the **Create** tab, with the name that is specified in the **API Identifier** field, i.e., '*VirusTotal v2.2.1_dev*'.

You cannot change the connector name and API identifier for a connector once it is set.

If you want to edit a custom connector that has been created using the Connector Wizard and not installed from Content Hub, and which has been published (installed), you are not required to clone the connector. Instead, you can choose to either edit, save, and publish the same version of the connector, or choose to add a new version to the connector. For example, if you click **Edit** on the 'Demo' connector (created using the Connector Wizard), the `Edit Connector` dialog displays the **Edit** and the **Add Version** options:

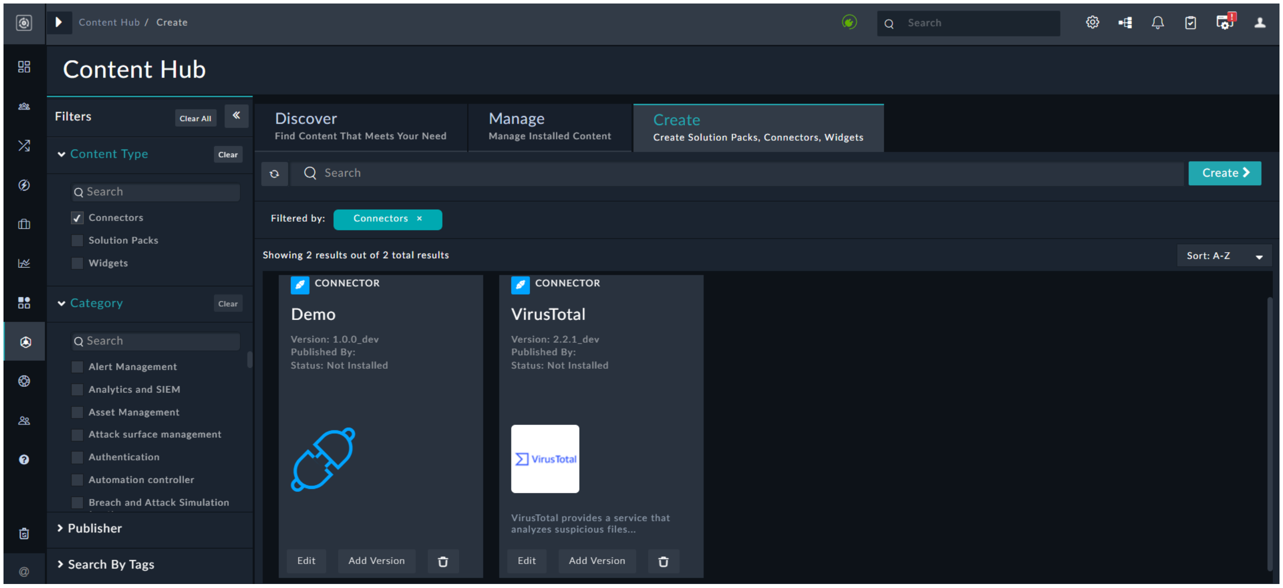


If a connector is published and is being used by other users, it is recommended that you add a version to the connector before editing the connector code, so that the changes you make do not cause any issues to existing operations done using the connector.

Once you create a connector using the "Connector Wizard", or if you are editing an installed connector, then those connectors are saved in the **Manage** tab from where you can edit the connector.

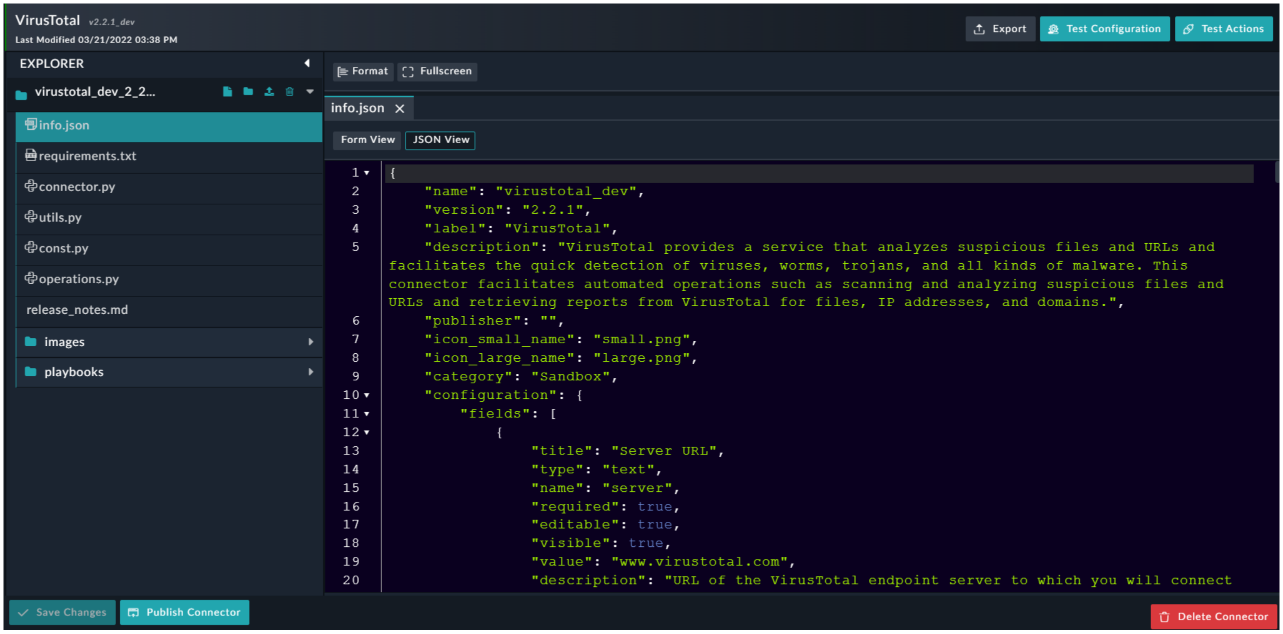
To edit a connector, do the following:

1. Open the **Manage** tab:



In the **Manage** tab, you will see connector cards with their details such as the connector version, the created and the last modified dates, whether the connector is installed or not, etc.
You might see more than one version of the connector if you are working on multiple versions of the connector, and also when you publish a connector a working copy of that version of that installed connector is maintained in the **Manage** tab.
To add a version for the connector, click **Add Version**, which will again display the `Clone Connector` dialog.

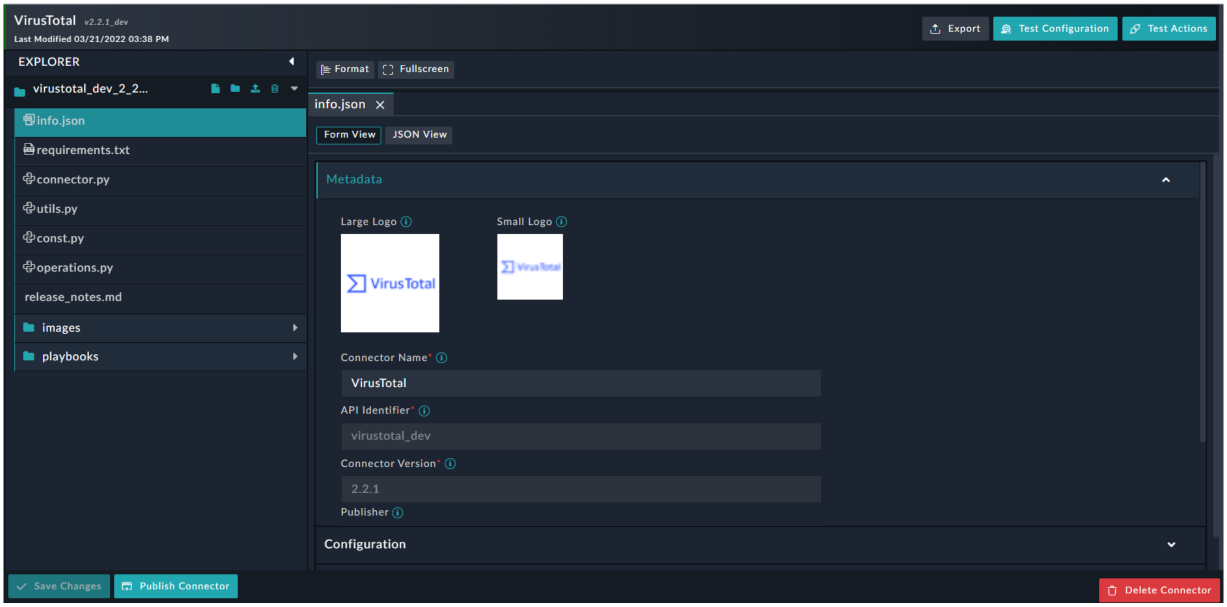
2. To edit a connector, click **Edit Connector**, which displays the code editor interface:



```
{
    "name": "virustotal_dev",
    "version": "2.2.1",
    "label": "VirusTotal",
    "description": "VirusTotal provides a service that analyzes suspicious files and URLs and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware. This connector facilitates automated operations such as scanning and analyzing suspicious files and URLs and retrieving reports from VirusTotal for files, IP addresses, and domains.",
    "publisher": "",
    "icon_small_name": "small.png",
    "icon_large_name": "large.png",
    "category": "Sandbox",
    "configuration": {
        "fields": [
            {
                "title": "Server URL",
                "type": "text",
                "name": "server",
                "required": true,
                "editable": true,
                "visible": true,
                "value": "www.virustotal.com",
                "description": "URL of the VirusTotal endpoint server to which you will connect
```

The files and folders that are present are similar to what is described in the Writing a custom connector manually section. A brief description of these files and folders follows:

- The `info.json` file contains information about the name and version of the connector, logo image file names, the configuration parameters, the set of functions supported by the connector, and their input parameters. The name of all the fields in the `info.json` file must be unique.
  From version 7.0.2 onwards, you can choose to edit the `info.json` file in the **Form** view, instead of editing the file in the raw JSON format, by clicking the **Form View** button:



Connector Name* VirusTotal

API Identifier* virustotal_dev

Connector Version* 2.2.1

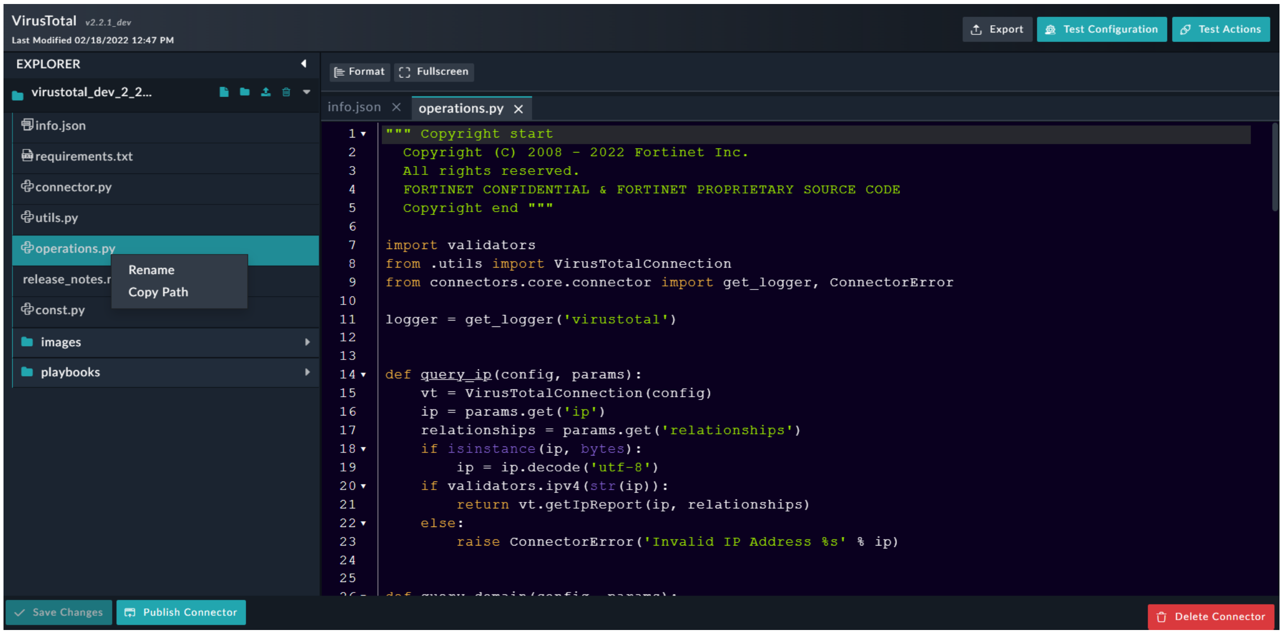
Publisher

  The **Form View** of the info.json file, contains Metadata, Configuration, and Actions information for the connector. Click the down (**v**) in the respective row to add or edit the information for each of these items.

  - **Metadata**: Contains information about the connector, including the name, version, logo, etc. of the connector.
  - **Configuration**: Contains the configuration parameters for the connector that is used to set up the connector configuration page.

    - **Actions**: Contains the actions that the connector can perform and its associated Metadata and parameters.
      For more information on editing these items in the Form View, see the Building a connector using the Connector Wizard section.
- From version 7.0.2 onwards, you can use the connector wizard to add actions to the connector or you can also use the **Form View** of the `info.json` file. In case you want to manually add an action to a connector, i.e., edit the raw JSON file, then you must add the operation parameters in the `info.json` file, and this in turn, creates a `.py` file per operation. Each operation `.py` file contains the default template format for the operation. Once the `info.json` is updated with the operation it is mapped in the `builtins.py`.
- The `connector.py` file extends the base connector class and implements the `check_health` and `execute` functions.
  Note: The `connector.py` template file provides an additional function called `dev_execute` that can be used during the development phase of the connector. Code changes do not always reflect without a service restart; therefore `dev_execute` reloads the function definition at every 'execute' call to ensure that the changes made at every save in the code get immediately reflected. You can change the `execute` function in `connector.py` as follows to call the `dev_execute` function instead:
  ```
  def execute(self, config, operations, params, *args, **kwargs):
        return self.dev_execute(config, operation, params)
  ```
  However, note that reloading the function at every function call is slow and performance-intensive. So, once the development is complete and the connector is ready to use, you must revert the code to the original. For more information on `connector.py`, see the connector.py section.
- The **images** directory contains the connector logo files.
- The **playbook** directory contains the `playbook.json` file, which contains the playbook collection that you want to include with your connector.
- The `requirements.txt` file lists the additional python libraries that are required by the connector. However, the connector dependencies are not installed by default when the connector is created. If you want to test the connector that is being developed, you must manually install the connector dependencies.
- The `release_notes.md` file (present for connectors that have released versions) contains information such as what's new and what's fixed, in a particular release of a connector.

3. Apart from the above files, it is recommended that you maintain a file (`utils.py`) that contains common functions such as DateTime conversion, str to list, etc., which can be used by multiple actions in the connector.
4. Edit the existing connector as required, and click **Save Changes**.
   The connector is saved in the "Draft" state.
   The left-pane of the code editor interface contains the name of the connector along with various icons using which you can perform actions such as creating new files and folders, and also uploading or deleting selected files or folders:

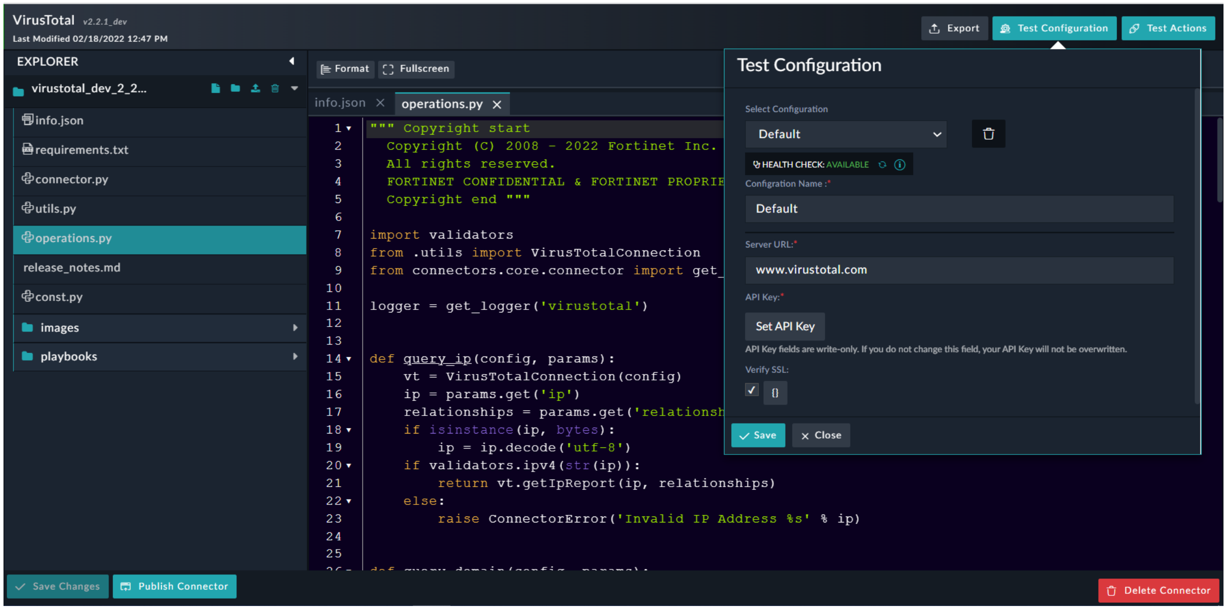


```
""" Copyright start
  Copyright (C) 2008 - 2022 Fortinet Inc.
  All rights reserved.
  FORTINET CONFIDENTIAL & FORTINET PROPRIETARY SOURCE CODE
  Copyright end """

import validators
from .utils import VirusTotalConnection
from connectors.core.connector import get_logger, ConnectorError

logger = get_logger('virustotal')


def query_ip(config, params):
    vt = VirusTotalConnection(config)
    ip = params.get('ip')
    relationships = params.get('relationships')
    if isinstance(ip, bytes):
        ip = ip.decode('utf-8')
    if validators.ipv4(str(ip)):
        return vt.getIpReport(ip, relationships)
    else:
        raise ConnectorError('Invalid IP Address %s' % ip)

```

You can also rename connector files and folders or copy its path, by right-clicking the selected file or folder row and selecting **Rename** or **Copy Path**.

You can also perform the following operations on the code editor interface:

- **Code Formatting**: To lint your code automatically and make the code more human-readable and error-free (programming and programming errors), select the entire code in the editor and click the **Format** button.
- **Full Screen**: To get a better working view and make the editor go full-screen, click the **Fullscreen** button. To exit the full screen, press `ESC`.
- **Export**: To export the connector as a `.tgz` file, click the **Export** button. You might want to import the exported connectors' `.tgz` file into another system.
- **Testing the configuration**: While developing a connector, you can perform a health check operation on the connector. Clicking the **Test Configuration** button opens a dialog containing the specified configuration parameters, where you can input the values, save the configuration, and perform the health check. If the configuration is valid, i.e., the configuration parameters and values specified are correct, then the health check will appear as "Available", else it will appear as "Disconnected".



```
""" Copyright start
  Copyright (C) 2008 - 2022 Fortinet Inc.
  All rights reserved.
  FORTINET CONFIDENTIAL & FORTINET PROPRIE
  Copyright end """

import validators
from .utils import VirusTotalConnection
from connectors.core.connector import get_

logger = get_logger('virustotal')


def query_ip(config, params):
    vt = VirusTotalConnection(config)
    ip = params.get('ip')
    relationships = params.get('relationsh
    if isinstance(ip, bytes):
        ip = ip.decode('utf-8')
    if validators.ipv4(str(ip)):
        return vt.getIpReport(ip, relationships)
    else:
        raise ConnectorError('Invalid IP Address %s' % ip)

```

Test Configuration

Select Configuration: Default

HEALTH CHECK: AVAILABLE

Configration Name :* Default

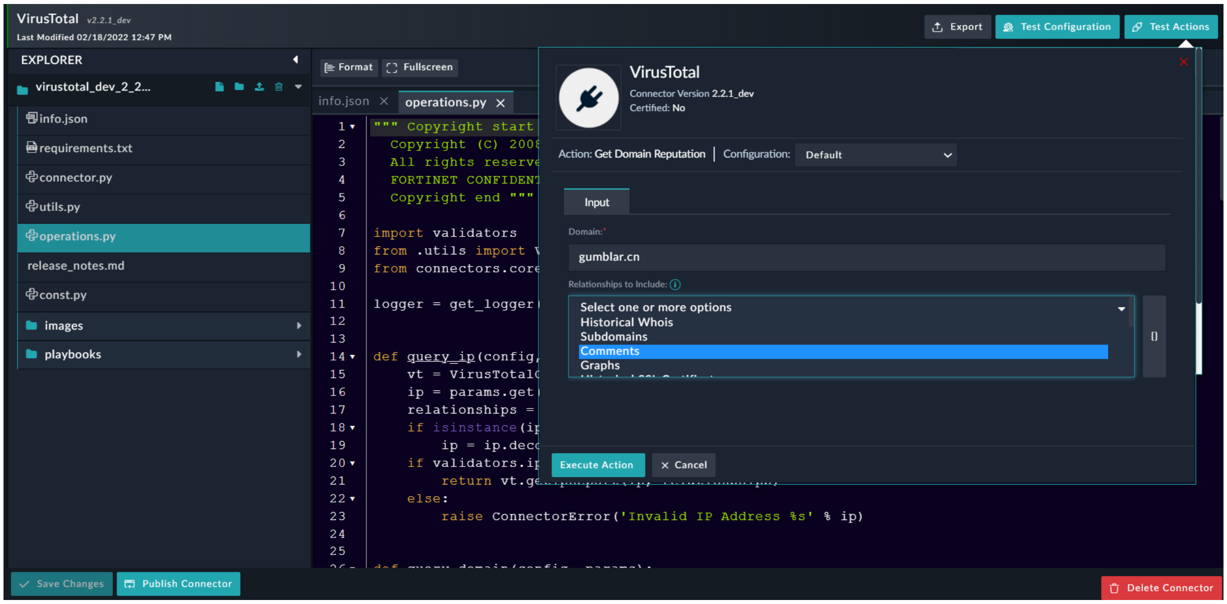
Server URL:* www.virustotal.com

API Key:* Set API Key

API Key fields are write-only. If you do not change this field, your API Key will not be overwritten.

Verify SSL: ☑

- **Testing the actions**: You can use the saved configuration for testing connector actions for their output/errors. Clicking the **Test Actions** button displays a list of actions for that connector. Click the action that you want to test to display a dialog containing the input parameters for that action. Enter the values for the input parameters and click **Execute Action** to view the output for that action.



```
""" Copyright start
  Copyright (C) 2008
  All rights reserve
  FORTINET CONFIDENT
  Copyright end """

import validators
from .utils import V
from connectors.core

logger = get_logger(


def query_ip(config,
    vt = VirusTotalC
    ip = params.get(
    relationships =
    if isinstance(ip
        ip = ip.deco
    if validators.ip
        return vt.ge
    else:
        raise ConnectorError('Invalid IP Address %s' % ip)
```

VirusTotal

Connector Version 2.2.1_dev

Certified: No

Action: Get Domain Reputation | Configuration: Default

Input

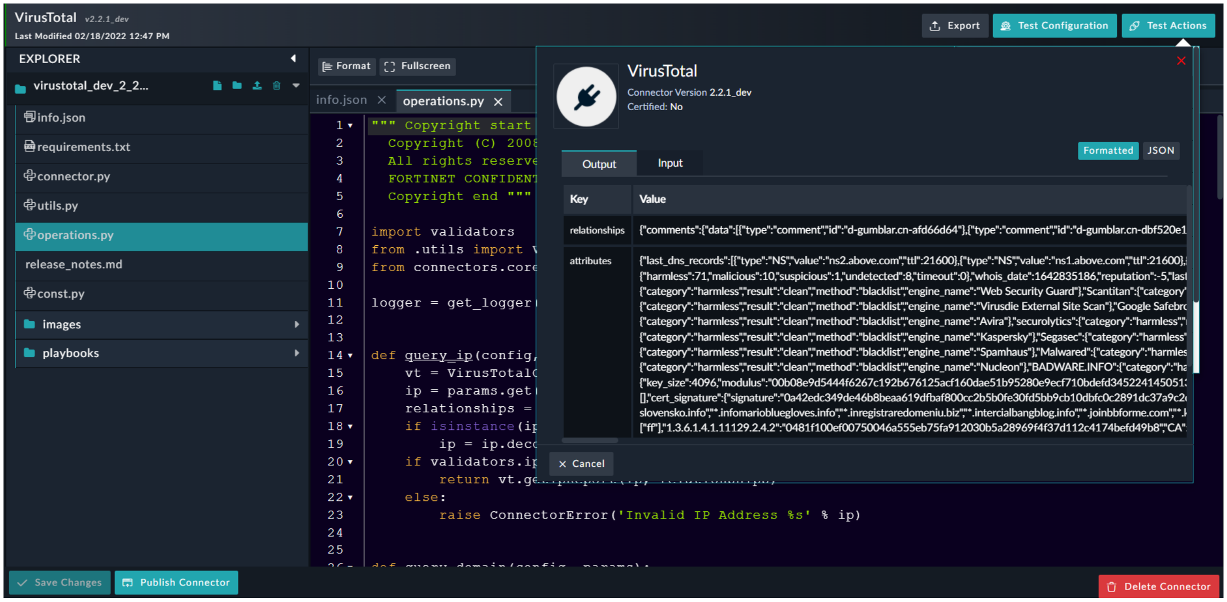
Domain:* gumblar.cn

Relationships to Include:

Select one or more options
Historical Whois
Subdomains
Comments
Graphs

Execute Action | Cancel

Once the connector action is executed, you can see the formatted output of the action, in a tabular format, as shown in the following image:



VirusTotal

Connector Version 2.2.1_dev

Certified: No

Output | Input

Formatted | JSON

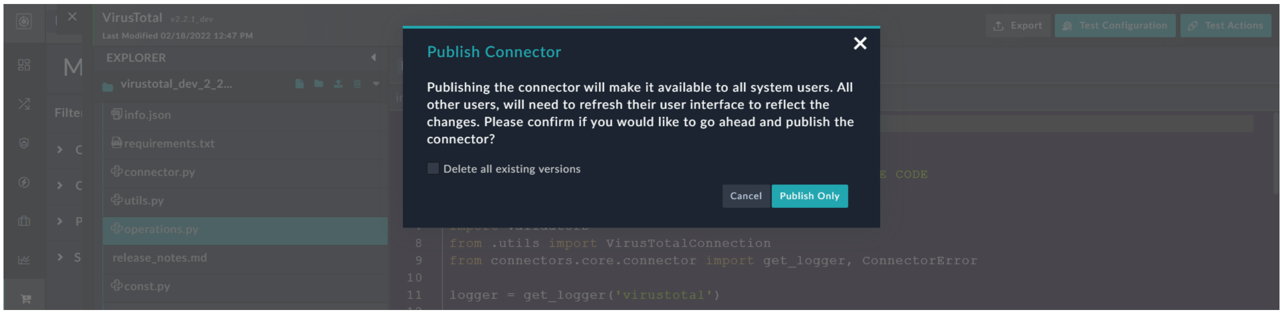
Key	Value
relationships	{"comments":{"data":[{"type":"comment","id":"d-gumblar.cn-afd66d64"},{"type":"comment","id":"d-gumblar.cn-dbf520e1
attributes	{"last_dns_records":[{"type":"NS","value":"ns2.above.com","ttl":21600},{"type":"NS","value":"ns1.above.com","ttl":21600}, {"harmless":71,"malicious":10,"suspicious":1,"undetected":8,"timeout":0},"whois_date":1642835186,"reputation":-5,"last {"category":"harmless","result":"clean","method":"blacklist","engine_name":"Web Security Guard"},"Scantitan":{"category" {"category":"harmless","result":"clean","method":"blacklist","engine_name":"Virusdie External Site Scan"},"Google Safebro {"category":"harmless","result":"clean","method":"blacklist","engine_name":"Avira"},"securolytics":{"category":"harmless"," {"category":"harmless","result":"clean","method":"blacklist","engine_name":"Kaspersky"},"Segasec":{"category":"harmless" {"category":"harmless","result":"clean","method":"blacklist","engine_name":"Spamhaus"},"Malwared":{"category":"harmles {"category":"harmless","result":"clean","method":"blacklist","engine_name":"Nucleon"},"BADWARE.INFO":{"category":"ha {"key_size":4096,"modulus":"00b08e9d5444f6267c192b676125acf160dae51b95280e9ecf710bdefd345224145051 [],"cert_signature":{"signature":"0a42edc349de46b8beaa619dfbaf800cc2b5b0fe30fd5bb9cb10dbfc0c2891dc37a9c2 slovensko.info","*.infomarioblueglovess.info","*.inregistraredomeniu.biz","*.intercialbangblog.info","*.joinbbforme.com","*. ["ff"],"1.3.6.1.4.1.11129.2.4.2":"0481f100ef00750046a555eb75fa912030b5a28969f4f37d112c4174befd49b8","CA"

Cancel

```
""" Copyright start
  Copyright (C) 2008
  All rights reserve
  FORTINET CONFIDENT
  Copyright end """

import validators
from .utils import V
from connectors.core

logger = get_logger(


def query_ip(config,
    vt = VirusTotalC
    ip = params.get(
    relationships =
    if isinstance(ip
        ip = ip.deco
    if validators.ip
        return vt.ge
    else:
        raise ConnectorError('Invalid IP Address %s' % ip)
```

- **Deleting the Connector**: To delete the connector, click the **Delete Connector**. button.

5. Once you have completed making the changes, click **Publish Connector** to display the `Publish Connector` dialog:



Publish Connector

Publishing the connector will make it available to all system users. All other users, will need to refresh their user interface to reflect the changes. Please confirm if you would like to go ahead and publish the connector?

□ Delete all existing versions

```
from .utils import VirusTotalConnection
from connectors.core.connector import get_logger, ConnectorError

logger = get_logger('virustotal')
```

Click **Publish Only** to publish the connector and make it available on the **Discover** tab for all the users of the system and use it in playbooks or run directly on records. However, a working copy for the same is also maintained in the **Create** tab.
To delete all existing versions of the connector from your system, select the **Delete all existing versions** checkbox.

Removing images of a connector installed from the Content Hub

If you are editing a connector that is part of the Content Hub, for example 'VirusTotal' and you want to remove the images of such a connector, do the following:

1. Open the connector in the code editor interface, and click the **images** folder.
2. Select the image that you want to delete and then click the **Delete** icon, then click **Delete** on the confirmation dialog.
3. In the code editor interface, click **info.json** and edit the `info.json` file to remove the file references of the deleted image. For example, if you have deleted the large logo for the connector (`large.png`), then in the `info.json` file, remove its reference:
   `"icon_large_name": "large.png"`
4. To save your changes, click **Save Changes**.

Debugging connectors

- You can add logger statements to test your connector's code paths. Use the `get_logger` utility function to initialize the logger. You can import `get_logger` from `connectors.core.connector`, and then declare the logger as `logger = get_logger('<connector name>')`. Then you can add `logger.error('<logger_message>')` to the `operation.py` file that you want to debug.
- If your connector's configuration or action fails, you can check the connector logs. All the connector logs are written to the `/var/log/cyops/cyops-integrations/connectors.log` file and follow the format: `%(asctime)s %(levelname)s %(name)s %(module)s %(funcName)s(): %(message)s`
  The default log level is `WARN`, however, you can change it to `INFO` or `DEBUG`.
  For example, to change the log level to `INFO`:
  Open the `/opt/cyops-integrations/integrations/configs/config.ini` file and set `connector_logger_level = 'INFO'`, and then restart the `uwsgi` service.

Writing a custom connector manually

Perform the following steps to write a custom connector manually:

1. Create a Connector Template Directory Structure with the required files.
2. Import the connector into FortiSOAR.
3. Check the health of the connector.
4. Add a connector operation to a playbook.

Connector Template Directory Structure

Create the following folder structure, with a folder with the connector name at the top and files within it:

connectorname folder

```
--+ playbooks
---+ __init__.py
---+ playbooks.json
--+ connector.py
--+ info.json
--+ images
--+ requirements.txt
--+ packages
---+ <package_name>
```

Python 3 is required for developing your connectors.

info.json

The `info.json` file contains information about the name and version of the connector, logo image file names, the configuration parameters, the set of functions supported by the connector, and their input parameters. The name of all the fields in the info.json file must be unique.

Ensure that the name of the connector in the `info.json` and the name of the connector folder matches exactly.

You can configure the following parameters, fields, and operations while writing connectors: description of connector and actions, tooltip, placeholder, conditional fields, backward compatibility (using visible_onchange parameter), validation using regex, label, apioperations, and grouping of fields. These parameters are explained in the **Notes** following the sample info.json file.

Following is an example of an `info.json` file:

```
{
 "name": "sampleConnector",
 "label": "sampleConnector",
 "description": "sampleConnector connector description",
 "publisher": "publisherName",
 "cs_approved": true/false,
 "cs_compatible": true/false,
 "version": "1.0.0",
 "category": "categoryType",
 "icon_small_name":"small_icon.jpeg",
 "icon_large_name":"large_icon.jpeg",
 "configuration": {
   "fields": [
     {
       "title": "fieldname",
       "required": true/false,
       "editable": true/false,
       "visible": true/false,
       "type": "text",
       "description": "text",
       "name": "user",
       "tooltip": "text for the tooltip",
       "placeholder": "placeholder text",
       "validation":{
            "pattern":"regex pattern for validation",
            "patternError":"text for the error message if validation fails"
          }
     }]
 },
 "operations": [
   {
     "operation": "function_template",
     "title": "Sample Function",
     "description": "description of the operation",
     "category": "categoryType",
     "annotation": "sample_annotation",
     "parameters": [{
       "title": "Sample Input1",
       "required": true/false,
       "editable": true/false,
       "visible": true/false,
       "type": "text",
       "name": "input1",
       "description": "text",
       "value": "default value1"
       "tooltip": "text",
     },
     {
       "title": "Sample Input2",
       "required": true/false,
       "editable": true/false,
       "visible": true/false,
       "type": "text",
       "name": "input2",
       "description": "text",
       "tooltip": "text",
       "value": "",
       "options": ["A", "B"],
       "onchange": {
            "A": [{
              "title": "User2",
              "required": true/false,
              "editable": true/false,
              "visible": true/false,
              "visible_onchange":false,
              "type": "text",
              "name": "user2",
              "description": "text",
                         "tooltip": "text",
              "value": "admin1"
            }],
            "B": [{
              "title": "User2",
              "required": true/false,
              "editable": true/false,
              "visible": true/false,
              "type": "integer",
              "name": "user2",
              "description": "text",
                         "tooltip": "text",
              "value": 12345
            },{
              "title": "Comment1",
              "required": true/false,
              "editable": true/false,
              "visible": true/false,
              "type": "text",
              "name": "comment1",
              "description": "text",
                         "tooltip": "text",
              "placeholder": "Add comment1",
              "value": "",
              "onchange": {
                "placeholder attribute": [{
                  "title": "placehoder attribute title",
                  "required": true/false,
                  "editable": true/false,
                  "visible": true/false,
                  "type": "checkbox",
                  "name": "placehoder attribute name",
                  "description": "text",
                             "tooltip": "text",
                  "value": false
                }
              ]}
            }]
       }
     }],
     "enabled": true,
     "output_schema": {"key1": "", "key2": []}
   },
   {
            "operation": "Sample Operation 2",
            "category": "containment",
            "annotation": "sample_op",
            "title": "title of the operation",
            "description": "description of the operation",
            "is_config_required": true/false,
            "parameters": [
                {
                    "title": "title of list",
                    "type": "text",
                    "name": "sample_list",
                    "required": true/false,
                    "editable": true/false,
                    "visible": true/false,
                    "description": "description of the operation",
                    "tooltip": "text"
       },
                {

                    "type": "label",
                    "label": "label text",
                    "visible": true
       },
                {
                    "title": "title of list",
                    "type": "text",
                    "name": "sample_list",
                    "description": "description of the operation",
                    "required": true/false,
                    "editable": true/false,
                    "visible": true/false,
                    "class": "group-element",
                    "tooltip": "text"
       },
                {
                    "title": "title of list",
                    "type": "text",
                    "name": "sample_list",
                    "description": "description of the operation",
                    "required": true/false,
                    "editable": true/false,
                    "visible": true/false,
                    "class": "group-element",
                    "tooltip": "text"
       },
                {
                    "title": "title of list",
                    "type": "text",
                    "name": "sample_list",
                    "description": "description of the operation",
                    "required": true/false,
                    "editable": true/false,
                    "visible": true/false,
                    "class": "group-element last",
                    "tooltip": "text"
       }
      ],
            "enabled": true,
            "output_schema": {}
    }
 ]
},
"ingestion_supported": true,
"ingestion_modes": [
        "scheduled"
],
"ingestion_preferences": {
        "modules": [
                "indicators"
        ],
        "launch_name":"DI Configuration"
},
"help_online": "link to online documentation"
```

Notes:

- The version of the connector must be in the **x.y.z** format, for example, **1.0.0**. The version must consist of valid integers, for example, "1.15.125" is a valid version.
- The `output_schema` defines the keys that are present in the output json on the execution of an operation. The `info.json` contains some common keys. However, the output json can have additional keys based on the input parameters. You can use these json keys to set the input for subsequent Playbook Steps, using the Dynamic Values. For more information, see the *Dynamic Values* section in "Playbooks."
- To support data ingestion, the `"ingestion_supported"` key must be set to `true`. You must also specify the ingestion method(s) that are supported by the connector using the `"ingestion_modes"` key. FortiSOAR supports the following ingestion methods: `"ingestion_modes": ["scheduled", "notification", "app_push"]`
  You can also customize the label of the button that is used to open the 'Data Ingestion Wizard'. By default, the label is set to "**Configure Data Ingestion**," which might not be appropriate in some cases, where the "Data Ingestion Wizard" must be run per configuration. If you want to change the default label, then add the `"launch_name"` parameter and its value in the `"ingestion_preferences"` key in the connector's `info.json` file. If the `"ingestion_preferences"` key is not present in the connector's `info.json` file, then you must first add the `"ingestion_preferences"` key, and then add the `"launch_name"` parameter and its value. For example:
  ```
  "ingestion_preferences": {
      "launch_name": "DI Configuration"
  }
  ```
- If you want to add online and/or offline documentation for your connector, add the following to your info.json:
  ```
  help_file: "name of pdf file in the connector folder"
  help_online: "link to the online documentation for the connector"
  ```
- Input types supported: text, password, checkbox, integer, decimal, datetime, phone, email, file, richtext, json, textarea, image, select, and multiselect.
  For `select` and `multiselect`, you can provide the list of inputs using the `options` key.
  For example,
  ```
  {
        "title": "Sample Field",
        "name": "sample",
        "required": "true",
        "editable": "true",
        "visible": "true",
        "type": "select",
        "options": ["A","B","C"]
  }
  ```
- `category` and `annotation` within *operations*: The category defines the category for the connector that you are adding, and it must be one of the following: investigation, remediation, containment, and miscellaneous.
  The annotation defines the operation or function that will be performed. An annotation is unique and belongs to only one category, i.e., you must not add an annotation to multiple categories.
  If you do not define any category in the `info.json` file, then by default, the annotation is added to the miscellaneous category.
  The category name must contain only lower-case alphabets. The annotation name must contain only lower-case

alphabets, underscores, and numbers.
Category and annotations must always come together.
Multiple operations within a connector can use the same annotation.

- description: You can add a description for the connector, which will be visible on the connector page in FortiSOAR and you can also add a description for the action or operation, which will be visible on the connector step page in the Playbook Designer.
- is_config_required: By default, the is_config_required key is set to true (default), which means that the connector uses the configuration specified in the connector configuration to execute an action. If you set the is_config_required key to false, then configurations are not required to execute an action and in this case, you can use dynamic fields to specify the configurations.
  Note: This key is applicable at the action level, i.e., limited to a single action.
- tooltip: You can add the tooltip parameter to any field to display information about that particular field. Note that if you do not want a tooltip against any field, then you must remove the tooltip parameter from that field in the info.json. You must not pass "" or null as values to the tooltip parameter.
- placeholder: You can add the placeholder parameter for text and select fields, which will display placeholders for those fields on the connector step page in the Playbook Designer.
- conditional fields (onchange): You can add the onchange parameter to fields, which you can use to display other fields or subfields conditionally based on the user input for that field. For example:

```
{
  "options": ["Now", "Yesterday"],
  "onchange": {
              "Now": [{
                "title": "Comment3",
                "required": true,
                "editable": true,
                "visible": true,
                "type": "text",
                "name": "comment3",
                "value": ""
              }],
              "Yesterday": [{
                "title": "Comment4",
                "required": true,
                "editable": true,
                "visible": true,
                "type": "text",
                "name": "comment4",
                "value": ""
              }]
  }
}
```

- visible_onchange: For backward compatibility of the connector you can use the visible_onchange parameter for conditional fields (onchange parameter).
  If you set visible_onchange to false, then this field will be hidden in FortiSOAR UI, and if you set visible_onchange to true, or if it is not present for any field, then this field is visible in FortiSOAR UI.
- validation: You can add regex validation to text and textarea fields. You can also add the error that will be displayed in case the validation fails. For example:

```
{
 "validation":{
           "pattern":"\\d+",
           "patternError":"this is not a number"
```

```
          }
}
```

- `label`: You can add the label field to display text on the connector UI. For example:

```
{
"type": "label",
"label": "this is my label",
"visible": true
}
```

- `grouping`: You can group fields based on your categorization using `"class": "group-element"`. For the last field in the group, use `"class": "group-element last"`.
  For example:

```
{
       "operation": "block_applications",
       "parameters": [
          {

              "type": "label",
              "label": "this is my label",
              "visible": true
       },
           {
               "title": "Applications Names(List Format)",
                "type": "text",
                "name": "app_list",
                "required": true,
                "editable": true,
                "visible": true,
                "class": "group-element",
                "tooltip": "Block Applications Names (List Format)"
       },
           {
               "title": "Applications Names(List Format)",
               "type": "text",
                "name": "app_list",
                "required": true,
                "editable": true,
                "visible": true,
               "class": "group-element last",
              "tooltip": "Block Applications Names (List Format)"
       }
     ]
 }
```

- `apiOperations`: You can fetch options for the `select` and `multiselect` fields from the API that you have defined in your operation. The parameters to this operation will be what users have entered in the configuration of this operation and the target field. apiOperations is not supported for the connector configuration.
  Note: To hide this operation in the from the **Actions** drop-down list in the Playbook Designer set `"visible": false` for this operation.
- `supports_remote`: Some connector actions cannot work when run on agents in a segmented network. Therefore, parameters of those actions must be marked as `"supports_remote": False`. For example:

```
"title": "Enable abc service",
"name": "abc_service",
"supports_remote":  False
```

- `conditional_output_schema`: Support for multi-option or dynamic output schema in connectors. In every operation that requires the dynamic output schema, you must add `conditional_output_schema` as a *key* in an array of objects. Each object will contain the condition, which you should add within `{{ }}` and then you must add the corresponding output schema for each condition. It is recommended that you add a default schema, using `"condition": "{{true}}"`, at the end of the defined conditions. The default schema will be used if none of the defined conditions are met.
  Notes:
  a. If you are using multiple conditions for evaluation such as a combination of `&&` and `||`, then you must wrap the whole condition within `()`.
  For example, `{{(command === 'ls' || command === 'ssh' )}}`
  `{{(command === 'ls' && port === 5895 )}}`
  b. If you are using the condition with the checkbox field, add the condition as:
  `{{ checkbox === true}}`
  c. If you are using the condition with the multi-select field, add the condition as:
  `{{multsel.toString() === (['option1','option2']).toString()}}`
  d. If you are using the condition with the `json` field, add the condition as:
  `{{jsonfieldname === '{\"key\":\"value\"}'}}`
  For example, `{{jsonfieldname === '{\"name\":\"fortinet\"}'}}`
  Important: To support versions earlier than the 4.12.0 release, `output_schema` will also always be present.
  Example of a `conditional_output_schema` definition:

```
{
       "conditional_output_schema": [
          {
                  "condition": "{{command === 'ls'}}",
                  "output_schema": {
                      "op_result": "",
                      "op_status": ""
                  }
              },
              {
                  "condition": "{{command === 'ssh'}}",
                  "output_schema": {
                      "result": "",
                      "status": ""
                  }
              },
              {
                  "condition": "{{true}}",
                  "output_schema": {
                      "result_default": "",
                      "status": ""
                  }
              }
     ]
 }
```

connector.py

The `connector.py` class extends the base connector class and implements the `check_health` and `execute` functions. Following is a skeleton of this class:

```
from connectors.core.connector import Connector, get_logger
logger = get_logger('<connector_name>')

class Template(Connector):
   def execute(self, config, operation, params, **kwargs):
       supported_operations = {'operation_1': function_template}
       return supported_operations[operation](config, params)

   def check_health(self, input):
       return True
```

Notes:

- All imports for the connector files should be relative. For example, if your `util.py` is parallel to `connector.py` and you want to import `util.py` then you must import it as: `import .util`.
- `get_logger` is a utility function to initialize the logger. You can import `get_logger` from `connectors.core.connector`, and then declare the logger as `logger = get_logger('<connector name>')`. All the connector logs are written to the `/var/log/cyops/cyops-integrations/connectors.log` file and follow the format: `%(asctime)s %(levelname)s %(name)s %(module)s %(funcName)s(): %(message)s`.
- In addition to the `execute` and `check_health` functions, the following optional and additional functions are also available:
  `on_add_config(self, config)`: Invoked when a new configuration is added to the connector. This is an optional function that can be overridden while setting up a new configuration.
  `on_update_config(selfself, old_config, new_config)`: Invoked when a configuration is updated for the connector. This is an optional function that can be overridden while setting up a configuration edit function.
  `on_delete_config(self, config)`: Invoked when a configuration is deleted from the connector. This is an optional function that can be overridden while setting up a configuration teardown function.
  `on_activate(self, config)` : Invoked when a configuration is activated.
  `on_deactivate(self, config)` : Invoked when a configuration is deactivated.
  `on_teardown(self)` : Invoked when a configuration is deleted. This is an optional function that can be overridden for dismantling a connector.
  You can use these functions to perform specific operations such as starting or stopping services at the relevant events. For example, you can use the `on_add_config()` function to start and stop a service when a configuration is added.

playbooks.json

The `playbook.json` file contains any playbook collection that you want to include with your connector. This could be a set of playbooks that demonstrate the usage of your connector.

images

The images directory must have the connector icon files. The 'icon_small_name' and 'icon_large_name' keys in the `info.json` must match the names of these icon files inside the `images` folder. Icon files can be in the `.jpg` or `.png` formats.

Once you have created all the files, bundle them into a *.tgz* file. For example, `tar -czvf sampleConnector.tgz sampleConnector/`.

requirements.txt and packages

If a connector requires additional python libraries, specify the libraries in the `requirements.txt` file.

If there is a dependency on any custom packages or if your instance does not have internet access to download packages from the internet, you can add the packages to the `packages` directory in the connector folder.

During a connector import, the framework first runs `pip install -r requirements.txt` followed by `pip install <package>` for every package in the `packages` directory. The commands are run in a separate thread and import is marked successful even when the dependency installation is still in progress. The dependency install logs are available at `/var/log/cyops/cyops-integrations/pipinstall_<timestamp>.log` on the FortiSOAR instance. If more than five *dependency* install log files get accumulated, the log files that are older than a day get deleted.

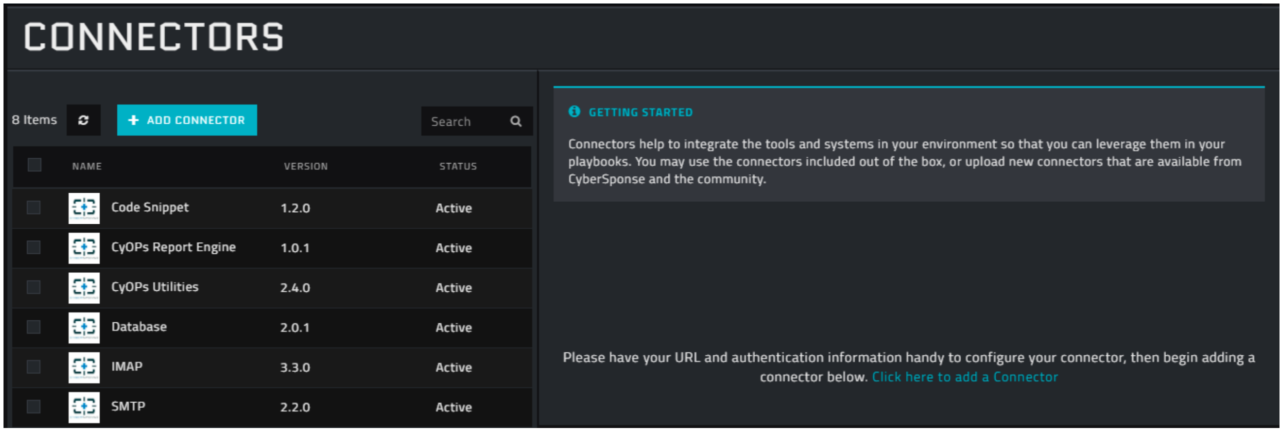
If the installation of a dependency, then you can install them again by invoking the REST APIs directly. Contact FortiSOAR Support for more details on the APIs.

Importing a connector into FortiSOAR

Use the "**Content Hub**" or "**Connector Store**" to install and configure connectors in FortiSOAR 5.0.0 and later. To install a connector, you must be assigned a role that has a minimum of `Create` access to the `Connectors` module. To configure connectors into FortiSOAR, you must be assigned a role that has a minimum of `Update` access to the `Connectors` module. For more information about the Content Hub, Connector Store, and the process of importing connectors into FortiSOAR, see the Introduction to connectors chapter.

Importing a connector into FortiSOAR prior to release 5.0.0

1. Log on to FortiSOAR.
2. On the left navigation pane, click **Automation** > **Connectors** and click **Add Connector**.



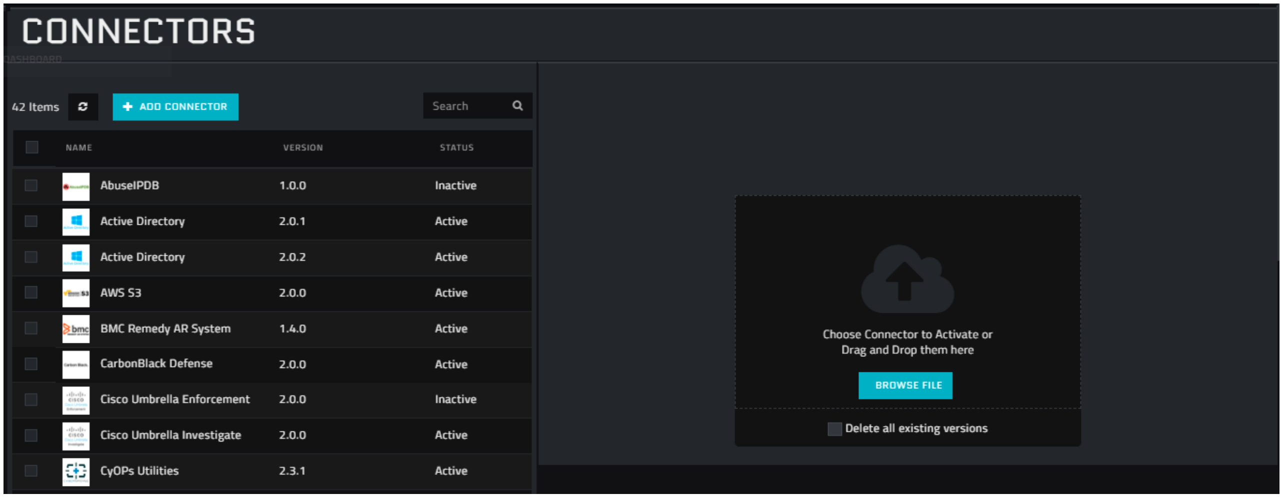
	NAME	VERSION	STATUS
□	Code Snippet	1.2.0	Active
□	CyOPs Report Engine	1.0.1	Active
□	CyOPs Utilities	2.4.0	Active
□	Database	2.0.1	Active
□	IMAP	3.3.0	Active
□	SMTP	2.2.0	Active

Connectors help to integrate the tools and systems in your environment so that you can leverage them in your playbooks. You may use the connectors included out of the box, or upload new connectors that are available from CyberSponse and the community.

Please have your URL and authentication information handy to configure your connector, then begin adding a connector below. Click here to add a Connector

3. Drag-and-drop the connector package file or click **Browse File** to import the *connector.tgz* file.
   FortiSOAR will prompt you to enter the configuration inputs that you have defined in the `info.json` file.
   Note: You can install different versions of a connector, enabling you to reference a specific version of a connector from a playbook. If you want to replace all previous versions of the connector, ensure that you click the **Delete all existing versions** checkbox while importing the new version of the connector. If you do not click the **Delete all existing versions** checkbox, then a new version of the connector is added. You must ensure that your playbooks

reference a correct and existing version of the connector.
Following is a sample image:



	NAME	VERSION	STATUS
□	AbuseIPDB	1.0.0	Inactive
□	Active Directory	2.0.1	Active
□	Active Directory	2.0.2	Active
□	AWS S3	2.0.0	Active
□	BMC Remedy AR System	1.4.0	Active
□	CarbonBlack Defense	2.0.0	Active
□	Cisco Umbrella Enforcement	2.0.0	Inactive
□	Cisco Umbrella Investigate	2.0.0	Active
□	CyOPs Utilities	2.3.1	Active

Choose Connector to Activate or
Drag and Drop them here

□ Delete all existing versions

FortiSOAR displays the `Uploading Connector` message and then displays the Connector Configuration popup.

4. To configure the connector, enter the required configuration details in the `Connector Configuration` popup.
The configuration details and the details of the connector specified in the `info.json` file are stored in the FortiSOAR database.
Note: You can add multiple configurations for your connector if you have more than one instance of your third-party server in your environment. You must, therefore, add a unique `Name` for each configuration.
If you have previous versions of a connector and you are configuring a newer version of that connector, with the same configuration parameters, then FortiSOAR fetches the configuration and input parameters of the latest available version of that connector. For example, if you have 1.0.0, 2.0.0, and 2.3.0 versions of the Fortinet FortiSIEM connector and you are configuring the 2.3.0 version of the Fortinet FortiSIEM connector, then while configuring the 2.3.0 version, FortiSOAR will fetch the configuration and input parameters from the 2.0.0 version of the Fortinet FortiSIEM connector. You can review the configuration and input parameters, and then decide to change them or leave them unchanged.
You can activate or deactivate a configured connector by clicking on the **Activate Connector** or **Deactivate Connector** Link.
You can check the **Mark As Default Configuration** option to make the selected configuration, the default configuration of this connector, on the particular FortiSOAR instance.
The `password` type fields in FortiSOAR include encryption and decryption. Passwords are encrypted before saving them into the database and decrypted when they are used in actions. In case of an upgrade, connectors that are already installed will work with stored passwords.
5. To save your configuration, click **Save**.
To view the list of actions that can be performed by the connector, click the **Actions** tab.
To view the playbook file that is bundled with the connector, click the **Playbooks** tab.
You can optionally perform a Health Check by clicking the **Refresh** icon that is present in the `Health Check` bar. The Health Check checks if the configuration parameters you have specified are correct and if connectivity can be established to the specified server, endpoint, or API.
If all the details are correct and the connectivity to the server can be established, then on the Connectors page, **Available** is displayed in the health check dialog.
If any or all the details are incorrect or if the connectivity to the server cannot be established then on the Connectors page, **Disconnected** is displayed in the health check dialog.
You can also click the **Refresh** icon that is present in the `Health Check` bar to perform the health check at any time.

Reimporting a connector

After importing a connector, any changes done in the python files of the connector automatically get reflected the next time you run a command on the connector. However, if changes are made in the `info.json`, then you need to reimport the connector, using the following command, for the updates to take effect:

```
/opt/cyops-integrations/.env/bin/python /opt/cyops-integrations/integrations/manage.py
reimport_connector -n <connector_name> -cv <connector_version> -migrate
```

If you only want to update the `info.json` changes and not retain the previous connector configuration, then you can omit the `-migrate` attribute as follows:

```
/opt/cyops-integrations/.env/bin/python /opt/cyops-integrations/integrations/manage.py
reimport_connector -n <connector_name> -cv <connector_version>
```

You can also reimport all connectors at a single time using the following command by omitting the `-n <connector_name>` and `-cv <connector_version>` attributes as follows:

```
/opt/cyops-integrations/.env/bin/python /opt/cyops-integrations/integrations/manage.py
reimport_connector -migrate
```

Check_Health function

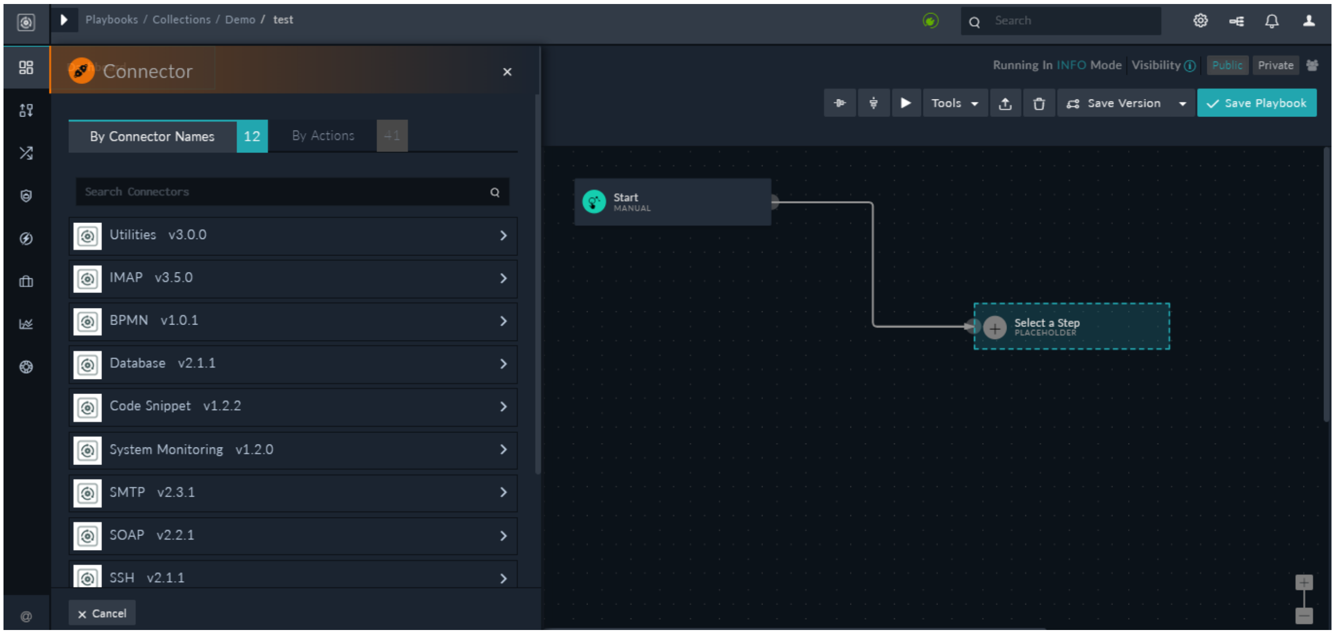
The `check_health` function of the connector is invoked when you click the **Refresh** icon. This function takes the dictionary of the configuration parameters as the input. For example: `{‘url’: ‘https://xyz.com’, 'user': 'admin', 'password': 'password'}`.

You must throw the `ConnectorError` from the function if you want the `check_health` function to fail in a given scenario, such as issues with connectivity or with the provided credentials. To throw the `ConnectorError`, you must import it from the `connectors.core.connector` module as follows:

```
from connectors.core.connector import ConnectorError
```

Add connector operation to a playbook

Once you have completed configuring and deploying your connector, you can add a connector operation to a playbook, by adding the connector as a step in the playbook, as shown in the following image:



You can install different versions of a connector, and while adding a connector operation, you specify a specific version of a connector within a Playbook. In case you have installed multiple connectors, and if the version of the connector specified in the playbook is not found, then the playbook by default uses the latest version. FortiSOAR checks for the latest version of the connector in the format "major version.minor version.patch version". For example, version 2.0.2 is a later version than 1.2.0.

The `execute` function of `connector.py` is called when an operation on the connector is invoked. This function takes the dictionary containing the `config`, `operation` and `params` fields. For example:

```
{'config': {'password': password, 'server_url': 'https://xyz.com', 'user': 'admin'},
'params': {'input1': 'value1', 'input2': 'value2'},
'operation': 'function_template'}
```

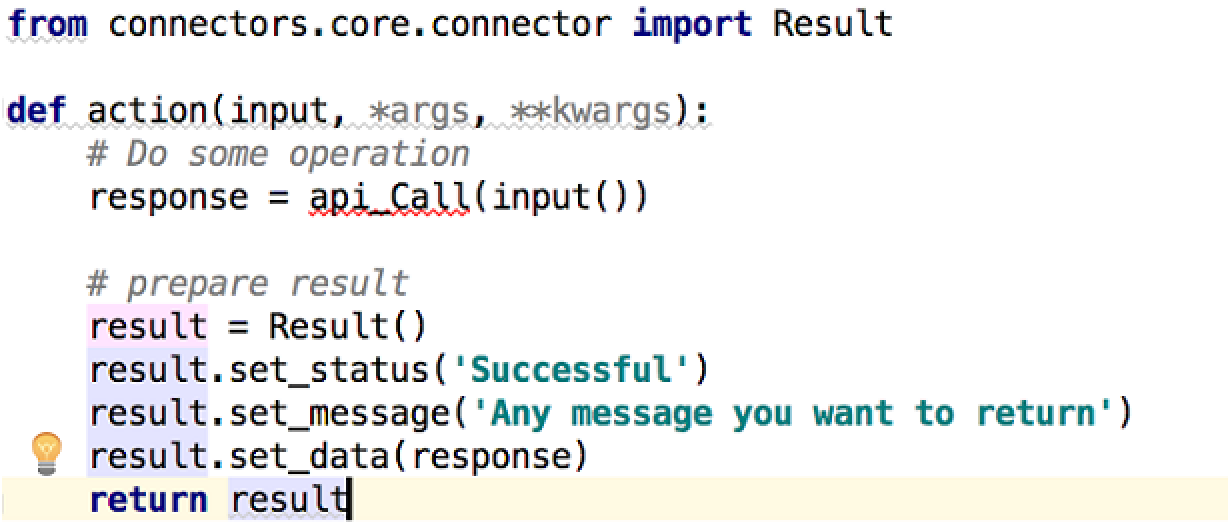
The return from the `execute` function is set in the `results.data` variable of the playbook step. A sample `execute` function is present in the connector.py section.

The `info.json` contains `output_schema`, which defines the keys that are present in the output json on the execution of an operation. The `info.json` contains some common keys. However, the output json can have additional keys based on the input parameters. You can use these json keys to set the input for subsequent Playbook Steps, using Dynamic Values. For more information, see the *Dynamic Values* section in the "Playbooks Guide."

Configuring a connector to return a response

If you want to return a response from any action of a connector, then you can import the `Result` class from the connectors framework to your `connector.py`, and then you can set message attributes such as status, message, etc.

Following is the `python` snippet for the same:



```
from connectors.core.connector import Result

def action(input, *args, **kwargs):
    # Do some operation
    response = api_Call(input())

    # prepare result
    result = Result()
    result.set_status('Successful')
    result.set_message('Any message you want to return')
    result.set_data(response)
    return result
```

Updating a connector configuration using the update_connnector_config() function

The `update_connnector_config()` function can be internally called from any connector to update the configuration of any connector by providing the name and version of that connector. Steps to be followed for updating a connector configuration:

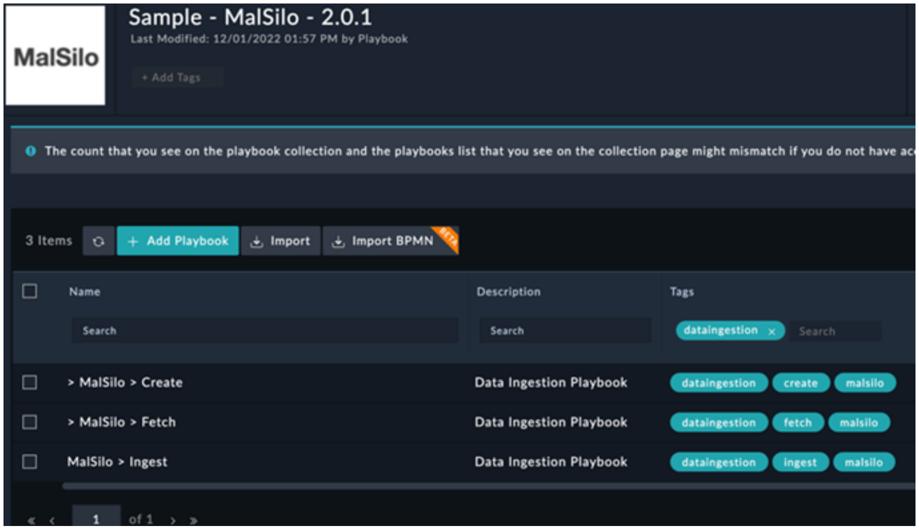
1. Import the `update_connnector_config()` function from `/opt/cyops-integrations/integrations/connectors/core/utils.py`.
2. Call the `update_connnector_config` function with the following parameters: `<connector_name>`, `<connector_version>`, `<update_config>`, and `<config_id>`.
   `update_connnector_config(<connector_name>,<connector_version>,<update_config>,<config_id>)`
   For example, `update_connnector_config("imap","3.2.0", {"username":"csdamin","password":"csadminpwd"},"5785130913212321")`
   Notes:
   If you do not provide the `<config_id>`, then the connector configuration that you have marked as default configuration will be updated.
   If you have not marked any connector configuration as the default configuration, and you have also not provided the `<config_id>`, then the following error is raised: `No configuration found to update. Please add a config_id or mark a configuration as the default.` in the `/var/log/cyops/cyops-integrations/connectors.log`. To resolve this error, either mark a connector configuration as the default configuration or provide the `<config_id>`.

Configuring a custom connector to support data ingestion

If your custom connector also supports data ingestion, you must make the following updates so that your connector works with the ingestion wizard in the FortiSOAR:

1. Update the `info.json` file as follows:
   a. `"ingestion_supported": true`
      This is a required parameter; if not included, the "Data Ingestion" button will not be displayed in the connector configuration.
   b. Specify the ingestion mode(s) to be used by your custom connector:
      `"ingestion_modes": ["scheduled", "notification", "app_push"]`
      This is a required parameter, and the value of "`ingestion_modes`" must be a list containing one or more of the following three elements:
      i. "`scheduled`": Specify this value if you want to setup ingestion on a schedule. The “Scheduling” page gets displayed in the Data Ingestion Wizard only if you add "`scheduled`" in `"ingestion_modes"`.
      ii. "`notification`": Specify this value if the connector supports a listener for real-time ingestion. The 'Exchange' connector is an example of a connector that supports a listener.
      iii. "`app_push`": Specify this value if the connector's ingestion is based solely on a push from the integrating application. The 'Splunk Add-on' is an example of an app_push integration.
   c. Specify additional preferences for your custom connector using the `"ingestion_preferences"` parameter:
      For example, in the Data Ingestion Wizard, to limit the module selection to specific modules only instead of all modules present on the instance, specify the list of supported modules explicitly using the “`modules`” key:
      `"ingestion_preferences": {"modules": ["threat_intel_feeds", "indicators"]`
      Or, to give a custom name to the launch button ("**Configure Data Ingestion**") used to open the 'Data Ingestion Wizard', use the "`launch_name`" key.
      For example:
      `"ingestion_preferences": {"launch_name": "My Custom Label"}`
2. Update the playbooks.
   The data ingestion process is essentially the running of a FortiSOAR playbook. The ingestion flow is comprised of two phases termed as "Fetch" and "Create". You can refer to the sample collection of ingestion-enabled connectors, such as FortiAnalyzer and FortiSIEM, which are included with FortiSOAR.
   All playbooks that contribute to the flow must adhere to the following conventions:
   a. They must be added as a part of the 'Sample Collection' shipped with the connector.
   b. All playbooks contributing to ingestion must contain the following tags:
      `<connector_name>`, `dataingestion`

For example, the MalSilo connector in the following image contains the tag `malsilo` and `dataingestion`:



Sample - MalSilo - 2.0.1

Last Modified: 12/01/2022 01:57 PM by Playbook

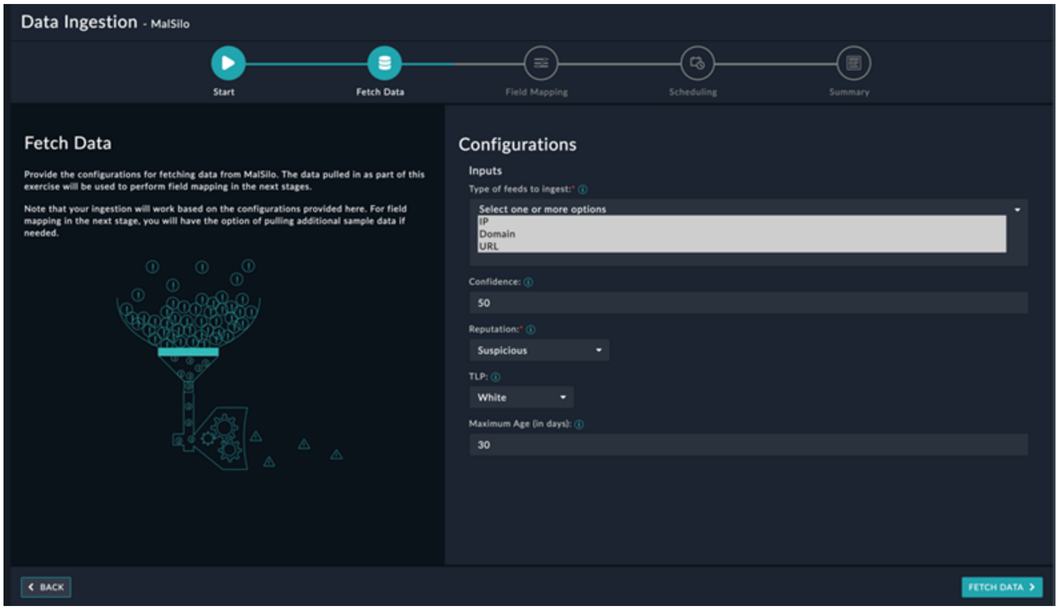
The count that you see on the playbook collection and the playbooks list that you see on the collection page might mismatch if you do not have ac

	Name	Description	Tags
□	> MalSilo > Create	Data Ingestion Playbook	dataingestion create malsilo
□	> MalSilo > Fetch	Data Ingestion Playbook	dataingestion fetch malsilo
□	MalSilo > Ingest	Data Ingestion Playbook	dataingestion ingest malsilo

c. **Fetch Playbook**: This playbook pulls data from the ingestion source.

   i. Must have an additional tag named "`fetch`"

   ii. The connector action step responsible for fetching the data must be named “**Fetch**”. The data ingestion wizard looks for a step with this name and displays its inputs as 'inputs' on the “Fetch Data” page of the Data Ingestion Wizard:



Data Ingestion - MalSilo

Fetch Data

Provide the configurations for fetching data from MalSilo. The data pulled in as part of this exercise will be used to perform field mapping in the next stages.

Note that your ingestion will work based on the configurations provided here. For field mapping in the next stage, you will have the option of pulling additional sample data if needed.

Configurations

Inputs

Type of feeds to ingest:* Select one or more options (IP, Domain, URL)

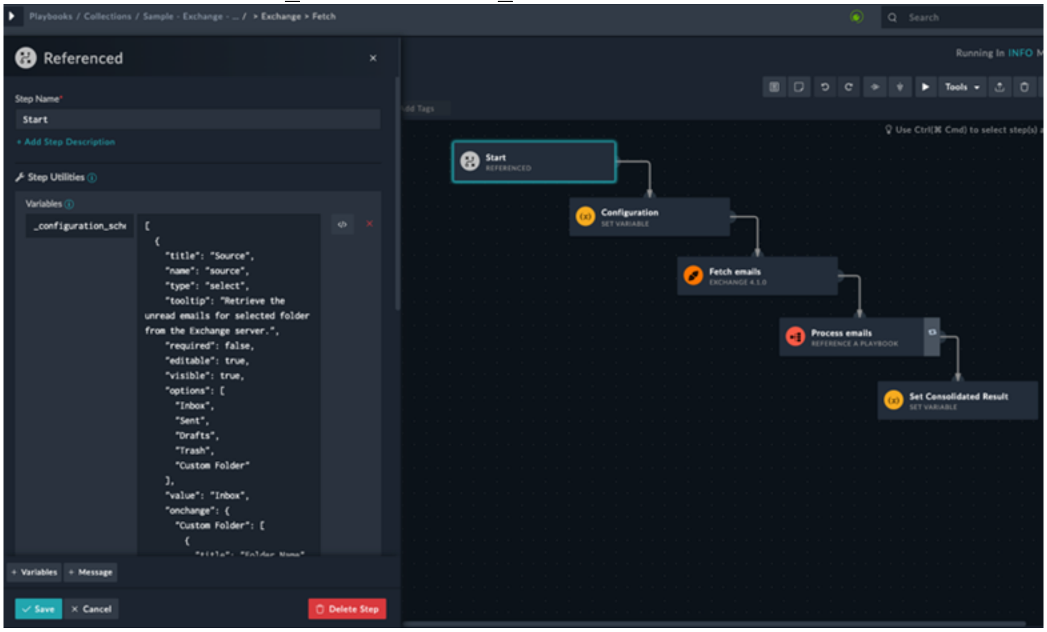
Confidence: 50

Reputation:* Suspicious

TLP: White

Maximum Age (in days): 30

   iii. If the inputs onthe “`Fetch Data`” page of the Data Ingestion Wizard are different from the inputs on the connector configuration step, then you need to add a "variable" called "`_configuration_schema`" to the “`Start`” step and define the "Fetch Data" inputs in the JSON format. The format of the JSON is the same as that for a connector action input. This variable, if present, is given preference over the '`Fetch`' step. An example of the "`_configuration_schema`" variable with JSON values:



Step Name*

Start

+ Add Step Description

Step Utilities

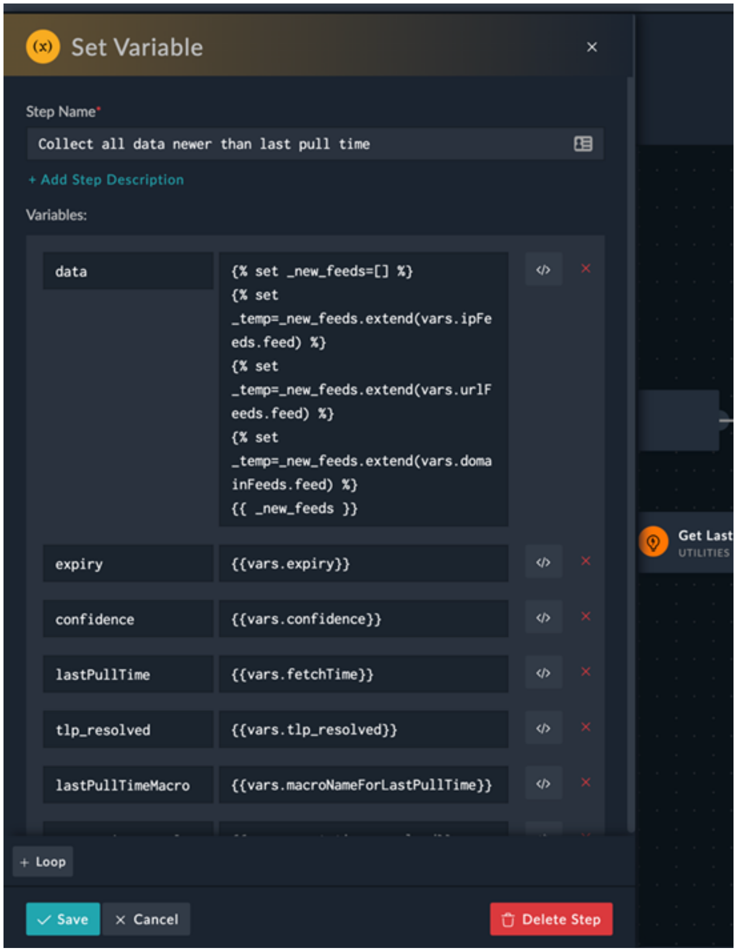
Variables

_configuration_schx

```
[
  {
    "title": "Source",
    "name": "source",
    "type": "select",
    "tooltip": "Retrieve the
unread emails for selected folder
from the Exchange server.",
    "required": false,
    "editable": true,
    "visible": true,
    "options": [
      "Inbox",
      "Sent",
      "Drafts",
      "Trash",
      "Custom Folder"
    ],
    "value": "Inbox",
    "onchange": {
      "Custom Folder": [
        {
          "title": "Folder Name"
```

iv. Must have a `Set Variable` step named "**Configuration**" as the second step of the playbook. All inputs provided by users in the "Fetch Data" step are saved in this playbook step.

v. The playbook must return data in a variable named "**data**" as the last step of the playbook.



Step Name*

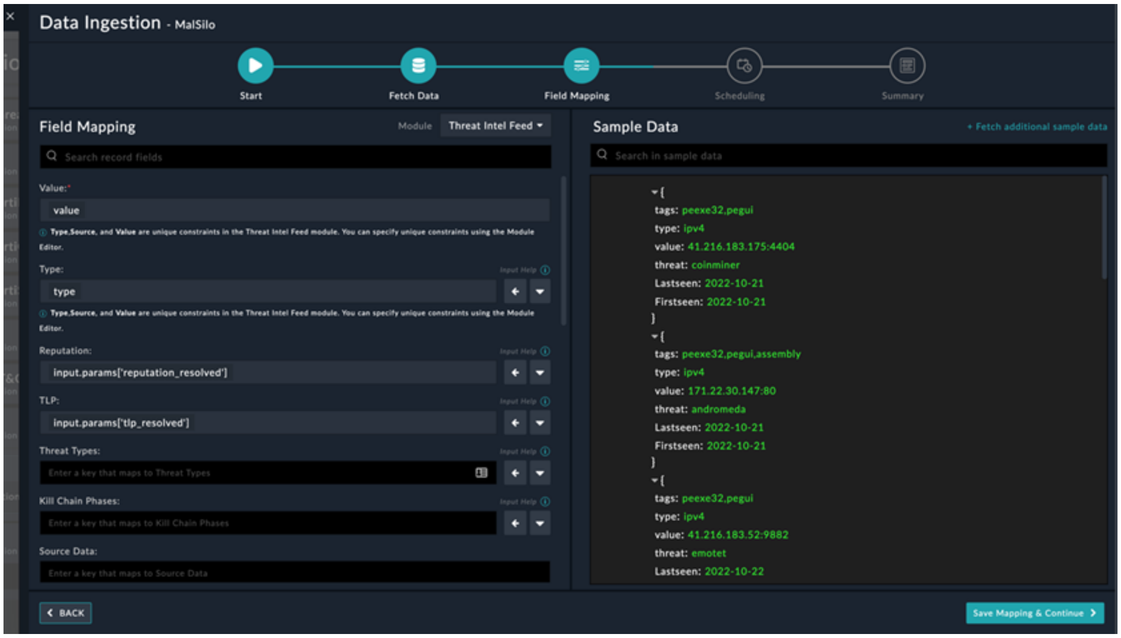
Collect all data newer than last pull time

Variables:

data	{% set _new_feeds=[] %}
	{% set _temp=_new_feeds.extend(vars.ipFeeds.feed) %}
	{% set _temp=_new_feeds.extend(vars.urlFeeds.feed) %}
	{% set _temp=_new_feeds.extend(vars.domainFeeds.feed) %}
	{{ _new_feeds }}

expiry	{{vars.expiry}}
confidence	{{vars.confidence}}
lastPullTime	{{vars.fetchTime}}
tlp_resolved	{{vars.tlp_resolved}}
lastPullTimeMacro	{{vars.macroNameForLastPullTime}}

`{{ vars.steps.<step_name>.data }}` of this playbook is what is presented in the "`Field Mapping`" screen of the Data Ingestion Wizard, and it is also used for record creation during the scheduled runs.



Value:* value

Type,Source, and Value are unique constraints in the Threat Intel Feed module. You can specify unique constraints using the Module Editor.

Type: type

Type,Source, and Value are unique constraints in the Threat Intel Feed module. You can specify unique constraints using the Module Editor.

Reputation: input.params['reputation_resolved']

TLP: input.params['tlp_resolved']

Threat Types:

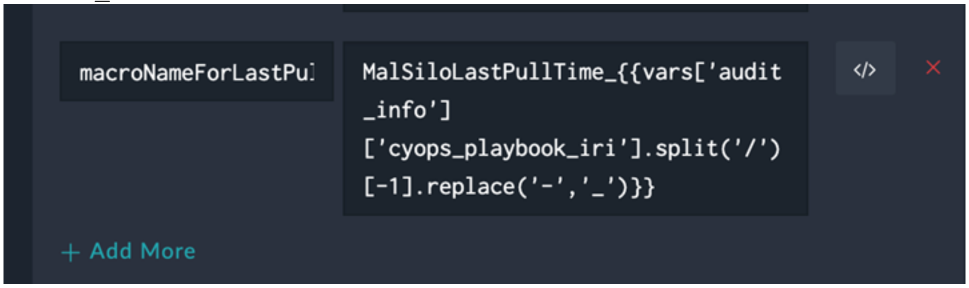
Kill Chain Phases:

Source Data:

```
{
  tags: peexe32,pegui
  type: ipv4
  value: 41.216.183.175:4404
  threat: coinminer
  Lastseen: 2022-10-21
  Firstseen: 2022-10-21
}
{
  tags: peexe32,pegui,assembly
  type: ipv4
  value: 171.22.30.147:80
  threat: andromeda
  Lastseen: 2022-10-21
  Firstseen: 2022-10-21
}
{
  tags: peexe32,pegui
  type: ipv4
  value: 41.216.183.52:9882
  threat: emotet
  Lastseen: 2022-10-22
```

d. **Create Playbook**: This playbook creates FortiSOAR records for every data element fetched from the data source.
   i. Must have an additional tag named "`create`".
   ii. Must have a "Create Record" step named "**Create**". The Data Ingestion wizard displays all the data mapping rules using this step. Any changes to the mapping done by the user are also saved back to this step in the playbook.

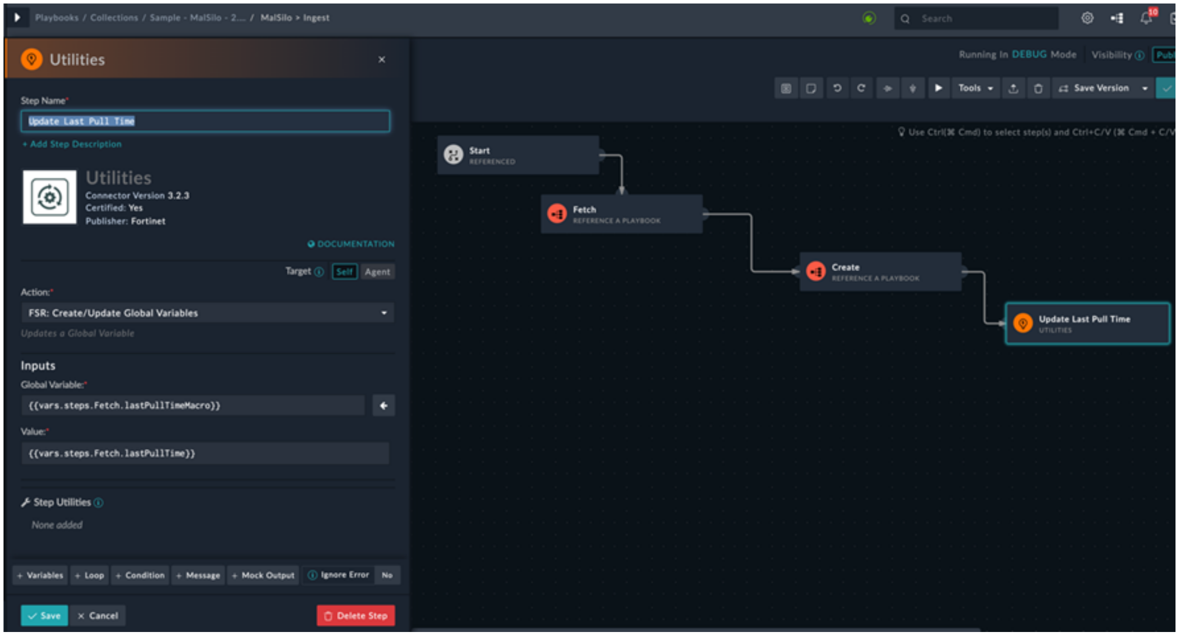
e. **Ingest Playbook**: This is a parent playbook that combines both the "Fetch" and "Create" phases.
   i. Must have an additional tag named "`create`".
   ii. Must invoke the "fetch" and "create" playbooks as "Reference Playbook" steps. The data returned from the "fetch" step must be passed as input to the "create" step.
   iii. For schedule-based ingestions, you can set the last successful pull time as the last step of this playbook, marking the successful completion of one cycle of ingestion. This ensures that subsequent runs pulls data created or modified after this time. This value can be stored as a FortiSOAR Global Variable. To make a global variable unique per configuration of the ingestion, you can append the `config_id` to the variable by using the following Jinja expression:
   `<Source>_ {{vars['audit_info']['cyops_playbook_iri'].split('/')[-1].replace('-','_')}}`



macroNameForLastPu]

```
MalSiloLastPullTime_{{vars['audit_info']['cyops_playbook_iri'].split('/')[-1].replace('-','_')}}
```

You can refer to any data ingestion collection, for example, the "Sample-MalSilo" data ingestion to see how to save global variables and use them while fetching data.



Step Name*: Update Last Pull Time

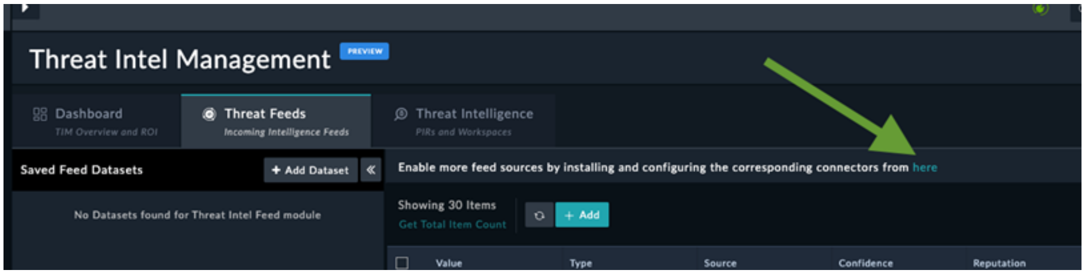
Action:* FSR: Create/Update Global Variables

Updates a Global Variable

Global Variable:* {{vars.steps.Fetch.lastPullTimeMacro}}

Value:* {{vars.steps.Fetch.lastPullTime}}

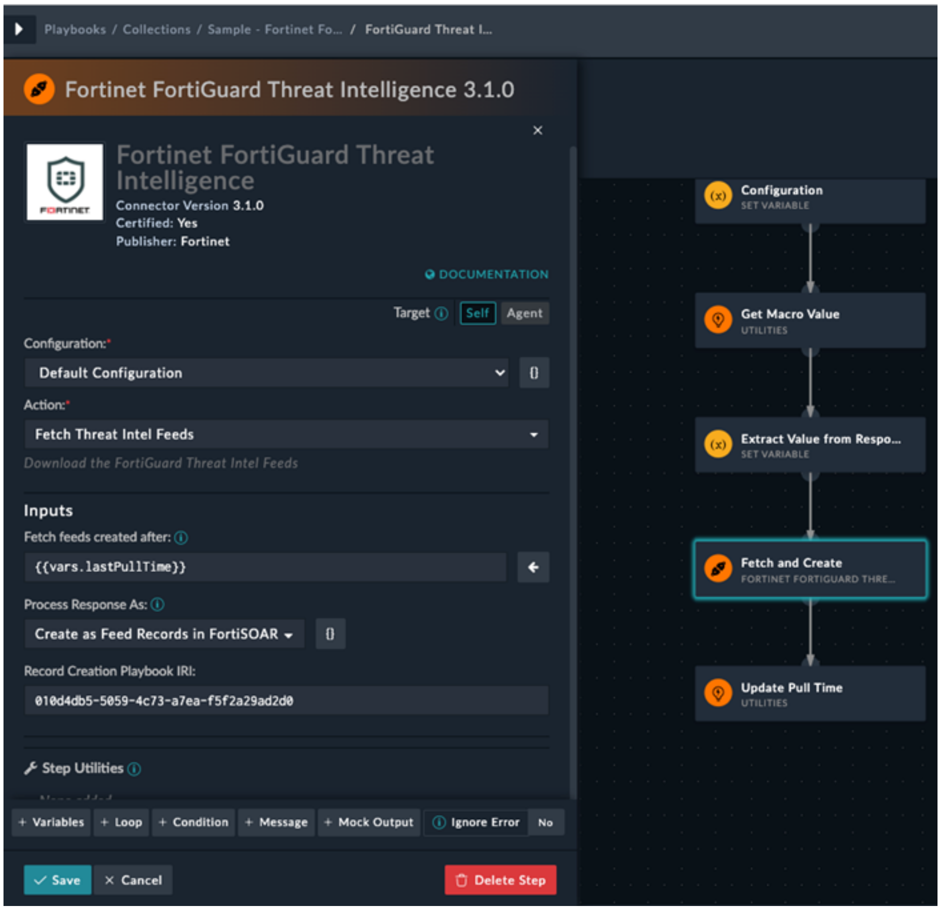
3. Updates specific to Threat Intelligence Management (TIM) ingestion:
   a. The TIM ingestion integrations must have an additional tag, "`ThreatIntel`" in the connector's info.json. This tag is used to display the connector in the listing for TIM-enabled connectors: `https:<fortisoar-url>/content-hub/all-content/?tag=ThreatIntel`



Enable more feed sources by installing and configuring the corresponding connectors from here

No Datasets found for Threat Intel Feed module

   b. Instead of the "Create Record" step, the playbooks must use the "Ingest Bulk Feed" step to handle larger volumes of feeds.
   c. Leverage pagination in APIs, as much as possible, while fetching data from the data source. Fetch and create one batch of data before fetching the next page. This reduces the memory requirements for data ingestion. If pagination is not supported by the data source, you can save on memory requirements by invoking the creation playbooks directly from the connector action that fetches data. To do so, take the IRI of the playbook that creates the feed records as an input to the "**Fetch and Create**" step. Note that the name of this step must also be kept the same.



Playbooks / Collections / Sample - Fortinet Fo... / FortiGuard Threat I...

Fortinet FortiGuard Threat Intelligence 3.1.0

Fortinet FortiGuard Threat Intelligence
Connector Version 3.1.0
Certified: Yes
Publisher: Fortinet

Target: Self

Configuration:* Default Configuration

Action:* Fetch Threat Intel Feeds

Download the FortiGuard Threat Intel Feeds

Inputs

Fetch feeds created after: `{{vars.lastPullTime}}`

Process Response As: Create as Feed Records in FortiSOAR

Record Creation Playbook IRI: `010d4db5-5059-4c73-a7ea-f5f2a29ad2d0`

Step Utilities

Ignore Error: No

Use the following function call in the connector to trigger these playbooks:

Import:

```
try:
   from integrations.crudhub import trigger_ingest_playbook
except:
   # ignore. lower FSR version
   # pass
```

Function Call:

```
trigger_ingest_playbook(batch, create_pb_id, parent_env=parent_env, batch_
size=1000, dedup_field="pattern")
```

Q&A Time

Q1. Why do I need three playbooks – Fetch/Create/Ingest and not just one?

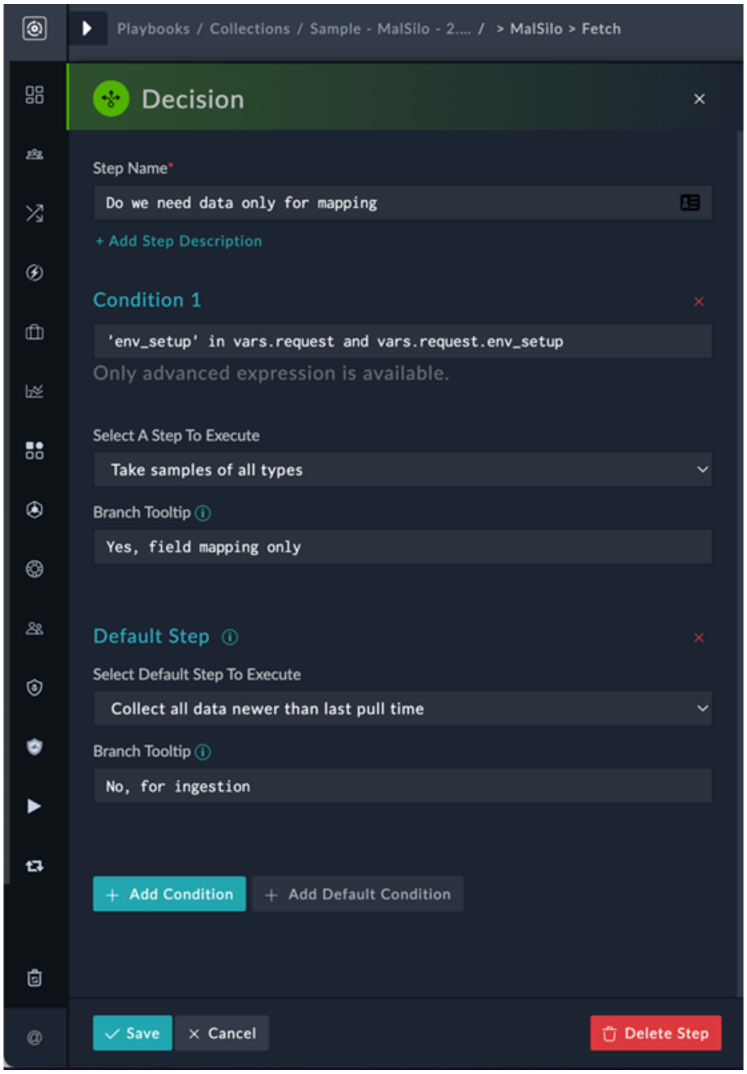
Answer:

The data ingestion collections get invoked from two flows:

1. For data mapping using the Data Ingestion Wizard.
2. For record creation through schedules or listener based notifications.
   For 1), the wizard invokes the playbook tagged with a "`fetch`" tag; hence, this playbook must only fetch data, and not create the corresponding records in FortiSOAR, as the user is still evaluating the queries and mappings.

For 2), the complete ingestion flow must run and save the last pulled time values; hence, in this case, the playbook with the "`ingest`" tag gets called.

However, you could still use a single playbook for the entire data ingestion and put all three tags, "`create`", "`fetch`", "`ingest`". When a playbook is triggered from the data ingestion wizard, it has a variable called "`env_setup`" in the request object. When the value of the "`env_setup`" request object is set to '**true**', it is used to skip the data creation steps.



Playbooks / Collections / Sample - MalSilo - 2.... / > MalSilo > Fetch

Decision

Step Name*: Do we need data only for mapping

+ Add Step Description

Condition 1

```
'env_setup' in vars.request and vars.request.env_setup
```

Only advanced expression is available.

Select A Step To Execute: Take samples of all types

Branch Tooltip: Yes, field mapping only

Default Step

Select Default Step To Execute: Collect all data newer than last pull time

Branch Tooltip: No, for ingestion

Q2. What happens to my configured ingestions when I upgrade the connector?

a. Will the ingestion use the latest connector version?

Answer: Yes. If the new version of the connector is compatible with the previously installed connector version, it is replaced automatically, and the ingestion starts using the latest installed connector version.

b. Are my previous mappings lost since the sample collection is now updated?

Answer: No. When you set up the data ingestion for an integration, it creates a clone of the sample collection. All user inputs from the "`Fetch Data`" and "`Field Mapping`" pages of the wizard are stored in the cloned collection. The cloned collection is the one that is scheduled for ingestion, and not the original sample collection.

c. Will any fixes/changes to the data ingestion be inherited by my existing ingestion configuration?

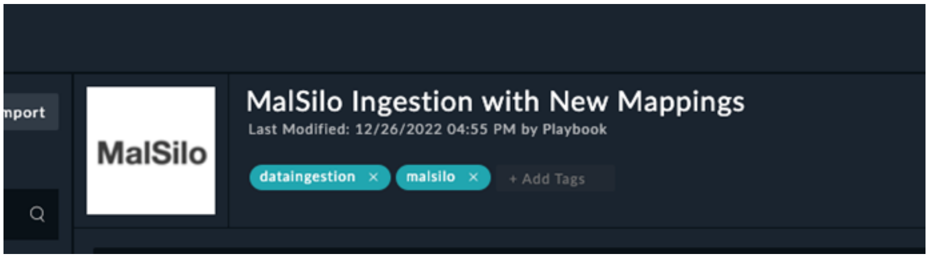
Answer: Yes, if the changes are in the connector actions.
No, if the changes are in the ingestion playbooks. You can re-configure the ingestion using the new default playbooks in the 'Sample Collection', or manually apply the changes to your cloned playbooks.

Q3. How can I use a custom collection other than the default sample collection?

Answer:

You can build your ingestion collection separately using the steps listed in the Configuring a custom connector to support data ingestion topic, and add the following two **tags** to the **playbook collection**:

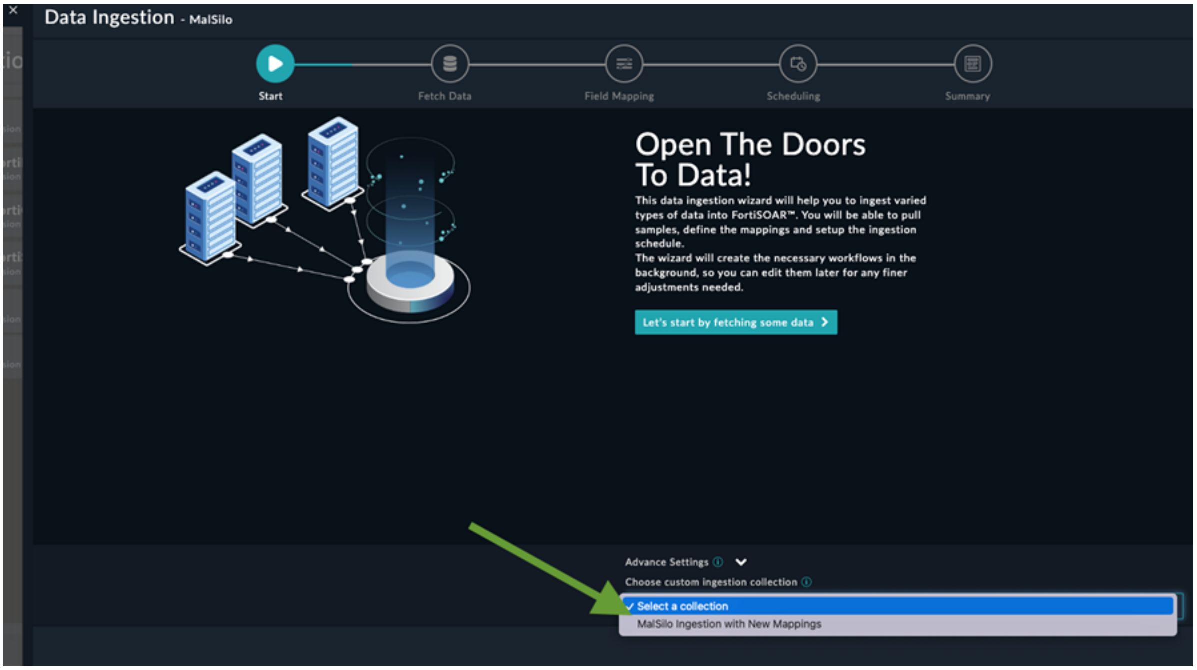
1. `"dataingestion"`
2. `"<connectorname>"`. For example, `"malsilo"`



MalSilo Ingestion with New Mappings

Last Modified: 12/26/2022 04:55 PM by Playbook

These custom collections are then available as alternates for setting up ingestion in the Data Ingestion Wizard. You can use this feature while building the default ingestion collection for an integration or ship it as a new Solution Pack with customized ingestion.



Data Ingestion - MalSilo

Open The Doors To Data!

This data ingestion wizard will help you to ingest varied types of data into FortiSOAR™. You will be able to pull samples, define the mappings and setup the ingestion schedule.
The wizard will create the necessary workflows in the background, so you can edit them later for any finer adjustments needed.

Choose custom ingestion collection: ✓ Select a collection

Start
Fetch Data
Field Mapping
Scheduling
Summary
Let's start by fetching some data
Advance Settings
MalSilo Ingestion with New Mappings

Building your own Pluggable Enrichment Playbook

The SOAR Framework Solution Pack (SP) version 2.0.0 introduces the 'Pluggable Enrichment' framework, using which you easily design pluggable enrichment playbooks for any integration and use them in the enrichment process without modifying the existing playbooks in '03-Enrich' collection. For detailed information about the SOAR Framework SP, see the SOAR Framework SP documentation in the Content Hub portal. Some threat intelligence connectors already contain pluggable enrichment playbooks; more information of which is available here.

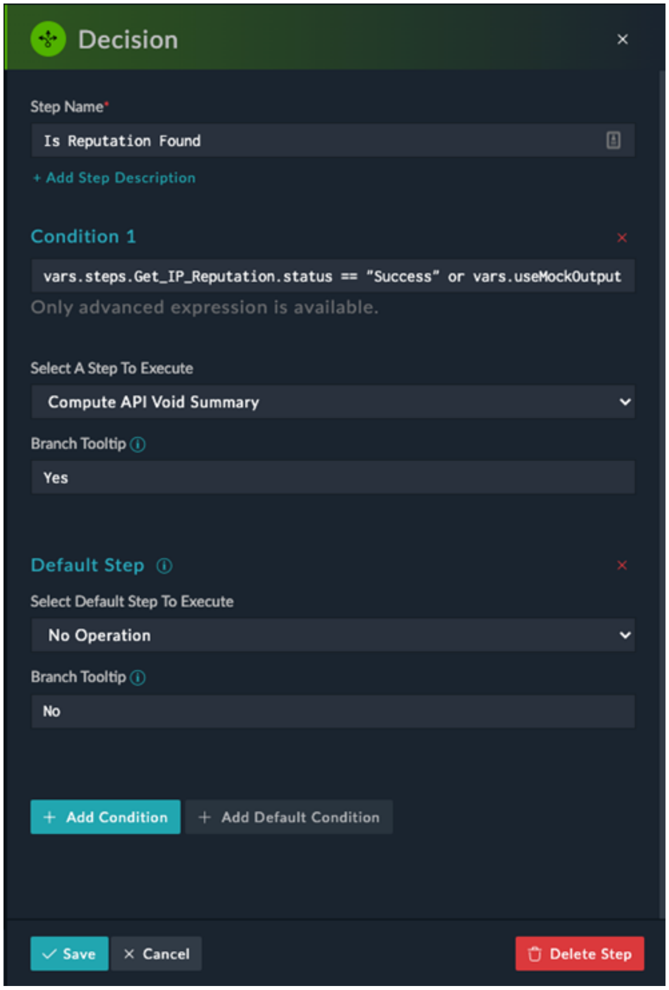
The following steps describe how to design your pluggable enrichment playbooks for 'IP Enrichment' using the 'API Void' integration:

1. Install and configure the API Void v1.0.1 connector.
2. Navigate to **Automation** > **Playbooks**, and search for API Void.
3. Select the 'Sample - APIVoid - 1.0.1' playbook collection.
4. Click **Add Playbook** to create a new pluggable enrichment playbook.
5. Specify the name of the new playbook, such as, "IP Address > API Void > Enrichment".
6. Choose **Referenced** as the 'Trigger' step, and click **Save** to save the playbook.
7. Add appropriate tags to your pluggable enrichment playbook. Since this example is for designing this pluggable enrichment playbook for 'IP Address', add the “`IP_Enrichment`” tag to the playbook.
   You can add the following tags to the pluggable enrichment playbook for different types of indicators:
   - **Domain** : `Domain_Enrichment`
   - **Email Address** : `Email_Enrichment`
   - **File** : `File_Enrichment`
   - **FileHash-MD5**, **FileHash-SHA1**, **FileHash-SHA256**: `FileHash_Enrichment`
   - **IP Address** : `IP_Enrichment`
   - **URL** : `URL_Enrichment`
   - **User** : `User_Enrichment`
8. The “Enrich Indicators (Type All)” parent playbook that is part of the "03 – Enrich" playbook collection calls the "IP Address > API Void > Enrichment" child pluggable enrichment playbook by passing the following parameters:
   a. “`indicator_value`” : Value of the IP address that needs to be enriched.
   b. “`style_colors`” : Value of default enrichment colors assigned to the indicator based on its reputation.
   For further details, see the Extending Default Indicator Enrichment Process topic.
9. Open the “IP Address > API Void > Enrichment” playbook in the playbook designer, and click **Tools** > **Edit Parameters**.
   In the **Parameters** popup, add the `indicator_value` and `style_colors` parameters.
10. Edit the “IP Address > API Void > Enrichment” playbook as follows:
    a. Add the **Set Variable** step” with the name as “`Configuration`”, and specify the following variables and their values:
       - `useMockOutput`: `{{globalVars.Demo_mode}}`
         To execute the playbook using mock output, set the global variable `Demo_mode` to `true`.
       - `indicator_value`: `{{vars.input.params['indicator_value']}}`
         The value of the indicator that you want to enrich. This value comes from the parent playbook.
       - `good_score`: `0`
         `suspicious_score`: `89-1`
         `malicious_score`: `100-90`
         After enrichment, the API returns a high-reliability value called “verdict,” which is used to predict the reputation of indicators.
         “good_score” is the range of good verdicts.
         “suspicious_score” is the range of suspicious verdicts.

"malicious_score" is the range of malicious verdicts.
You can update the values of these parameters based on your requirements.

b. Add the **Connector** step, and search for the `APIVoid v1.0.1` connector. Specify the following values for this step:
    i. In the **Step Name** field, specify `Get IP Reputation`.
    ii. From the **Action** drop-down list, select **Get IP Reputation**.
    iii. In the **IP Address** field, specify `{{vars.indicator_value}}`.

c. Add the **Decision** step for checking whether or not the reputation is found:
    i. In the **Step Name** field, specify `Is Reputation Found`.
    ii. Click **Add Condition**:
        i. Define 'Condition 1' as follows:
        `(vars.steps.Get_IP_Reputation.status == "Success" or vars.useMockOutput)`
        ii. From the **Select A Step To Execute** drop-down list, select **Compute API Void Summary**.
        iii. In the **Branch Tooltip** field, specify `Yes`.
    iii. Click **Add Default Condition** to define the default step as follows:
        i. From the **Select A Step To Execute** drop-down list, select **No Operation**.
        ii. In the **Branch Tooltip** field, specify `No`.



Decision

Step Name*
Is Reputation Found
+ Add Step Description

Condition 1
vars.steps.Get_IP_Reputation.status == "Success" or vars.useMockOutput
Only advanced expression is available.

Select A Step To Execute
Compute API Void Summary

Branch Tooltip
Yes

Default Step

Select Default Step To Execute
No Operation

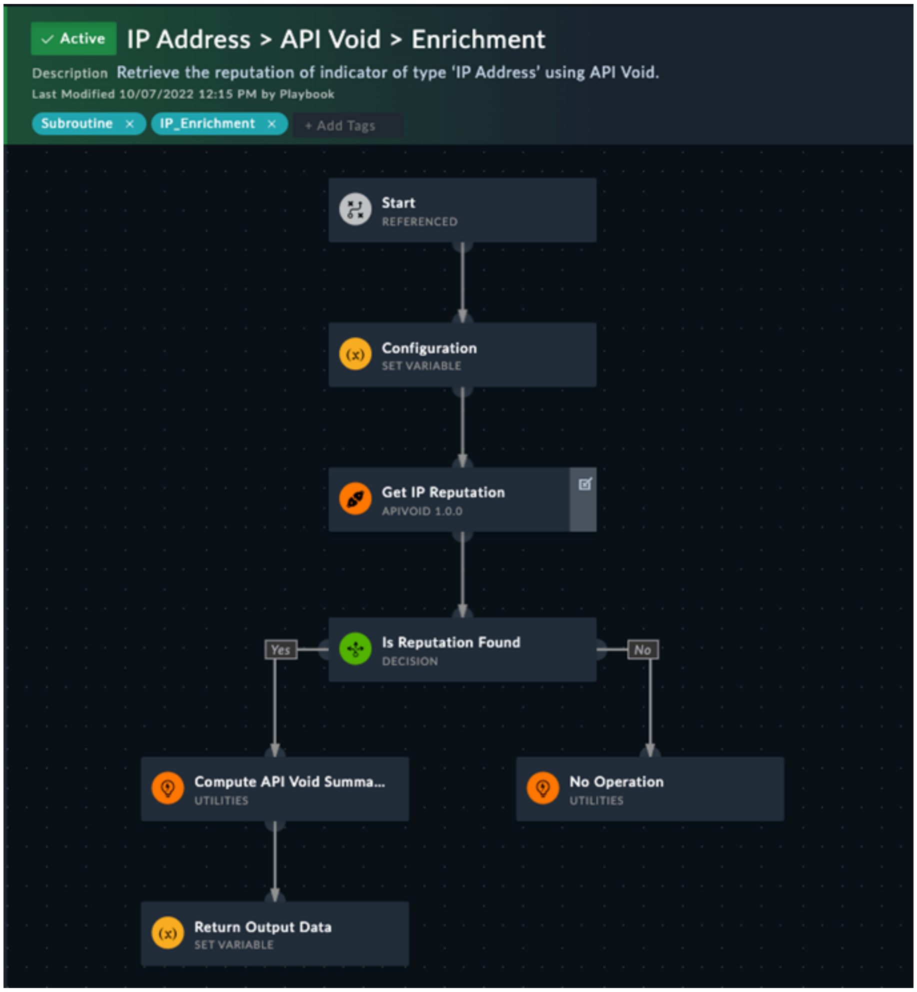
Branch Tooltip
No

d. Add the **Utilities** step and edit it as follows:
   i. In the **Step Name** field, specify `Compute API Void Summary`.
   ii. From the **Action** drop-down list, select **Utils: Format as RichText (Markdown)**.
   iii. In the **Inputs** field, specify the code that you want to render as the 'Enrichment Summary' in the **Description** field of the indicator. You can refer to samples of the code from any pluggable enrichment playbook, for example, pluggable enrichment playbooks in the API Void sample playbook collection, "IP Address > API Void > Enrichment".

e. Add another **Utilities** step and edit it as follows:
   i. In the **Step Name** field, specify `No operation`.
   ii. From the **Action** drop-down list, select **Utils: No Operation**.

f. Add the **Set Variable** step" with the name "`Return Output Data`", and set the return values to the parent playbook, 'Enrich Indicators (Type All)', present in the '03 – Enrich' playbook collection. See the apivoid connector document for the return values that can be sent to the parent playbook.

Sample "IP Address > API Void > Enrichment" pluggable enrichment playbook:



IP Address > API Void > Enrichment

Description Retrieve the reputation of indicator of type 'IP Address' using API Void.

Last Modified 10/07/2022 12:15 PM by Playbook

FortiSOAR Connector SDK

The FortiSOAR Connector SDK consists of a simulator for writing and testing your connector. You can use this CLI to help you develop your custom connector, without having a ForiSOAR™ instance.

For information on how to set up and use the FortiSOAR Connector SDK, see Fortinet Developer Network (FNDN). You can also download the Connector SDK from FNDN and use it to develop a custom connector. You must log into FNDN to access the Connector SDK.

Data Ingestion

FortiSOAR has a dedicated data ingestion wizard that facilitates data ingestion from external SIEM solutions and other third-party sources like threat intelligence platforms, email solutions, etc. The wizard also takes care of scheduling data ingestion into FortiSOAR, if the connector is enabled for scheduling.

The Data Ingestion wizard eases data ingestion configuration and mapping of fields between the two systems. The wizard assists in fetching sample data from the source, mapping fields from the source data into a FortiSOAR module, and setting up an ingestion schedule if required and if the connector is enabled for scheduling. Based on the inputs provided in the wizard, the system dynamically generates the ingestion workflows without the user being exposed to the details of playbooks and easing the process of configuring ingestion.

The Data Ingestion wizard allows the mapping of drop-down (picklist) fields and items such that you can map picklist items to a range of values, lesser than, and greater than values. The Data Ingestion wizard allows you to define the frequency at which you want to pull content into FortiSOAR from third-party integration, such as SIEMs that allow you to define the polling frequency. By default, the frequency of pulling content is set at five minutes.

FortiSOAR and its connectors come with sample playbooks for ingesting data from various SIEM sources, Threat Intelligence Platforms, Vulnerability Management Tools, Configuration Management Databases, etc. Connectors that you can use to ingest data through this wizard are enabled by default in FortiSOAR.

Data ingestion is generally mapped to the alerts or incident modules, and from release 7.2.0 onwards, the Incident Response modules get added once you have installed the SOAR Framework Solution Pack (SP). Therefore, for data ingestion to work correctly you must install this solution pack. *From release 7.2.0 onwards, the SOAR Framework Solution Pack is installed by default with the fresh installations of FortiSOAR*. For detailed information about the SOAR Framework SP, see the SOAR Framework SP documentation.

Data ingestion also has the following features:

- Supports fetching of data for each configuration of your connector, i.e., if you have two configurations present in your connector, then you can pull ingestion data for each configuration.
- Supports multiple queries for pulling data based on your requirement; however, you must specify one query to schedule the pulling of data from the connector into FortiSOAR.
- Supports easy and efficient ingestion of data from various sources using the intuitive UI of the Data ingestion wizard.
- Supports monitoring of the data ingestion that you have set up and provides information on which connectors are configured for using the Data Ingestion Wizard on the **Data Ingestion** tab of the `Connectors` page. The Data Ingestion also provides you other information such as what is the status of a configuration, what is the schedule for ingestion, when data was last pulled using that configuration, etc.
- Supports inserting or upserting records in bulk during data ingestion, which improves the performance since all the records are created or upserted in one request.

Modes of Data Ingestion

Data Ingestion can work in any of the following three modes:

- **Notification Based**: Some connectors such as IMAP, Exchange, Syslog, etc have a Notification Service that gets instantly notified when a message arrives on the server. The notification service, in turn, triggers a FortiSOAR playbook to create FortiSOAR records from the fetched data.
- **Schedule Based**: Some connectors such as QRadar, Anomali, ServiceNow, etc use the Fetch APIs of the integration, i.e., for example, the fetch API of Anomali, along with a user-defined query to fetch data from the product into FortiSOAR. These fetch playbooks are scheduled to run by default, every 5 mins, or can also be scheduled to run according to a user-defined interval.
  Note that most integrations that support a Notification Based ingestion also support a Schedule Based ingestion. You must, however, configure only one of the two since otherwise both would pull the data from the same source and there might be data loss due to conflicts.
- **App Push**: Some connectors such as Splunk have an add-on that is provided by FortiSOAR and which can be installed on the server-side to push data from the integration to FortiSOAR. The add-on is configured with a user-password or appliance-based authentication in FortiSOAR and it triggers FortiSOAR playbooks to create the records in FortiSOAR.

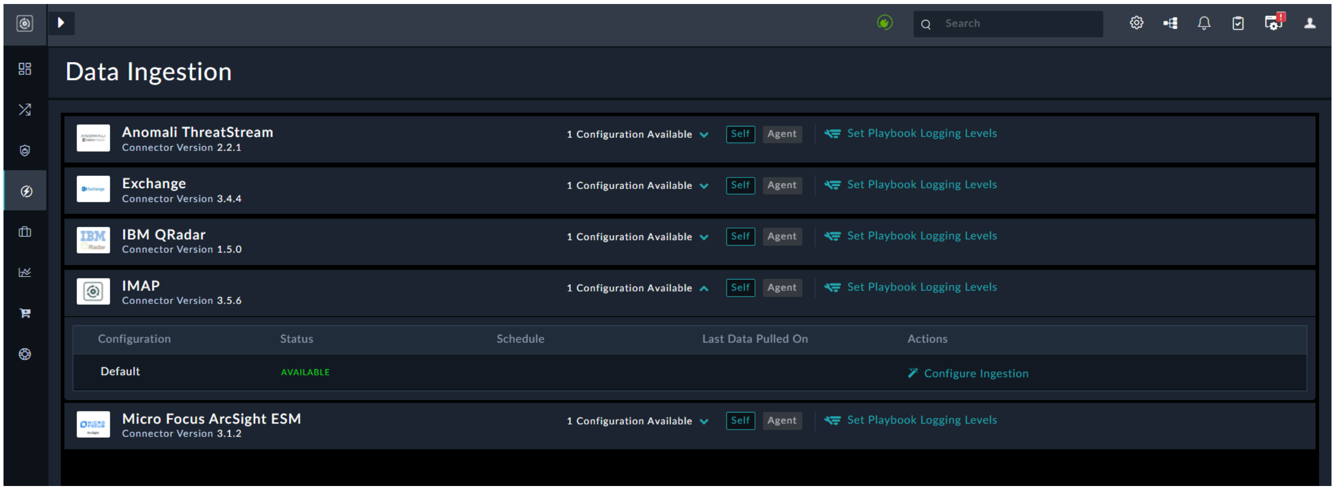
Permissions required for using the Data Ingestion Wizard

For permissions on using connectors, see the Introduction to connectors chapter. Apart from this, to use data ingestion, you must be assigned a role that has `Read` and `Update` permissions on the `Connectors` module and `Create`, `Read`, `Update`, and `Execute` permissions on the `Playbooks` module. If you want to schedule the data ingestion, then you are also required to have `Create`, `Update` and `Read` permissions on the `Schedules` module.

Data Ingestion Wizard

The **Data Ingestion** page displays connectors that are configured for using the Data Ingestion Wizard, i.e., you can use the data ingestion wizard to ingest data into FortiSOAR for these connectors.

To view the data ingestion page, log on to FortiSOAR, and on the left navigation pane, click **Automation** > **Data Ingestion**, which displays all the connectors that are installed on your system and are enabled for data ingestion, along with the number of configurations available for that connector.



Data Ingestion

Anomali ThreatStream
Connector Version 2.2.1
1 Configuration Available

Exchange
Connector Version 3.4.4
1 Configuration Available

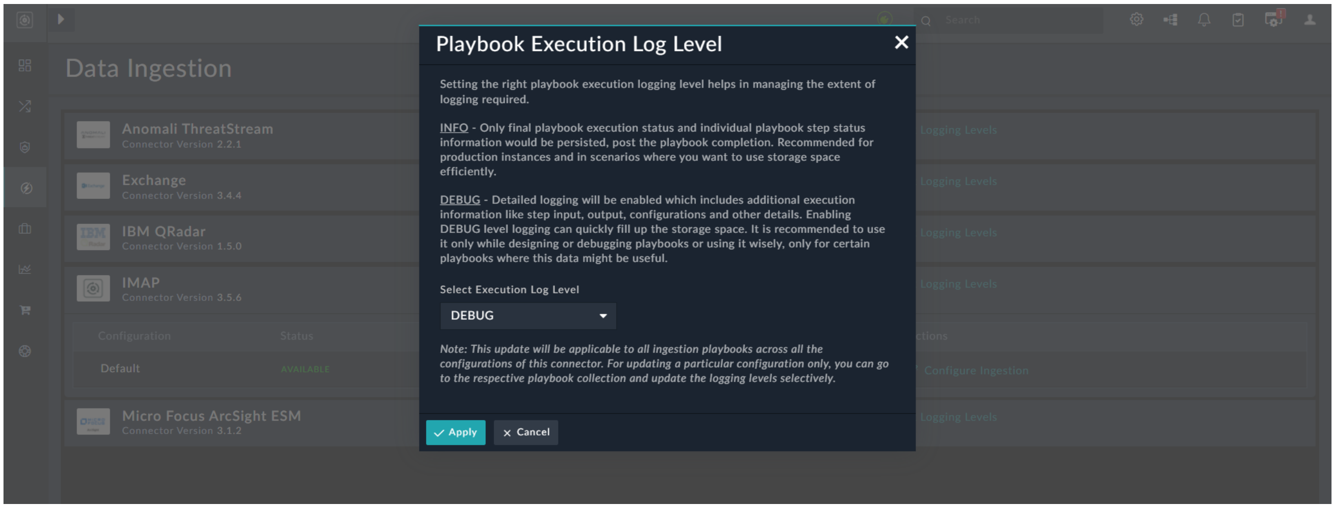
IBM QRadar
Connector Version 1.5.0
1 Configuration Available

IMAP
Connector Version 3.5.6
1 Configuration Available

Configuration	Status	Schedule	Last Data Pulled On	Actions
Default	AVAILABLE			Configure Ingestion

Micro Focus ArcSight ESM
Connector Version 3.1.2
1 Configuration Available

From version 7.0.2 onward, you can set the logging levels (`INFO` or `DEBUG`) for your ingestion playbook by clicking **Set Playbook Logging Levels**, which displays the `Playbook Execution Log Level` dialog:



Playbook Execution Log Level

Setting the right playbook execution logging level helps in managing the extent of logging required.

INFO - Only final playbook execution status and individual playbook step status information would be persisted, post the playbook completion. Recommended for production instances and in scenarios where you want to use storage space efficiently.

DEBUG - Detailed logging will be enabled which includes additional execution information like step input, output, configurations and other details. Enabling DEBUG level logging can quickly fill up the storage space. It is recommended to use it only while designing or debugging playbooks or using it wisely, only for certain playbooks where this data might be useful.

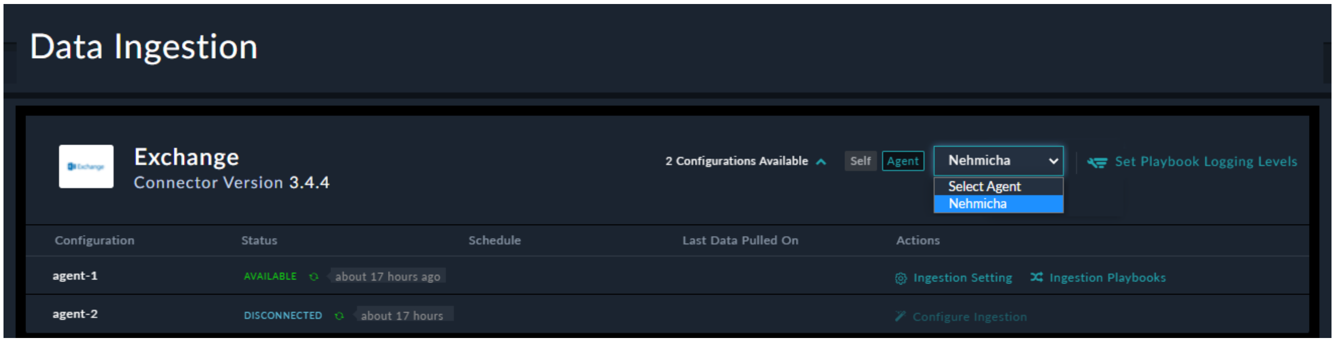
Select Execution Log Level: DEBUG

Note: This update will be applicable to all ingestion playbooks across all the configurations of this connector. For updating a particular configuration only, you can go to the respective playbook collection and update the logging levels selectively.

From the **Select Execution Log Level** field, select the logging level that you want to set for this playbook and click **Apply**. By default, the `INFO` log level is set for ingestion playbooks; however, if you want extensive logging for ingestion playbooks, then retain the logging level as `DEBUG`, and click **Apply**.

Changes made to the logging level apply to all ingestion playbooks across all the configurations of the particular connector. If you want to update only a particular configuration of the connector, then you have to go to the respective playbook collection and selectively update the logging levels.

You can also run the data ingestion wizard on agent configurations. To display the details of the configurations, click the `<number of configurations> configurations available` link. To view the configuration details of the current node, click **Self**. To view the agent configuration details, click **Agent**, and then from the drop-down list, select the name of the agent whose configuration details you want to view, as shown in the following image:



Data Ingestion

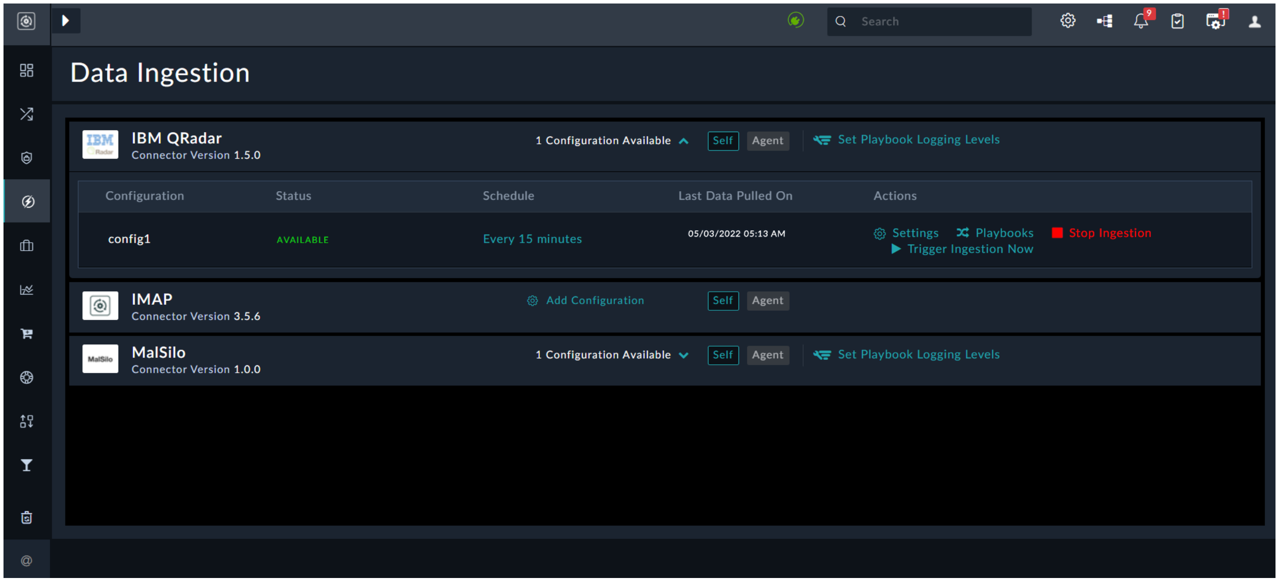
Exchange
Connector Version 3.4.4

Configuration	Status	Schedule	Last Data Pulled On	Actions
agent-1	AVAILABLE about 17 hours ago			Ingestion Setting Ingestion Playbooks
agent-2	DISCONNECTED about 17 hours			Configure Ingestion

Details that are displayed for the configurations are as follows:

- **Status** of a configuration, whether it is **Available** or **Disconnected**.
- **Schedule** for running the ingestion, i.e., what is the frequency at which the third-party integration will be polled to ingest data into FortiSOAR.
- **Last Data Pulled on** contains the DateTime when the data was last pulled into FortiSOAR using this configuration.
- **Actions** contain the following options:
  Configure Ingestion: Clicking this link displays the Data Ingestion Wizard. This option is displayed when you are configuring data ingestion for the first time for a particular configuration.
  Ingestion Settings: Clicking this link displays the Data Ingestion Wizard with the configured ingestion settings. This

option is displayed when you have already configured data ingestion for a particular configuration.
Ingestion Playbooks: Clicking this link opens the ingestion playbooks collection in a new window.
Trigger Ingestion Now: Clicking this link immediately starts the data ingestion process. Subsequent data ingestion occurs based on the data ingestion schedule you have set.
Start Ingestion / **Stop Ingestion**: Clicking this link either starts ingestion for a newly-added configuration or stops ingestion for a particular configuration based on the ingestion settings you have configured.



Data Ingestion

IBM QRadar
Connector Version 1.5.0
1 Configuration Available

Configuration	Status	Schedule	Last Data Pulled On	Actions
config1	AVAILABLE	Every 15 minutes	05/03/2022 05:13 AM	Settings Playbooks Stop Ingestion Trigger Ingestion Now

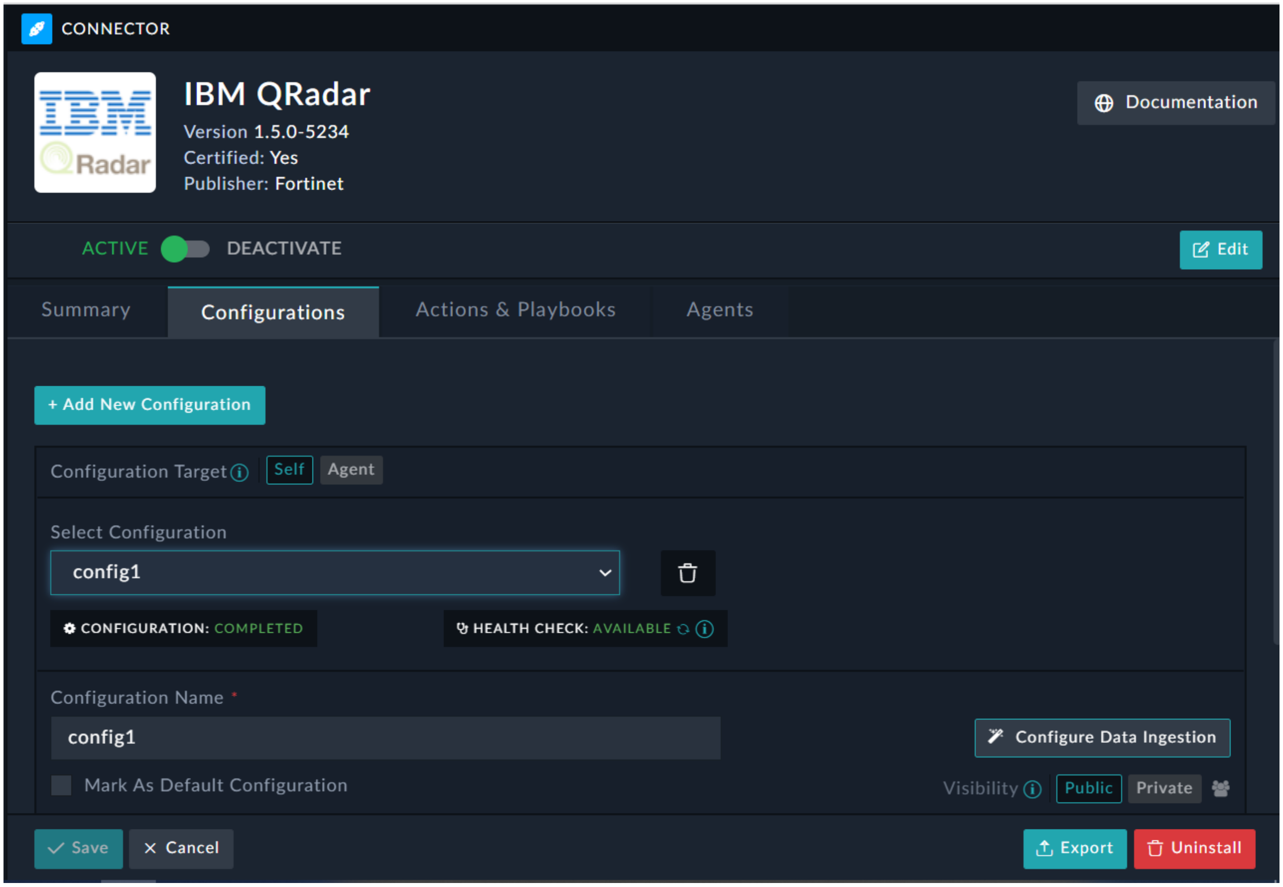
IMAP
Connector Version 3.5.6

MalSilo
Connector Version 1.0.0
1 Configuration Available

If you have upgraded to FortiSOAR 6.0.0 or later from an earlier version, then before you reconfigure ingestion for the same connector configuration, you must deactivate the earlier ingestion playbooks that are present in the ingestion collection for the connector. The links to the ingestion playbooks that were created prior to the upgrade will be present on the **System Fixtures** page in the "Ingestion Playbooks" section will not be visible in the Data Ingestion tab (new in version 6.0.0) of the "Connectors" page. If your data ingestion is schedule-based, then you must also stop or delete the earlier schedules for the connector.

Process of ingesting data using the Data Ingestion Wizard

1. Log on to FortiSOAR.
2. On the left navigation pane, click **Data Ingestion** (see Data Ingestion) or go to **Content Hub** or **Automation** > **Connectors** and click the **Manage** tab, which lists the installed connectors in the card view. For more information, see Data Ingestion.
3. On the **Manage** tab, click the connector card using which you want to ingest data. For our example, we have chosen IBM QRadar.
4. On the `Connector Configuration` popup, from the **Select Configuration** drop-down list select the configuration for which you want to configure the ingestion, and then click **Configure Data Ingestion**.



Version 1.5.0-5234
Certified: Yes
Publisher: Fortinet

Configuration Target: Self

Select Configuration: config1

CONFIGURATION: COMPLETED

HEALTH CHECK: AVAILABLE

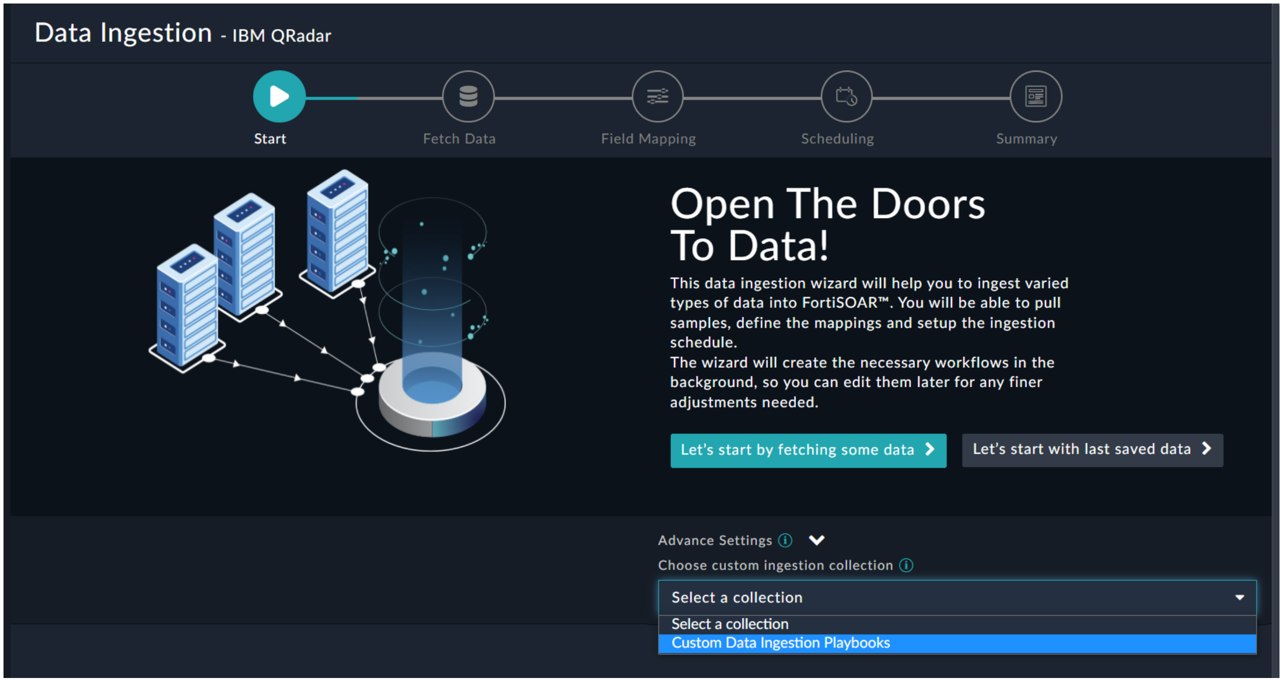
Configuration Name *: config1

□ Mark As Default Configuration

Visibility: Public

Only connectors that have been enabled for data ingestion will have the **Configure Data Ingestion** button.
Note: To ingest data, your connector must be in the **Available** state.

5. Click **Configure Data Ingestion** to display the `Start` screen of the Data Ingestion Wizard.


Data Ingestion - IBM QRadar
Start
Fetch Data
Field Mapping
Scheduling
Summary
Open The Doors To Data!
Let's start by fetching some data
Let's start with last saved data
Advance Settings
Select a collection
Custom Data Ingestion Playbooks

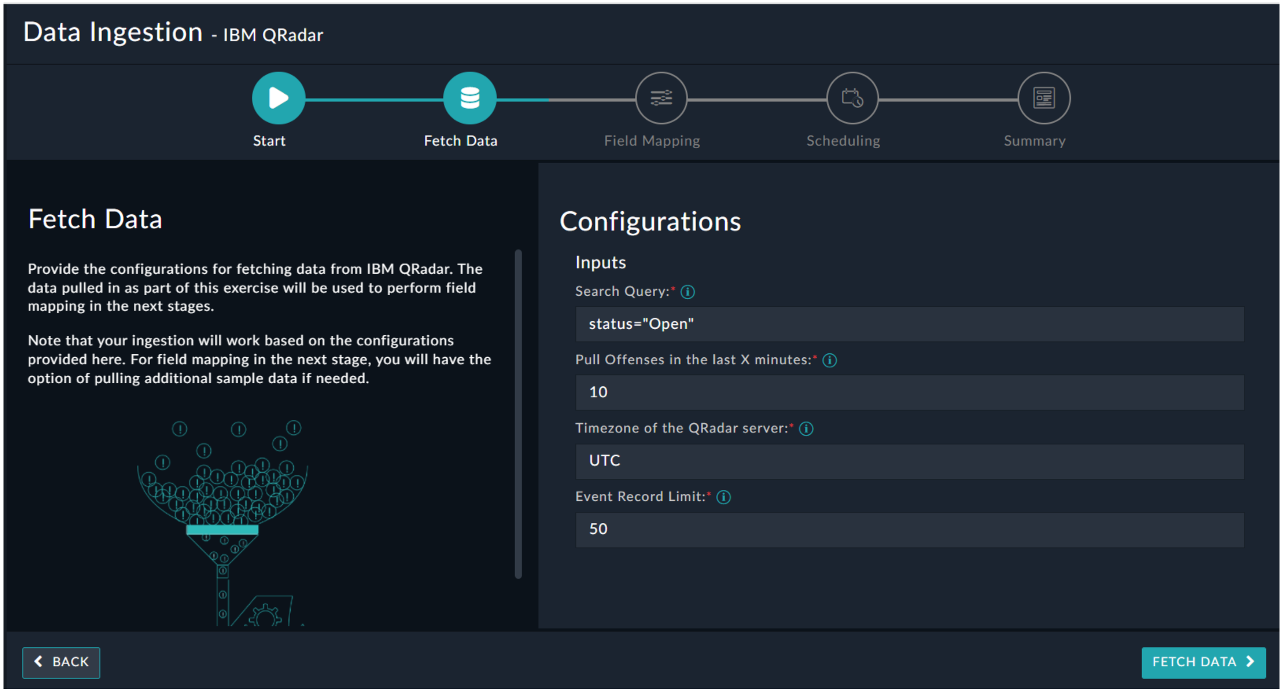
This data ingestion wizard will help you to ingest varied types of data into FortiSOAR™. You will be able to pull samples, define the mappings and setup the ingestion schedule.
The wizard will create the necessary workflows in the background, so you can edit them later for any finer adjustments needed.

Choose custom ingestion collection: Select a collection

If you are running data ingestion for the first time, click the **Let's Start by fetching some data** button to display the

`Fetch Sample Data` screen. However, if you have previously run ingestion using the IBM QRadar connector, you will also see the **Let's start with last saved data** button. Release 7.2.0 has enhanced data ingestion to save the pulled results from the previously run ingestion and use the same instead of re-fetching the results. Clicking **Let's start with last saved data** moves directly to the `Field Mapping` screen, skipping the `Fetch Sample Data` screen, as the data pulled from the previous run ingestion will be used. In this case, a **Skip** button is also added to the `Fetch Data` screen.
Release 7.2.0 also provides you with the option of selecting a custom playbook collection, other than the sample playbook collection that is bundled with the connector, to be used for ingesting data. Examples for the need of custom playbook collections could be in the case of a War Room that is set up with the unified communication feature, where we need communication to link with the war room record whenever a relevant email is ingested, which can be achieved by a custom data ingestion playbooks collection, or in case of MSSP environments where custom data mapping can be referenced for onboarding new customers, etc. Therefore, from release 7.2.0 onwards, you can click **Advance Settings** and from the **Choose custom integration collection** drop-down list, you can select playbooks from different collections that support data ingestion to be used for data ingestion or upload a custom data ingestion playbook collection. For playbook collections to be listed in the **Choose custom integration collection** drop-down list, ensure that playbook collection has a `#dataingestion` tag and the ingestion playbooks have the `#<nameOfConnector>` and `#dataingestion` tags, for example, `#qradar #dataingestion`. If you select a custom data ingestion playbook collection, then data is fetched using the selected playbook collection.
Note: When you use a custom playbook collection for data ingestion, then in this case the sample playbook collection gets overridden by the custom playbook collection in the ingestion collection, i.e., in the playbooks collection that is shown when you click Playbooks in the connector row on the Data Ingestion page.

6. Sample data is required to create a field mapping between your connector data and FortiSOAR. The sample data is pulled from connector actions or ingestion playbooks. By default, FortiSOAR ingests data using an "Ingestion Playbook" that is included by default with each connector that is enabled for data ingestion.
   A connector would generally require some configuration details, which you will require to fill in on this screen. For example, in the case of IBM QRadar, in the **Search Query** field, enter the query based on which you want to fetch data into FortiSOAR. In the **Pull Offenses in Last X Min**, enter the number of minutes for which you want to retrieve created or modified offenses from IBM QRadar, for example, **10**, in this case, the data ingestion wizard will fetch offenses that are created or modified in the last 10 minutes from IBM QRadar. In the **Timezone of the QRadar server**, specify the timezone of the QRadar server; this is used to convert for pulling offenses in the last X minutes, and in the **Event Record Limit**, specify the maximum offenses records you want to fetch, and then click **Fetch Data**.



Data Ingestion - IBM QRadar

Fetch Data

Provide the configurations for fetching data from IBM QRadar. The data pulled in as part of this exercise will be used to perform field mapping in the next stages.

Note that your ingestion will work based on the configurations provided here. For field mapping in the next stage, you will have the option of pulling additional sample data if needed.

Configurations

Inputs

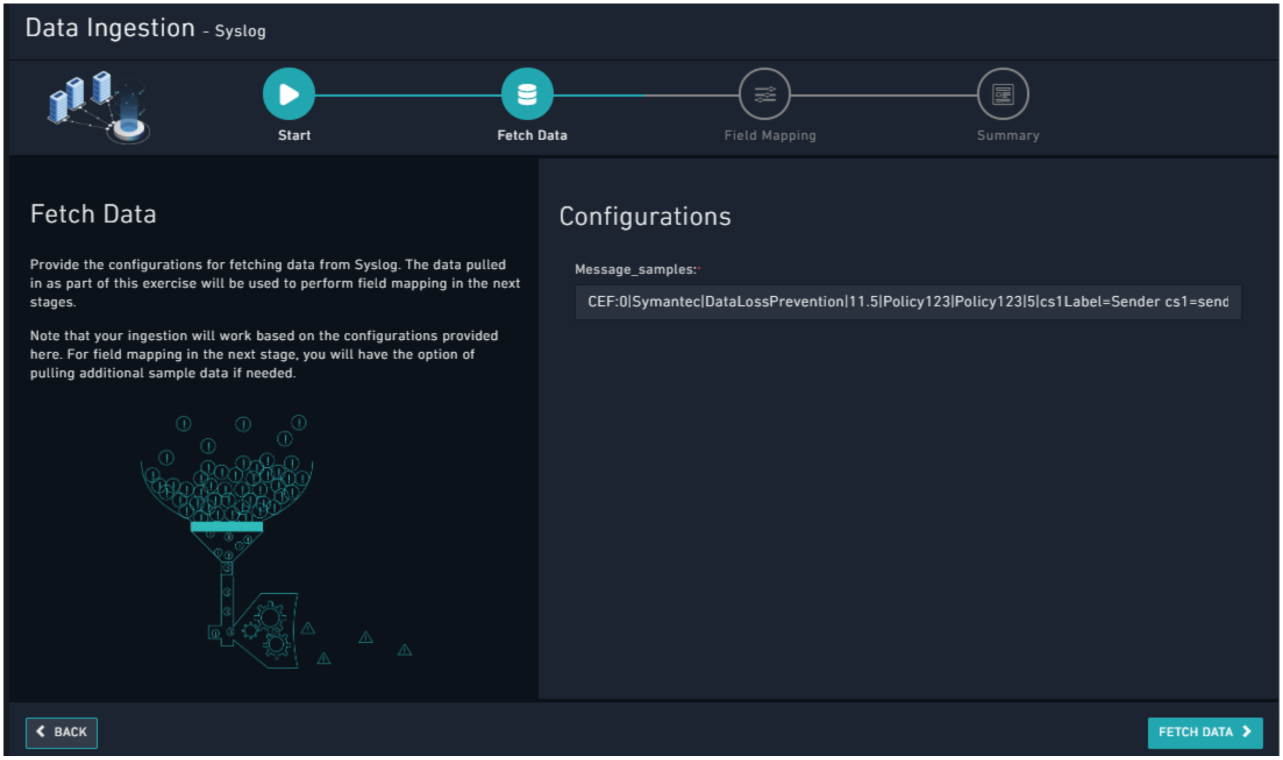
Search Query:* status="Open"

Pull Offenses in the last X minutes:* 10

Timezone of the QRadar server:* UTC

Event Record Limit:* 50

Another example is **Syslog** where you will see the **Message_samples** field in which you can add samples of messages in the CEF Format using which you want to map the fields:



Data Ingestion - Syslog

Fetch Data

Provide the configurations for fetching data from Syslog. The data pulled in as part of this exercise will be used to perform field mapping in the next stages.

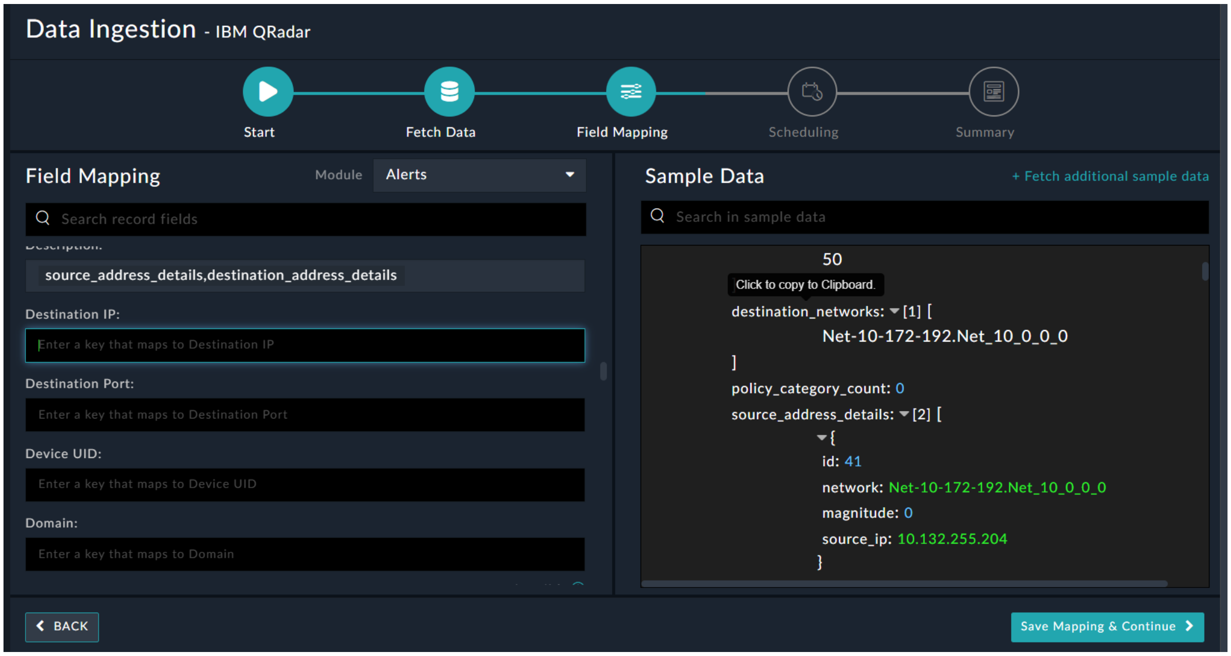
Note that your ingestion will work based on the configurations provided here. For field mapping in the next stage, you will have the option of pulling additional sample data if needed.

Configurations

Message_samples:* CEF:0|Symantec|DataLossPrevention|11.5|Policy123|Policy123|5|cs1Label=Sender cs1=send

7. On the **Field Mapping** screen, map the fields of the sample data to the fields present in FortiSOAR as follows:
   a. The Field Mapping screen displays the Sample Data on the right side of the screen and the Field Mapping (FortiSOAR fields) on the left side of the screen. The sample data is in the form of a Key-Value pair.
   From the **Module** drop-down list that appears next to Field Mapping, select the FortiSOAR module for which you want to map the fields. The default module will be already be selected, for example, **Alerts**.
   Note: If you select any module other than the default module, you will require to remap all the fields.

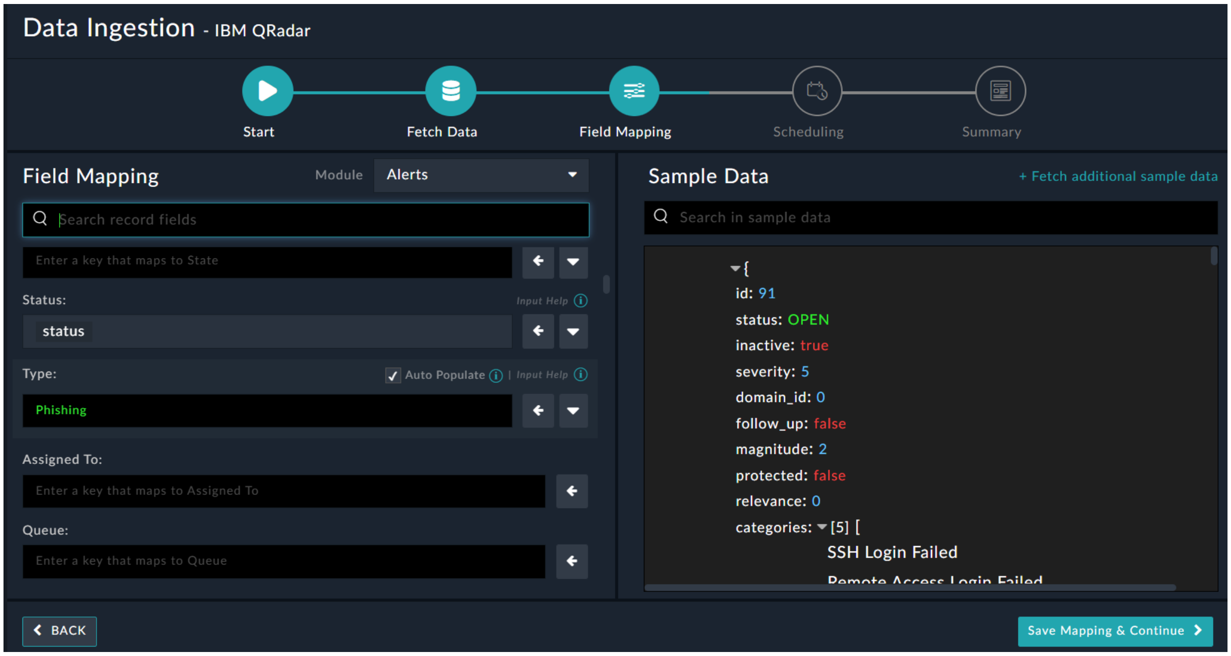
b. From the **Sample Data** fields, map the fields onto the fields present in the **Alerts** module as follows:
Important: Some fields such as Name and some picklists can come pre-mapped with their keys. You do not require to re-map these fields unless you want to override their default values. You can search for fields in the record and the sample data.
To map a field, click the key in the sample data to add jinja for the field. For example, to map the `destination_networks` field from the IBM QRadar sample data to the `Destination IP` field in FortiSOAR, click **Destination IP** and then click **destination_networks**:


Data Ingestion - IBM QRadar
Start
Fetch Data
Field Mapping
Scheduling
Summary
Field Mapping
Module
Alerts
Search record fields
Description:
Destination IP:
Enter a key that maps to Destination IP
Destination Port:
Enter a key that maps to Destination Port
Device UID:
Enter a key that maps to Device UID
Domain:
Enter a key that maps to Domain
Sample Data
+ Fetch additional sample data
Search in sample data
Click to copy to Clipboard.
BACK
Save Mapping & Continue

source_address_details,destination_address_details

```
50
destination_networks: [1] [
        Net-10-172-192.Net_10_0_0_0
]
policy_category_count: 0
source_address_details: [2] [
        {
        id: 41
        network: Net-10-172-192.Net_10_0_0_0
        magnitude: 0
        source_ip: 10.132.255.204
        }
```

To view the key of a field, double click in the field, for example, if you click the newly-mapped description field, its key `{{vars.sourcedata["destination_networks"]}}` will be displayed. Once the key (which is in jinja) is displayed, this field will always be displayed with the key value.
Note: If bulk insert or upsert is supported for the connector, then the key value changes from `{{vars.sourcedata["name of the field"]}}` to `{{vars.item["name of the field"]}}`. For example, `{{vars.item["destination_networks"]}}`.
From version 7.0.0 onwards, if your administrator has enabled any 'Lookup' or 'Picklist' type of field to accept the values generated from the recommendation engine, then you will see an **Auto populate** checkbox appearing beside this field. For example, the 'Type' field in the following image:



Data Ingestion - IBM QRadar

Field Mapping   Module: Alerts

Enter a key that maps to State

Status: status

Type: ☑ Auto Populate | Input Help: Phishing

Assigned To: Enter a key that maps to Assigned To

Queue: Enter a key that maps to Queue

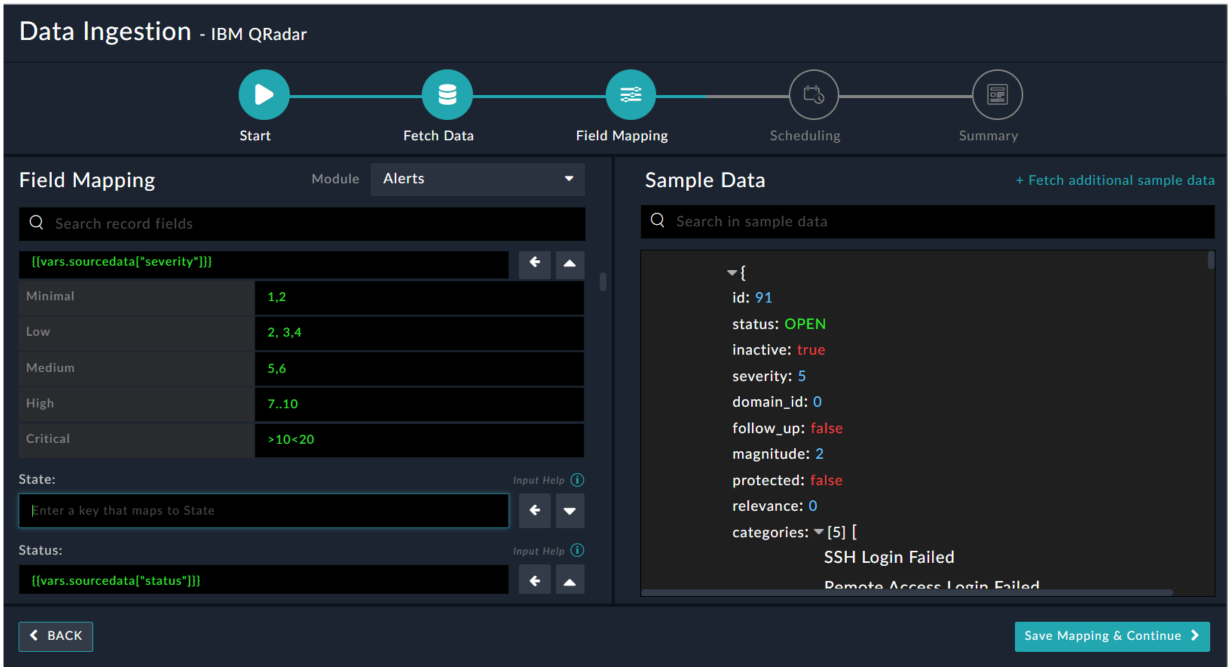
Sample Data

```
{
  id: 91
  status: OPEN
  inactive: true
  severity: 5
  domain_id: 0
  follow_up: false
  magnitude: 2
  protected: false
  relevance: 0
  categories: [5] [
      SSH Login Failed
      Remote Access Login Failed
```

If you select the **Auto populate** checkbox, and users have not specified any values for such fields, then the value of such fields gets auto-populated with the values from the recommendation engine that is based on learning from past similar records.

Picklists can come pre-populated with their defined defaults if they are available. You can also map picklists using the wizard.

To map a picklist, such as **Status**, select the picklist whose value you want to map. For example, `severity` from the IBM QRadar sample data can be mapped to the `Severity` picklist in FortiSOAR by clicking `severity`. FortiSOAR would already have mapped this picklist by default, and it will appear as `severity` whose key value is `{{vars.sourcedata["severity"]}}`. Next, you must map the items of the Reputation picklists. To view and map the items of the Severity picklist in FortiSOAR, click the down arrow (**v**). The Severity picklist has items such as Minimal, Low, Medium, Hight, and Critical. These picklist items can be mapped as per the values defined in the source. You can map more than one picklist item in FortiSOAR can map to a single value in the source. For example, you can map both **2** to both Minimal and Low in FortiSOAR. You can also map one picklist item in FortiSOAR to two values or more values in the source, for example, Low can be 2,3,4. In this case, if the **severity** field in IBM QRadar has values 2, 3, or 4, then all cases with confidence 2 to 4 will be mapped to **Low** severity in FortiSOAR. You can also specify the mapping of a range of values, lesser than, and greater than values. For example, **High** can be mapped `7..10` in FortiSOAR, which means that if the **severity** field in IBM QRadar contains any value between 7 to 10 they will be mapped as **High** severity in FortiSOAR. Similarly, if you map **Critical** to `>10<20` in FortiSOAR this means that if the **severity** field in IBM ThreatStream contains any value greater than 10 and less than 20, then they will be mapped to **Critical** severity in FortiSOAR.



Field Mapping

Module: Alerts

{{vars.sourcedata["severity"]}}

Minimal	1,2
Low	2, 3,4
Medium	5,6
High	7..10
Critical	>10<20

State: Enter a key that maps to State

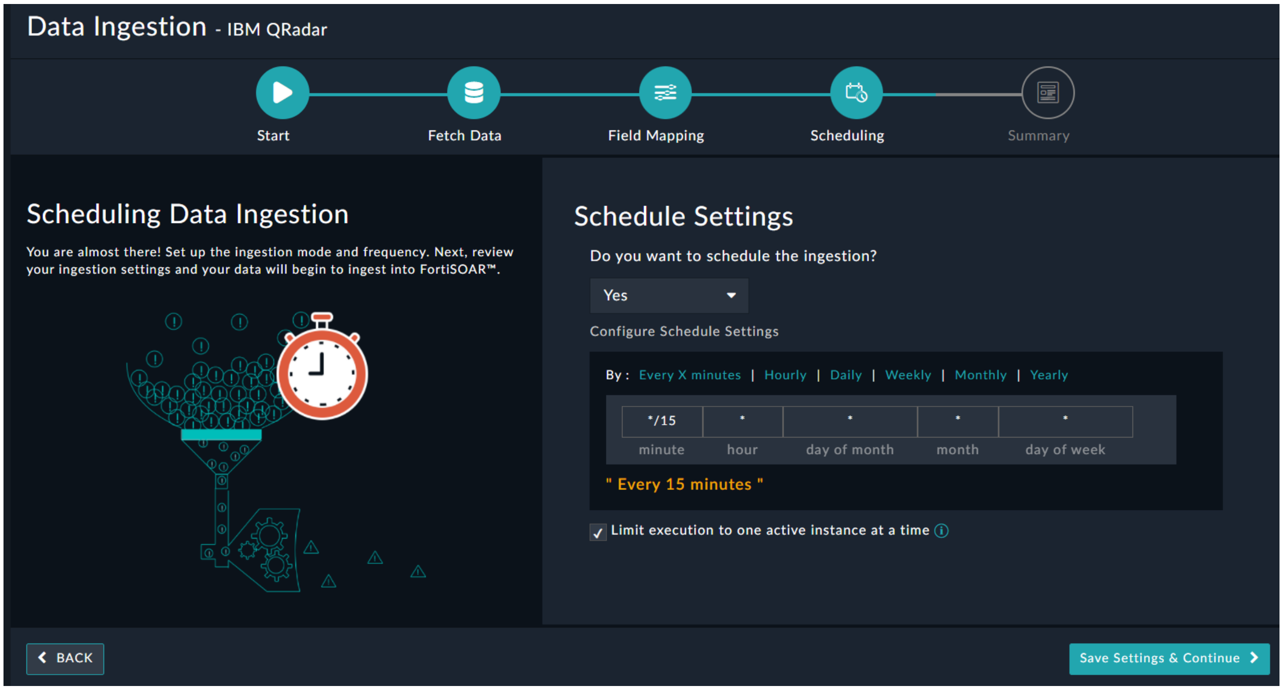
Status: {{vars.sourcedata["status"]}}

Sample Data

```
{
id: 91
status: OPEN
inactive: true
severity: 5
domain_id: 0
follow_up: false
magnitude: 2
protected: false
relevance: 0
categories: [5] [
    SSH Login Failed
    Remote Access Login Failed
```

Note: Multiple expressions of type "number" must be separated by a space. Also, in the case of a range, there must be two dots between the numbers.
You can also specify string values in a picklist such as Open, In Progress, Closed, etc. Multiple expressions of type "string" must be comma-separated.
In case you want to use another query to fetch additional data to create a comprehensive mapping, click the **+ Fetch additional sample data** link that appears in the Sample Data header. Clicking the **+ Fetch additional sample data** link opens the Configurations dialog in which you can change the configuration such as, updating the **Search_query**, and then click **Fetch Data**. The data ingestion wizard fetches the data based on the updated configuration on a new page in the `Sample Data` section. You can continue mapping based on the newly fetched data.
Once you are satisfied with the mappings, click **Save Mapping & Continue**.

8. (Optional) If your connector is enabled for scheduling, such as IBM QRadar, then you will be shown the **Scheduling** screen, using which you can specify the schedule for data ingestion from the connector into FortiSOAR, i.e., you can specify the frequency of polling a third-party integration, such as SIEM, so that the content gets pulled from the third-party integration into FortiSOAR. By default, scheduling is set to pull data every 5 minutes.
To configure scheduling, from the **Do you want to schedule the ingestion?** drop-down list, select **Yes**.
In the `Configure Schedule Settings` section, specify the Cron expression for the schedule. For example, if you want to pull data from IBM QRadar every 15 minutes, click **Every X Minute** and in the minute box enter `*/15`.



Data Ingestion - IBM QRadar

Scheduling Data Ingestion

You are almost there! Set up the ingestion mode and frequency. Next, review your ingestion settings and your data will begin to ingest into FortiSOAR™.

Schedule Settings

Do you want to schedule the ingestion?

Yes

Configure Schedule Settings

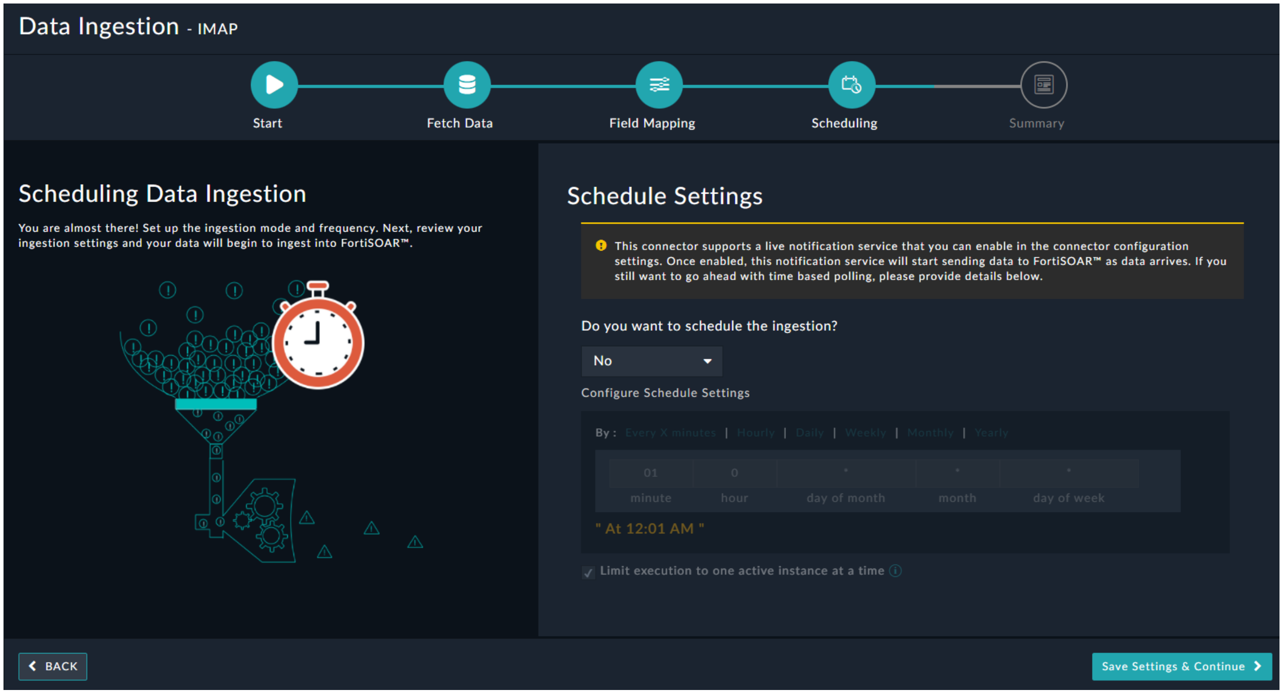
By : Every X minutes | Hourly | Daily | Weekly | Monthly | Yearly

minute	hour	day of month	month	day of week
*/15	*	*	*	*

" Every 15 minutes "

☑ Limit execution to one active instance at a time

Note: If you select the **Limit execution to one active instance at a time** checkbox, then do not rerun the workflow (schedule) if the previous schedule is still running. Some connectors such as IMAP support data ingestion using both a scheduled-based pull as well as instant notification using the Notification Service. You must, however, configure only one of the two, since both would otherwise pull the data from the same source and there could be data loss due to conflicts. Therefore, the `Scheduling` screen for the IMAP connector displays the following warning:



Data Ingestion - IMAP

Scheduling Data Ingestion

You are almost there! Set up the ingestion mode and frequency. Next, review your ingestion settings and your data will begin to ingest into FortiSOAR™.

Schedule Settings

This connector supports a live notification service that you can enable in the connector configuration settings. Once enabled, this notification service will start sending data to FortiSOAR™ as data arrives. If you still want to go ahead with time based polling, please provide details below.

Do you want to schedule the ingestion?

No

Configure Schedule Settings

By : Every X minutes | Hourly | Daily | Weekly | Monthly | Yearly

minute	hour	day of month	month	day of week
01	0	*	*	*

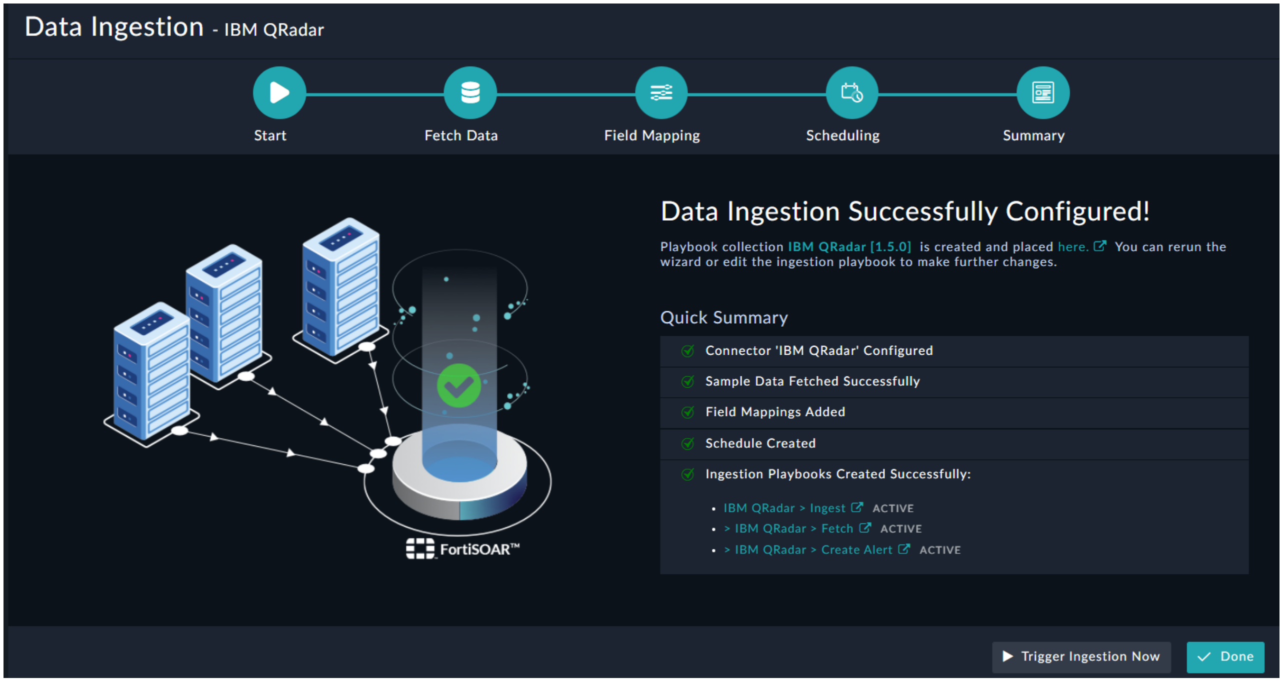
" At 12:01 AM "

☑ Limit execution to one active instance at a time

Only if you want to ingest data using a scheduled-based pull, then you should use the `Scheduling` screen to specify the schedule for data ingestion from the IMAP connector into FortiSOAR.

Once you are satisfied with the scheduling, click **Save Settings & Continue**.

9. The **Summary** screen displays a summary of the mapping done, and it also contains links to the Ingestion playbooks.



Data Ingestion Successfully Configured!

Playbook collection IBM QRadar [1.5.0] is created and placed here. You can rerun the wizard or edit the ingestion playbook to make further changes.

Quick Summary

- ✓ Connector 'IBM QRadar' Configured
- ✓ Sample Data Fetched Successfully
- ✓ Field Mappings Added
- ✓ Schedule Created
- ✓ Ingestion Playbooks Created Successfully:
  - IBM QRadar > Ingest ACTIVE
  - > IBM QRadar > Fetch ACTIVE
  - > IBM QRadar > Create Alert ACTIVE

Click **Trigger Ingestion Now** to immediately begin the process of data ingestion. Subsequent data ingestion occurs based on the data ingestion schedule you have set.
Click **Done** to complete the data ingestion and exit the Data Ingestion Wizard.

Notes for Data Ingestion Playbooks:

- The **Summary** step contains links to playbooks that contain the data mapping based on the mappings you have specified in the wizard. You can click the links and open the playbook in the Playbook Designer.
  For example, to open the playbook that fetches data from the Anomali ThreatStream connector, click **Anomali ThreatStream > Ingest** link, which opens the `Anomali ThreatStream > Ingest` playbook in the Playbook Designer.
- You can open the Data Ingestion Playbooks collection by clicking **here** in the Summary step.
- You can modify these data ingestion playbooks or the mappings in the playbooks using the Playbook Designer and those modifications are reflected in the mappings, i.e., if you open the "Data Ingestion Wizard" again you will see the modified content.

Notes for Data Ingestion

- If you are using the IMAP connector for data ingestion, then you must ensure that you configure either a scheduled-based pull, using Scheduling in the Data Ingestion Wizard, or the instant notification method using the Email Notification Service, by clicking the Enable Email Notification Service in the IMAP connector configuration pane. If you configure both then both the methods would pull the data from the same source and there could be data loss due to conflicts.
- If you are using the IMAP connector for data ingestion and at the field mapping stage you change the default module for data ingestion, which is set as **Alerts** to any other module, for example, **Emails**, then the playbook used for ingesting data remain active and no records will be created. To get the data ingestion to work correctly, do the following:
  Open the `> IMAP > Create Record` playbook and click the **Upsert File Indicators** step. In the Upsert File Indicators step, you need to initialize the IRI value of the module that you want to use for data ingestion instead of

the alert record IRI. To achieve this, click the **emailRecordIRI** field, and use "Dynamic Values" to enter the IRI value of the required module or enter the following Jinja value: `{{vars.steps.Create_Record['@id']}}`.
For example, if you have changed the module used for data ingestion from Alerts to Emails, then to initialize the `emailRecordIRI` variable instead of `alertRecordIRI`, in the **emailRecordIRI** field, enter the `{{vars.steps.Create_Record['@id']}}` value.
Also, note that If you select any module other than the default module, you will require to remap all the fields.

- If you are using the Syslog connector for data ingestion, then you must ensure playbooks used for data ingestion provide input in the CEF format, otherwise, you require to modify the playbooks. Since CEF is the most common format for Syslog messages, the default data ingestion playbooks parse the message as CEF. If the Syslog messages from your server are non-CEF complaint, you can modify the "Parse CEF" step in the `> Syslog> Fetch` playbook that is part of the playbooks shipped with the Syslog connector, before you configure Data Ingestion. The Syslog connector also provides "Parse Message" action to parse RFC 3164 and RFC 5424 formatted messages and convert these message formats to CEF format, instead of using the "Parse CEF" step.
- From version 7.0.2 onwards the ingestion wizard has been enhanced to not fail if there are no `#fetch` or `#create` playbooks. There can be connectors that require only the `#ingest` playbook that contains a single action, which pulls data and creates records. Such connectors do not require the `#fetch` playbook since data is not required for custom module mapping and the `#create` playbook is not required since the 'Create Record' step is not needed. Therefore, in this case, the Data Ingestion Wizard works as follows:
  - The Data Ingestion Wizard is loaded even if only the 'Ingest' playbook is present; and in this case, the wizard displays only the `Scheduling` screen.
  - If the 'Fetch' playbook is not present, the wizard does not display the `Fetch Data` screen.
  - If the 'Create' playbook is not present or the 'Create Record' step is not present, the wizard does not display the `Field Mapping` screen.

Troubleshooting

The Fetch Data screen in case of IMAP displays the`ERROR :: create failed: [ALREADYEXISTS] Folder name ... error

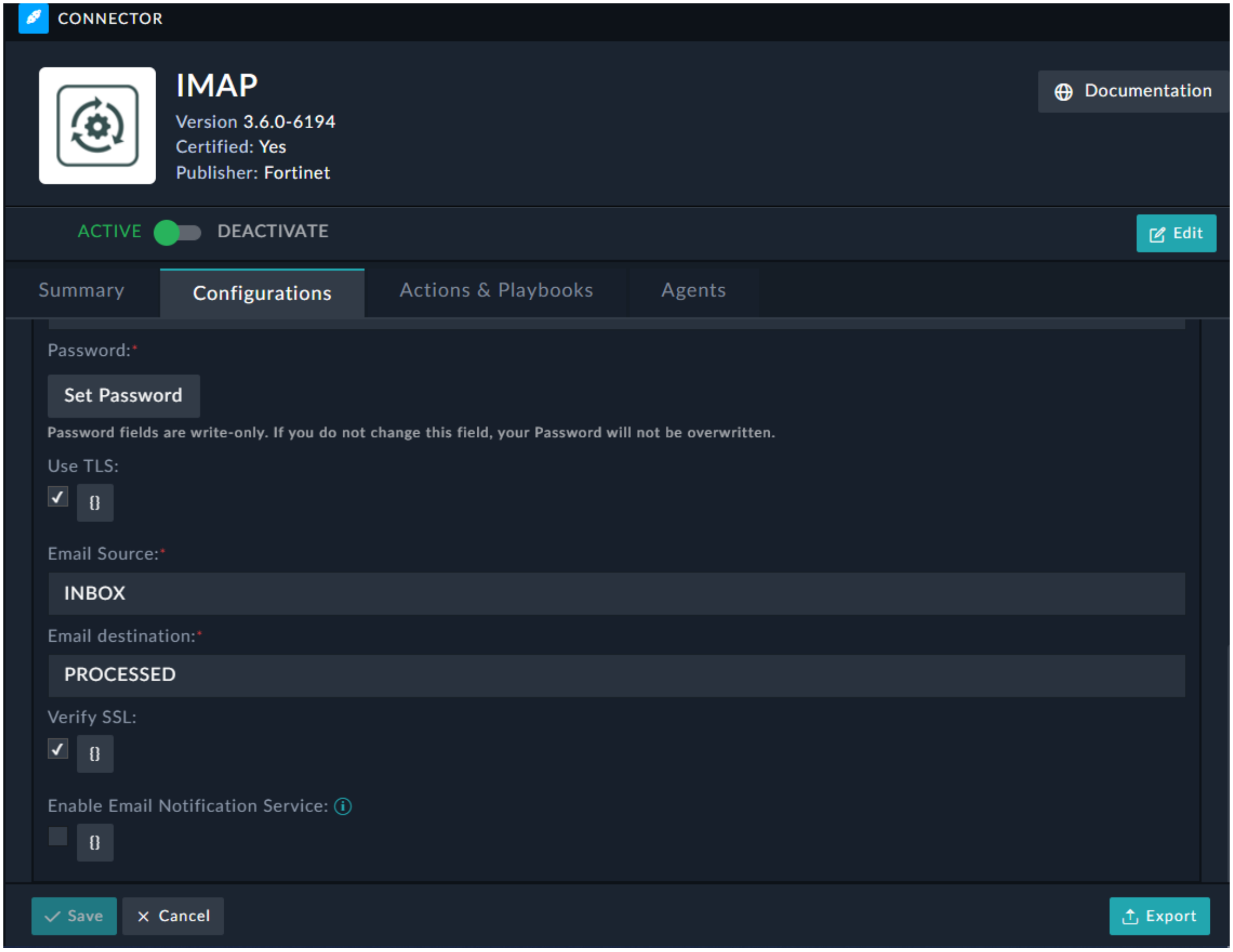
This error occurs when you are running the Data Ingestion Wizard using the IMAP connector. Click **Fetch Data** on the `Fetch Data` screen and an error such as "`> IMAP > Fetch has failed because CS_INTEGRATION-5: Error occurred while executing the connector action ERROR:: create failed [ALREADYEXISTS] Folder name conflicts with existing folder name........`" is displayed.

The above error can occur due to the following reasons:

- This issue occurs if the **Email destination** folder specified on the server is similar to an existing folder. Mailbox names are case sensitive on the creation of folders using the IMAP client, for example, if you specify `MyFolder` in the **Email destination** field and a folder named `myfolder` already exists on the server, then this error will be displayed on the Fetch Data screen when you are running the Data Ingestion Wizard.
- This error also occurs if the folder that you have specified does not have permission for the IMAP client.

Resolution

- In the IMAP configuration pane, update the folder name in the **Email destination** field and save the configuration:



CONNECTOR

IMAP
Version 3.6.0-6194
Certified: Yes
Publisher: Fortinet

Documentation

ACTIVE DEACTIVATE

Edit

Summary | Configurations | Actions & Playbooks | Agents

Password:*

Set Password

Password fields are write-only. If you do not change this field, your Password will not be overwritten.

Use TLS:
☑ {}

Email Source:*
INBOX

Email destination:*
PROCESSED

Verify SSL:
☑ {}

Enable Email Notification Service: ⓘ
□ {}

✓ Save ✕ Cancel Export

- Ensure that you provide appropriate permissions to the email destination folder for the IMAP client.

Post upgrade to FortiSOAR 6.0.0 or later from version 5.1.0 does not save any data ingestion configurations

If you have upgraded your FortiSOAR version from 5.1.0 to 6.0.0 or later, you will see data ingestion configurations that you had saved in version 5.1.0 are not available in version 6.0.0. You will not see any schedules, ingestion playbook link, or actions button on the `Data Ingestion` page. However, your data ingestion will continue to run. This is because support for fetching data for each configuration of your connector has been introduced in version 6.0.0 and therefore, the older data ingestion configurations and playbooks have been deprecated.

Resolution

Rerun the data ingestion wizard, i.e., reconfigure your data ingestion if required. Note that your previously configured data ingestion will continue to run, however, you will not be able to see the historical data. If you do require to reconfigure the data ingestion, then you must ensure that you stop your previous configured data ingestion, otherwise both the current and the old data ingestion will continue to run.

FORTINET®

www.fortinet.com