How-tos for EclecticIQ Platform

Hands-on articles on specific platform features

Last generated: March 08, 2017

Last generated on Mar 8, 2017

Table of contents

How-tos and tutorials — EclecticIQ Platform

Summary: This section is dedicated to how-tos and tutorials about EclecticIQ Platform. Hands-on, example-driven documentation that addresses specific features and user scenarios in a pragmatic way.

Browse the table for the topics you want to look up.
You can also use the drop-down menu on the left-hand navigation sidebar to access the articles or to go to a different section.

Title	Excerpt
How to check system health	System health gives you a clear basic overview of the overall platform health, as well as the operational status of its components.
How to configure a different database in OpenTAXII	By default, OpenTAXII uses SQLite as a database. You can change this setting to configure a different database, for example PostgreSQL.
How to configure incoming feeds	This summary page gives you an overview of the available how-to and tutorial articles about incoming feeds. They describe how to configure content types, transport types, and all the required optio...
How to configure Anubis Cyberfeed incoming feeds	Set up and configure AnubisNetworks Infections Detection Cyberfeed incoming feeds.
How to configure Group-IB accounts incoming feeds	Set up and configure Group-IB accounts incoming feeds.
How to configure Group-IB cards incoming feeds	Set up and configure Group-IB cards incoming feeds.
How to configure Group-IB IMEIs incoming feeds	Set up and configure Group-IB IMEIs incoming feeds.
How to configure Intel 471 incoming feeds	Set up and configure Intel 471 incoming feeds.
How to configure EclecticIQ JSON incoming feeds	Set up and configure EclecticIQ JSON incoming feeds.
How to configure PDF incoming feeds	Set up and configure PDF incoming feeds.

Title	Excerpt
How to configure STIX incoming feeds	Set up and configure STIX 1.0, 1.1, 1.1.1 and 1.2 incoming feeds.
How to configure text incoming feeds	Set up and configure plain text incoming feeds.
How to configure ThreatGRID incoming feeds	Set up and configure ThreatGRID incoming feeds.
How to configure Threat Recon incoming feeds	Set up and configure Threat Recon incoming feeds.
How to configure outgoing feeds	This summary page gives you an overview of the available how-to and tutorial articles about outgoing feeds. They describe how to configure content types, transport types, and all the required optio...
How to configure ArcSight CEF outgoing feeds	Set up and configure ArcSight CEF outgoing feeds.
How to configure EclecticIQ CSV outgoing feeds	Set up and configure EclecticIQ CSV outgoing feeds.
How to configure EclecticIQ JSON outgoing feeds	Set up and configure EclecticIQ JSON outgoing feeds.
How to configure Snort alerts outgoing feeds	Set up and configure Snort alerts outgoing feeds.
How to configure Snort logs outgoing feeds	Set up and configure Snort logs outgoing feeds.
How to configure STIX 1.2 outgoing feeds	Set up and configure Snort logs outgoing feeds.
How to create a money mule TTP	Create a money mule TTP entity to investigate fraudulent activities and to identify the actors involved in them.
How to enable audit logging in Kibana	Enable audit logging to examine system events and user access to understand what happened and when, where in the platform, the results it produced, and who/what triggered it.

Title	Excerpt
How to enrich entities with extracts	Enrichment extracts augment the quality of the intelligence you obtain from cyber data analysis. Enrich entities and integrate entity extracts with additional raw data to access a broader context a...
How to install the platform via an RPM package	This step-by-step tutorial walks you through a fresh installation of the platform onto a virtual server via an RPM package.
How to make API calls with a script	Make calls to the EclecticIQ API using our simple 'papi' script.
How to merge entities	Merge almost identical entities into a master entity and rewire relationships to reduce data noise.
How to monitor the platform	As a system administrator, you can use tools like Celery and Supervisor to monitor platform tasks to check day-to-day operations and to investigate, in case an issue occurs.
How to organize tags with taxonomies	The Taxonomy page displays an overview of the tags used to label entities in the platform. Besides using tags to organize entities, you can design taxonomies to structure the tags, and to create a ...
How to reindex Elasticsearch	You may need to reindex Elasticsearch for several reasons: from changes to data types or data analysis, to updating the Elasticsearch schema by adding or removing fields. Whenever a change in the d...
How to reindex the graph database	You may need to reindex the graph database for several reasons: from changes to data types or data analysis, to updating the data schema by adding or removing fields. Whenever a change in the data ...
How to report sightings through the API	Create and update sighting entities programmatically by making calls to the EclecticIQ API.
How to retrieve outgoing feeds through the API	Fetch outgoing feeds either manually through the platform GUI or programmatically via the API.
How to search logs for issues in Kibana	Search Kibana to retrieve log data about errors, warnings, or issues concerning specific platform components.
How to setup Nginx client certificate verification	Set up and configure SSL client certificate verification in Nginx.
How to shut down the platform	Graciously shut down the platform by first stopping its core services and processes.
How to split MISP STIX packages	Split MISP STIX packages into their corresponding embedded STIX packages by using the splitter command line utility.

Title	Excerpt
How to upgrade Elasticsearch and Kibana	Elasticsearch and Kibana are two third-party products EclecticIQ Platform uses. Elasticsearch was upgraded from version 1.7.1 to version 2.3.3, and Kibana was upgraded from version 4.1.1 to version...
How to upgrade to Neo4j 2.3.1	Neo4j is one of the third-party products EclecticIQ Platform uses. It was upgraded from version 2.2.4 to version 2.3.1. Follow these steps to migrate to the new version of the graph database.
How to address logging issues in Kibana	Inspect Kibana and Logstash configurations to identify and troubleshoot logging issues.
How to work with the Elasticsearch sightings enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the Elasticsearch sightings enricher, view e...
How to work with enrichers	This summary page offers an overview of the available how-to and tutorial articles about configuring and working with enrichers. They describe how to set up enricher rules and tasks, as well as how...
How to work with exclusions	The exclusion list filters out unwanted and/or unnecessary data that would otherwise clutter the platform.
How to work with exposure	Exposure shows you what your organization is doing with the ingested cyber threat intelligence, so that you can evaluate its usage to define courses of actions and other preventive or reactive proc...
How to work with the Flashpoint AggregINT enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the Flashpoint AggregINT enricher, view enri...
How to work with the Flashpoint Blueprint enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the Flashpoint Blueprint enricher, view enri...
How to work with the Flashpoint Thresher enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the Flashpoint Thresher enricher, view enric...
How to work with the Fox-IT InTELL Portal enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the Fox-IT InTELL Portal enricher, view enri...
How to work with the Intel 471 enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the Intel 471 enricher, view enrichment extr...
How to work with the OpenResolve enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the OpenResolve enricher, view enrichment ex...

Title	Excerpt
How to work with the PassiveTotal enrichers	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run PassiveTotal whois, passive DNS, IP and doma...
How to work with the PyDat enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the PyDat enricher, view enrichment extracts...
How to work with relationships	The Neighborhood tab in the entity detail pane includes a small graph canvas showing close relationships of the entity to other entities, as well as related extracts, datasets, workspaces, and tasks.
How to work with the RIPEstat GeoIP enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the RIPEstat GeoIP enricher, view enrichment...
How to work with the RIPEstat Whois enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the RIPEstat Whois enricher, view enrichment...
How to work with the ThreatGRID enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the ThreatGRID enricher, view enrichment ext...
How to work with the VirusTotal enricher	Raw data enrichment extracts improve the quality of the intelligence you obtain from external sources and use for cyber data analysis. Configure and run the VirusTotal enricher, view enrichment ext...

Feedback

No one reads manuals, ever. We know.
Yet, we strive to give you clear, concise, and complete documentation that helps you get stuff done neatly.

We are committed to crafting good documentation, because life is too short for bad doc.
We appreciate your comments, and we'd love to hear from you: if you have questions or suggestions, drop us a line and share your thoughts with us!

The Product Team

How to split MISP STIX packages

Summary: Split MISP STIX packages into their corresponding embedded STIX packages by using the splitter command line utility.

Issue

MISP XML files usually include multiple STIX XML packages. Each embedded STIX package holds data defining an entity object. The parent MISP XML file serves as a container for the embedded STIX packages. When a MISP XML file holds a large number of STIX packages, it may cause ingestion errors.

To address this issue and to correctly ingest all the valid STIX content, you can split the source MISP XML package into its constituent embedded STIX packages. This process removes the MISP XML container layer, and it outputs one XML file per STIX XML sub-package. The resulting STIX XML packages can then be ingested and processed by the platform.

Solution

The EclecticIQ Platform ships with a command line utility that splits the embedded STIX packages into separate XML files: `split-misp-stix-packages`.

The `split-misp-stix-packages` script is in the `.../platform-api/scripts` directory.

To run the script correctly, follow these recommendations:

- You need to run `split-misp-stix-packages` from within the CentOS virtual environment the platform runs on. Currently supported: **CentOS Linux 7 (1511)** `(https://lists.centos.org/pipermail/centos-announce/2015-december/021518.html)`.
- Explicitly point to the Python interpreter inside the virtual environment.
  Example:

```
# Python interpreter included with the
# virtual environment the platform runs on.
/opt/.../platform-api/venv/bin/python

# Directory in the platform virtual environment where the
# MISP XML splitter is located.
/opt/.../platform-api/scripts/split-misp-stix-packages
```

Usage

To view the built-in help, run the following command(s):

```
# Enter this command:
$ split-misp-stix-packages --help

# The help is displayed:
Usage: split-misp-stix-packages [OPTIONS] FILE.xml

Options:
  --output-directory DIRECTORY  Output directory  [required]
  --output-base-name TEXT       File name template to be used for output files
  --debug
  --help                        Show this message and exit.
```

Option	Description
`--output-directory`	*Required* — Defines the file splitter output directory the embedded STIX packages are saved to after extracting them from the MISP XML wrapper.
`--output-base-name`	*Optional* — By default, STIX packages are named `package-<int>.xml`, where `<int>` is a sequentially progressive numeric value starting at zero. If you want, you can specify a different name than `package`. It is not possible to modify the hyphen or the numeric part of the file name.
`--debug`	*Optional* — In case of errors, you can use this option to return a verbose output.
`--help`	*Optional* — Displays the built-in help.

Example

In this example, we are using the following dummy names for files and directories:

- `<platform_virtual_environment_username>`: user name to access the CentOS virtual environment the platform runs on. Currently supported: **CentOS Linux 7 (1511)** `(https://lists.centos.org/pipermail/centos-announce/2015-december/021518.html)`.
- `<platform_virtual_environment_password>`: password to access the CentOS virtual environment the platform runs on. Currently supported: **CentOS Linux 7 (1511)** `(https://lists.centos.org/pipermail/centos-announce/2015-december/021518.html)`.
- `misp-out misp-stix-package.xml`: example MISP XML file to extract the embedded STIX packages from.

- `temp`: directory where the source `misp-out misp-stix-package.xml` file is temporarily stored.
- `misp-out`: output directory where the embedded STIX packages, originally in the MISP XML file, are saved as separate XML files.

These are the main steps:

- To perform several tasks during the procedure, you need root-level permissions.
  To obtain administration rights, run the following command(s):

```
$ sudo su -
```

- Create an output directory where the STIX packages can be saved as separate XML files.
- Go to the directory where the MISP XML file you want to split is located.
- Run the file splitter utility.

```
# SSH authentication in the EclecticIQ Platform virtual environment.
$ sudo su <platform_virtual_environment_username>
password: <platform_virtual_environment_password>

# Create a new directory; the MISP XML sub-packages will be saved here.
$ mkdir misp-out

# Go to the directory where you saved to source MISP STIX XML package.
# Example: "temp".
$ cd temp

# Run the split utility tool. Specify:
# - The output directory for the split sub-packages.
# - The source MISP STIX XML package you want to split.
$ split-misp-stix-packages --output-directory misp-out misp-stix-package.xml

# Log message at the end of a successful operation,
# where "%d" is an integer.
%d output files written
```

How to configure outgoing feeds

Summary: This summary page gives you an overview of the available how-to and tutorial articles about outgoing feeds. They describe how to configure content types, transport types, and all the required options you need to set when you create outgoing feeds to distribute and share acquired cyber threat intelligence through EclecticIQ Platform.

Browse the table for the topics you want to look up.
You can also use the drop-down menu on the left-hand navigation sidebar to access the articles or to go to a different section.

Title	Excerpt
How to configure ArcSight CEF outgoing feeds	Set up and configure ArcSight CEF outgoing feeds.
How to configure EclecticIQ CSV outgoing feeds	Set up and configure EclecticIQ CSV outgoing feeds.
How to configure EclecticIQ JSON outgoing feeds	Set up and configure EclecticIQ JSON outgoing feeds.
How to configure Snort alerts outgoing feeds	Set up and configure Snort alerts outgoing feeds.
How to configure Snort logs outgoing feeds	Set up and configure Snort logs outgoing feeds.
How to configure STIX 1.2 outgoing feeds	Set up and configure Snort logs outgoing feeds.
How to retrieve outgoing feeds through the API	Fetch outgoing feeds either manually through the platform GUI or programmatically via the API.

How to configure ArcSight CEF outgoing feeds

Summary: Set up and configure ArcSight CEF outgoing feeds.

In the EclecticIQ Platform you can configure outgoing feeds to share and distribute cyber threat intelligence in several formats. Share knowledge and promote collaboration to support an ecosystem where partners work together to identify threats, and define an effective course of action to ensure their assets are protected.

This article describes how to configure **ArcSight CEF** outgoing feeds, so that you can distribute selected intelligence through the EclecticIQ Platform.

Configure the general options

✔ On the forms, input fields marked with an asterisk are required.

- On the top navigation bar, click the ✚ *plus* button.
- On the **Create new** sidebar, click **Data management > Outgoing feed**.

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button.
- Under **Configuration** on the drop-down menu, click **Data management**, and then **Outgoing feeds**.

The **Outgoing feeds** page displays an overview of the configured outgoing feeds to publish and distribute selected intel from the platform to external parties, services, and systems.

On the top-right corner of the screen, click the **+ Outgoing feed** button. - The **✚ > Data management > Outgoing feeds > Create** form page includes several input fields to help you define *what* you want to share and *how*, that is, the data content type and the data transport type.

- Under **Feed name**, enter a name for the feed you are creating. It should be descriptive and easy to remember.
- **Transport type**: from the drop-down menu select the appropriate transport type to publish the data through the outgoing feed. This can vary, based on the carrier used to distribute the data.
- Depending on the selected transport type, you may need to specify additional settings under **Transport configuration**.
  For example:
  - A URL endpoint corresponding to the API service exposing the data source for the incoming feed.
  - A valid API key to grant you access to the feed data source.
  - Any required login credentials to obtain access to the feed data source.

- **Content type**: from the drop-down menu select **ArcSight CEF** and configure the appropriate parameters under **Content configuration**, when applicable.
- **Dataset**: from the drop-down menu select one or more datasets as data sources for the outgoing feed.
- **Update strategy**: from the drop-down menu select the preferred method to update the data:
  - **Append**: every time the outgoing feed task runs, new data is appended to the existing data.
    When new intelligence is made available through the outgoing feed, it is added after/below the existing data from the same feed.
  - **Replace** every time the outgoing feed task runs, existing data is deleted and it is replaced with new data.
    When new intelligence is made available through the outgoing feed, it overwrites and replaces existing data from the same feed.

Set a schedule

- Under **Execution schedule** you can define how often you want to run the outgoing feed task:
- **None**: no schedule is defined. You need to manually trigger the task to publish data through the outgoing feed.
- **Minute**: the outgoing feed task run automatically.
  You define the execution interval in 5-minute increments from the corresponding drop-down menu.
- **Hour**: the outgoing feed task task run automatically every hour.
  You define how long after the beginning of an hour the task should run from the corresponding drop-down menu.
- **Day**: the outgoing feed task task run automatically once a day.
  You define the time of the day when the task should run from the corresponding drop-down menu.
- **Week**: the outgoing feed task task run automatically once a week.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menu.
- **Month**: the outgoing feed task task run automatically once a month.
  You define the day of the calendar month and time of the day when the task should run from the corresponding drop-down menu.
  Keep in mind that not all months of the year have 31 days.
- Select the **Active** checkbox to make the feed available immediately after creating it.

Set a TLP override

- The **Override TLP** options overwrite the **TLP** `(https://www.us-cert.gov/tlp)` color code associated with the outgoing feed entities with the one you set here. The selected TLP value is assigned to all the entities in the outgoing feed.

You can override the original or the current TLP color code of an entity, an incoming feed, or an outgoing feed. When working as a filter, TLP colors select a decreasing range: if you set a TLP color as a filter the enricher, the feed, or the returned filtered results include all the entities flagged with the selected TLP color code, as well as all the entities whose TLP color indicates that they are progressively lower risk, less sensitive, and suitable for disclosure to broader audiences.
For example, if you select green the filtered results include entities with a TLP color set to green, as well as entities with a TLP color set to white, and entities with no TLP color code flag.

- The **Filter TLP color** radio buttons allow including in the outgoing feed data only an entity subset, based on the selected **TLP** `(https://www.us-cert.gov/tlp)` value. If you set a TLP color as a filter, the feed includes all the entities flagged with the selected TLP color code, as well as the entities whose TLP color indicates that they are suitable for progressively broader audiences. For example, if you select green, the feed includes entities with a TLP color set to green and entities with a TLP color set to white.

Set reliability and relevancy

- **Source reliability**: from the drop-down menu select an option to flag the level of reliability of the source. It helps analysts assess how much confidence they can reasonably have in an intel source.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code)`.
  Example: *B - Usually reliable*
- **Relevancy threshold (%)** allows you to set a filter to include in the outgoing feed data only the entities whose relevancy value is higher than the one defined here.

Set extract filters

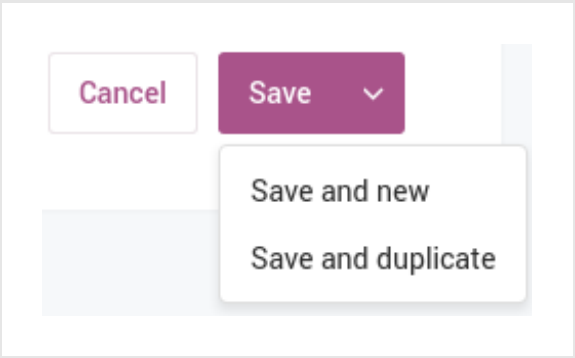
- **Allowed extract states**: from the drop-down menu select one or more extract states to include in the outgoing feed data only the entities whose extract states match the selections defined here.
- **Extract types**: from the drop-down menu select one or more extract types to include in the outgoing feed data only the entities whose extracts types match the selections defined here.
- **Enrichment extract types**: from the drop-down menu select one or more enrichment extract types to include in the outgoing feed data only the entities whose enrichment extracts types match the selections defined here.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating right away a new item of the same type; for example, a dataset, a feed, a rule, or a task.

- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.



Configure the content type

When you select the **ArcSight CEF** content type for an outgoing feed, you need to configure the following content type parameters:

- **Extract types**: from the drop-down menu select one or more extract types to include in the outgoing feed data only the entities whose extracts match the selected types.
- **Enrichment extract types**: from the drop-down menu select one or more enrichment extract types to include in the outgoing feed data only the entities whose enrichment extracts match the selected types.

##

Content type	Allowed transport types
ArcSight CEF	FTP upload
	HTTP download
	Mount point upload
	Send email
	Syslog push
	TAXII inbox
	TAXII poll

FTP upload

If you want to make the outgoing feed data available through FTP, from the **Transport type** drop-down list select **FTP upload**.
Under **Transport configuration**, configure the following settings:

- **FTP server URL**: the target `ftp://` location to upload the outgoing feed content to, so as to make it available for download.
- **Username**: a valid user name to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.
- **Password**: a valid password to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.

HTTP download

> **Warning:** The HTTP upload/download transport type requires basic access authentication.

If you want to make the outgoing feed data available through an HTTP URL, from the **Transport type** drop-down list select **HTTP download**.
Under **Transport configuration**, configure the following settings:

- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).

Mount point upload

If the source of the outgoing feed is located on a local or network unit, from the **Transport type** drop-down list select the **Mount point upload**
option.

Under **Transport configuration**, configure the following settings:

- **Mount point path**: the path to the local or network unit where the source data for the outgoing feed is stored.

Send email

Warning: Email needs to be correctly configured in the platform system settings for this transport option to work.

If you want to make the outgoing feed data available by email, from the **Transport type** drop-down list select **Send email**.
Under **Transport configuration**, configure the following settings:

- **Mail subject**: enter a short, descriptive subject for the outgoing email notifications.
- **Platform groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list. All the members of the selected group(s) will receive email notifications with the outgoing feed data.
- **Platform users**: if you want to further limit the outgoing feed email recipient targets, from the drop-down list you can select one or more users. In this case, only the selected users belonging to the designated groups will receive email notifications with the outgoing feed data.

Syslog push

If you want to make the outgoing feed data available through a syslog push service, from the **Transport type** drop-down list select **Syslog push**.
Under **Transport configuration**, configure the following settings:

- **Syslog server host**: specify the IP address or the host name of the server handling syslog message log communications.
- **Syslog server port**: specify the port number of the server handling syslog message log communications. Make sure the port is open, and that data traffic through the port is not blocked by, for example, a firewall.

  Typical port settings for the TCP protocol:

  - *601* for syslog-conn
  - *6514* for syslog over TCP with TLS

  Typical port settings for the UDP protocol:

  - *514* for syslog
- **Protocol**: from the drop-down menu select the transmission protocol, either **TCP** or **UDP**.

TAXII inbox

Warning: Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.

The **TAXII inbox** transport type requires Cabby. For further details, see the **official Cabby documentation** (https://cabby.readthedocs.org/en/latest/), the **Cabby public repo on GitHub** (https://github.com/eclecticiq/cabby), and the **Cabby download page** (https://pypi.python.org/pypi/cabby/).

If you want to make the outgoing feed data available through a TAXII server and push email notifications to TAXII clients, from the **Transport type** drop-down list select **TAXII inbox**.
Under **Transport configuration**, configure the following settings:

- **Inbox service URL**: specify a valid URL address to determine the service location where the available **TAXII data collections** (https://taxiiproject.github.io/documentation/sample-use/#data-collections) are stored.
  Example:
  `https://example.com/taxii-inbox`
- **Destination collection name**: specify a valid collection as the source for the outgoing feed data.
  Example:
  `collection.Default`
- **Taxii version**: select the TAXII version your system supports:
  - Either **1.0** (https://taxiiproject.github.io/releases/1.0/)
  - Or **1.1** (https://taxiiproject.github.io/releases/1.1/)
- **EclecticIQ authentication URL**: the URL exposing the platform authentication and authorization service. The platform autorization endpoint is `/auth`.
  Example:
  `https://<platform.host>/auth`
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **SSL certificate**: paste here a valid SSL certificate, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines.
- **SSL key**: paste here a valid SSL private key, including the `-----BEGIN RSA PRIVATE KEY-----` and `-----END RSA PRIVATE KEY-----` lines.
- **SSL key password**: enter here the password to unlock the SSL key.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

TAXII poll

Warning: Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.

The **TAXII poll** transport type requires Cabby. For further details, see the **official Cabby documentation** (https://cabby.readthedocs.org/en/latest/), the **Cabby public repo on GitHub** (https://github.com/eclecticiq/cabby), and the **Cabby download page** (https://pypi.python.org/pypi/cabby/).

If you want to make the outgoing feed data available through polling — where a TAXII client polls the TAXII server to request information and data updates — from the **Transport type** drop-down list select **TAXII poll**.

- Make sure that at least one dataset is selected under **Dataset** to allow TAXII clients to request information and updates about the specified **TAXII data collection(s)** (https://taxiiproject.github.io/documentation/sample-use/#data-collections).
- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

How to configure EclecticIQ CSV outgoing feeds

Summary: Set up and configure EclecticIQ CSV outgoing feeds.

In the EclecticIQ Platform you can configure outgoing feeds to share and distribute cyber threat intelligence in several formats. Share knowledge and promote collaboration to support an ecosystem where partners work together to identify threats, and define an effective course of action to ensure their assets are protected.

This article describes how to configure **EclecticIQ CSV** outgoing feeds, so that you can distribute selected intelligence through the EclecticIQ Platform.

Configure the general options

✔ On the forms, input fields marked with an asterisk are required.

- On the top navigation bar, click the ✚ *plus* button.
- On the **Create new** sidebar, click **Data management > Outgoing feed**.

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button.
- Under **Configuration** on the drop-down menu, click **Data management**, and then **Outgoing feeds**.

The **Outgoing feeds** page displays an overview of the configured outgoing feeds to publish and distribute selected intel from the platform to external parties, services, and systems.

On the top-right corner of the screen, click the **+ Outgoing feed** button. - The **✚ > Data management > Outgoing feeds > Create** form page includes several input fields to help you define *what* you want to share and *how*, that is, the data content type and the data transport type.

- Under **Feed name**, enter a name for the feed you are creating. It should be descriptive and easy to remember.
- **Transport type**: from the drop-down menu select the appropriate transport type to publish the data through the outgoing feed. This can vary, based on the carrier used to distribute the data.
- Depending on the selected transport type, you may need to specify additional settings under **Transport configuration**.
  For example:
  - A URL endpoint corresponding to the API service exposing the data source for the incoming feed.
  - A valid API key to grant you access to the feed data source.
  - Any required login credentials to obtain access to the feed data source.

- **Content type**: from the drop-down menu select **EclecticIQ Entities CSV** or **EclecticIQ Extracts CSV** and configure the appropriate parameters under **Content configuration**, when applicable.
- **Dataset**: from the drop-down menu select one or more datasets as data sources for the outgoing feed.
- **Update strategy**: from the drop-down menu select the preferred method to update the data:
  - **Append**: every time the outgoing feed task runs, new data is appended to the existing data.
    When new intelligence is made available through the outgoing feed, it is added after/below the existing data from the same feed.
  - **Replace** every time the outgoing feed task runs, existing data is deleted and it is replaced with new data.
    When new intelligence is made available through the outgoing feed, it overwrites and replaces existing data from the same feed.
  - **Diff**: every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at extract-level, i.e. any extracts that have been added to or removed from entities in the set, or at entity-level, i.e. any entities that have been added to or removed from the set. Depending on the selected CSV content option, each row in the CSV output contains information about one entity or one extract.
    An extra diff column is added to the output to indicate if a row, and therefore either an extract or an entity, has been added to or removed from the set.
    This option allows you to identify any changes in a feed between two task runs without downloading the whole feed every time.

Set a schedule

- Under **Execution schedule** you can define how often you want to run the outgoing feed task:
- **None**: no schedule is defined. You need to manually trigger the task to publish data through the outgoing feed.
- **Minute**: the outgoing feed task run automatically.
  You define the execution interval in 5-minute increments from the corresponding drop-down menu.
- **Hour**: the outgoing feed task task run automatically every hour.
  You define how long after the beginning of an hour the task should run from the corresponding drop-down menu.
- **Day**: the outgoing feed task task run automatically once a day.
  You define the time of the day when the task should run from the corresponding drop-down menu.
- **Week**: the outgoing feed task task run automatically once a week.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menu.
- **Month**: the outgoing feed task task run automatically once a month.
  You define the day of the calendar month and time of the day when the task should run from the corresponding drop-down menu.
  Keep in mind that not all months of the year have 31 days.
- Select the **Active** checkbox to make the feed available immediately after creating it.

Set a TLP override

- The **Override TLP** options overwrite the **TLP** (https://www.us-cert.gov/tlp) color code associated with the outgoing feed entities with the one you set here. The selected TLP value is assigned to all the entities in the outgoing feed.

You can override the original or the current TLP color code of an entity, an incoming feed, or an outgoing feed. When working as a filter, TLP colors select a decreasing range: if you set a TLP color as a filter the enricher, the feed, or the returned filtered results include all the entities flagged with the selected TLP color code, as well as all the entities whose TLP color indicates that they are progressively lower risk, less sensitive, and suitable for disclosure to broader audiences.
For example, if you select green the filtered results include entities with a TLP color set to green, as well as entities with a TLP color set to white, and entities with no TLP color code flag.

- The **Filter TLP color** radio buttons allow including in the outgoing feed data only an entity subset, based on the selected **TLP** (https://www.us-cert.gov/tlp) value. If you set a TLP color as a filter, the feed includes all the entities flagged with the selected TLP color code, as well as the entities whose TLP color indicates that they are suitable for progressively broader audiences. For example, if you select green, the feed includes entities with a TLP color set to green and entities with a TLP color set to white.

Set reliability and relevancy

- **Source reliability**: from the drop-down menu select an option to flag the level of reliability of the source. It helps analysts assess how much confidence they can reasonably have in an intel source.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** (https://en.wikipedia.org/wiki/admiralty_code).
  Example: *B - Usually reliable*
- **Relevancy threshold (%)** allows you to set a filter to include in the outgoing feed data only the entities whose relevancy value is higher than the one defined here.

Set extract filters

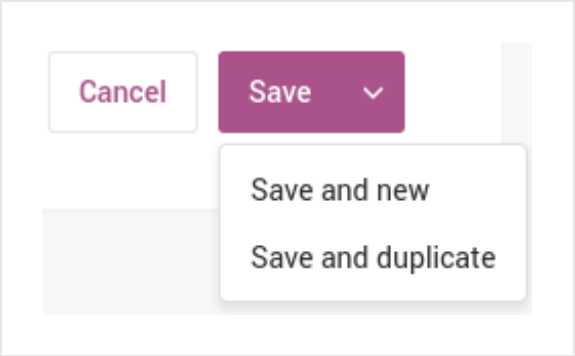
- **Allowed extract states**: from the drop-down menu select one or more extract states to include in the outgoing feed data only the entities whose extract states match the selections defined here.
- **Extract types**: from the drop-down menu select one or more extract types to include in the outgoing feed data only the entities whose extracts types match the selections defined here.
- **Enrichment extract types**: from the drop-down menu select one or more enrichment extract types to include in the outgoing feed data only the entities whose enrichment extracts types match the selections defined here.

- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating right away a new item of the same type; for example, a dataset, a feed, a rule, or a task.
- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.



Configure the content type

When you select the **EclecticIQ CSV** content type for an outgoing feed, you need to configure the following content type parameters.

From the drop-down menu select one of the following options to define the preferred structure for the output data and the resulting layout in the CSV output:

- **EclecticIQ Entities CSV**: in the resulting CSV with column headers, each row holds information referring to one entity.
  For example, an indicator, a TTP, and so on.
- **EclecticIQ Extracts CSV**: in the resulting CSV with column headers, each row holds information referring to one extract.
  For example, an IP address, a hash, a geographic location name, and so on.

Warning: If you select **EclecticIQ Extracts CSV**, you need to choose at least one extract type from the **Extract types** drop-down list, and at least one enrichment extract type from the **Enrichment extract types** drop-down list.

If you select **EclecticIQ Extracts CSV**, by default the outgoing feed includes only *first level*, *original* extracts:

- First level: the extracted data is inside a CybOX object.
- Original: the value is extracted as is, that is, the extract holds the actual value found in the CybOX object.

You can include also *second level*, *derived* extracts by selecting one or both checkboxes under **Content configuration**:

- **Include derived extracts**: the extracted data is the result of an analysis of the original value found inside a STIX field.
- **Include secondary extracts**: the source of the extracted data is a value inside a STIX field, not a value inside a CybOX object.

Derivation and levels

Derivation — **Original** vs **Derived** extracts — and levels — level **1** and level **2** extracts — work together to make it easier to identify and analyze the extracts you need to act on by including them in your prevention and/or detection toolchains.

You can filter extracts by derivation and levels to discard unwanted noise. You can zero in on specific bits of information to examine them, and to investigate any relationships they may have with other entities or extracts.

Level 1	The data origin is a value extracted from a field inside a *CybOX* object. Example: a URI in a CybOX `URIObj` field.
Level 2	The data origin is a value extracted from a field inside a *STIX* object. Example: a URI in a STIX `Reference` field.
Original	The original data is extracted and stored in the resulting extract as is. Example: a URI.
Derived	The original data is processed, and then a subset of that data is extracted and stored in the resulting extract. Example: a domain that is part of a URI.

Extracts are bits of information that contribute with additional context to an entity description. The platform can flag extracts to automate processes such as:

- Add potentially malicious threats to a prevention and/or a detection system;
- Exclude non-malicious extracts that do not represent a potential threat for the organization.

A set of rules can handle the flags and route extracts to a prevention and/or a detection system, or mark them as ignorable and filter them out to reduce unwanted data noise.

Original + level 1

Derivation	**Original**

Level	**1**

- **Original/1**: the extracted data is directly retrieved as is from a CybOX object embedded in a STIX indicator.
- **Original**: the value is extracted as is, that is, the extract holds the actual value found in the CybOX object. For example, a URI value extracted from:

```
<URIObj:Value condition="Equals">http://x4z9arb.cn/4712</URIObj:Value>
```

- **1**: the extracted data is inside a CybOX object.
  For example, a URI in a CybOX object embedded in a STIX indicator.

When the platform flags an extract as **Original/1**, it handles it as follows:

- It assigns the extract an initially *low maliciousness* level;
- It flags it as a *level 1* extracted data to indicate that it originates from a CybOX object. Therefore, it is directly related to its source, and it is probably relevant.;
- It marks it as a potential threat that needs to be added to a detection and/or prevention system.

Derived + level 2

Derivation	**Derived**
Level	**2**

- **Derived/2**: the source of the extracted data is a value inside a STIX field, not a value inside a CybOX object.
- **Derived**: the extracted data is the result of an analysis of the original value found inside a STIX field. For example, a domain name extracted from a URI:

```
<!-- The original extract value, in this example a URI -->
<stixCommon:Reference>https://technet.microsoft.com/library/security/2887505</stixComm
on:Reference>

<!-- The derived extract obtained from the URI, that is, a domain -->
technet.microsoft.com
```

- **2**: the source of the extracted data is a value inside a STIX field, not a value inside a CybOX object.
  For example, a URI in a STIX field like a header, a title, or a reference.

When the platform flags an extract as **Derived/2**, it handles it as follows:

- It does not assign the extract any maliciousness level;
- It flags it as a *level 2* extracted data to indicate that it does not originate from a CybOX object, but from a STIX field. Therefore, it is more distant from its source, and possibly not relevant.;
- It does not mark it for inclusion in a detection and/or prevention system.

Configure transport and content types

Content type	Allowed transport types
EclecticIQ CSV	FTP upload
	HTTP download
	Mount point upload
	Send email
	Syslog push
	TAXII inbox
	TAXII poll

FTP upload

If you want to make the outgoing feed data available through FTP, from the**Transport type** drop-down list select **FTP upload**.
Under **Transport configuration**, configure the following settings:

- **FTP server URL**: the target `ftp://` location to upload the outgoing feed content to, so as to make it available for download.
- **Username**: a valid user name to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.
- **Password**: a valid password to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.

HTTP download

> **Warning:** The HTTP upload/download transport type requires basic access authentication.

If you want to make the outgoing feed data available through an HTTP URL, from the**Transport type** drop-down list select **HTTP download**.
Under **Transport configuration**, configure the following settings:

- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).

Mount point upload

If the source of the outgoing feed is located on a local or network unit, from the **Transport type** drop-down list select the **Mount point upload**
option.

Under **Transport configuration**, configure the following settings:

- **Mount point path**: the path to the local or network unit where the source data for the outgoing feed is stored.

Send email

> **Warning:** Email needs to be correctly configured in the platform system settings for this transport option to work.

If you want to make the outgoing feed data available by email, from the**Transport type** drop-down list select **Send email**.
Under **Transport configuration**, configure the following settings:

- **Mail subject**: enter a short, descriptive subject for the outgoing email notifications.
- **Platform groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list. All the members of the selected group(s) will receive email notifications with the outgoing feed data.
- **Platform users**: if you want to further limit the outgoing feed email recipient targets, from the drop-down list you can select one or more users. In this case, only the selected users belonging to the designated groups will receive email notifications with the outgoing feed data.

TAXII inbox

Warning: Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.

The **TAXII inbox** transport type requires Cabby. For further details, see the **official Cabby documentation** (https://cabby.readthedocs.org/en/latest/), the **Cabby public repo on GitHub** (https://github.com/eclecticiq/cabby), and the **Cabby download page** (https://pypi.python.org/pypi/cabby/).

If you want to make the outgoing feed data available through a TAXII server and push email notifications to TAXII clients, from the **Transport type** drop-down list select **TAXII inbox**.
Under **Transport configuration**, configure the following settings:

- **Inbox service URL**: specify a valid URL address to determine the service location where the available **TAXII data collections** (https://taxiiproject.github.io/documentation/sample-use/#data-collections) are stored.
  Example:
  `https://example.com/taxii-inbox`
- **Destination collection name**: specify a valid collection as the source for the outgoing feed data.
  Example:
  `collection.Default`
- **Taxii version**: select the TAXII version your system supports:
  - Either **1.0** (https://taxiiproject.github.io/releases/1.0/)
  - Or **1.1** (https://taxiiproject.github.io/releases/1.1/)
- **EclecticIQ authentication URL**: the URL exposing the platform authentication and authorization service. The platform autorization endpoint is `/auth`.
  Example:
  `https://<platform.host>/auth`
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **SSL certificate**: paste here a valid SSL certificate, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines.
- **SSL key**: paste here a valid SSL private key, including the `-----BEGIN RSA PRIVATE KEY-----` and `-----END RSA PRIVATE KEY-----` lines.
- **SSL key password**: enter here the password to unlock the SSL key.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

TAXII poll

Warning: Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.

The **TAXII poll** transport type requires Cabby. For further details, see the **official Cabby documentation** (https://cabby.readthedocs.org/en/latest/), the **Cabby public repo on GitHub** (https://github.com/eclecticiq/cabby), and the **Cabby download page** (https://pypi.python.org/pypi/cabby/).

If you want to make the outgoing feed data available through polling — where a TAXII client polls the TAXII server to request information and data updates — from the **Transport type** drop-down list select **TAXII poll**.

- Make sure that at least one dataset is selected under **Dataset** to allow TAXII clients to request information and updates about the specified **TAXII data collection(s)** (https://taxiiproject.github.io/documentation/sample-use/#data-collections).
- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

How to configure EclecticIQ JSON outgoing feeds

Summary: Set up and configure EclecticIQ JSON outgoing feeds.

In the EclecticIQ Platform you can configure outgoing feeds to share and distribute cyber threat intelligence in several formats. Share knowledge and promote collaboration to support an ecosystem where partners work together to identify threats, and define an effective course of action to ensure their assets are protected.

This article describes how to configure **EclecticIQ JSON** outgoing feeds, so that you can distribute selected intelligence through the EclecticIQ Platform.

Configure the general options

✔ On the forms, input fields marked with an asterisk are required.

- On the top navigation bar, click the ✚ *plus* button.
- On the **Create new** sidebar, click **Data management > Outgoing feed**.

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button.
- Under **Configuration** on the drop-down menu, click **Data management**, and then **Outgoing feeds**.

The **Outgoing feeds** page displays an overview of the configured outgoing feeds to publish and distribute selected intel from the platform to external parties, services, and systems.

On the top-right corner of the screen, click the **+ Outgoing feed** button. - The **✚ > Data management > Outgoing feeds > Create** form page includes several input fields to help you define *what* you want to share and *how*, that is, the data content type and the data transport type.

- Under **Feed name**, enter a name for the feed you are creating. It should be descriptive and easy to remember.
- **Transport type**: from the drop-down menu select the appropriate transport type to publish the data through the outgoing feed. This can vary, based on the carrier used to distribute the data.
- Depending on the selected transport type, you may need to specify additional settings under **Transport configuration**.
  For example:
  - A URL endpoint corresponding to the API service exposing the data source for the incoming feed.
  - A valid API key to grant you access to the feed data source.
  - Any required login credentials to obtain access to the feed data source.

- **Content type**: from the drop-down menu select **EclecticIQ JSON** and configure the appropriate parameters under **Content configuration**, when applicable.
- **Dataset**: from the drop-down menu select one or more datasets as data sources for the outgoing feed.
- **Update strategy**: from the drop-down menu select the preferred method to update the data:
  - **Append**: every time the outgoing feed task runs, new data is appended to the existing data.
    When new intelligence is made available through the outgoing feed, it is added after/below the existing data from the same feed.
  - **Replace** every time the outgoing feed task runs, existing data is deleted and it is replaced with new data.
    When new intelligence is made available through the outgoing feed, it overwrites and replaces existing data from the same feed.
  - **Diff**: every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at extract-level, i.e. any extracts that have been added to or removed from entities in the set, or at entity-level, i.e. any entities that have been added to or removed from the set. Depending on the selected CSV content option, each row in the CSV output contains information about one entity or one extract.
    An extra diff column is added to the output to indicate if a row, and therefore either an extract or an entity, has been added to or removed from the set.
    This option allows you to identify any changes in a feed between two task runs without downloading the whole feed every time.

Set a schedule

- Under **Execution schedule** you can define how often you want to run the outgoing feed task:
- **None**: no schedule is defined. You need to manually trigger the task to publish data through the outgoing feed.
- **Minute**: the outgoing feed task run automatically.
  You define the execution interval in 5-minute increments from the corresponding drop-down menu.
- **Hour**: the outgoing feed task task run automatically every hour.
  You define how long after the beginning of an hour the task should run from the corresponding drop-down menu.
- **Day**: the outgoing feed task task run automatically once a day.
  You define the time of the day when the task should run from the corresponding drop-down menu.
- **Week**: the outgoing feed task task run automatically once a week.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menu.
- **Month**: the outgoing feed task task run automatically once a month.
  You define the day of the calendar month and time of the day when the task should run from the corresponding drop-down menu.
  Keep in mind that not all months of the year have 31 days.
- Select the **Active** checkbox to make the feed available immediately after creating it.

Set a TLP override

- The **Override TLP** options overwrite the **TLP** (https://www.us-cert.gov/tlp) color code associated with the outgoing feed entities with the one you set here. The selected TLP value is assigned to all the entities in the outgoing feed.

You can override the original or the current TLP color code of an entity, an incoming feed, or an outgoing feed. When working as a filter, TLP colors select a decreasing range: if you set a TLP color as a filter the enricher, the feed, or the returned filtered results include all the entities flagged with the selected TLP color code, as well as all the entities whose TLP color indicates that they are progressively lower risk, less sensitive, and suitable for disclosure to broader audiences.
For example, if you select green the filtered results include entities with a TLP color set to green, as well as entities with a TLP color set to white, and entities with no TLP color code flag.

- The **Filter TLP color** radio buttons allow including in the outgoing feed data only an entity subset, based on the selected **TLP** (https://www.us-cert.gov/tlp) value. If you set a TLP color as a filter, the feed includes all the entities flagged with the selected TLP color code, as well as the entities whose TLP color indicates that they are suitable for progressively broader audiences. For example, if you select green, the feed includes entities with a TLP color set to green and entities with a TLP color set to white.

Set reliability and relevancy

- **Source reliability**: from the drop-down menu select an option to flag the level of reliability of the source. It helps analysts assess how much confidence they can reasonably have in an intel source.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** (https://en.wikipedia.org/wiki/admiralty_code).
  Example: *B - Usually reliable*
- **Relevancy threshold (%)** allows you to set a filter to include in the outgoing feed data only the entities whose relevancy value is higher than the one defined here.

Set extract filters

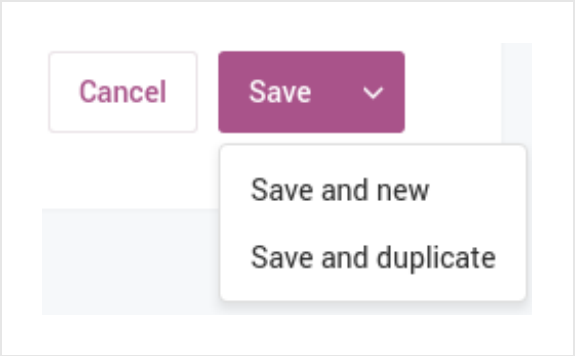
- **Allowed extract states**: from the drop-down menu select one or more extract states to include in the outgoing feed data only the entities whose extract states match the selections defined here.
- **Extract types**: from the drop-down menu select one or more extract types to include in the outgoing feed data only the entities whose extracts types match the selections defined here.
- **Enrichment extract types**: from the drop-down menu select one or more enrichment extract types to include in the outgoing feed data only the entities whose enrichment extracts types match the selections defined here.

- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating right away a new item of the same type; for example, a dataset, a feed, a rule, or a task.
- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.



Configure the content type

When you select the **EclecticIQ JSON** content type for an outgoing feed, you need to configure the following content type parameters:

- **Override producer**: select this checkbox to replace the original producer identity with the one defined in the platform. Leave it deselected to include the original producer of the information.

##

Content type	Allowed transport types
EclecticIQ JSON	FTP upload
	HTTP download
	Mount point upload
	Send email
	TAXII inbox
	TAXII poll

FTP upload

If you want to make the outgoing feed data available through FTP, from the **Transport type** drop-down list select **FTP upload**.
Under **Transport configuration**, configure the following settings:

- **FTP server URL**: the target `ftp://` location to upload the outgoing feed content to, so as to make it available for download.
- **Username**: a valid user name to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.
- **Password**: a valid password to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.

HTTP download

> **Warning:** The HTTP upload/download transport type requires basic access authentication.

If you want to make the outgoing feed data available through an HTTP URL, from the **Transport type** drop-down list select **HTTP download**.
Under **Transport configuration**, configure the following settings:

- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).

Mount point upload

If the source of the outgoing feed is located on a local or network unit, from the **Transport type** drop-down list select the **Mount point upload**
option.

Under **Transport configuration**, configure the following settings:

- **Mount point path**: the path to the local or network unit where the source data for the outgoing feed is stored.

Send email

> **Warning:** Email needs to be correctly configured in the platform system settings for this transport option to work.

If you want to make the outgoing feed data available by email, from the**Transport type** drop-down list select **Send email**.
Under **Transport configuration**, configure the following settings:

- **Mail subject**: enter a short, descriptive subject for the outgoing email notifications.
- **Platform groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list. All the members of the selected group(s) will receive email notifications with the outgoing feed data.
- **Platform users**: if you want to further limit the outgoing feed email recipient targets, from the drop-down list you can select one or more users. In this case, only the selected users belonging to the designated groups will receive email notifications with the outgoing feed data.

TAXII inbox

> **Warning:** Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.
>
> The **TAXII inbox** transport type requires Cabby. For further details, see the **official Cabby documentation** `(https://cabby.readthedocs.org/en/latest/)`, the **Cabby public repo on GitHub** `(https://github.com/eclecticiq/cabby)`, and the **Cabby download page** `(https://pypi.python.org/pypi/cabby/)`.

If you want to make the outgoing feed data available through a TAXII server and push email notifications to TAXII clients, from the **Transport type** drop-down list select **TAXII inbox**.
Under **Transport configuration**, configure the following settings:

- **Inbox service URL**: specify a valid URL address to determine the service location where the available **TAXII data collections** `(https://taxiiproject.github.io/documentation/sample-use/#data-collections)` are stored.
  Example:
  `https://example.com/taxii-inbox`
- **Destination collection name**: specify a valid collection as the source for the outgoing feed data.
  Example:
  `collection.Default`

- **Taxii version**: select the TAXII version your system supports:
  - Either **1.0** (https://taxiiproject.github.io/releases/1.0/)
  - Or **1.1** (https://taxiiproject.github.io/releases/1.1/)
- **EclecticIQ authentication URL**: the URL exposing the platform authentication and authorization service. The platform autorization endpoint is `/auth`.
  Example:
  `https://<platform.host>/auth`
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **SSL certificate**: paste here a valid SSL certificate, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines.
- **SSL key**: paste here a valid SSL private key, including the `-----BEGIN RSA PRIVATE KEY-----` and `-----END RSA PRIVATE KEY-----` lines.
- **SSL key password**: enter here the password to unlock the SSL key.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

TAXII poll

> **Warning:** Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.
>
> The **TAXII poll** transport type requires Cabby. For further details, see the **official Cabby documentation** (https://cabby.readthedocs.org/en/latest/), the **Cabby public repo on GitHub** (https://github.com/eclecticiq/cabby), and the **Cabby download page** (https://pypi.python.org/pypi/cabby/).

If you want to make the outgoing feed data available through polling — where a TAXII client polls the TAXII server to request information and data updates — from the **Transport type** drop-down list select **TAXII poll**.

- Make sure that at least one dataset is selected under **Dataset** to allow TAXII clients to request information and updates about the specified **TAXII data collection(s)** (https://taxiiproject.github.io/documentation/sample-use/#data-collections).
- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

How to configure STIX 1.2 outgoing feeds

Summary: Set up and configure Snort logs outgoing feeds.

In the EclecticIQ Platform you can configure outgoing feeds to share and distribute cyber threat intelligence in several formats. Share knowledge and promote collaboration to support an ecosystem where partners work together to identify threats, and define an effective course of action to ensure their assets are protected.

This article describes how to configure **STIX 1.2** version **1.2** `(https://stixproject.github.io/data-model/1.2/)` outgoing feeds, so that you can distribute selected intelligence through the EclecticIQ Platform.

Configure the general options

✔ On the forms, input fields marked with an asterisk are required.

- On the top navigation bar, click the ✚ *plus* button.
- On the **Create new** sidebar, click **Data management > Outgoing feed**.

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button.
- Under **Configuration** on the drop-down menu, click **Data management**, and then **Outgoing feeds**.

The **Outgoing feeds** page displays an overview of the configured outgoing feeds to publish and distribute selected intel from the platform to external parties, services, and systems.

On the top-right corner of the screen, click the **+ Outgoing feed** button. - The **✚ > Data management > Outgoing feeds > Create** form page includes several input fields to help you define *what* you want to share and *how*, that is, the data content type and the data transport type.

- Under **Feed name**, enter a name for the feed you are creating. It should be descriptive and easy to remember.
- **Transport type**: from the drop-down menu select the appropriate transport type to publish the data through the outgoing feed. This can vary, based on the carrier used to distribute the data.
- Depending on the selected transport type, you may need to specify additional settings under **Transport configuration**.
  For example:
  - A URL endpoint corresponding to the API service exposing the data source for the incoming feed.
  - A valid API key to grant you access to the feed data source.
  - Any required login credentials to obtain access to the feed data source.

- **Content type**: from the drop-down menu select **STIX 1.2** and configure the appropriate parameters under **Content configuration**, when applicable.
- **Dataset**: from the drop-down menu select one or more datasets as data sources for the outgoing feed.
- **Update strategy**: from the drop-down menu select the preferred method to update the data:
  - **Append**: every time the outgoing feed task runs, new data is appended to the existing data.
    When new intelligence is made available through the outgoing feed, it is added after/below the existing data from the same feed.
  - **Replace** every time the outgoing feed task runs, existing data is deleted and it is replaced with new data.
    When new intelligence is made available through the outgoing feed, it overwrites and replaces existing data from the same feed.

Set a schedule

- Under **Execution schedule** you can define how often you want to run the outgoing feed task:
- **None**: no schedule is defined. You need to manually trigger the task to publish data through the outgoing feed.
- **Minute**: the outgoing feed task run automatically.
  You define the execution interval in 5-minute increments from the corresponding drop-down menu.
- **Hour**: the outgoing feed task task run automatically every hour.
  You define how long after the beginning of an hour the task should run from the corresponding drop-down menu.
- **Day**: the outgoing feed task task run automatically once a day.
  You define the time of the day when the task should run from the corresponding drop-down menu.
- **Week**: the outgoing feed task task run automatically once a week.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menu.
- **Month**: the outgoing feed task task run automatically once a month.
  You define the day of the calendar month and time of the day when the task should run from the corresponding drop-down menu.
  Keep in mind that not all months of the year have 31 days.
- Select the **Active** checkbox to make the feed available immediately after creating it.

Set a TLP override

- The **Override TLP** options overwrite the **TLP** `(https://www.us-cert.gov/tlp)` color code associated with the outgoing feed entities with the one you set here. The selected TLP value is assigned to all the entities in the outgoing feed.

You can override the original or the current TLP color code of an entity, an incoming feed, or an outgoing feed. When working as a filter, TLP colors select a decreasing range: if you set a TLP color as a filter the enricher, the feed, or the returned filtered results include all the entities flagged with the selected TLP color code, as well as all the entities whose TLP color indicates that they are progressively lower risk, less sensitive, and suitable for disclosure to broader audiences.
For example, if you select green the filtered results include entities with a TLP color set to green, as well as entities with a TLP color set to white, and entities with no TLP color code flag.

- The **Filter TLP color** radio buttons allow including in the outgoing feed data only an entity subset, based on the selected **TLP** `(https://www.us-cert.gov/tlp)` value. If you set a TLP color as a filter, the feed includes all the entities flagged with the selected TLP color code, as well as the entities whose TLP color indicates that they are suitable for progressively broader audiences. For example, if you select green, the feed includes entities with a TLP color set to green and entities with a TLP color set to white.

Set reliability and relevancy

- **Source reliability**: from the drop-down menu select an option to flag the level of reliability of the source. It helps analysts assess how much confidence they can reasonably have in an intel source.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code)`.
  Example: *B - Usually reliable*
- **Relevancy threshold (%)** allows you to set a filter to include in the outgoing feed data only the entities whose relevancy value is higher than the one defined here.

Set extract filters

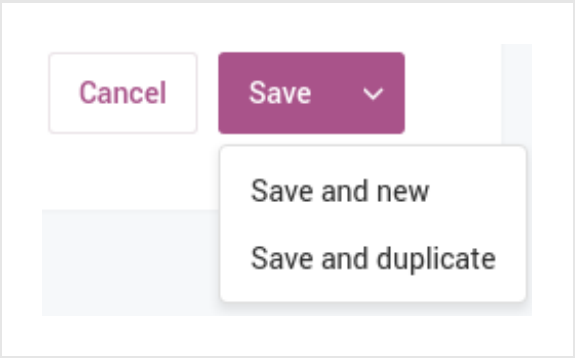
- **Allowed extract states**: from the drop-down menu select one or more extract states to include in the outgoing feed data only the entities whose extract states match the selections defined here.
- **Extract types**: from the drop-down menu select one or more extract types to include in the outgoing feed data only the entities whose extracts types match the selections defined here.
- **Enrichment extract types**: from the drop-down menu select one or more enrichment extract types to include in the outgoing feed data only the entities whose enrichment extracts types match the selections defined here.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating right away a new item of the same type; for example, a dataset, a feed, a rule, or a task.

- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.



Configure the content type

When you select the **STIX 1.2** content type for an outgoing feed, you need to configure the following content type parameters:

- **Override producer**: select this checkbox to replace the original producer identity with the one defined in the platform. Leave it deselected to include the original producer of the information.

##

Content type	Allowed transport types
STIX 1.2	FTP upload
	HTTP download
	Mount point upload
	Send email
	TAXII inbox
	TAXII poll

FTP upload

If you want to make the outgoing feed data available through FTP, from the**Transport type** drop-down list select **FTP upload**.
Under **Transport configuration**, configure the following settings:

- **FTP server URL**: the target `ftp://` location to upload the outgoing feed content to, so as to make it available for download.

- **Username**: a valid user name to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.
- **Password**: a valid password to authenticate and be granted the necessary authorization to upload the outgoing feed content to the designated FTP server location.

HTTP download

> **Warning:** The HTTP upload/download transport type requires basic access authentication.

If you want to make the outgoing feed data available through an HTTP URL, from the**Transport type** drop-down list select **HTTP download**.
Under **Transport configuration**, configure the following settings:

- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).

Mount point upload

If the source of the outgoing feed is located on a local or network unit, from the **Transport type** drop-down list select the **Mount point upload**
option.

Under **Transport configuration**, configure the following settings:

- **Mount point path**: the path to the local or network unit where the source data for the outgoing feed is stored.

Send email

> **Warning:** Email needs to be correctly configured in the platform system settings for this transport option to work.

If you want to make the outgoing feed data available by email, from the **Transport type** drop-down list select **Send email**.
Under **Transport configuration**, configure the following settings:

- **Mail subject**: enter a short, descriptive subject for the outgoing email notifications.
- **Platform groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list. All the members of the selected group(s) will receive email notifications with the outgoing feed data.
- **Platform users**: if you want to further limit the outgoing feed email recipient targets, from the drop-down list you can select one or more users. In this case, only the selected users belonging to the designated groups will receive email notifications with the outgoing feed data.

TAXII inbox

> **Warning:** Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.
>
> The **TAXII inbox** transport type requires Cabby. For further details, see the **official Cabby documentation** (`https://cabby.readthedocs.org/en/latest/`), the **Cabby public repo on GitHub** (`https://github.com/eclecticiq/cabby`), and the **Cabby download page** (`https://pypi.python.org/pypi/cabby/`).

If you want to make the outgoing feed data available through a TAXII server and push email notifications to TAXII clients, from the **Transport type** drop-down list select **TAXII inbox**.
Under **Transport configuration**, configure the following settings:

- **Inbox service URL**: specify a valid URL address to determine the service location where the available **TAXII data collections** (`https://taxiiproject.github.io/documentation/sample-use/#data-collections`) are stored.
  Example:
  `https://example.com/taxii-inbox`
- **Destination collection name**: specify a valid collection as the source for the outgoing feed data.
  Example:
  `collection.Default`
- **Taxii version**: select the TAXII version your system supports:
  - Either **1.0** (`https://taxiiproject.github.io/releases/1.0/`)
  - Or **1.1** (`https://taxiiproject.github.io/releases/1.1/`)
- **EclecticIQ authentication URL**: the URL exposing the platform authentication and authorization service. The platform autorization endpoint is `/auth`.
  Example:
  `https://<platform.host>/auth`
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content

- **Password**: a valid password to authenticate and be granted the necessary authorization to access the source location of the outgoing feed content
- **SSL certificate**: paste here a valid SSL certificate, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines.
- **SSL key**: paste here a valid SSL private key, including the `-----BEGIN RSA PRIVATE KEY-----` and `-----END RSA PRIVATE KEY-----` lines.
- **SSL key password**: enter here the password to unlock the SSL key.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

TAXII poll

> **Warning:** Before configuring a TAXII transport type for an incoming or outgoing feed, make sure the appropriate TAXII service is correctly configured in the platform sysem settings.
>
> The **TAXII poll** transport type requires Cabby. For further details, see the **official Cabby documentation** (`https://cabby.readthedocs.org/en/latest/`), the **Cabby public repo on GitHub** (`https://github.com/eclecticiq/cabby`), and the **Cabby download page** (`https://pypi.python.org/pypi/cabby/`).

If you want to make the outgoing feed data available through polling — where a TAXII client polls the TAXII server to request information and data updates — from the **Transport type** drop-down list select **TAXII poll**.

- Make sure that at least one dataset is selected under **Dataset** to allow TAXII clients to request information and updates about the specified **TAXII data collection(s)** (`https://taxiiproject.github.io/documentation/sample-use/#data-collections`).
- **Public**: default setting: deselected. Leave this checkbox deselected to make the outgoing feed available only to specific groups. You can select these groups from the **Authorized groups** drop-down list.
- **Authorized groups**: restricts access to the outgoing feed only to thegroups you select from the drop-down list.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

How to create a money mule TTP

Summary: Create a money mule TTP entity to investigate fraudulent activities and to identify the actors involved in them.

Money mules are middlemen who carry out illegal transactions on behalf of a criminal third party. Money mules may not always be aware that they are engaging in criminal activities aimed at committing fraud. They are part of a larger scheme designed to carry out fraudulent transactions involving money or goods.

In a fraudulent financial transaction, money mules are responsible for laundering the illicitly obtained money such as proceeds from phishing, malware or email scams. They transfer the money using money orders or cryptocurrencies, which provide an effective layer of obfuscation.

To identify and to track these actors and their behavioral patterns, fraud and risk teams can create TTP entities that describe the actors, their behaviors, and the victims. Analysts can add relationships with other entities on the fly, as well as let the platform process the data to generate meaningful intelligence providing valuable context to their investigation.

In the EclecticIQ Platform, you always record a money mule as a TTP entity where you need to include at least:

- An actor (the money mule).
  The TTP entity describes the money mule as a malicious actor by defining the context the money mule operates in as accurately as possible.
- A victim (for example, a bank account).
  You define and describe the victim of a money mule in the **Characteristics > Targeted Victim** section.
  A victim can be an individual, a commercial or financial entity, or an object like an email address.
- An intended effect of the criminal behavior (for example, fraud).
  You select the intended effect a money mule aims to achieve in the **Intended effects** section.
  Such an effect can be fraud, theft, money laundering, and so on.

Create a money mule TTP

To create a TTP entity describing a money mule, do the following:

- On the left-hand navigation sidebar, click **Editor**.
- On the editor page, click the **+ Entity** button.
- From the drop-down menu select **TTP**.

The entity editor is displayed, and you can proceed to create the new TTP entity.

> ✔ On the forms, input fields marked with an asterisk are required.

Title

Specify the name of the new entity. It should be descriptive and easy to remember.
For example: *Money mule related to IBAN <bank_account_number>*

Analysis

it is a free-text input field to include non-structured information such as additional context, references, links, and so on.
For example, you can add contextual details that can help identify the money mule or the location they operate in.

Confidence

From the drop-down menu select an option to assign the entity a confidence value.
it flags the estimated level of confidence concerning the maliciousness of the (potential) threat the entity represents.
Allowed values:

- **Unknown**
- **None**
- **Low**
- **Medium**
- **High**

Intended effects

From the drop-down menu select an option to specify the purpose or the goal the cyber threat aims at achieving.
Fraud is a very common effect money mules and their associates intend to achieve.

Characteristics

This field allows you to add extra details to more accurately describe the entity; for example, by specifying the threat type, the resources it uses to spread and to reach the intended target, or any connections with other entities.
The one characteristic you want to include in a money mule TTP entity is **Targeted Victim**.

Create a targeted victim

Use the **Characteristics > Targeted Victim** section to record information about the individual, the organization, or the resources affected by the money mule's behavior:

- Under **Characteristics**, click **✚ Characteristic**, and then select **Targeted Victim**.
- The **Targeted Victim** editor opens. It is based on the **CIQ standard** `(https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ciq)` and its **specifications** `(http://docs.oasis-open.org/ciq/v3.0/specs/ciq-specs-v3.html)`. The Customer Information Quality specification aims at providing an open and standard data model to accurately and consistently describe a party such as an individual or an organization, as well as attributes like roles and relationships.
  Apart from drop-down menus and checkboxes, where available, the editor input fields accept free-text as an input.
  No field is mandatory.

Name

Specify the name of the targeted victim. It should be descriptive and easy to remember.
For example: *IBAN <bank_account_number>*

Specification

Defines the targeted victim type.
From the drop-down menu select an option to define the type of victim:

- **Account**
- **Person**
- **Organization**
- **Electronic address**

Targeted systems

Defines the type of infrastructure, system or equipment affected by the money mule's behavior.
The main groups are:

- **Enterprise systems**, for example, the database of a financial institution
- **Industrial control systems**, for example, controllers and other devices driving and monitoring a power plant or an oil pipeline
- **Third-party services**, for example, a cloud-based storage or database services
- **User**, for example, a laptop or a USB stick.

Targeted information

Defines the type of information the money mule illicitly handles.
The main groups are:

- **Information Assets**, for example, the database of a financial institution
- **Authentication Cookies**, for example, persistent cookies granting user access to social media sites, online shopping sites, as well as corporate intranets.

Specify the targeted victim type

- Under **Characteristics > Targeted Victim > Specification**, click **✚ Fields**.
  The available types allow you to describe affected individuals, organizations, and assets.

Account

Account type

Defines the type of account.
For example, *bank* or *online*.

Account status

Defines the current status of the account.

For example, *active* or *blocked*.

Account specification

This section takes a set of predefined keys you can select from the drop-down menu, along with the corresponding values you enter as free-text in the input fields.

- Click the ✚ **add** link to add account details.
- Click the ✚ **more** link to add a new empty row to populate.

Key	Value
Account ID	The account number. Example: *NL30INGB0123456789*
Issuing Authority	The financial institution that issues the account. Example: *ABC Bank*
Account Type	The type of account. Example: *debit* or *savings*
Account Branch	The local branch office or the retail location of the bank responsible for issuing the account. Example: *Utrecht center*
Issuing Country Name	The name of country where the account was issued. Example: *The Netherlands*

Person

Person name

This section takes a set of predefined keys you can select from the drop-down menu, along with the corresponding values you enter as free-text in the input fields.

- Click the ✚ **add** link to add personal details.
- Click the ✚ **more** link to add a new empty row to populate.

Key	Value
Preceding Title	Example: *His*, *Her*
Title	Example: *Rogueness*, *Excellence*, *Pandit*, *Sheikh*
First Name	Example: *Peter*
Middle Name	Example: *Brandon*
Last Name	Example: *Quill*
OtherName Name	Example: *Guardian of the Galaxy*
Alias Name	Example: *Star-Lord*

Key	Value
Generation Identifier	Example: *Jr.*, *Sr.*, *The Younger*, *The Elder*, *XXVIII*
Degree	Example: *BSc Ethical Hacking*

Organization

Organization name

This section takes a set of predefined keys you can select from the drop-down menu, along with the corresponding values you enter as free-text in the input fields.

- Click the ✚ **add** link to add details about the organization.
- Click the ✚ **more** link to add a new empty row to populate.

Key	Value
Name Only	The name the organization is commonly referred to. Example: *Wey-Yu*
Type Only	The entity definition of the organization. Example: *Inc*, *LLC*, *Ltd*
Full Name	The full name of the organization. Example: *Weyland-Yutani Corporation, Inc.*

Electronic address

Electronic address

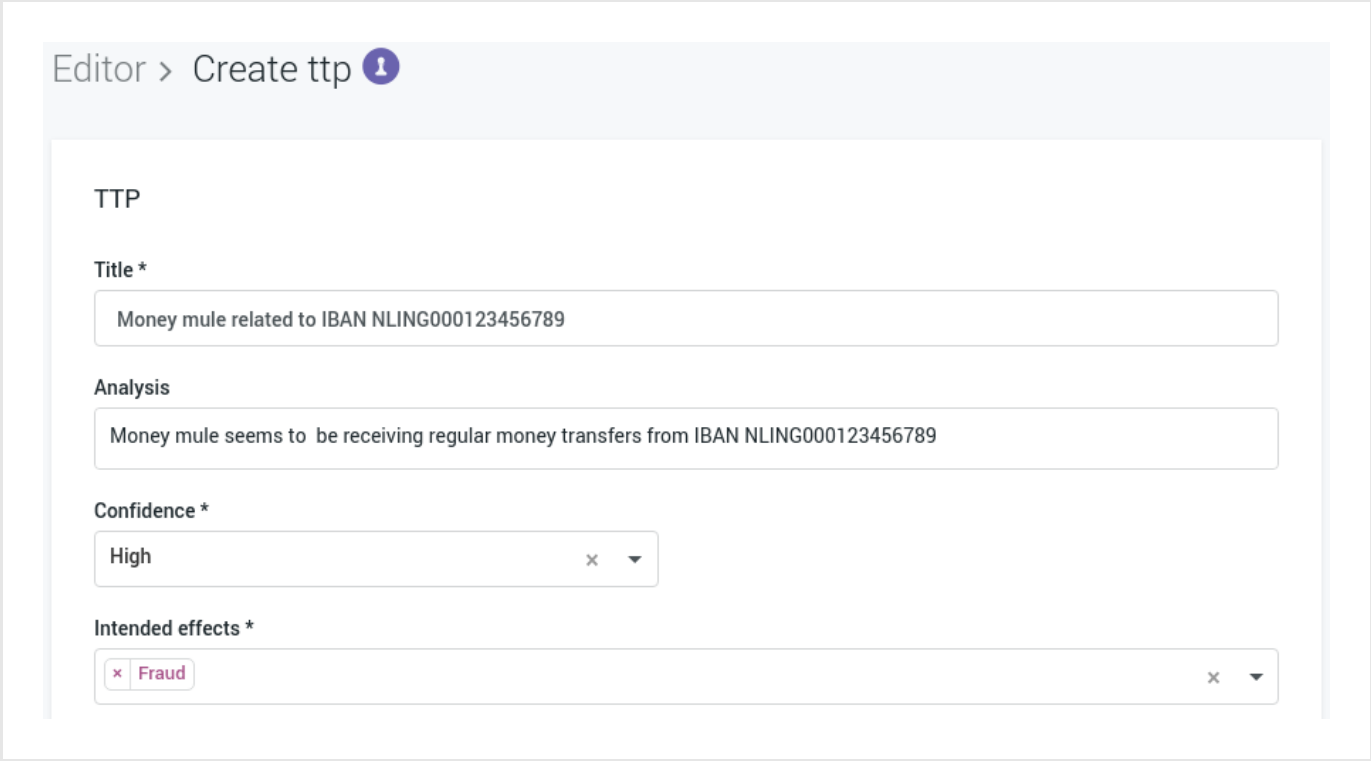
This section takes a set of predefined keys you can select from the drop-down menu, along with the corresponding values you enter as free-text in the input fields.

- The key corresponds to the service provider, for example Google, Yahoo, Skype, ICQ, and so on.
- The associated value needs to be a valid format for the selected service provider, for example:
  - Google: *larry@gmail.com*
  - Yahoo: *melinda@yahoo.com*
  - Skype: *<any_valid_skype_username>*
- Click the ✚ **add** link to add details about electronic contact channels like email, chat, instant messaging, IRC, and so on.
- Click the ✚ **Fields** to add a new empty row to populate.

Next steps

To complete the money mule TTP entity creation, follow the standard steps and procedures you normally use to create entities in the editor, tag them, add relationships, and enrich them with extracts.

Example



Targeted Victim

Name

IBAN NL30INGB0123456789

Specification

Account

Account type

Bank account

Account status

Active

Account specification (5)

Type *

Account ID

Value *

NL30INGB0123456789

Type *

Issuing Authority

Value *

ING Bank

Type *

Account Type

Value *

Debit

Type *

Account Branch

Value *

Utrecht center local branch

Specification

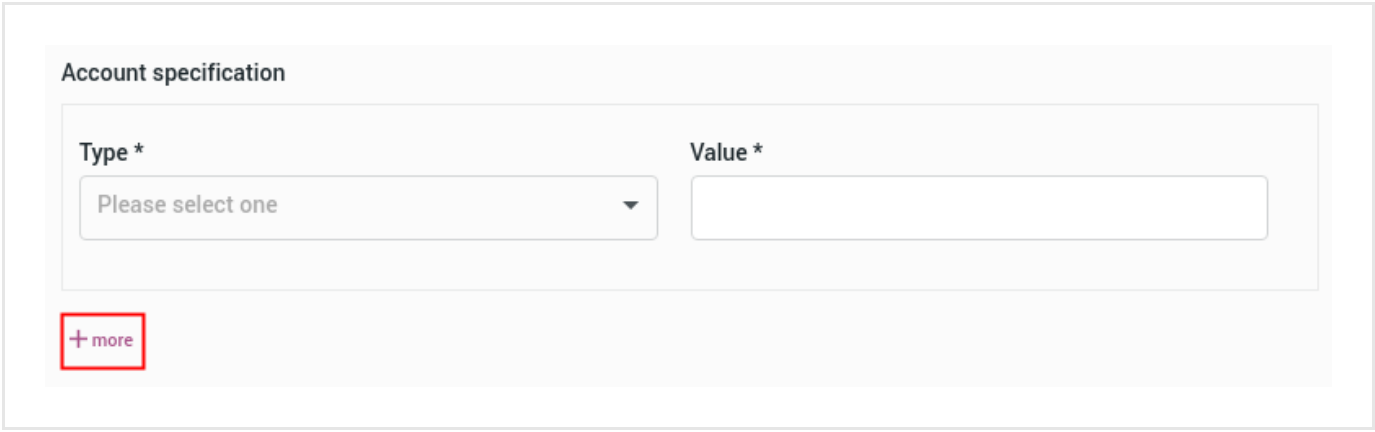
Account

Account type

Account status

Account specification

+ add



How to organize tags with taxonomies

Summary: The Taxonomy page displays an overview of the tags used to label entities in the platform. Besides using tags to organize entities, you can design taxonomies to structure the tags, and to create a controlled tag corpus to improve information retrieval.

Taxonomies are structured categories. Categorization criteria take into account document content and the meaning it conveys. Taxonomies make it easier for you to maintain content, and they help users find what they're looking for. They provide a hierarchical framework to structure tags and to point out relationships among tags. In turn, tag relationships form a reference grid that makes content easier to navigate and to retrieve.

The main benefits of implementing a taxonomy are:

- Label information in a structured way to make it easier to navigate and to retrieve.
- Provide a reference framework to control entity tagging in the platform, so that tags remain meaningful and consistent.
- Deliver more accurate seach results.

The Taxonomy feature

You can use the **Taxonomy** feature to define specific categories to organize entity tags. You can create as many taxonomies as you need, and you can hierarchically relate taxonomy entries with parent-child relationships.

Predefined taxonomies

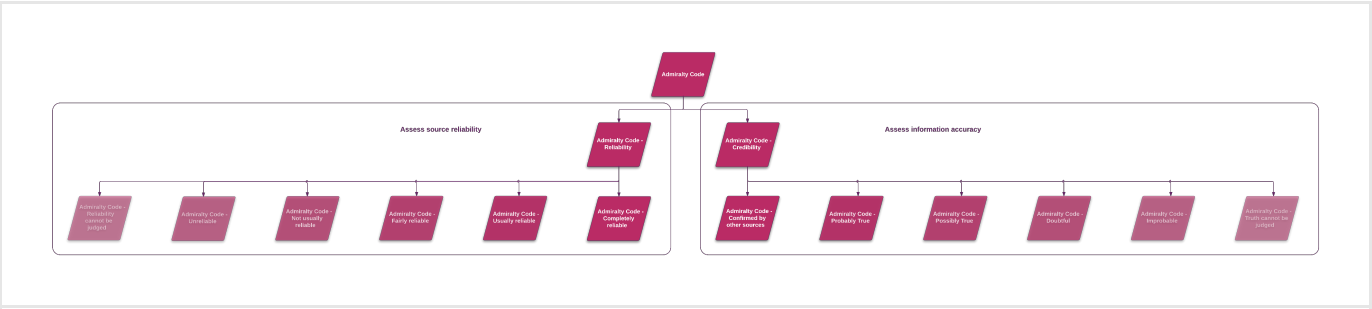
The EclecticIQ Platform ships with the following predefined taxonomy entry sets:

- Admiralty code: based on the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code)`, it assesses the reliability of the source, and the accuracy level of the information.
- Kill chain phase: defines the point(s) in the **kill chain** `(http://www.net-security.org/article.php?id=2220&p=1)` where it is possible to intervene with a mitigation action.

Admiralty code

Use the Admiralty Code taxonomy to label entities with tags that define the level of reliability of the entity source, and the level of accuracy of the entity data. The Admiralty Code taxonomy makes it easier to filter entities and information based on criteria like relevance and credibility.

Entity source reliability	Entity data accuracy
Reliable	Confirmed by other sources
Usually reliable	Probably True
Fairly reliable	Possibly True
Not usually reliable	Doubtful
Unreliable	Improbable
Cannot be judged	Truth cannot be judged



Kill chain

In the context of cyber threat defense, a kill chain aims at encouraging proactive defense, and at implementing adequate courses of action as early as possible in the chain.

The kill chain provides a model to:

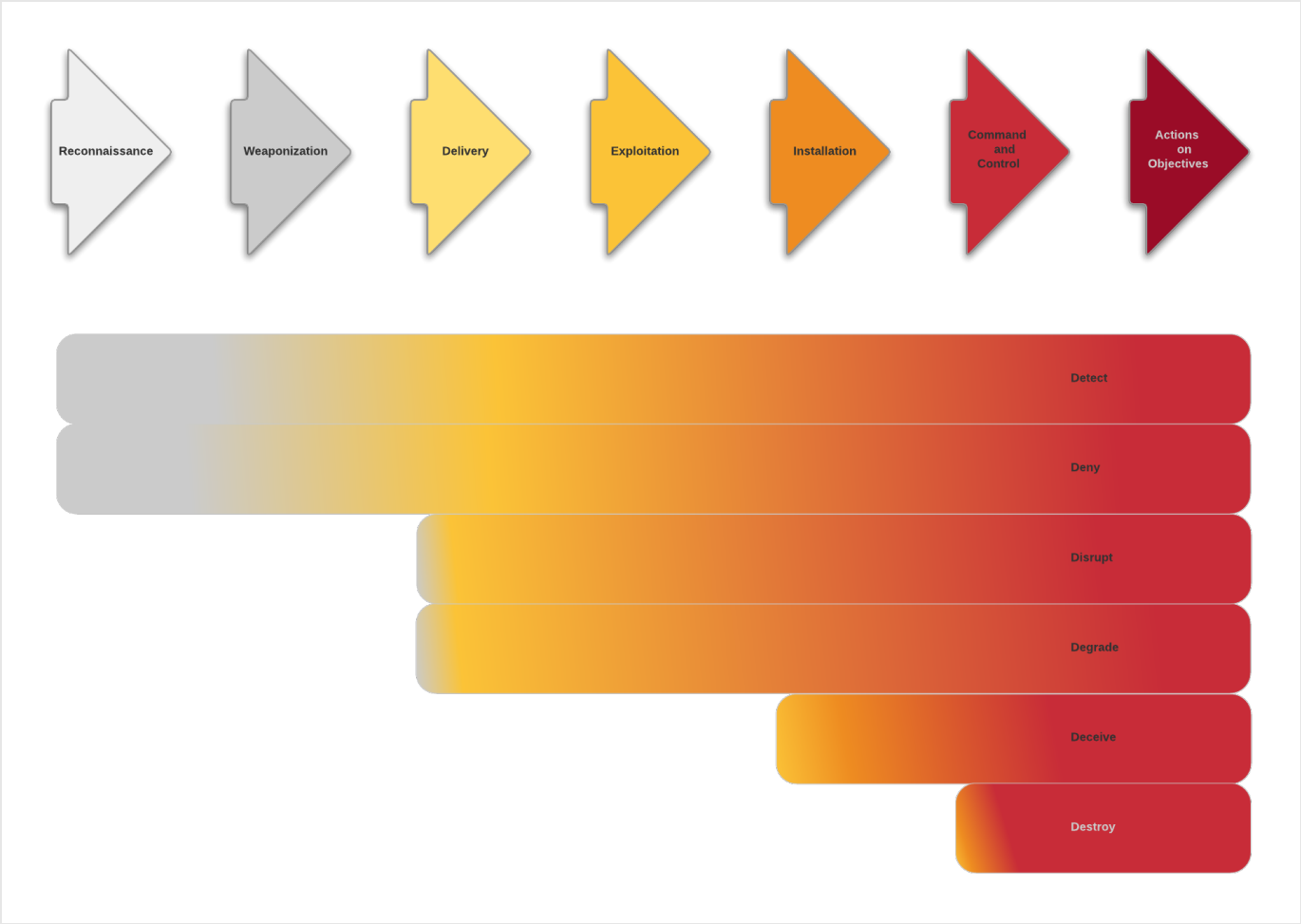
- Identify the root cause of an intrusion, and quantify the damage it causes.
- Plan a defensive course of action to neutralize it.

Kill chain phase	Description
Reconnaissance	Research, identification and selection of targets, often represented as crawling Internet websites such as conference proceedings and mailing lists for email addresses, social relationships, or information on specific technologies.
Weaponization	Coupling a remote access trojan with an exploit into a deliverable payload, typically by means of an automated tool (weaponizer). Increasingly, client application data files such as Adobe Portable Document Format (PDF) or Microsoft Office documents serve as the weaponized deliverable.

Kill chain phase	Description
Delivery	Transmission of the weapon to the targeted environment. The three most prevalent delivery vectors for weaponized payloads by APT actors, as observed by the Lockheed Martin Computer Incident Response Team (LM-CIRT) for the years 2004-2010, are email attachments, websites, and USB removable media.
Exploitation	After the weapon is delivered to victim host, exploitation triggers intruders' code. Most often, exploitation targets an application or operating system vulnerability, but it could also more simply exploit the users themselves or leverage an operating system feature that auto-executes code.
Installation	Installation of a remote access trojan or backdoor on the victim system allows the adversary to maintain persistence inside the environment.
Command and Control (C2)	Typically, compromised hosts must beacon outbound to an Internet controller server to establish a C2 channel. APT malware especially requires manual interaction rather than conduct activity automatically. Once the C2 channel establishes, intruders have "hands on the keyboard" access inside the target environment.
Actions on Objectives	Only now, after progressing through the first six phases, can intruders take actions to achieve their original objectives. Typically, this objective is data exfiltration which involves collecting, encrypting and extracting information from the victim environment; violations of data integrity or availability are potential objectives as well. Alternatively, the intruders may only desire access to the initial victim box for use as a hop point to compromise additional systems and move laterally inside the network.

(Source: *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*, by Eric M. Hutchins, Michael J. Cloppert, Rohan M. Amin, Ph.D. Paper presented at the 6th Annual International Conference on Information Warfare and Security, Washington, DC, 2011.

Course of action	Description
Detect	For example use analytics, auditing, logging tools, and intrusion detection systems (IDS) to detect the intrusion.
Deny	For example use patching, firewall rules, access control lists (ACL), and intrusion prevention systems (IPS) to deny exploitation.
Disrupt	For example use data execution prevention (DEP) and intrusion prevention systems to block or otherwise disturb exploitation.
Degrade	For example use queuing or a tarpit to hinder or otherwise reduce exploitation.
Deceive	For example use DNS redirection or a honeypot to divert exploitation to a decoy.
Destroy	Take control of the attacker's system, and completely neutralize it.



Create a taxonomy entry

✔ On the forms, input fields marked with an asterisk are required.

To create a new taxonomy entry to categorize entity tags, do the following:

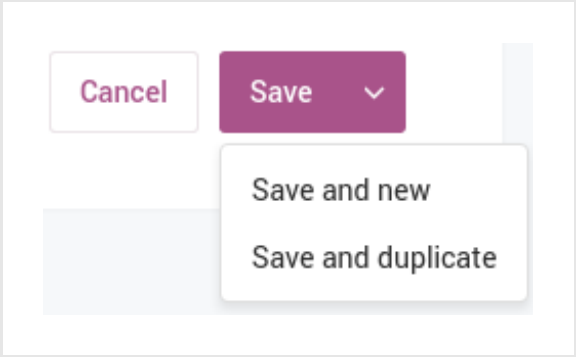
- On the left-hand navigation sidebar click **Taxonomy**.
  On the **Taxonomy** page, an overview of the existing entries is displayed in a table.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.
- Click the ✚ **Entry** button.

- On the **Taxonomy > Create new taxonomy** page, fill out the input fields to define the new taxonomy entry, and whether it is a *parent*, top-level entry, or a *child* entry, i.e. subordinate to a parent:
  - **Name**: define a name for the taxonomy entry. The name you specify here corresponds to the tag name you can assign to entities.
  - **Description**: enter a short explanation of what the entry represents or refers to.
  - **Parent**: if you are creating a subordinate/child taxonomy entry, from the drop-down menu select the parent entry you want to relate the child to.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.
- The newly created taxonomy entry detail pane is displayed, where you can review the entry information. Child entry details on this pane include a clickable link to the corresponding parent.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating right away a new item of the same type; for example, a dataset, a feed, a rule, or a task.
- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.



Edit a taxonomy entry

To edit an existing taxonomy entry, do the following:

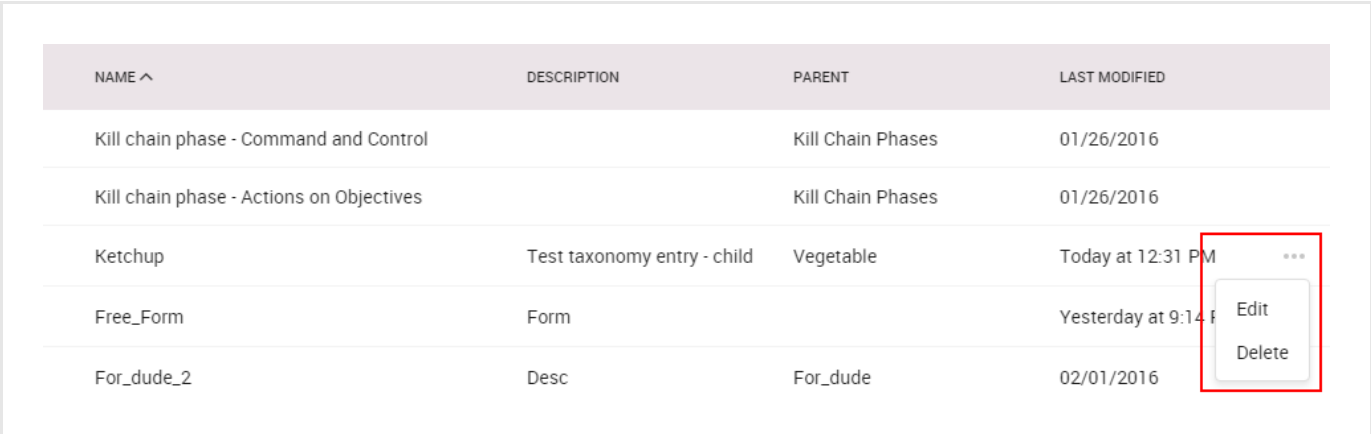
- On the left-hand navigation sidebar click **Taxonomy**.
  On the **Taxonomy** page, an overview of the existing entries is displayed in a table.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.
- On the overview table, click the dotted menu icon.
- From the drop-down menu select **Edit**.

- On the **Taxonomy > Edit Taxonomy** page, edit the name, the description, or the parent-child hierarchy relationship as needed.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Delete a taxonomy entry

To delete an existing taxonomy entry, do the following:

- On the left-hand navigation sidebar click **Taxonomy**.
  On the **Taxonomy** page, an overview of the existing entries is displayed in a table.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (**^**) or a downward-pointing (**ˇ**) arrow in the header indicates ascending and descending sort order, respectively.
- On the overview table, click the dotted menu icon.
- From the drop-down menu select **Delete**.



- On the confirmation pop-up dialog, click **Delete** to confirm the action.
- The taxonomy entry is deleted.

If you delete a parent entry in the taxonomy section, any children related to the removed parent are kept in the taxonomy section, but they lose the parent-child relationship and they become top-level taxonomy entries.

Last generated on Mar 8, 2017

How to work with exposure

Summary: Exposure shows you what your organization is doing with the ingested cyber threat intelligence, so that you can evaluate its usage to define courses of actions and other preventive or reactive procedures within the organization.

What is exposure

In the 2004 Pixar movie **The Incredibles** (`http://www.imdb.com/title/tt0317705/`), Helen Parr goes to see Edna Mode only to find out that Bob Parr, her husband, has resumed superhero work. And he's been gone from home for a few days. When Edna repeatedly asks Helen *"Do you know where he is?"*, Helen does not know what to answer. Previously in the movie, she had witnessed some changes in her husband's behavior that should have alerted her, but she did not use that information.

This is exposure in a nutshell.

When platform entities are flagged as exposed, your organization is not making the most of the available cyber threat intelligence (CTI) to drive an effective course of action. Intelligence is either underutilized, or it is simply just sitting there.

Exposure helps you assess how your organization uses and leverages CTI: how is CTI affecting the organization? Is the organization using CTI to drive processes aiming to detect, deter, and defeat attacks and to minimize risk? What is working well, and what can be done to improve intel utilization?

Exposure gives you a comprehensive and user-friendly overview that helps you answer these questions by showing you how your organization uses existing CTI, and what it can do to use CTI more efficiently.

Configure exposure

You can configure Exposure to be as generic or as specific as you need:

- On the top navigation bar click **Exposure**.
- On the left-hand navigation sidebar click click **Settings**.
- On the **Exposure > Settings** page click **Edit Exposure Settings** to change exposure behavior.

On the configuration page you can define which entities you want to watch for exposure, as well as set filters to minimize unwanted data noise:

- **Entity types**: from the drop-down menu select Entity types to include one or more entity types in the exposure configuration.
  The entity types you add here are tracked to assess their exposure.

- **Extract types**: from the drop-down menu select one or more observable types.
  This option filters the selected entities to include in the exposure configuration only entities with at least one observable type matching the selection(s) you specify here.
- **Confidence values**: from the drop-down menu select one or more confidence values.
  This option filters the selected observable types to include in the exposure configuration only observables whose maliciousness confidence value matches the selection you specify here.
  This filter furter limits the scope of the exposure configuration by watching only entities with at least one observable type matching your selection(s) under **Extract types**. Moreover, the maliciousness confidence level of the matching observable type(s) needs to correspond to least one of the values specified here.
  Confidence corresponds to the value you set under **Rules > Extract > ✚ Rule > Action > Mark as malicious > Confidence**.
- **Entity age**: it defines a time interval ranging from now, that it, the current time, to a point in the past.
  It is an integer and it represents days.
  Only the entities that fall inside this range and that are not older than the number of days specified here are tracked to assess their exposure.
- **Relevancy threshold**: *Relevancy* is a numerical value based on the current time and the estimated start time of the threat. You can use it to sort and filter entities. *0%* = low relevancy — *100%* = high relevancy. Its value is 100% when the current time (*now*) is included between the threat start and end times. Otherwise, its value is 0. If the estimated end time is not available, relevancy is calculated using the estimated start time and the half-life value.
- **Only active entities**: if you select this checkbox, only published entities are tracked to assess their exposure.
  Draft entities are excluded.
- **Show enrichment extracts**: if you select this checkbox, enrichment observables are included and displayed, when available.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

After configuring exposure behavior, you should configure which outgoing feeds should share and distribute exposure information to external systems and devices, so that the data can trigger appropriate actions and responses as part of a concerted course of action.

- On the top navigation bar click **Exposure**.
- On the left-hand navigation sidebar click click **Outgoing feeds**.

On the **Exposure > Outgoing feeds** page you can define how to publish the ingested CTI to minimize exposure. For example, if you are publishing an outgoing feed to an external detection system, the feed data stream is used to detect potential threats.

On this page you map outgoing feeds to the purpose they serve in the context of an integration with external tools and systems.
Within exposure an unused outgoing feed, or a wrongly mapped outgoing feed — for example, an outgoing feed marked as **Detect** but used to distribute CTI to a relevant community, instead — is flagged as exposed.

For each outgoing feed in the overview, you can select one or more checkboxes to map feed usage as appropriate:

- **Detect**: the outgoing feed is published to an external detection system. The feed data is used to detect potential threats that have infiltrated your organization.
- **Prevent**: the outgoing feed is published to an external prevention system. The feed data is used to prevent potential threats from attacking your organization.

- **Community**: the outgoing feed is published to an external information distribution system. The feed is used to share CTI with other parties within or outside the organization.
- **N.A.**: the outgoing feed is not published to any external system.

View exposure

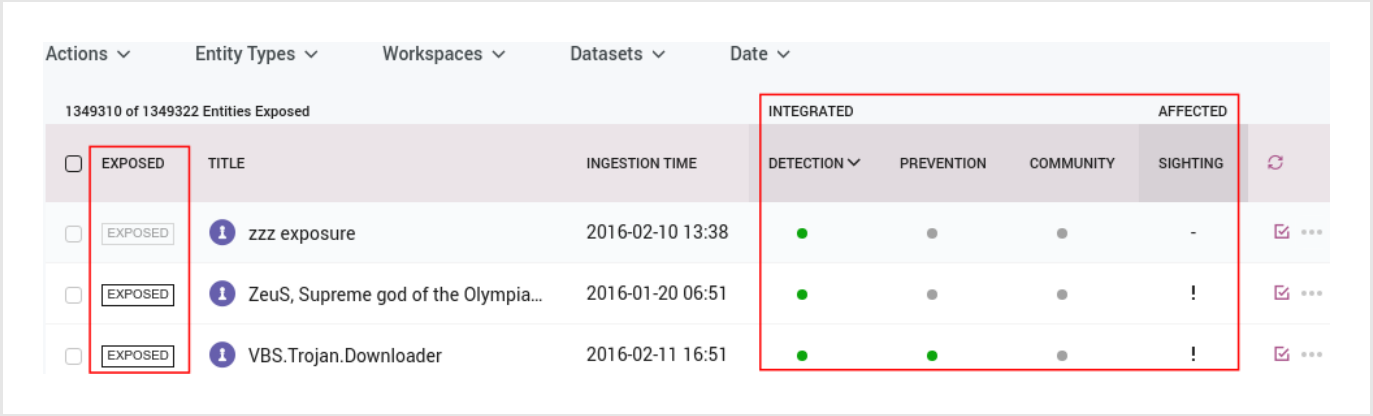
Exposed entities are ingested and processed. However, their intelligence value is not used to produce any follow-up actions.
For example, triggering a detection event in a malware detection application downstream in the system; or a prevention event such as creating a firewall rule; or a community event such as sending a notification message to inform other parties about the possible threat the entity represents.
The entities hold intelligence value that is not consumed. In other words, nobody is doing anything with this information.

To view exposed entities, do the following:

- On the top navigation bar click **Exposure**.
- It shows an overview of all exposed entities.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (**^**) or a downward-pointing (**ˇ**) arrow in the header indicates ascending and descending sort order, respectively.
- To enable the quick filters, click **Show filters**.
- To disable the quick filters, click **Hide filters**.
- On the left-hand navigation sidebar click a filter group name to expand the corresponding sub-nodes:
  - **Entity type**: select one or more checkboxes to view exposure details for the specified entity types.
  - **Date**: select a time interval to view exposure details for the entities ingested between the specified start and end dates.
  - **Dataset**: select one or more checkboxes to view exposure details for the entities belonging to the specified datasets.

You can stack and combine filters as needed. For example, you can create a filter to view exposure details for indicators belonging to the X, Y, and Z datasets, ingested in the first two weeks of last month.

The **Exposure** view shows the following exposure-specific information:



- **Exposed**: the **Exposed** label indicates that the entity is exposed, that is, it is not used in any detection, prevention, or community integrations or processes.
- **Detection**: the entity and the intelligence value it holds are being consumed in an integration with an external system. In this case, with a detection system. If the dot is green, the entity information is used to carry out a follow-up action. It can be a detection follow-up; for example, it can trigger adjusting the settings of a malware detection application accordingly. It can be a prevention follow-up; for example, it can instrument a third-party system to block a range of malicious IP addresses or domain names. Or it can produce a community follow-up; for example, creating and publishing a report to notify other parties about the possible threat the entity represents.
- **Prevention**: the entity and the intelligence value it holds are being consumed in an integration with an external system. In this case, with a prevention system. If the dot is green, the entity information is used to carry out a follow-up action. It can be a detection follow-up; for example, it can trigger adjusting the settings of a malware detection application accordingly. It can be a prevention follow-up; for example, it can instrument a third-party system to block a range of malicious IP addresses or domain names. Or it can produce a community follow-up; for example, creating and publishing a report to notify other parties about the possible threat the entity represents.
- **Community**: the entity and the intelligence value it holds are being consumed in an integration with an external system. In this case, with an information distribution system. If the dot is green, the entity information is used to carry out a follow-up action. It can be a detection follow-up; for example, it can trigger adjusting the settings of a malware detection application accordingly. It can be a prevention follow-up; for example, it can instrument a third-party system to block a range of malicious IP addresses or domain names. Or it can produce a community follow-up; for example, creating and publishing a report to notify other parties about the possible threat the entity represents.
- **Sighting**: a **!** flag means that the entity has been seen in a secured domain, and there should be a sighting entity recording the occurrence.
- Click the ↻ icon to refresh and update the view.

✔ If an entity has been sighted, it is by default exposed no matter the level of integration with external detection, prevention or information distribution systems.

Override entity exposure

You can manually override the configured exposure settings for an entity. The **Override exposure** option allows you to reverse the **Detection**, **Prevention**, and **Sighting** exposure values, and to set them to their opposites.

The entity exposure override history is stored in reverse chronological order, based on the time when the change was applied.

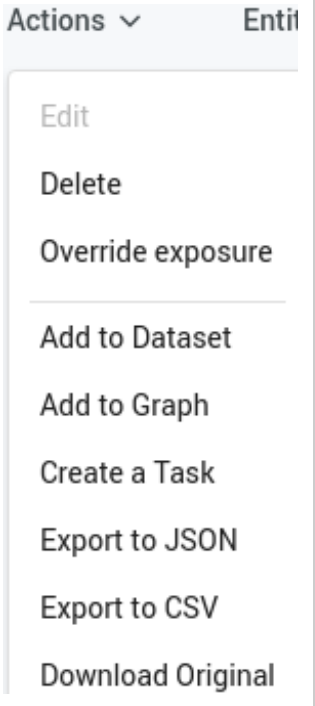
To manually change the exposure state of an entity, do the following:

- On the top navigation bar click **Exposure**.
- On the **Exposure** page click the dotted menu icon on the row corresponding to the entity whose exposure settings you want to override.
- From the context menu select **Override exposure**.
- On the **Override exposure state** dialog window, select the **Override exposure state to** (**ON** or **OFF**, depending on the current exposure value) checkboxes to reverse the current exposure value — from **ON** to **OFF** or the other way around — of **Detection**, **Prevention**, and **Sighting**.
- If you want, you can specify a start date for the override value(s) to become effective: from the drop-down menu select select the desired start date.
- When you are done, close the dialog window. Your settings are automatically saved.

After confiming and saving a manual exposure override, the override value persists until new content is generated, and the entity is updated.

Edit entity exposure

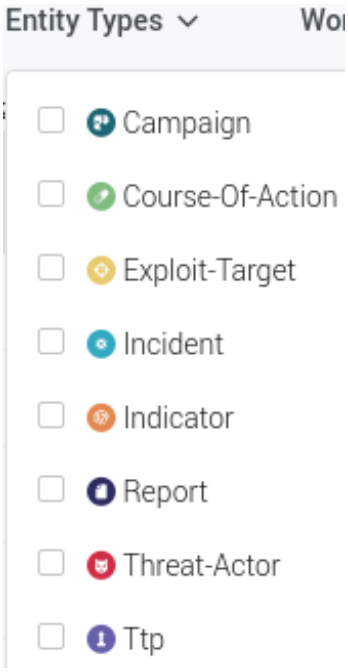
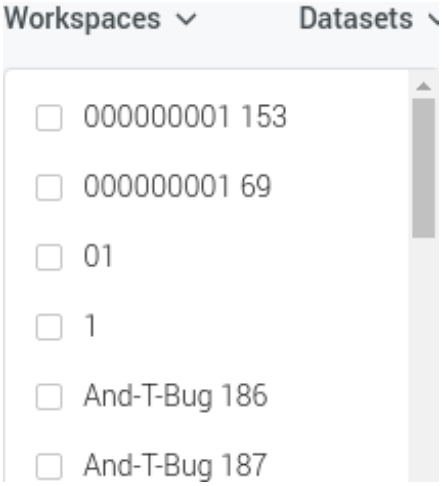
You can manipulate entities by selecting one of the options available in the **Actions** menu above the table view. Apart from the extra **Override exposure** option, the **Actions** menu is the same as the corresponding menu on a workspace **Entities** tab.

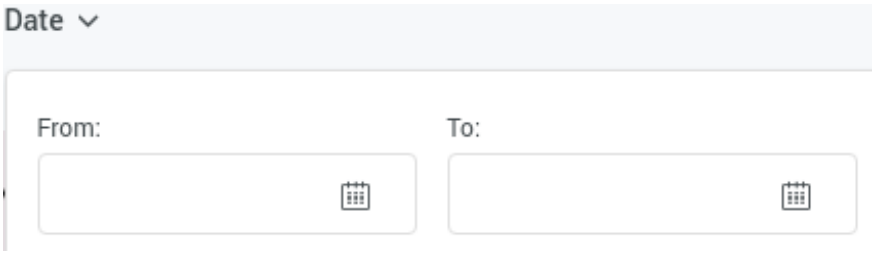
Actions	

Filter exposure

You can narrow down the displayed results by specifying a search string in the filter input field. Alternatively, click one or more quick filters above the table view to select and filter by specific:

- Entity types
- Worskpaces
- Datasets
- Date ranges

Entity Types	
Workspaces	
Datasets	

Date	

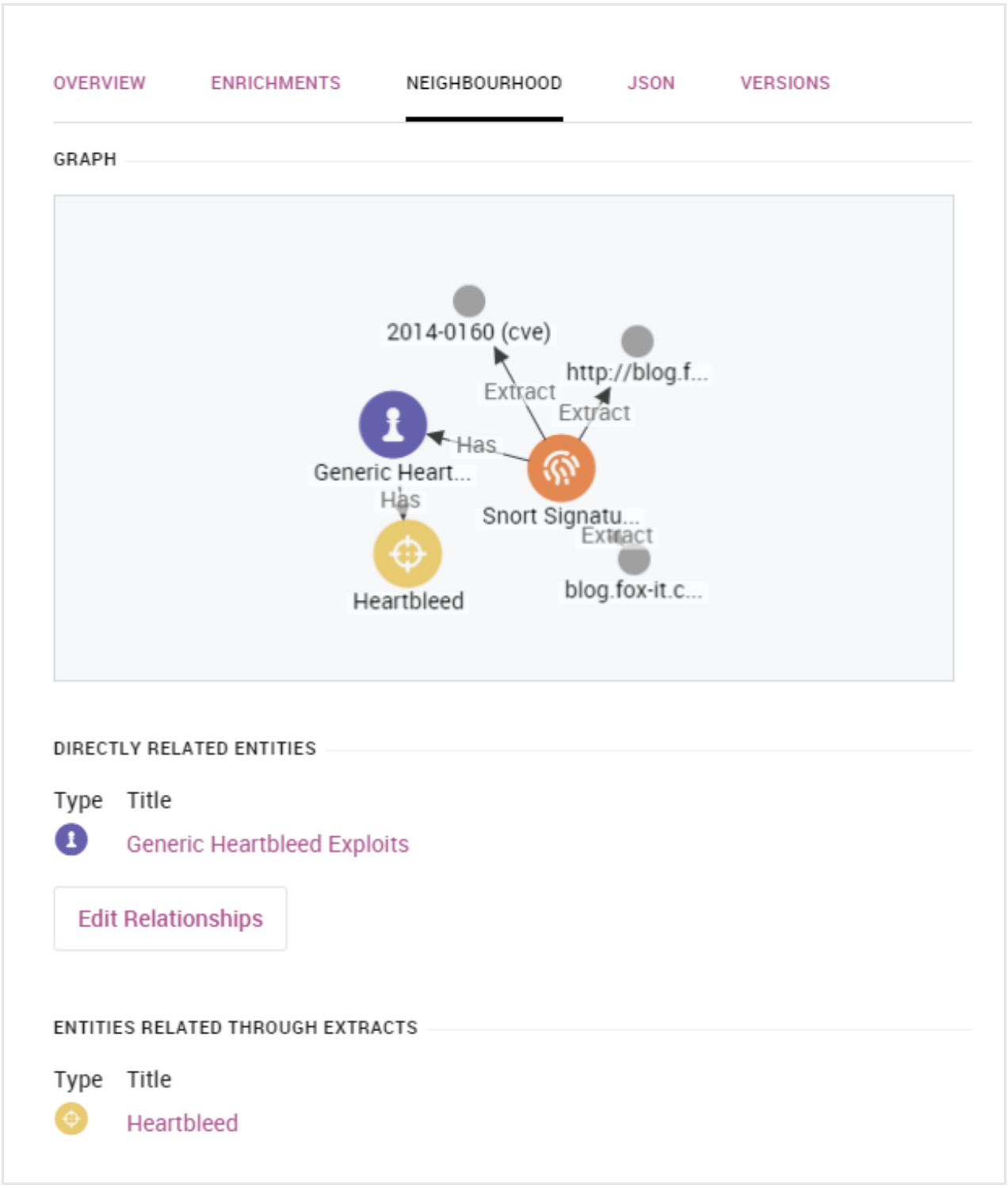
How to work with relationships

Summary: The Neighborhood tab in the entity detail pane includes a small graph canvas showing close relationships of the entity to other entities, as well as related extracts, datasets, workspaces, and tasks.

Go to the Neighborhood graph

During an analysis you may want to quickly inspect an entity to check relationships to other entities and extracts. Normally, you would load the selected entity onto the graph, open the graph, and proceed with the inspection.

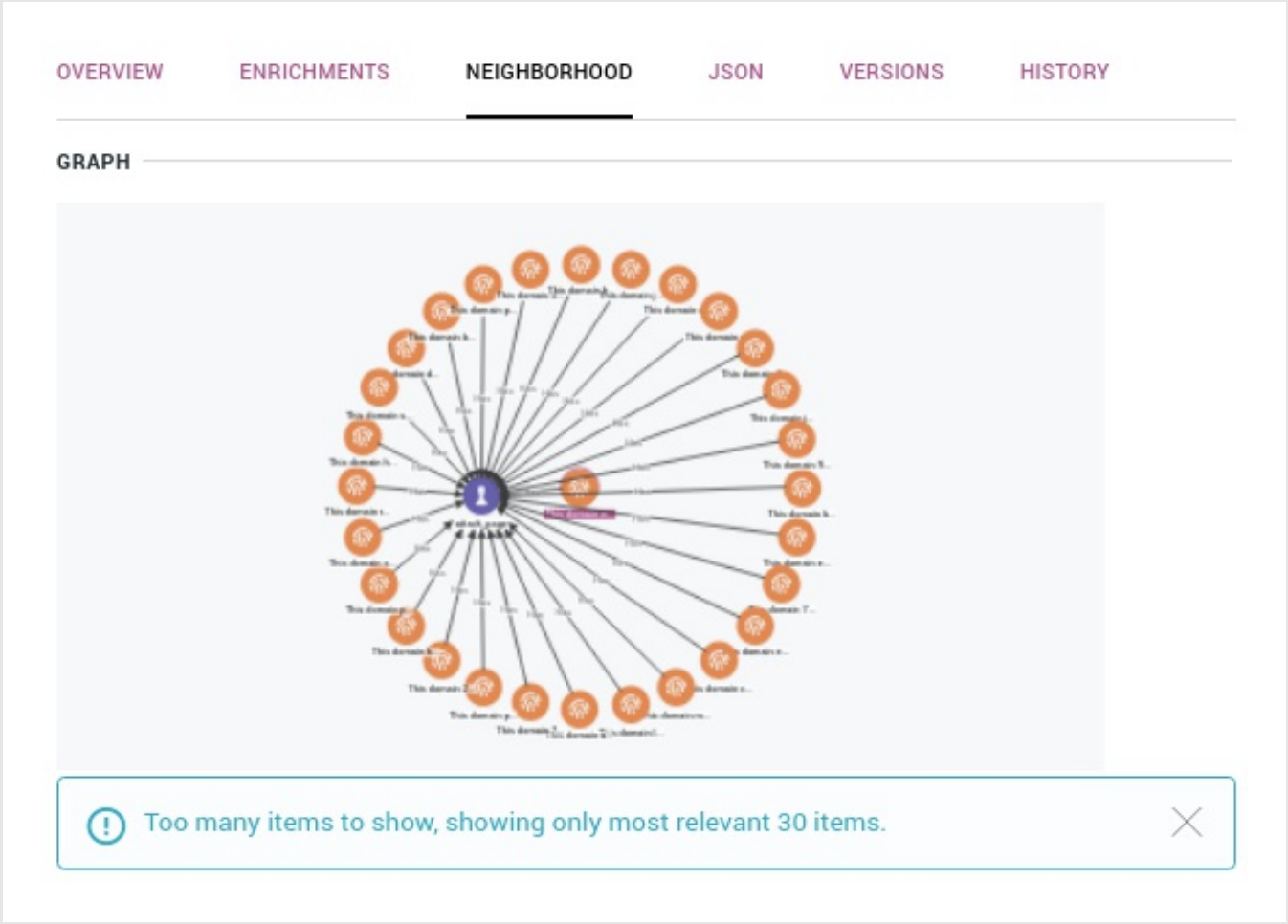
Without leaving the entity detail pane, the **Neighborhood** tab is a faster alternative: click it to see a small graph displaying close-range relationships the entity has with nearby entities and extracts.



Click the embedded graph to load the entity and its neighborhood relationships onto the graph canvas, where you can further analyze the data.

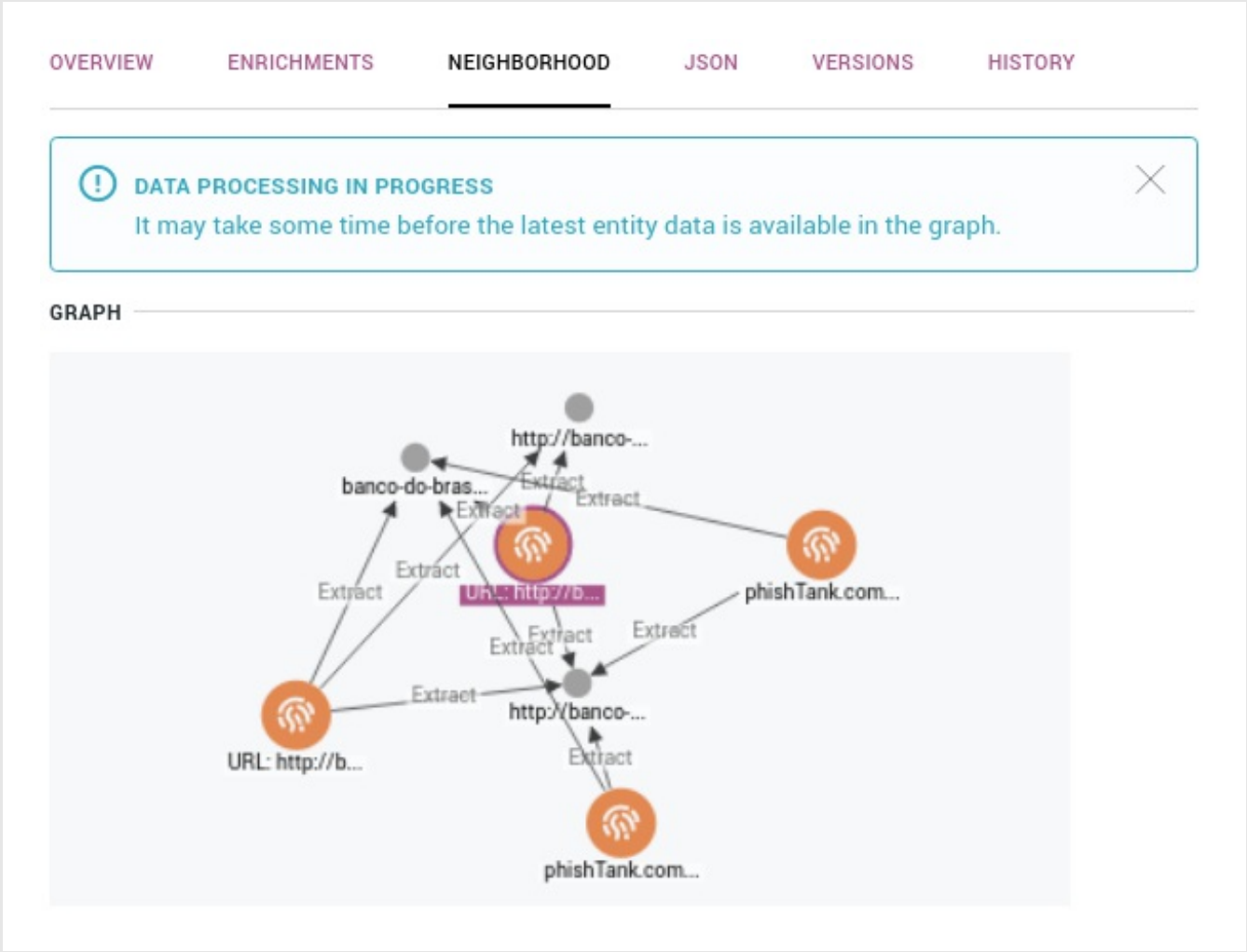
If more than 100 relationships are found for the entity, only the 30 most recently created relationships are displayed on the **Neighborhood** graph. In this case, a notification message is displayed to inform the user:

> Too many items to show, showing only most relevant 30 items.



The embedded graph is a snapshot of the graph canvas view. The embedded snapshot is refreshed when accessing the **Neighborhood** tab, but it is not updated in real time. When the entity relationship landscape changes, for example after adding or removing relationships, the embedded graph goes out of sync. In this case, a notification message is displayed to inform the user:

DATA PROCESSING IN PROGRESS — It may take some time before the latest entity data is available in the graph.

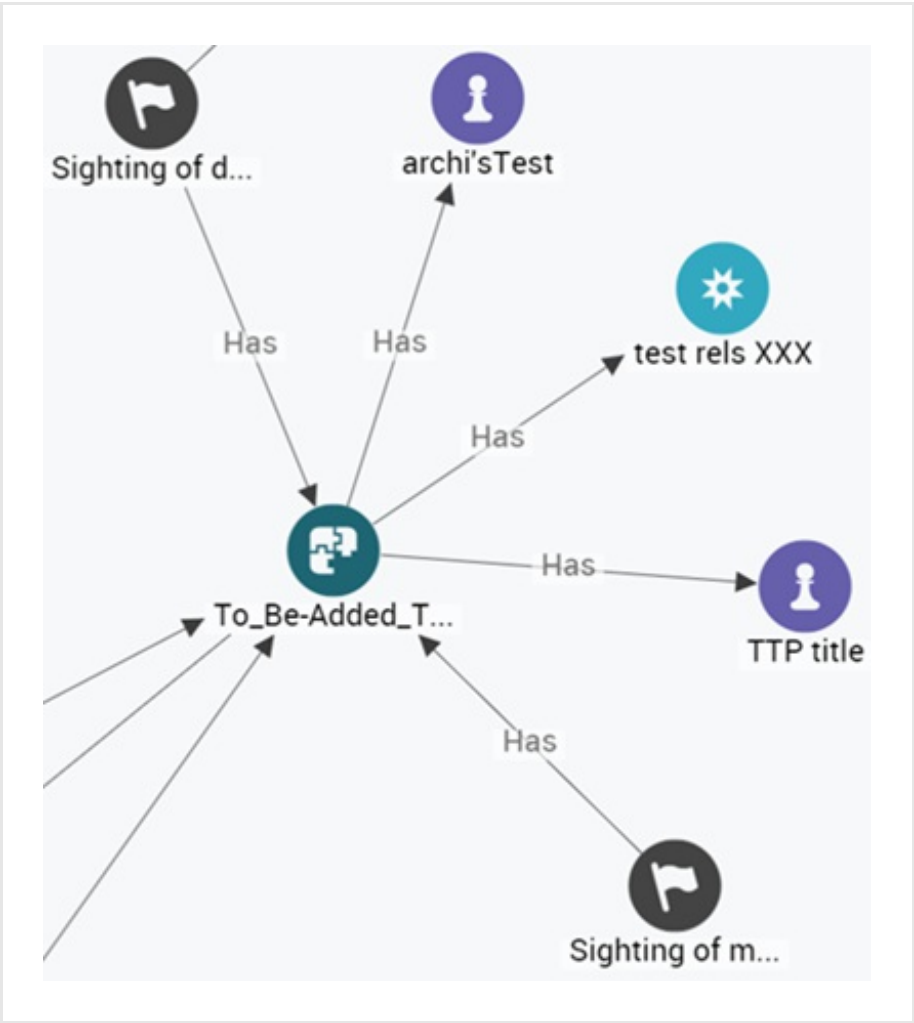


The embedded graph is back in sync after the platform graph has completed indexing. The time this task requires varies, depending on the size of the graph queue.

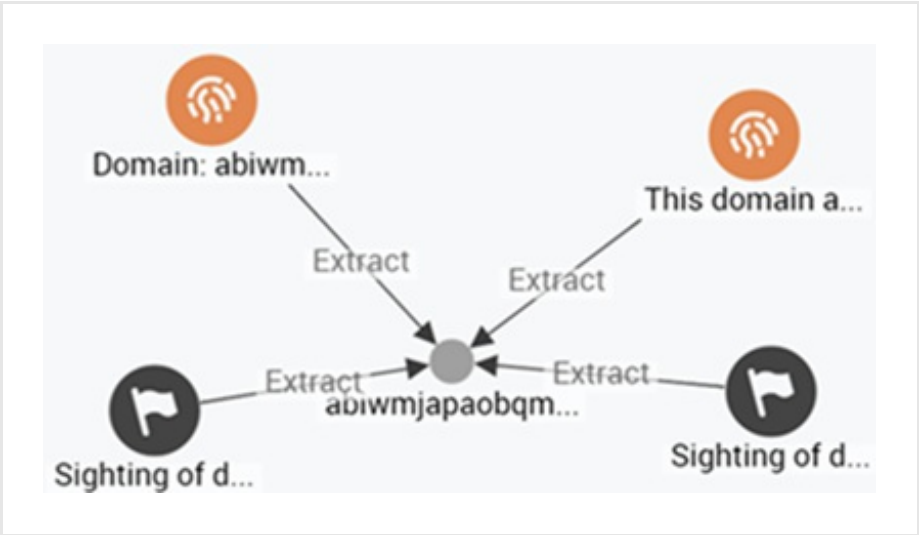
Explore the entity neighborhood

View relationships

In the visual graph representation you can inspect any relationships the entity may have with other entities in the platform. Relationships can be *direct* — the entities are immediately related to each other — or *indirect* — the entities are related through a shared entity or a shared extract.



Entities with direct relationships



Entities with indirect relationships

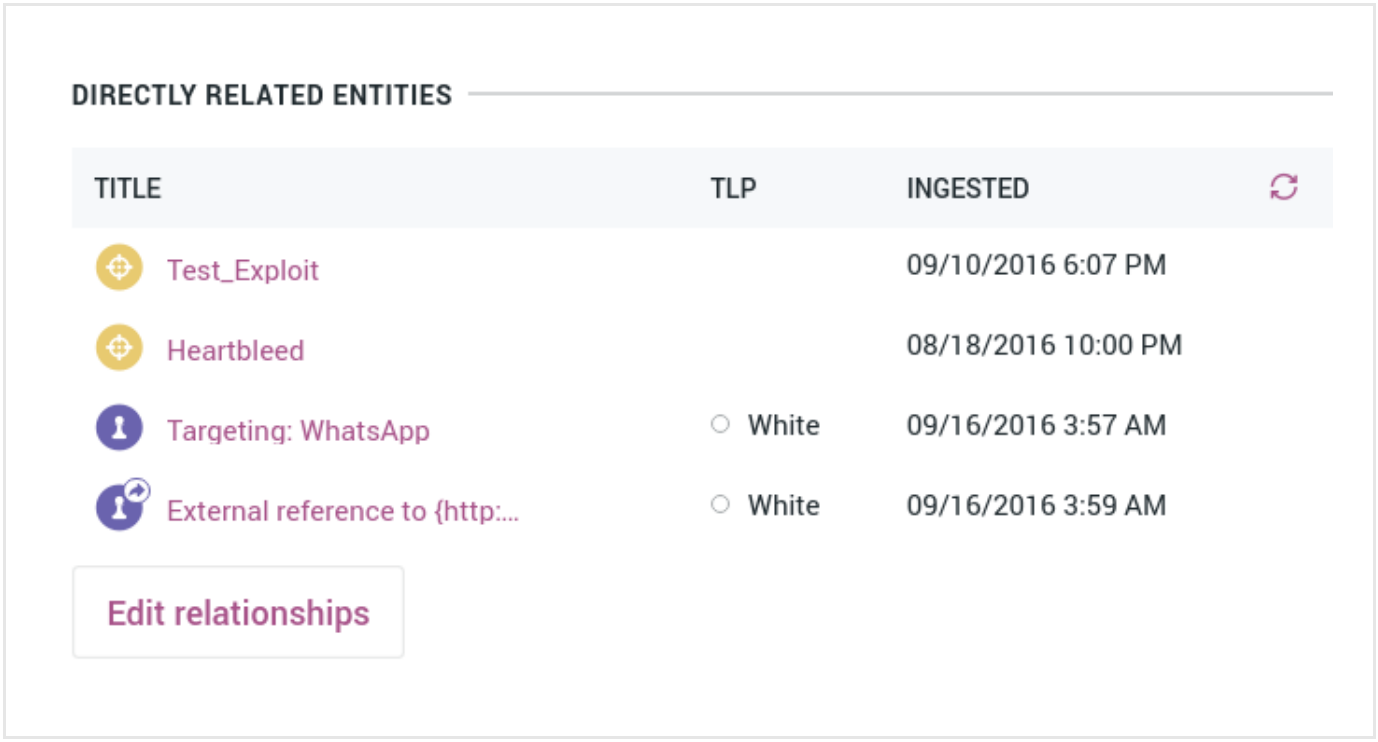
To visually examine the entity more closely, click the small graph to launch the larger and more powerful platform graph.

To edit entity relationships, click **Edit relationships**.

Directly related entities

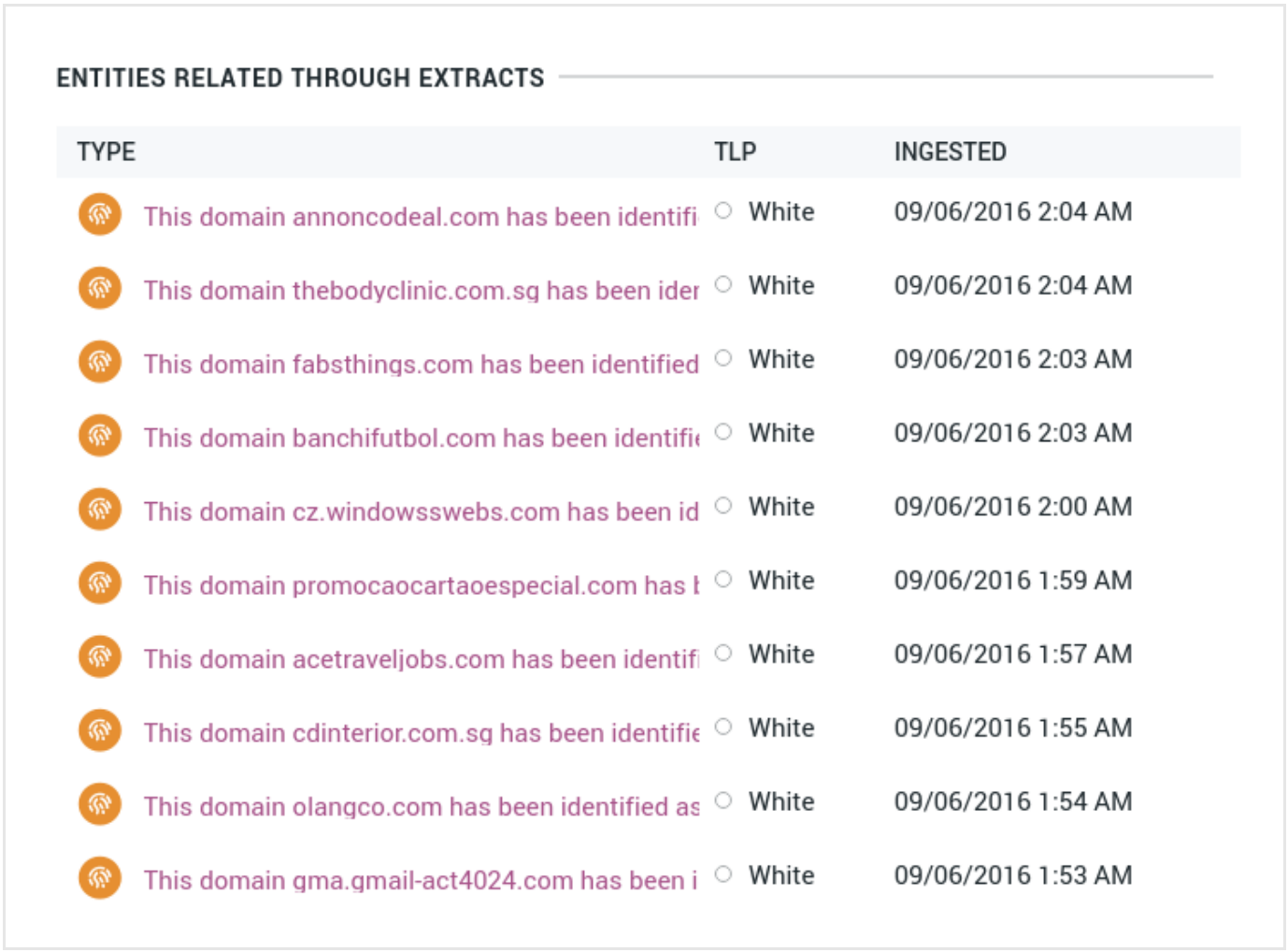
Entities that are directly related to the active entity displayed on the entity pane are listed under **Directly related entities**, and they are sorted by entity type.

Besides the entity name, you can see the TLP color code assigned to the entity, if available, and the entity ingestion time.
To refresh the view, if necessary, click the refresh icon: 🔄.



Entities related through extracts

Entities that are indirectly related through extracts to the active entity displayed on the entity pane are listed under **Entities related through extracts**, and they are sorted by entity type.
Besides the entity name, you can see the TLP color code assigned to the entity, if available, and the entity ingestion time.
To refresh the view, if necessary, click the refresh icon: 🔄.



Related datasets

Entities can belong to one, more, or no datasets. Any datasets the entity is part of are listed here.

Related workspaces

Entities can belong to one, more, or no workspaces. Any workspaces the entity is part of are listed here.

Related tasks

Actionable user tasks associated with the entity are listed here You can create tasks and assign them to yourself or to other users to request follow-ups, for example further investigation or a call for action.

Edit relationships

You can update the entity information by adding and removing relationships. To do so, do the following:

- To *add a relationship*, click the **Edit relationships** button under the **Directly related entities** section.
- **Edit relationships** changes to **✚ Relationship**, and a drop-down menu becomes available. The menu options vary, depending on the entity type displayed in the entity detail pane.
- From the menu, select the option corresponding to the relationship you want to create. The **Search an entity** pop-up dialog is displayed.

- In the **Search an entity** pop-up dialog, click the checkbox(es) to select one or more entities to relate to the entity displayed in the entity detail pane.

> You can refine the search results by specifying a search string in the filter input field. Alternatively, click any column header to select and filter by specific:
> - Entity types
> - Source types
> - Datasets
> - Date ranges.

- Click **Select**.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.
- To *remove a relationship*, click the **x** symbol on the row reporting the relationship you want to remove.
- The row and the corresponding relationship are immediately removed; therefore, this action cannot be undone.

Edit relationships for a campaign

Select this menu option…	**… to create this relationship**
Associated campaigns	Outgoing relationship — Relates the campaign to the selected campaign(s) in the **Search an entity** pop-up dialog.
Attributions	Outgoing relationship — Relates the campaign to the selected threat-actor(s) in the **Search an entity** pop-up dialog.
Related incidents	Outgoing relationship — Relates the campaign to the selected incident(s) in the **Search an entity** pop-up dialog.
Related TTPs	Outgoing relationship — Relates the campaign to the selected TTP(s) in the **Search an entity** pop-up dialog.
Indicator –> Related campaigns	Incoming relationship — Relates the selected indicator(s) in the **Search an entity** pop-up dialog to the campaign.
Report –> Campaigns	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the campaign.
Threat actor –> Associated campaigns	Incoming relationship — Relates the selected threat-actor(s) in the **Search an entity** pop-up dialog to the campaign.
Eclecticiq sighting –> Campaign	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the campaign.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for a course of action

Select this menu option…	… to create this relationship
Related Exploit Targets	Outgoing relationship — Relates the course of action to the selected exploit target(s) in the **Search an entity** pop-up dialog
Related Incidents	Outgoing relationship — Relates the course of action to the selected incident(s) in the **Search an entity** pop-up dialog
Related Courses of Action	Outgoing relationship — Relates the course of action to the selected course(s) of action in the **Search an entity** pop-up dialog
Exploit Target –> Potential Courses of Action	Incoming relationship — Relates the selected exploit target(s) in the **Search an entity** pop-up dialog to the course of action.
Indicator –> Suggested Courses of Action	Incoming relationship — Relates the selected indicator(s) in the **Search an entity** pop-up dialog to the course of action. Recommends carrying out a course of action to respond to an indicator.
Incident –> Courses of Action Requested	Incoming relationship — Relates the selected indicator(s) in the **Search an entity** pop-up dialog to the course of action. Requests to carry out a course of action to respond to an incident.
Incident –> Courses of Action Taken	Incoming relationship — Relates the selected indicator(s) in the **Search an entity** pop-up dialog to the course of action. Reports the course of action carried out as a response to an incident.
Report –> Courses of Action	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the course of action.
Eclecticiq sighting –> Course of Action	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the course of action.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for an exploit target

Select this menu option…	… to create this relationship
Potential Courses of Action	Outgoing relationship — Relates the exploit target to the selected potential course(s) of action in the **Search an entity** pop-up dialog
Related exploit targets	Outgoing relationship — Relates the exploit target to the selected exploit target(s) in the **Search an entity** pop-up dialog

Select this menu option…	… to create this relationship
Course of action –> Related Exploit Targets	Incoming relationship — Relates the selected course(s) of action in the **Search an entity** pop-up dialog to the exploit target.
Report –> Exploit Targets	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the exploit target.
TTP –> Exploit Targets	Incoming relationship — Relates the selected TTP(s) in the **Search an entity** pop-up dialog to the exploit target.
Eclecticiq sighting –> Exploit Target	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the exploit target.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for an incident

Select this menu option…	… to create this relationship
Related Indicators	Outgoing relationship — Relates the incident to the selected indicator(s) in the **Search an entity** pop-up dialog.
Leveraged TTPs	Outgoing relationship — Relates the incident to the selected TTP(s) in the **Search an entity** pop-up dialog.
Attributed threat actors	Outgoing relationship — Relates the incident to the selected threat-actor(s) in the **Search an entity** pop-up dialog.
Related incidents	Outgoing relationship — Relates the incident to the selected incident(s) in the **Search an entity** pop-up dialog.
Courses of Action Requested	Outgoing relationship — Relates the incident to the selected course(s) of action in the **Search an entity** pop-up dialog that are requested to respond to the incident.
Courses of Action Taken	Outgoing relationship — Relates the incident to the selected course(s) of action in the **Search an entity** pop-up dialog that are carried out as a response to the incident.
Campaign –> Related Incidents	Incoming relationship — Relates the selected campaign(s) in the **Search an entity** pop-up dialog to the incident.
Course of Action –> Related Incidents	Incoming relationship — Relates the selected course(s) of action in the **Search an entity** pop-up dialog to the incident.
Report –> Incidents	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the incident.

Select this menu option…	… to create this relationship
Eclecticiq sighting –> Incident	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the incident.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for an indicator

Select this menu option…	… to create this relationship
Indicated TTPs	Outgoing relationship — Relates the indicator to the selected TTPs(s) in the **Search an entity** pop-up dialog.
Suggested Courses of Action	Outgoing relationship — Relates the indicator to the selected course(s) of action in the **Search an entity** pop-up dialog. Recommends carrying out a course of action to respond to the indicator.
Related indicators	Outgoing relationship — Relates the indicator to the selected indicator(s) in the **Search an entity** pop-up dialog.
Related campaigns	Outgoing relationship — Relates the indicator to the selected campaign(s) in the **Search an entity** pop-up dialog.
Incident –> Related indicators	Incoming relationship — Relates the selected incident(s) in the **Search an entity** pop-up dialog to the indicator.
Report –> Indicators	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the indicator.
Eclecticiq sighting –> Indicator	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the indicator.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for a report

Select this menu option…	… to create this relationship
Indicators	Outgoing relationship — Relates the report to the indicator(s) in the **Search an entity** pop-up dialog.
TTPs	Outgoing relationship — Relates the report to the selected TTP(s) in the **Search an entity** pop-up dialog. Recommends carrying out a course of action to respond to the report.
Exploit targets	Outgoing relationship — Relates the report to the selected exploit target(s) in the **Search an entity** pop-up dialog.

Select this menu option…	… to create this relationship

Incidents	Outgoing relationship — Relates the report to the selected incident(s) in the **Search an entity** pop-up dialog.
Courses of Action	Outgoing relationship — Relates the report to the selected course(s) of action in the **Search an entity** pop-up dialog.
Campaigns	Outgoing relationship — Relates the report to the selected campaign(s) in the **Search an entity** pop-up dialog.
Threat Actors	Outgoing relationship — Relates the report to the selected threat actor(s) in the **Search an entity** pop-up dialog.
Eclecticiq sighting –> Report	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the report.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for a sighting

Select this menu option…	… to create this relationship
Campaign	Outgoing relationship — Relates the sighting to the selected campaign(s) in the **Search an entity** pop-up dialog.
Course of Action	Outgoing relationship — Relates the sighting to the selected course(s) of action in the **Search an entity** pop-up dialog.
Exploit target	Outgoing relationship — Relates the sighting to the selected exploit target(s) in the **Search an entity** pop-up dialog.
Indicator	Outgoing relationship — Relates the sighting to the selected indicator(s) in the **Search an entity** pop-up dialog.
Incident	Outgoing relationship — Relates the sighting to the selected incident(s) in the **Search an entity** pop-up dialog.
Report	Outgoing relationship — Relates the sighting to the selected report(s) in the **Search an entity** pop-up dialog.
Threat actor	Outgoing relationship — Relates the sighting to the threat actor(s) in the **Search an entity** pop-up dialog.
TTP	Outgoing relationship — Relates the sighting to the selected TTP(s) in the **Search an entity** pop-up dialog.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for a threat actor

Select this menu option…	… to create this relationship
Observed TTPs	Outgoing relationship — Relates the threat actor to the selected TTP(s) in the **Search an entity** pop-up dialog.
Associated campaigns	Outgoing relationship — Relates the threat actor to the selected campaign(s) in the **Search an entity** pop-up dialog.
Associated actors	Outgoing relationship — Relates the threat actor to the selected threat actor(s) in the **Search an entity** pop-up dialog.
Campaign –> Attributions	Incoming relationship — Relates the selected campaign(s) in the **Search an entity** pop-up dialog to the threat actor.
Incident –> Attributed threat actors	Incoming relationship — Relates the selected incident(s) in the **Search an entity** pop-up dialog to the threat actor.
Report –> Threat actors	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the threat actor.
Eclecticiq sighting –> Threat actor	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the threat actor.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Edit relationships for a TTP

Select this menu option…	… to create this relationship
Exploit targets	Outgoing relationship — Relates the TTP to the selected exploit target(s) in the **Search an entity** pop-up dialog.
Related TTPs	Outgoing relationship — Relates the TTP to the selected TTP(s) in the **Search an entity** pop-up dialog.
Campaign –> Related TTPs	Incoming relationship — Relates the selected campaign(s) in the **Search an entity** pop-up dialog to the TTP.
Indicator –> Indicated TTPs	Incoming relationship — Relates the selected indicator(s) in the **Search an entity** pop-up dialog to the TTP.
Incident –> Leveraged TTPs	Incoming relationship — Relates the selected incident(s) in the **Search an entity** pop-up dialog to the TTP.

Select this menu option…	… to create this relationship
Report –> TTPs	Incoming relationship — Relates the selected report(s) in the **Search an entity** pop-up dialog to the TTP.
Eclecticiq sighting –> TTPs	Incoming relationship — Relates the selected sighting(s) in the **Search an entity** pop-up dialog to the TTP.

When you are done, click **Save** to store your changes, or **Cancel** to discard them.

View related datasets

When a dataset is related to an entity, it shares data with it. Datasets and entities can be related in the following ways:

- The entity is included in the dataset.
- The entity and the dataset share common extracts.
- The dataset contains an entity that bears a direct or indirect relationship with the active entity displayed in the entity detail pane.

Any datasets that are related to the entity displayed in the entity pane are listed under the **Related datasets** section, sorted in ascending order by dataset name. To change the sort order, based on dataset name or on the total number of entities included in each dataset, click the corresponding column headers.

View related workspaces

When an entity is related to a workspace, it is included in it. Any workspaces that are related to the entity displayed in the entity pane are listed under the **Related workspaces** section, sorted in ascending order by workspace name. Besides the name, you can see the most recent workspace modification date/time, and whether or not you are a workspace collaborator.

To change the sort order, based on workspace name, modification date/time or collaboration flag, click the corresponding column headers.

View related tasks

Under **Related tasks** you can see what is being done with the entity information. **Related tasks** lists any actions that have been requested, are in progress, or have been carried out as a response to the entity.

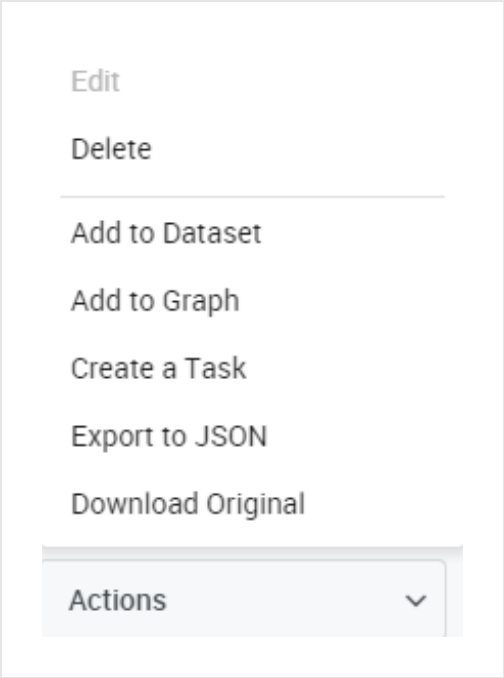
The tasks are sorted in ascending order by task name. Besides the name, you can see the task progress status, who is assigned to take care of it, and the deadline within which it needs to be completed.

To change the sort order, based on task name, status, assigned person or deadline, click the corresponding column headers.

Manipulate the entity

You can edit and modify entity data by choosing the desired option from the **Actions** pop-up menu on the bottom half of each entity detail pane tab. For example you can load it onto the graph to further examine it in a visual environment, you can create a task to take action and respond to the entity, or you can download it.

Grayed-out menu options are disabled in the current context, and are not available.



Select this menu option…	**…to carry out this action**
Edit	Makes the entity detail pane fields writable, so that you can edit and modify the entity.
Delete	Permanently removes the entity from the platform. If you confirm the entity deletion in the pop-up confirmation dialog, the operation is completed, and it cannot be undone.
Add to dataset	Adds the entity to the specified dataset(s).
Add to graph	Adds the selected entity to the graph to further examine it in a visual environment.
Create a Task	Makes the entity data actionable by creating a task, and by assigning it to a user.

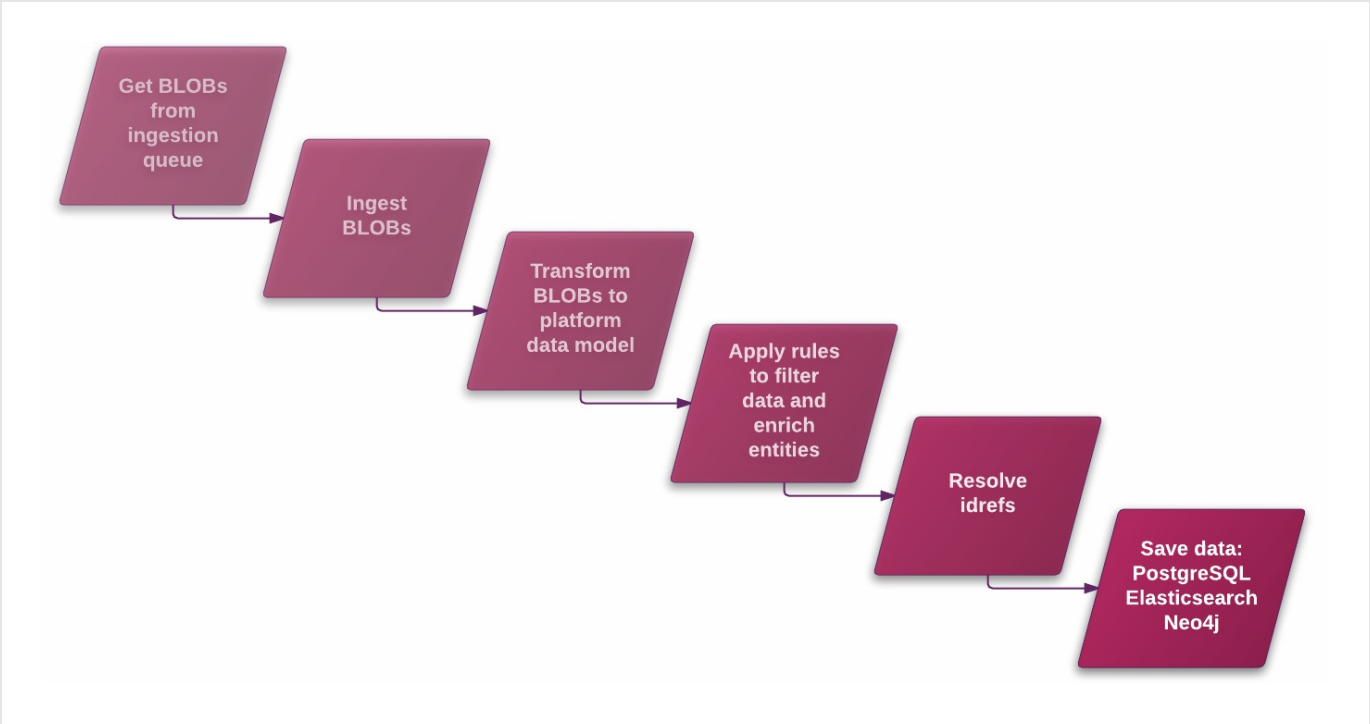
Select this menu option…	…to carry out this action
Export to JSON	Exports the entity as a JSON object that can be saved, for example locally.
Download Original	Downloads the selected entity in its original data format.

How to enrich entities with extracts

Summary: Enrichment extracts augment the quality of the intelligence you obtain from cyber data analysis. Enrich entities and integrate entity extracts with additional raw data to access a broader context and gain deeper insight into threat scenarios.

Ingestion

Data ingestion into the platform is a multi-step process:



- Incoming data flows in and it is added to the ingestion queue.
- BLOBs are fetched from the ingestion queue to be processed.
- BLOBs are processed:
  - Data is deduplicated;
  - Data is normalized;
  - Data is transformed to the platform internal data model:
    - Entities
    - Observables
    - Relationships.
- Rules and filters enrich entities, create relationships, flag and tag entities, and so on.

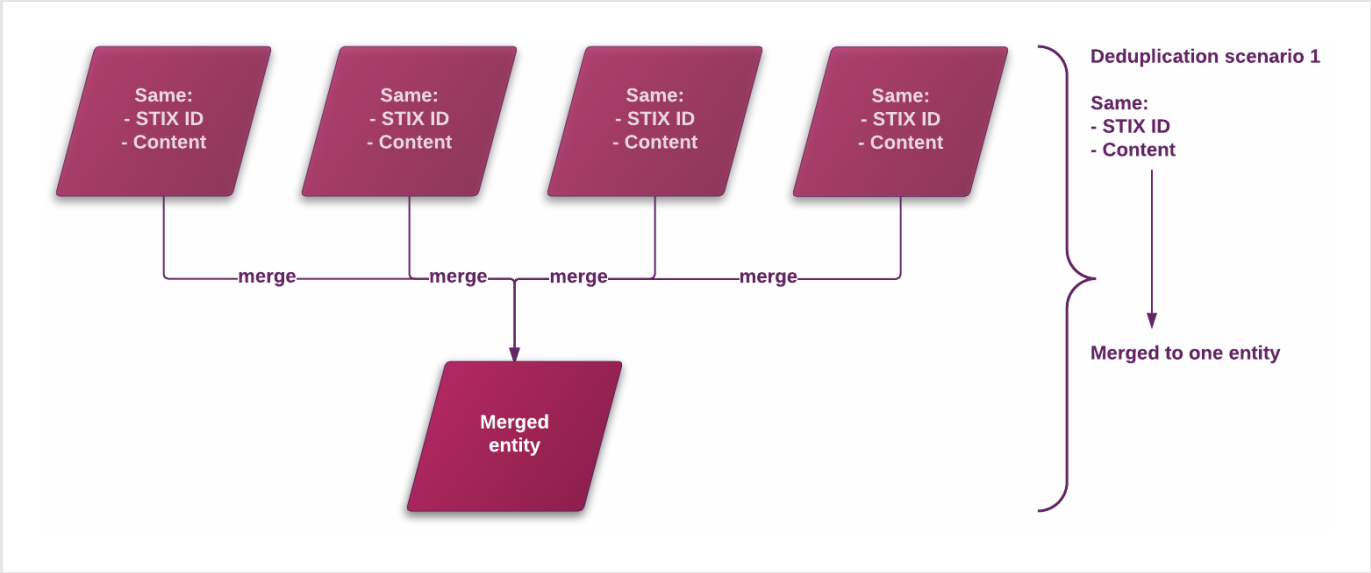
- The platform tries to resolve ID references by looking for the data the IDs refer to.
- The ingested entities are saved to the databases.

Deduplication

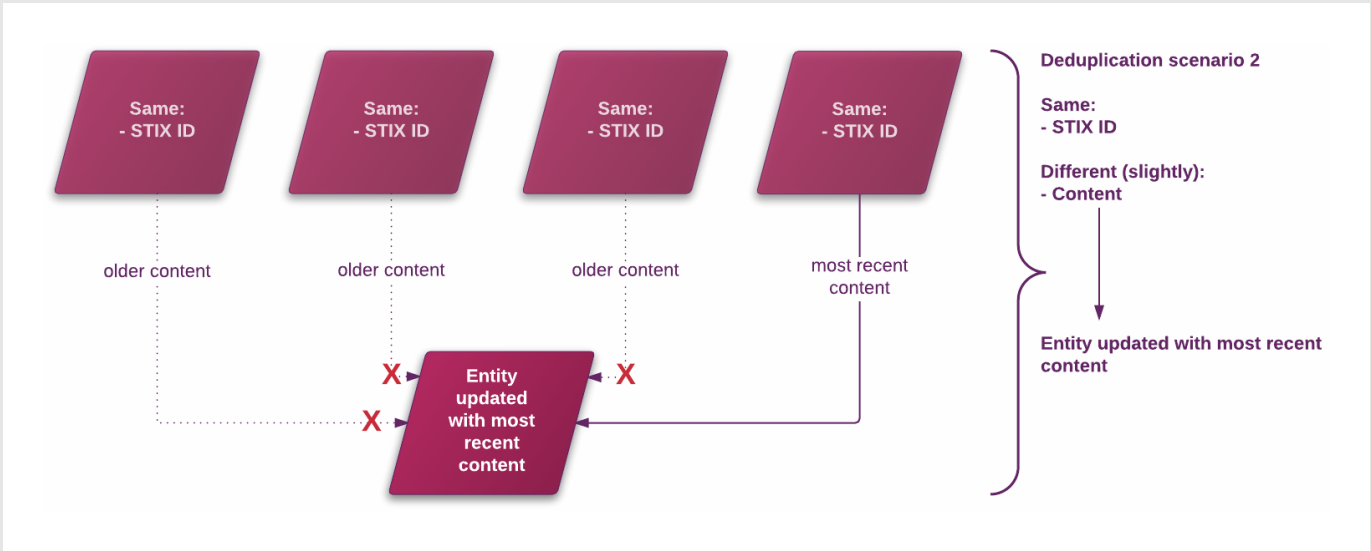
When checking newly ingested entities vs. existing ones, entity-level deduplication handles the following scenarios:

- Multiple entities sharing the *same STIX ID* and having *identical content* are handled like identical copies.
  In this case, identical copies of the same entity are merged to one entity.
- Multiple entities sharing the *same STIX ID* and having *slightly different content* are handled like versions of the same entity.
  In this case, the most recent content is used to update the existing entity. This avoids creating redundant copies of the same entity in the system.
- Multiple entities sharing the *same STIX ID* and having *identical content*, but *different timestamps* are handled like chronological versions of the same entity.
  In this case, the existing entity timestamp is updated to the most recent value without creating a new version of the entity.

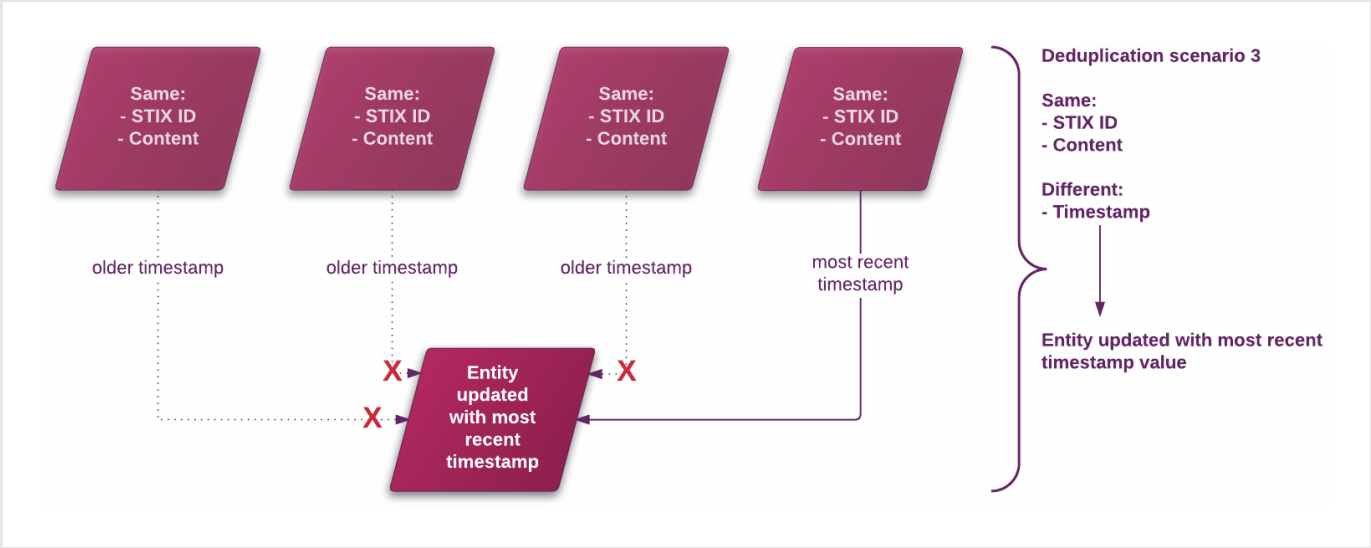
Before deduplication	**After deduplication**
Multiple entities	They are merged to one entity
Same STIX ID	
Identical content	
(identical copies)	



Before deduplication	After deduplication
Multiple entities	The existing entity is updated with the most recent content
Same STIX ID	
Slightly different content	
(versions of the same entity)	



Before deduplication	After deduplication
Multiple entities	The existing entity timestamp is updated to the most recent value
Same STIX ID	
Identical content	
Different timestamps	
(chronological versions of the same entity)	



Filtering and enriching

Data extraction produces observables from data retrieved in embedded CybOX objects. This process contribute to data fusion across the whole platform dataset.

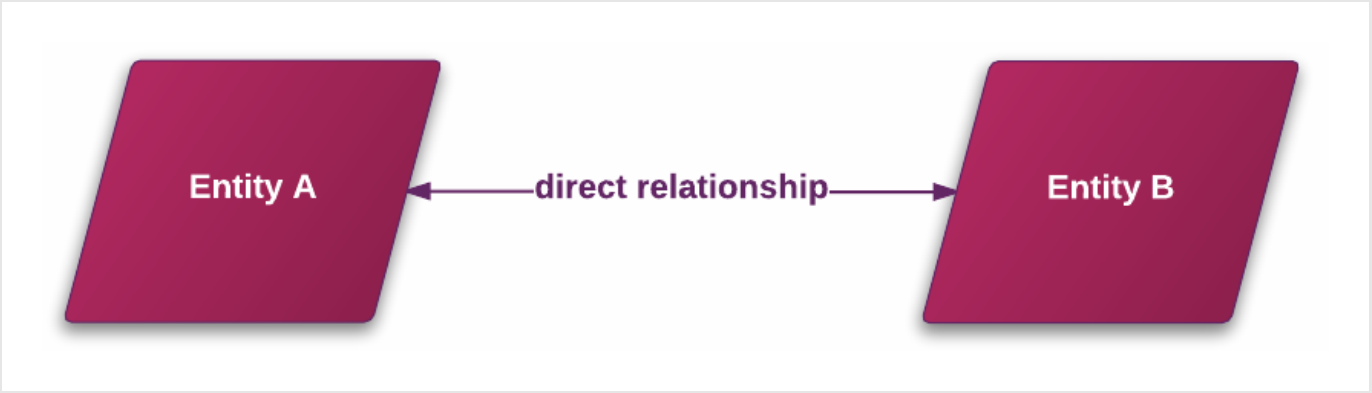
- Enrichment rules sift through data to augment it with enrichment extracts, that is, observables, obtained from the available enrichment sources, and to exclude specific data based on the defined filtering rules.
- You can further refine the quality of the extracted data by applying ad-hoc rules to classify extracts as safe or malicious.

STIX and CyBOX objects can both include placeholder references to external objects and data in the `idref` field.
By default, the platform attempts to resolve `idref`s at entity level and at nested object level by looking for the data matching the `idref` value.
If it finds matching data, it creates either a relationship between entities — entity-level, STIX `idref` resolution — or it replaces the `idref` placeholder value with the corresponding actual data — nested object-level, CybOX `idref` resolution.

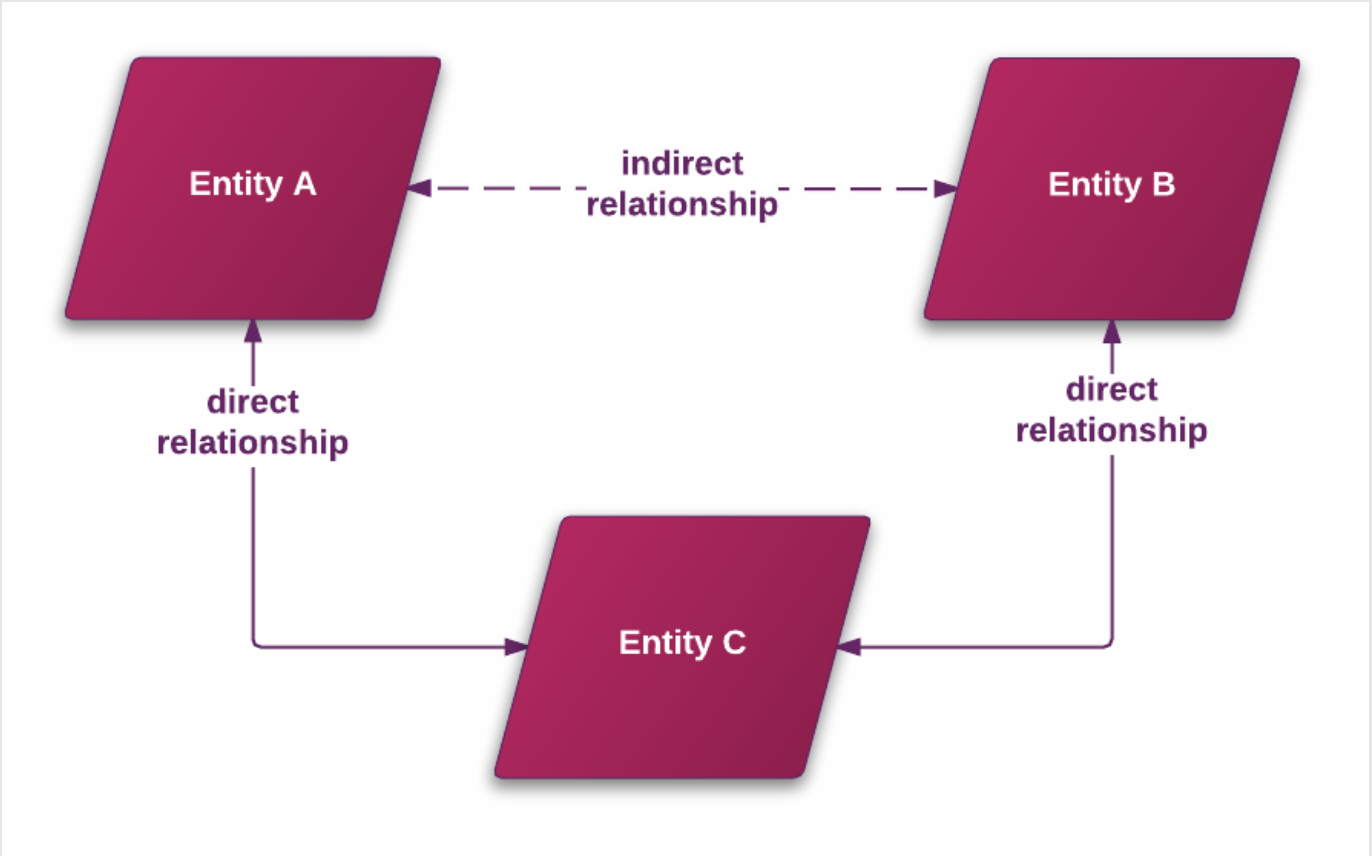
idref resolution — Entity level

Entities and observables can reference external data through the STIX and CybOX`idref` field.
When the `idref --> id` reference is at entity level (STIX), the platform creates an entity relationship between the entities.

- When a STIX `idref` directly references an entity, the platform creates a direct relationship to associate the two entities.



- When a STIX `idref` indirectly references an entity, for example by establishing a relationship with the target entity through a connecting entity or an observable, the platform creates an indirect relationship to associate the two entities.

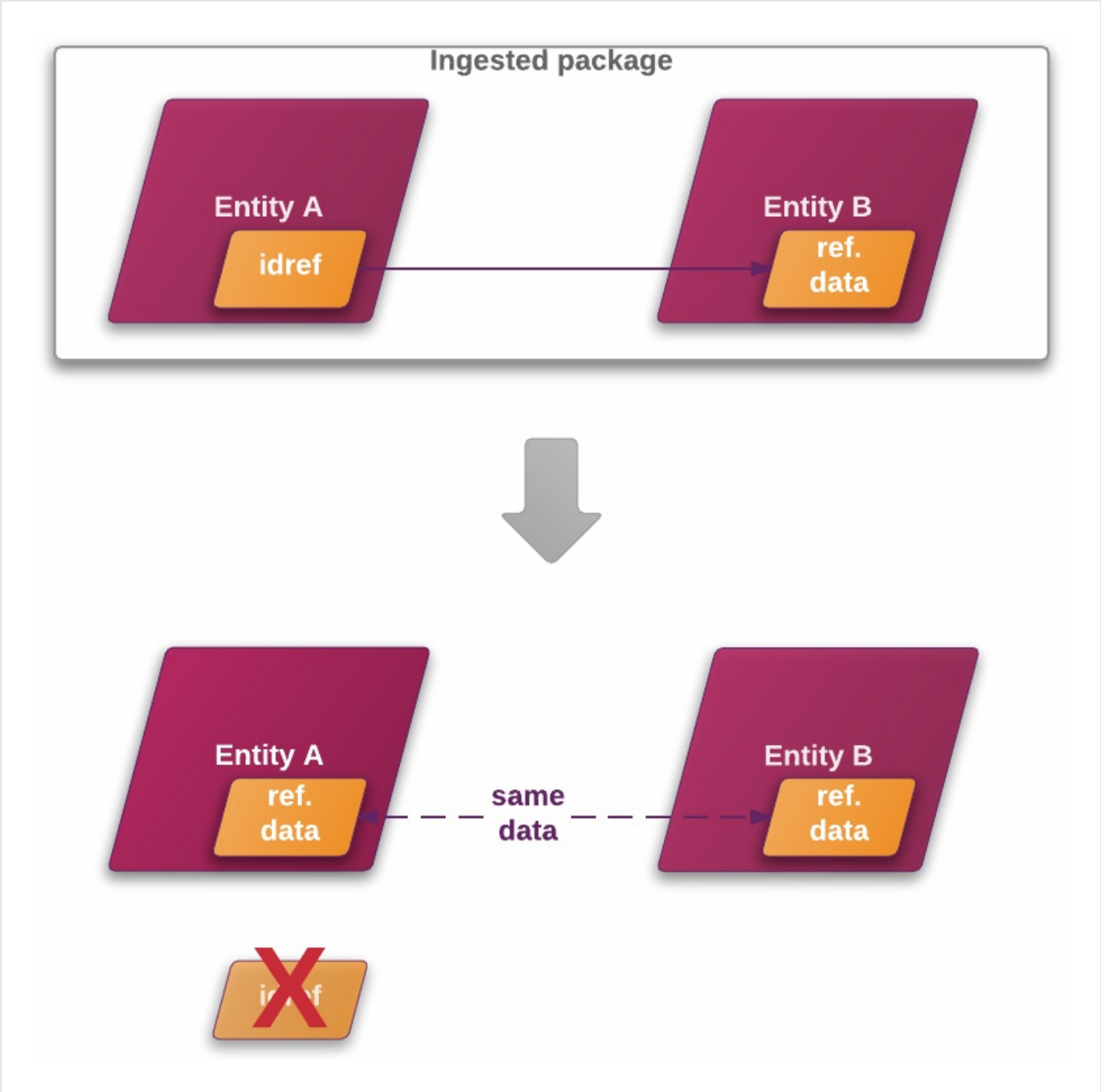


- When a STIX `idref` references an entity of the same type and content as the entity it belongs to, the only differences being either the timestamp, or the version reference values between the two entities, the platform processes the entity with the more recent timestamp or with the higher version value as an update of the other entity.

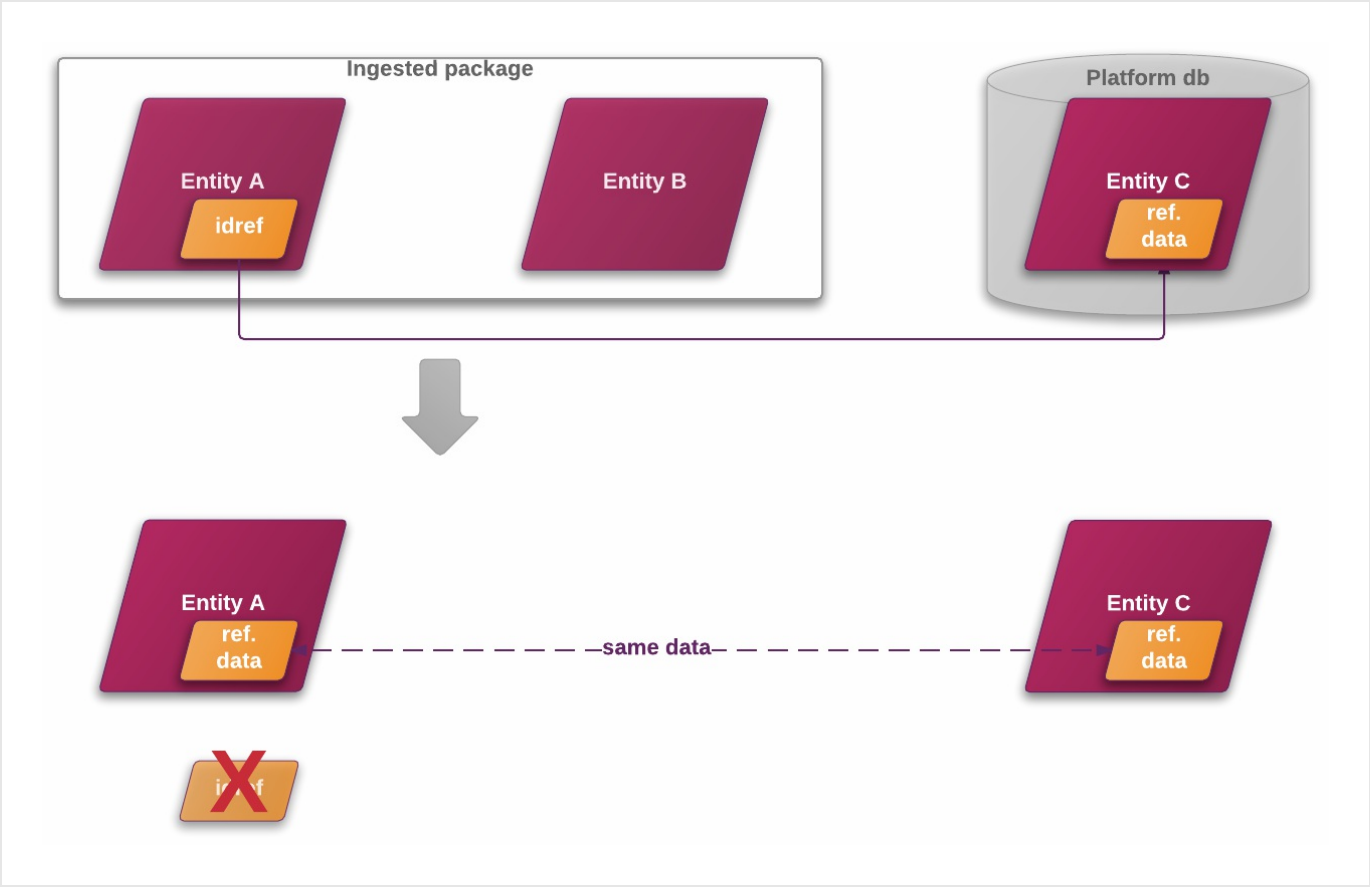
idref resolution — Nested objects

When the `idref --> id` reference is at nested object level (CybOX), that is, when an entity includes an embedded CybOX observable object with an `idref`, the platform attempts to resolve the CybOX `idref` by looking for the referenced data:

- If the CybOX `idref` references data available inside the same ingested package or data stored in the platform database, the newly ingested `idref` is removed and it is replaced with the existing referenced data.



Referenced data is inside the same ingested package



Referenced data is outside the same ingested package, but available in the platform database

- If the `idref` references unavailable data at the moment, the reference is resolved if/when the referenced data becomes available:
  - The `idref` is converted to an empty placeholder TTP or indicator entity that is populated when/if the originally referenced data becomes available.
  - The empty placeholder entity is indexed; therefore, it is searchable, and it can be loaded onto the graph.

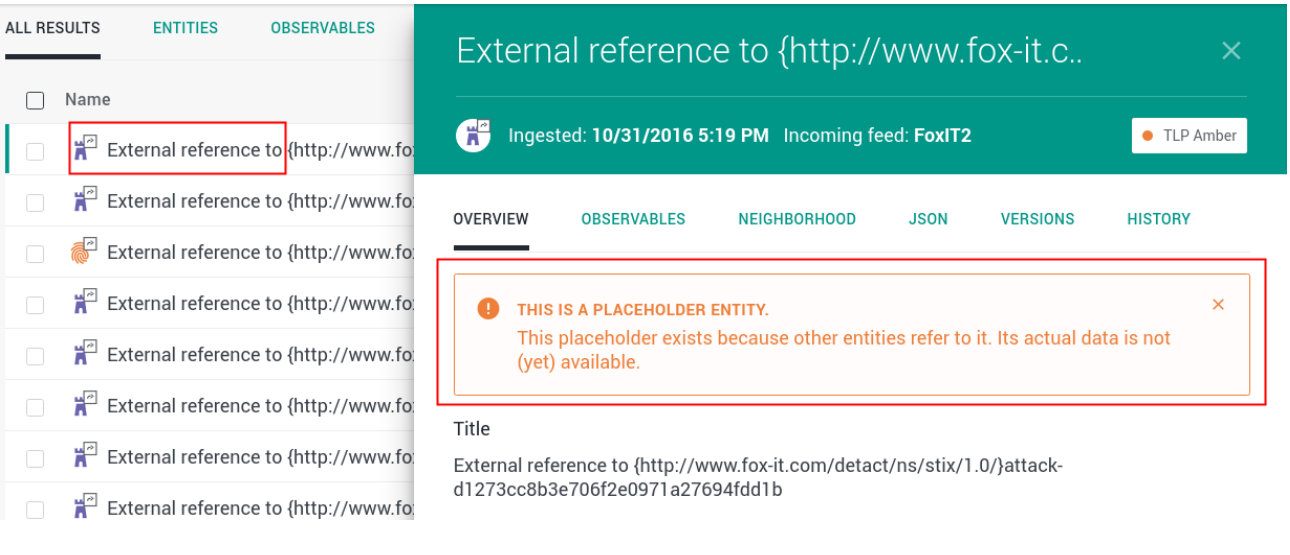
Referenced data is unavailable, an empty placeholder entity points to it in case it becomes available in the future

The process works identically in the opposite direction: if CybOX data is ingested, the platform looks for an existing `idref` pointing to it.

- If it finds a matching `idref`, the existing `idref` is removed and it is replaced with the newly ingested referenced data.
- If no matching `idref` is available, nothing happens. If a matching `idref` becomes available at a later time, it will be resolved then by replacing it with the corresponding referenced data.

Example of an empty placeholder entity

This is an empty TTP placeholder, as shown in the entity detail pane:



This is the corresponding JSON representation:

```
{
  "entities": [

    ...

      "data": {
        "id": "{http://www.example.com/stix/1.0/}attack-
d1273cc8b3e108h2e0971a27694fdd1e",
        "type": "ttp",
        "title": "External reference to {http://www.example.com/stix/1.0/}attack-
d1273cc8b3e108h2e0971a27694fdd1e"
      }
    }
  ],

    ...

}
```

Data saving

Last but not least, the platform saves the ingested entities to the databases in the following order:

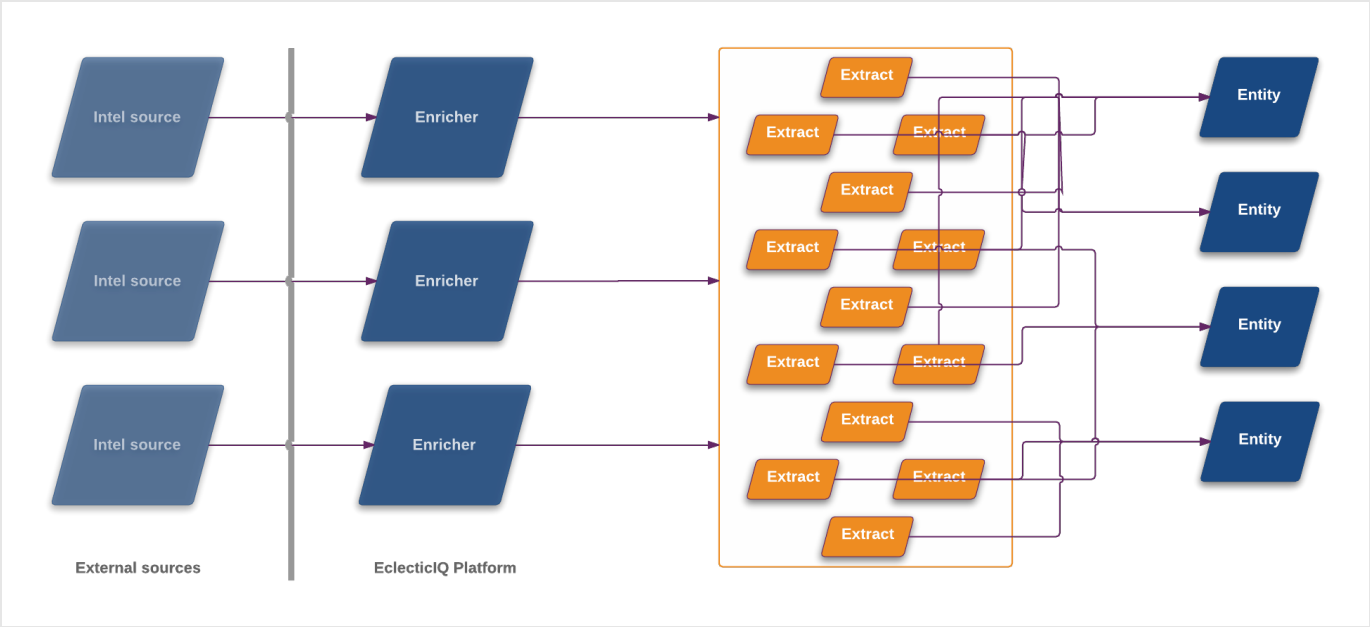
- Entity store (PostgreSQL)
- Search store (Elasticsearch)
- Graph store (Neo4j)

Enriching entities via extracts

The platform can ingest cyber threat intelligence through incoming feeds, by manually uploading one or more files, or by creating an entity in the entity editor.

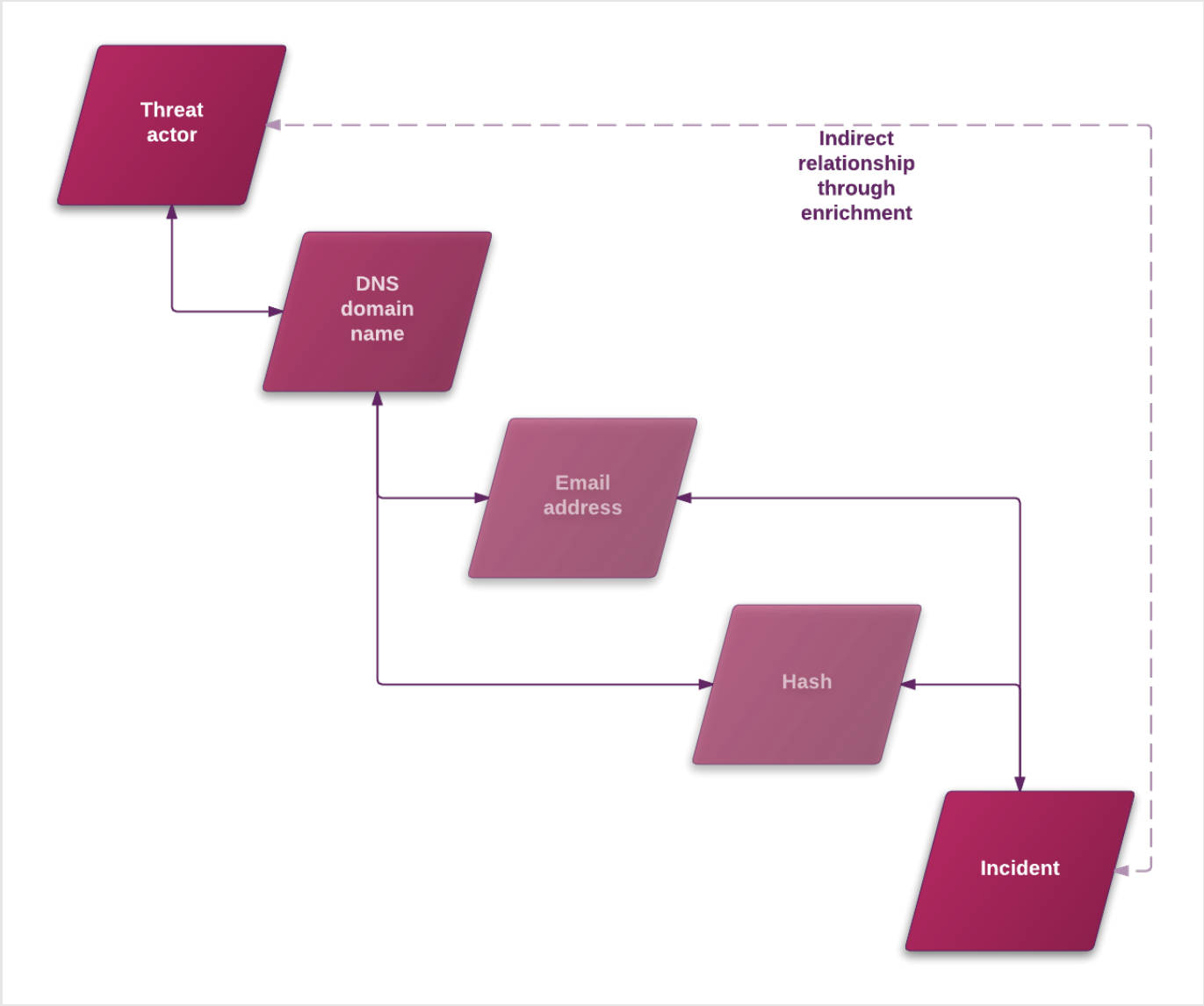
After ingesting and saving entities to the database, you can integrate the existing information with additional details. The extra information is raw data that augments the entity intelligence value by adding more context and meaning to it. The data is extracted from different sources such as feeds, reports, database searches, curated intel distribution lists, and so on.

The platform uses enrichers to fetch and extract the data. Enricher rules sift through the data to link it to relevant entities as enrichment observables.
This process does not alter core entity data: each bit of enriching information is saved to observables, which are related to entities.

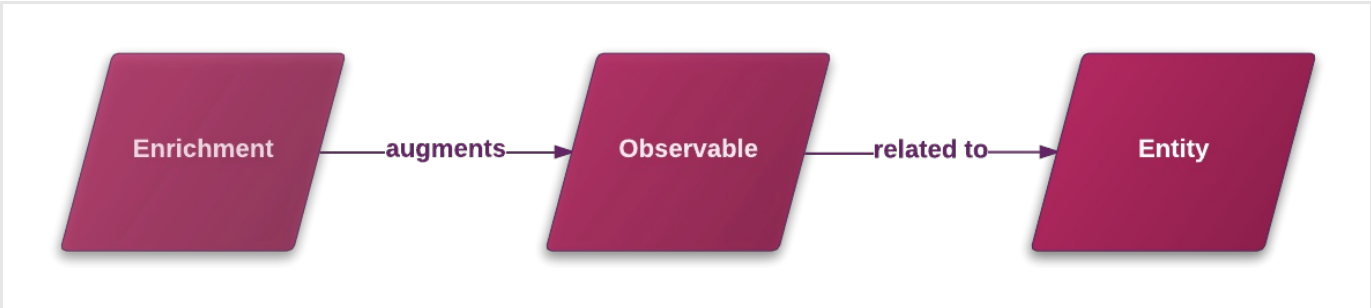


For example, let's assume a scenario where an analyst is investigating a threat actor entity. The entity includes some observables, and one of them is a DNS domain name.
The analyst looks up the domain name by running it through a whois service. The lookup results include an email address.
During the investigation, the analyst retrieves also a file hash related to the domain name. An examination reveals that the hash is related to an incident. Information about the incident includes the same email address detail the DNS domain name returned.
There is an indirect relationship between the threat actor and the incident that would not have been noticeable without extra context, which in this example is provided by the hash.

Enrichments help get a broader and sharper picture: by adding meaningful context, they help discover broader, indirect relationships that are not immediately visible.



Enrichments augment extracts with raw data information related to entities:



Extracts from data URI and raw artifacts

The extracted entity data that is stored inside observables ranges from short, simple data such as email addresses, domain names, IP addresses, and so on, to binary data. When an entity contains binary data, the data can be included as either a *data URI* or a *CybOX raw artifact* element.

During ingestion, extraction logic handles binary data URI and raw artifact objects embedded in CybOX objects in the following way:

- **data URIs** (`https://en.wikipedia.org/wikidata_uri_scheme`) are extracted and stored as entity attachments and new hash values:
  - The data URI value is recalculated to a new hash: `uri-hash-sha256`.
    The SHA-256 hash value for `uri-hash-sha256` is calculated over the UTF-8 encoding of the data URI string.
    The `uri-hash-sha256` hash substitute allows for entity correlation among entities containing the same data URI.
  - The binary data/raw content embedded in the data URI is decoded and processed:
    - The extracted binary data content is stored as an entity attachment similar to the CybOX `Raw_Artifact` object.
    - The extracted content is hashed using SHA-512, SHA-256, SHA-1, and MD5.
      Each resulting hash is added to the relevant entities as an observable.

Example

A data URI with image content nested inside a CybOX object generates the following output:

- 1 `uri-hash-sha256` hash to facilitate entity correlation
- 4 calculated hash observables: `hash-sha512`, `hash-sha256`, `hash-sha1`, and `hash-md5`
- 1 embedded JSON entity attachment (`raw-artifact`) with the extracted binary data

The following example shows a sample input along with the corresponding output.

```
dataUriExtractionSample(

        input={
            
        },

        output={

            # Recalculated hash of the original URI:
            ('uri-hash-sha256:'
             'd16ae5d51dda6f58995171aa23c0fa5e'
             '6dcd9c777cf9c251c4be3b1d62fdf670'),

            # Multiple hashes of the decoded content:
            'hash-md5:3eacd0132310ea44cad756b378a3bc07',

            'hash-sha1:e2216a7e9b73f5cb0279351c78ce61c33475cea7',

            ('hash-sha256:'
             'bb229a48bee31f5d54ca12dc9bd960c6'
             '3a671f0d4be86a054c1d324a44499d96'),

            ('hash-sha512:'
             'bd9ab35dde3a5242b04c159187732e13'
             'b0a6da50ddcff7015dfb78cdd68743e1'
             '91eaf5cddedd49bef7d2d5a642c21727'
             '2a40e5ba603fe24ca676a53f8c417c5d'),

            # (Attachment) Raw artifact as embedded JSON with the content:
            ('raw-artifact:{"content": '
             '"R0lGODlhAQABAAAAACH5BAEKAAEALAAAAAABAAEAAAICTAEAOw==", '
             '"content_encoding": "base64", "type": "image/png"}'),
        }),
```

Enrichers

Enrichers poll external data sources to provide additional context and detail to augment — hence, enrich — the intelligence value of the entities stored in the platform.

The platform ships with several built-in, ready-to-use enrichers to obtain geolocation IP and whois details, DNS domain and malware information, as well as other relevant data to help analysts draw a sharper and more comprehensive picture of the cyber threat relationships and the cyber threat scenarios under investigation.

Enrichers automatically augment all the entities that accept the enricher's content type as an observable. In other words, the observable types an entity supports define the applicable enrichers an entity can use.

Enricher types

Enricher	API endpoint	I
Elasticsearch sightings	`http://<elasticsearch_url>:9200/<schema_resource>`	Searches an external Elasticsea criteria are processed to automa
Fox-IT InTELL Portal	`https://cybercrime-portal.fox-it.com/`	Based on Fox-IT InTELL, the pc a range of sources like forums a suspicious activity.
Intel 471	`https://api.intel471.com/v1/`	Besides data on compromised I Intel 471 focuses on providing fi and groups.
OpenDNS OpenResolve	`http://api.openresolve.com/{}/{}`	OpenResolve by OpenDNS offe to retrieve reverse-DNS lookup
PyDat	`http://10.0.1.60:8000/ (example)`	**PyDat** (`https://github.com/` locally, and it can work together (`https://github.com/mitrec with-elasticsearch`) to provi passive DNS lookup informatior organization, country, city, stree
RIPEstat GeoIP	`https://stat.ripe.net/data/geoloc/data.json?resource={IP_address}`	Geolocation IP information from **API** (`https://stat.ripe.net` longitude, country, and city.
RIPEstat Whois	`https://stat.ripe.net/data/whois/data.json?resource={IP_address}`	Whois information from the RIPI **API** (`https://github.com/ri` including inet number, name, or telephone.
Cisco AMP Threat Grid	`https://panacea.threatgrid.com/api/v2/`	Polls data from the Cisco AMP T a range of cyber threat data like network streams, and hash files
VirusTotal	`https://www.virustotal.com/vtapi/v2/{}`	Polls data from the VirusTotal A domains (passive DNS) and IP a against 60+ antimalware produc additional metadata information

Enricher	API endpoint	I
Flashpoint AggregINT	`https://endlesstunnel.info/v3`	Polls data from the Flashpoint A hosts, domains, IP addresses, a thematic datasets focusing on h groups, communities in conflict, **CBRN** (`https://en.wikiped` produces enrichment observabl user name of the author of a po UTC date and time of a post in **I** (`https://en.wikipedia.org/` (`https://tools.ietf.org/ht`
Flashpoint Blueprint	`https://endlesstunnel.info/v3`	Polls data from the Flashpoint A geolocation and IP ranges, as w search thematic datasets focusi supremacist groups, state actor (`https://en.wikipedia.org/` enrichment observables like city latitude/longitude or IP address a hit, user name uniquely match
Flashpoint Thresher	`https://endlesstunnel.info/v3`	Polls data from the Flashpoint A datasets focusing on hackers, te **CBRN** (`https://en.wikiped` produces enrichment observabl
PassiveTotal Whois	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTot** (`https://api.passivetotal.` `getv2whoisquery`). It provides associated with an IP address o details. Analysts can retrieve re telephone, and email details. Th queries to obtain, for example, r same individual or the same cor
PassiveTotal Passive DNS	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTot** (`https://api.passivetotal.` `getv2dnspassivequery`). It pro cross-referencing IP addresses over time. Analysts can examine IP addresses over time. They ca more domain names that may b

Enricher	API endpoint	I
PassiveTotal IP/Domain	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTot** (`https://api.passivetotal.` `getv2enrichmentquery`). It pro queried IP address or domain n name, any sub-domains, inet de **(ASN)** (`https://en.wikipedia.org/` as well as geolocation informati look for further connections that
PassiveTotal Malware	`https://api.passivetotal.org/v2`	Polls data from the **PassiveTot** (`https://api.passivetotal.` `getv2enrichmentmalwarequer` to the queried host or domain, s sha1, hash-sha256, hash-sha51 malware entries are also tagge `enrichment_extracts.meta.c` the value you set under **Rules > malicious**; `enrichment_extra` corresponds to the value you se **Confidence > Malicious - Low**
Splunk sightings	`http://10.0.1.22:8089/ (example)`	Based on the search queries de matching data in the specified S and saved to the platform as sig

Enricher input

The overview shows the supported observable data types you can use as input for the enrichers.
These are the value types the *enrichment_extracts.kind* search query field returns.

Enricher	Supported kinds (observable types)
Elasticsearch sightings	ipv4, ipv6, domain, host, uri, hash-md5, hash-sha1, hash-sha256, hash-sha512
Fox-IT InTELL Portal	ipv4, ipv6, domain, host, uri, hash-md5, hash-sha1, hash-sha256
Intel 471	ipv4, ipv6, domain, host, uri, email, actor-id, hash-md5, hash-sha256
OpenDNS OpenResolve	ipv4, ipv6, domain, host
PyDat	ipv4, ipv6, domain
RIPEstat GeoIP	ipv4, ipv6

Enricher	Supported kinds (observable types)
RIPEstat Whois	ipv4, ipv6
Cisco AMP Threat Grid	ipv4, ipv6, domain, host, uri, hash-md5, hash-sha1, hash-sha256, hash-sha512, winregistry
VirusTotal	ipv4, ipv6, domain, uri, hash-md5, hash-sha1, hash-sha256
Flashpoint AggregINT	ipv4, ipv6, domain, host, uri, email, actor-id, hash-md5, hash-sha1, hash-sha256, hash-sha512
Flashpoint Blueprint	ipv4, ipv6, domain, host, uri, email, actor-id, hash-md5, hash-sha1, hash-sha256, hash-sha512
Flashpoint Thresher	ipv4, domain, host, uri, hash-sha1, file
PassiveTotal Whois	ipv4, ipv6, domain, host
PassiveTotal Passive DNS	ipv4, ipv6, domain, host
PassiveTotal IP/Domain	ipv4, ipv6, domain, host
PassiveTotal Malware	domain, host
Splunk sightings	domain, email, hash-md5, hash-sha1, hash-sha256, hash-sha256, host, ipv4, ipv6, uri

Enricher output

Enrichers automatically augment all the entities that accept the enricher's content type as an observable. In other words, the observable types an entity supports define the applicable enrichers an entity can use.

The overview describes the output each enricher generates. The resulting enrichment observables are associated to the entities they bear relationships to.

Enricher	Generated output
Elasticsearch sightings	Creates sightings from matching results returned from a search in an external Elasticsearch instance.
Fox-IT InTELL Portal	Enriches observables with relevant contextual information from forums, chats, and IRC channels.
Intel 471	Enriches observables with data focusing on threat actor information.

Enricher	Generated output
OpenDNS OpenResolve	Enriches observables with reverse-DNS lookup information.
PyDat	Enriches observables with whois data, current IP resolution and passive DNS information.
RIPEstat GeoIP	Enriches observables with geolocation information related to IP addresses: coordinates, country, and city.
RIPEstat Whois	Enriches observables with whois information related to IP addresses.
Cisco AMP Threat Grid	Enriches observables with information like IP addresses, domains, host names, hashes, and Windows registry keys.
VirusTotal	Enriches observables with information like IP addresses, domains, and hash files.
Flashpoint AggregINT	Enriches observables with information like IP addresses, domains, host names, and hash files.
Flashpoint Blueprint	Enriches observables with information like IP addresses, domains, host names, and URLs.
Flashpoint Thresher	Enriches observables with information like IP addresses, domains, URLs, hashes, and files.
PassiveTotal Whois	Enriches observables with **whois** `(https://www.riskiq.com/products/learn-threat-research-and-analysis/)` information.
PassiveTotal Passive DNS	Enriches observables with **passive DNS** `(https://www.riskiq.com/products/learn-threat-research-and-analysis/)` information.
PassiveTotal IP/Domain	Enriches observables with **enrichment** `(https://passivetotal.readthedocs.io/en/latest/api.html#enrichment-request)` information.
PassiveTotal Malware	Enriches observables with **malware enrichment** `(https://passivetotal.readthedocs.io/en/latest/api.html?highlight=malware#enrichment-request)` information.
Splunk sightings	Creates sightings for matching input observables, based on the search result items retrieved in the specified Splunk instance.

Enrich entities

You can enrich entities in the following ways:

- Automatically, or
- Manually.

Enrichment rules and enrichment tasks drive the enrichment process to:

- Poll selected and trustworthy intelligence data sources;
- Retrieve relevant, accurate, and reliable data to augment platform entities with additional bits of information that provide additional context.

Rules

Enrichment rules define what to do with the retrieved enrichment data.
Rules act like filters, and they set the logical constraints defining:

- The platform data sources to augment with the enrichment information. Data sources can be incoming feeds, as well as other enrichers.
- Within the selected platform data sources, the entity type(s) to augment with the enrichment information.
- The enrichers to use to fetch the enrichment data.

Tasks

Enrichment tasks define process execution by setting the following options:

- The data fetching mechanism; for example, an API endpoint exposing the enrichment data service.
- Specific data sources; for example, datasets targeting threat actors like hackers and terrorist groups.
- Data rate limit and monthly execution cap values to control the amount of polled data.
- A source reliability flag for the incoming enrichment data to simplify assessing the quality of the retrieved data.

Automatically enrich entities

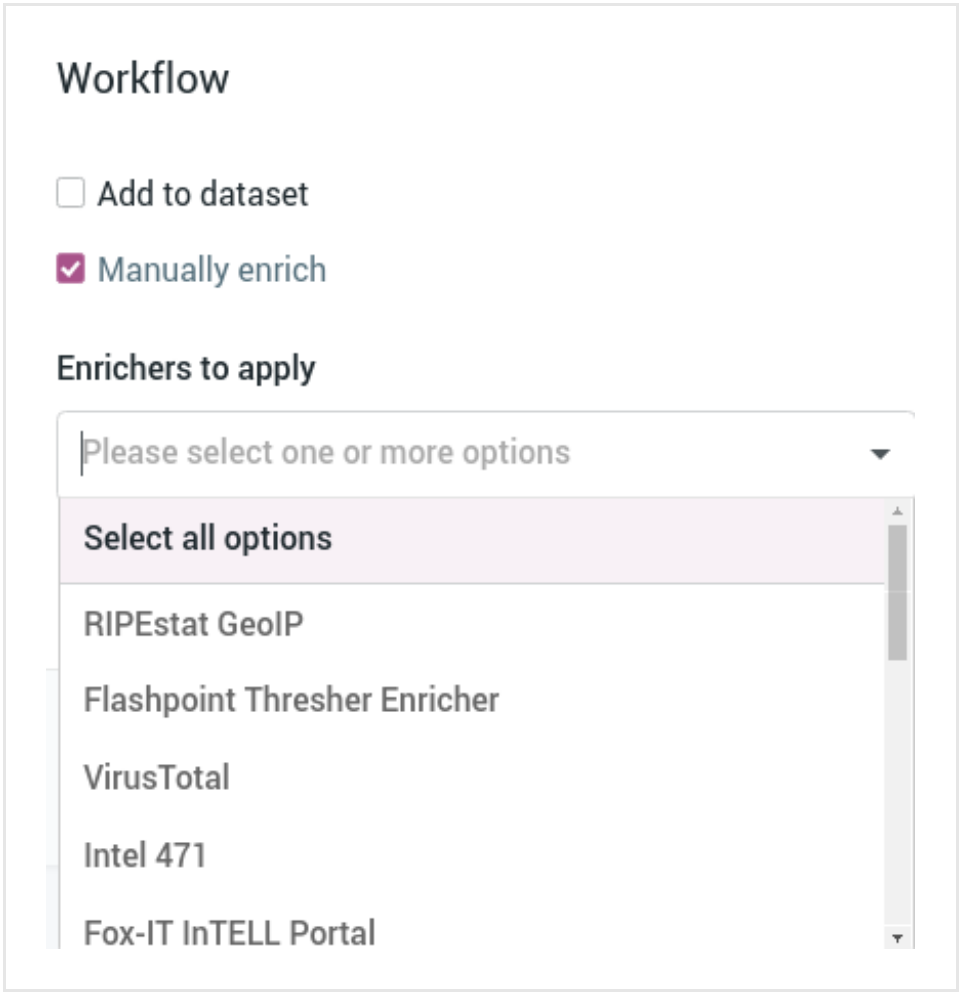
To automatically enrich entities, make sure enricher tasks are active, and the necessary enrichment rules are configured.

Rules give you control over the type of information you want to retrieve or exclude, and what you want to do with it. You can assign one or more enricher sources to specific extract types. You can set multiple filters to cover usage scenarios as needed. You can then examine the returned enrichment extract data, as well as route it to other devices that enforce cyber threat detection or prevention.

Manually enrich entities

To adjust enrichment behavior to manually apply it to the entities you want to enrich, do the following:

- Open an entity in editing mode.
  For example, on the top navigation bar click **Browse > Published** to display an overview of the published entities available in the platform.
- On the row corresponding to the entity you want to manually enrich, click the dotted menu icon to display the context menu.
- From the drop-down menu select **Edit**.
- At the bottom of the entity editor page, select the **Manually enrich** checkbox.
  A new input field with a drop-down menu becomes available.
- From the drop-down menu select one or more enrichers you want to apply to the entity.



- When you are done, click **Save draft** to store your changes without publishing the entity, **Publish** to release the new version of the entity including your changes, or **Cancel** to discard the changes.

Alternatively, you can manually enrich an entity by selecting it; for example, from a dataset, from **Browse** or from **Discovery**.
An overlay slides in from the side of the screen to display the entity detail pane.

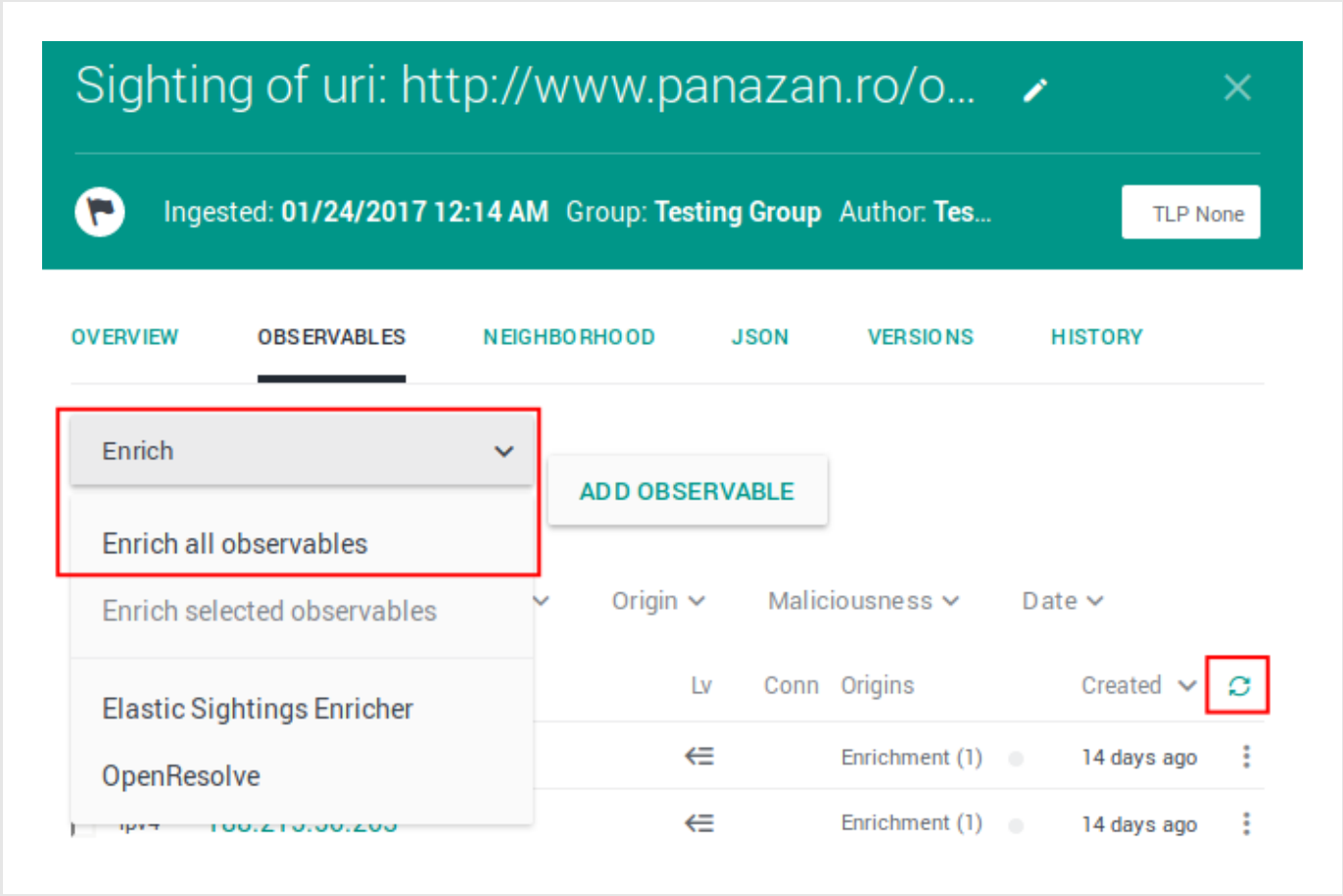
- On the entity detail pane, click **Observables**.
- The **Observables** tab shows an overview of the enrichment observables the entity has been augmented with.

To manually enrich the entity observables:

- Click the ↻ refresh icon to trigger a task run that polls all the enrichers configured for the entity.

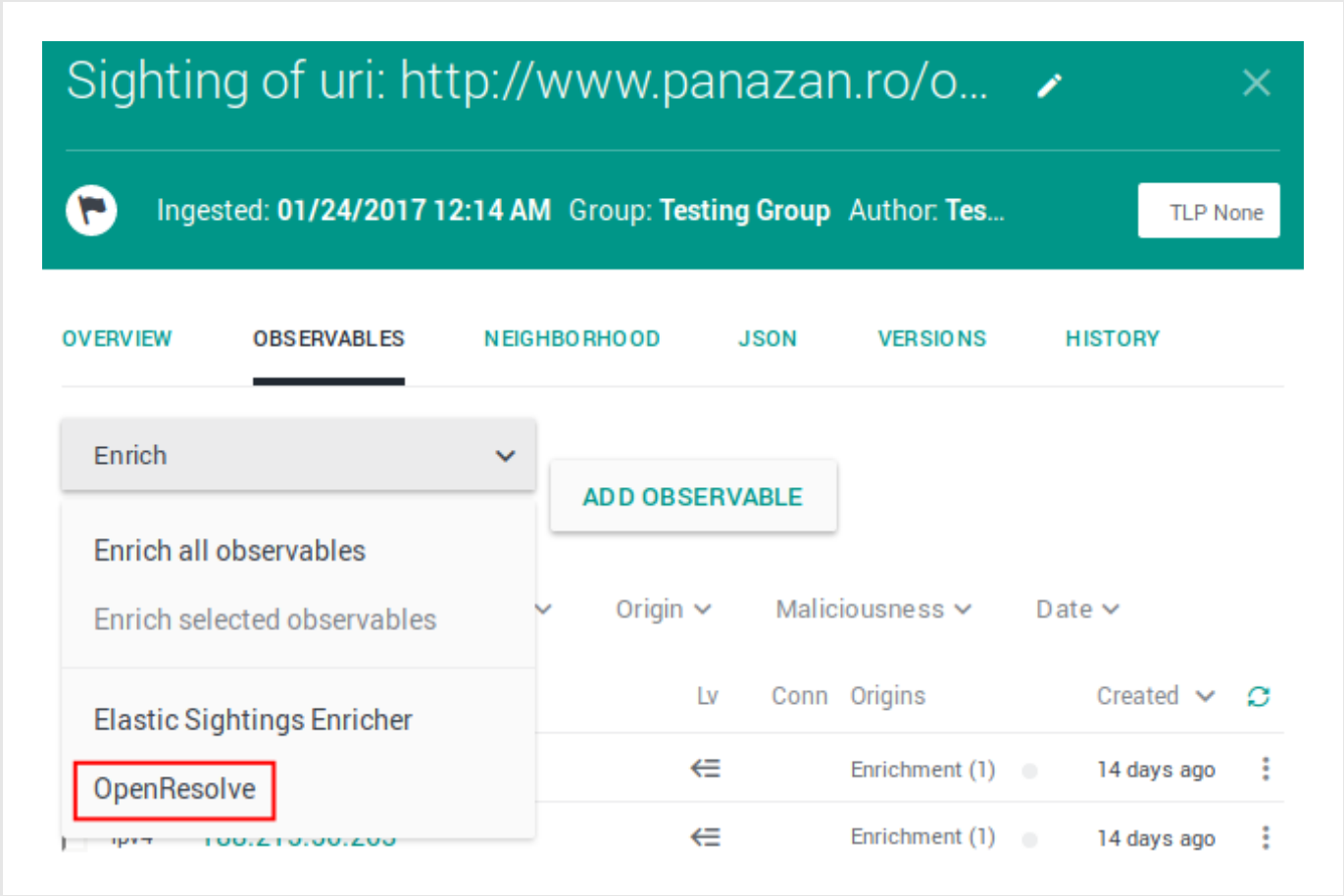
Alternatively:

- From the **Enrich** drop-down menu, select **Enrich all observables**.
- The platform polls all applicable enrichers for the entity, and it enriches all the entity observables with the retrieved data.



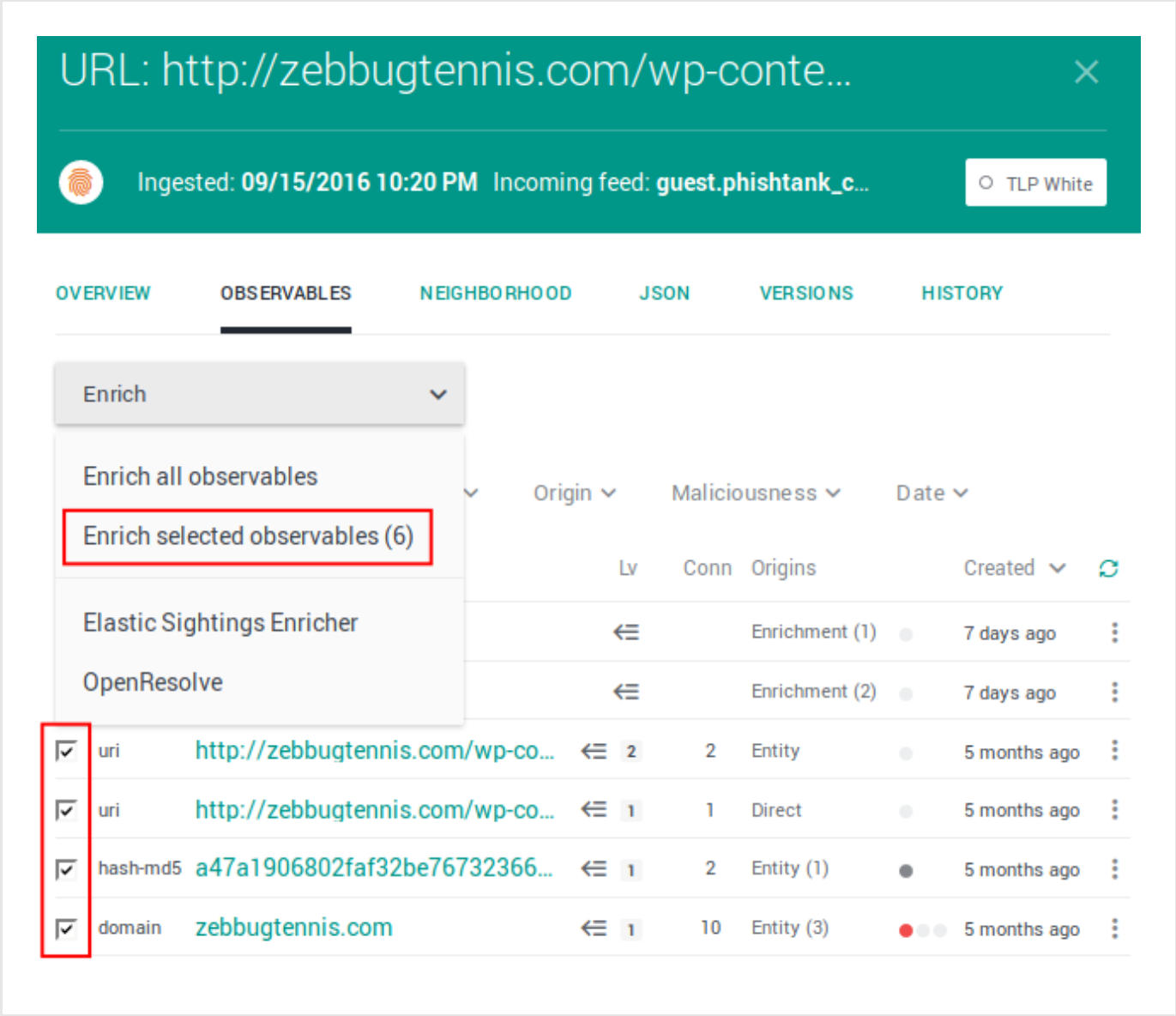
To poll a specific enricher:

- Select it from the **Enrich** drop-down menu, and then click it.
- The platform polls the specified enricher for the entity, and it enriches all the entity observables with the retrieved data.



To enrich only specific observables:

- On the **Observables** tab, select the checkboxes corresponding to the observables you want to enrich.
- From the **Enrich** drop-down menu, select **Enrich selected observables**.
- The platform polls all applicable enrichers for the entity, and it enriches the selected entity observables with the retrieved data.



The available enricher tasks in the drop-down menu are automatically filtered to show only the applicable enrichers for the entity.
Enrichers automatically augment all the entities that accept the enricher's content type as an observable. In other words, the observable types an entity supports define the applicable enrichers an entity can use.

Work with enricher rules

View enricher rules

To view enricher rules, do the following:

- On the top navigation bar click **+ > Rules > Enrichment**

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button next to the user avatar image.
- From the drop-down menu select **Rules**.
- On the left-hand navigation sidebar click **Enrichment**.
- The default view is **Rules > Enrichment**. It shows an overview of the configured enricher rules.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.
- To view the details of a specific rule, click an area on the row corresponding to the rule you want to examine. An overlay slides in from the side of the screen to display the rule detail pane.

Add enricher rules

To add a new enricher rule, do the following:

- On the top navigation bar click **✚ > Rules > Enrichment**

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button next to the user avatar image.
- From the drop-down menu select **Rules**.
- On the left-hand navigation sidebar click **Enrichment**.
- The default view is **Rules > Enrichment**. It shows an overview of the configured enricher rules.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.
- Click the **+ Rule** button.

> ✔ On the forms, input fields marked with an asterisk are required.

On the **Rules > Enrichment > Create** page, fill out the fields to create the new enricher rule:

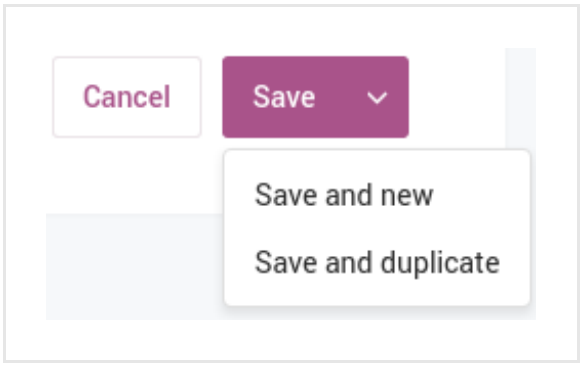
- **Name**: define a name to identify the rule. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.
- Click **✚ add** or **✚ more** to add a filtering option.
- **Source**: from the drop-down menu select the incoming feed or the enricher whose observables you want to augment with additional information.
- **Entity types**: from the drop-down menu select the entity type whose observables you want to enrich with additional information.

- **TLP**: from the drop-down menu select the TLP color code you want to use to filter enrichment data. **TLP** `(https://www.us-cert.gov/tlp)` provides an intuitive reference to assess how sensitive information is, focusing in particular on how serious it is, and whom it should or should not be shared with.
- Click **+ add** or **+ more** to add a new filtering option, for example to include another incoming feed or a different entity type. A filter can take only one source and one entity type at a time, but you can set up rules with as many filters as you need.
- **Enrichers**: from the drop-down menu select one or more enrichers to apply the rule to. When a rule is applied to one or more enrichers, it filters the enrichment data polled from the enricher, source based on the specified rule filters and criteria.
- Select the **Active** checkbox to enable the rule immediately after creating it.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Save options

Besides committing current data by clicking **Save**, you can also click the downward-pointing arrow on the **Save** button to display a context menu with additional save options:

- **Save and new**: saves the current data for the active item, and it allows you to start creating right away a new item of the same type; for example, a dataset, a feed, a rule, or a task.
- **Save and duplicate**: saves the current data for the active item, and it creates a pre-populated copy of the same item, which you can use as a template to speed up manual creation work.



Edit enricher rules

To edit enricher rules, do the following:

- On the top navigation bar click **+ > Rules > Enrichment**

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button next to the user avatar image.
- From the drop-down menu select **Rules**.
- On the left-hand navigation sidebar click **Enrichment**.

- The default view is **Rules > Enrichment**. It shows an overview of the configured enricher rules.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.

To edit the details of a specific rule, do the following:

- Click an area on the row corresponding to the rule you want to examine. An overlay slides in from the side of the screen to display the rule detail pane.
- On the detail pane, click **Edit**.

Alternatively:

- Click the dotted menu icon on the row corresponding to the enricher you want to configure or modify.
- From the drop-down menu select **Edit**.

> ✔ On the forms, input fields marked with an asterisk are required.

- **Name**: define a name to identify the rule. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.
- **Source**: from the drop-down menu select the incoming feed or the enricher whose observables you want to augment with additional information.
- **Entity types**: from the drop-down menu select the entity type whose observables you want to enrich with additional information.
- **TLP**: from the drop-down menu select the TLP color code you want to use to filter enrichment data.
  TLP `(https://www.us-cert.gov/tlp)` provides an intuitive reference to assess how sensitive information is, focusing in particular on how serious it is, and whom it should or should not be shared with.
- Click **✚ add** or **✚ more** to add a new filtering option, for example to include another incoming feed or a different entity type.
- **Enrichers**: from the drop-down menu select one or more enrichers to apply the rule to. They are external data providers that are polled to obtain relevant enricher raw data; for example, whois lookup, reverse DNS, or GeoIP information.
- Select the **Active** checkbox to enable the rule immediately after creating it.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Delete enricher rules

To delete an enricher rule, do the following:

- On the top navigation bar click **✚ > Rules > Enrichment**

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button next to the user avatar image.
- From the drop-down menu select **Rules**.
- On the left-hand navigation sidebar click **Enrichment**.
- The default view is **Rules > Enrichment**. It shows an overview of the configured enricher rules.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.
- Click an area on the row corresponding to the rule you want to delete. An overlay slides in from the side of the screen to display the rule detail pane.
- Click **Delete** on the rule detail pane.

Alternatively:

- Click the dotted menu icon on the row corresponding to the rule you want to delete.
- From the drop-down menu select **Delete**.
- On the confirmation pop-up dialog, click **Delete** to confirm the action.
- The rule is deleted.

Work with enricher tasks

View enricher tasks

To view enricher tasks, do the following:

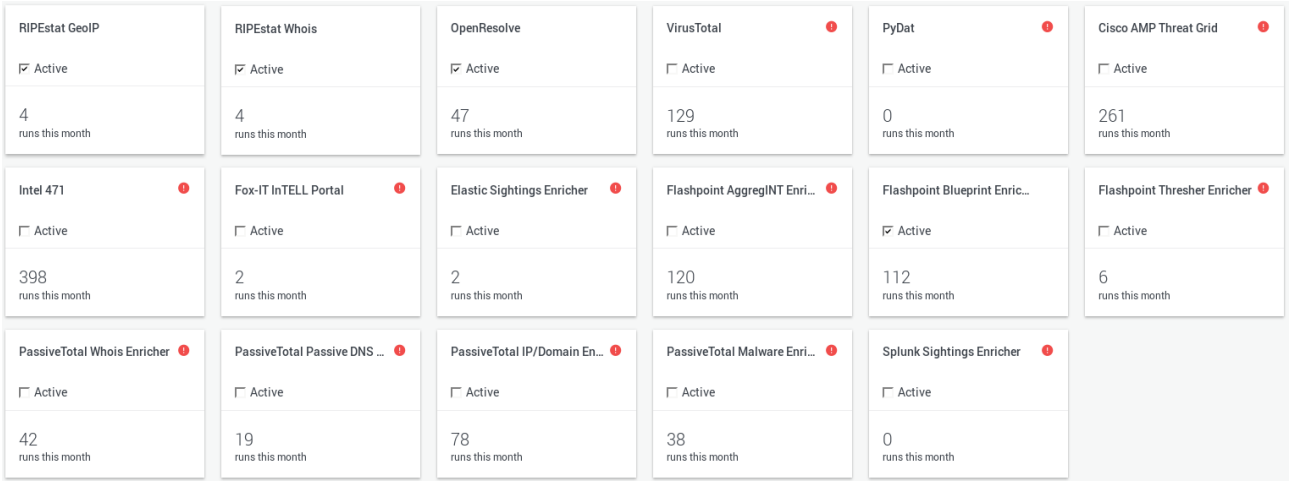
- On the top navigation bar click ✚ **> Data management > Datasets > Enrichment**

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button next to the user avatar image.
- From the drop-down menu select **Data management**.
- On the left-hand navigation sidebar click **Enrichment**.
- Click the enricher you want to examine.
- On the enricher detail page, you can view all the details about the selected enricher, including the rules driving the enricher behavior, recently executed enriching tasks, and the state.
- You can click the state value or an enrichment rule to display additional information.

When the state value returns **FAILURE**, click the link to view the task execution traceback and to begin troubleshooting.

The **Data management > Enrichment** view shows all configured enrichers polling third-party and/or external services to acquire additional information to integrate observables with, so that they can provide more context to the cyber threat entities they belong to.



Edit enricher tasks

To configure and/or to edit an enricher task, do the following:

- On the top navigation bar click **+ > Data management > Datasets > Enrichment**

Alternatively:

- On the top navigation bar, click the ⚙ *configuration and settings* button next to the user avatar image.
- From the drop-down menu select **Data management**.
- On the left-hand navigation sidebar click **Enrichment**.
- Click the enricher you want to configure or modify.
- On the enricher detail page, click the **Edit** button.

✔ On the forms, input fields marked with an asterisk are required.

- **Name**: the name used to identify the enricher. It should be descriptive and easy to remember.
- **Description**: additional textual details. If you want, you can add a short description to provide more information and context.

- **Cache validity (sec)**: defines for how long enrichment data remains stored in the cache. The value is expressed in seconds.
- **Rate limit (per sec)**: sets the maximum allowed number of requests/executions per second.
- **Monthly execution cap (executions)**: sets a maximum allowed number of requests/executions per month.
  Together with rate limiting, execution cap helps control data traffic for the enricher; for example, when the API or the service you are connecting to enforces usage limits.
- **Source reliability**: from the drop-down menu select an option to flag the level of reliability of the source. It helps analysts assess how much confidence they can reasonably have in an intel source.
  Values in this menu have the same meaning as the first character in the **two-character Admiralty System code** `(https://en.wikipedia.org/wiki/admiralty_code)`.
  Example: *B - Usually reliable*
- **Active**: checkbox. Select the **Active** checkbox to enable the enricher task immediately after editing and saving it.
  If you select the checkbox, the rule is executed automatically. If you deselect it, you need to run the rule manually.
- Under **Parameters**, define the specific configuration options for the selected enricher, where applicable.
- When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Warning:
Some enricher tasks include an additional API key field where you specify the API key issued by the source of the enricher, along with the necessary authentication and authorization credentials. Contact the intel service provider whose data you want to use as a source for the enricher to request an API key and any other required credentials.

You need to install and set up PyDat locally. The product does not work outside a local network. You need to configure the host before you can access PyDat features through the API endpoint. See also:

- **Mitre blog on PyDat** `(http://www.mitre.org/capabilities/cybersecurity/overview/cybersecurity-blog/using-whois-and-passive-dns-for-intelligence)`
- **PyDat GitHub repo** `(https://github.com/mitrecnd/whodat)`

How to make API calls with a script

Summary: Make calls to the EclecticIQ API using our simple 'papi' script.

Dependencies

The script we are going to use in this example relies on the following dependencies:

- **HTTPie** `(http://httpie.org/)`
- **jq** `(https://stedolan.github.io/jq/)`

The default values for user name and password in the script are:

- `USERNAME=test`
- `PASSWORD=test`

How the script works

You can use the `papi.sh` script to make calls to the EclecticIQ Platform API from the command line. The script uses *HTTPie* and *jq* to perform the following actions:

- It communicates with the `auth` API endpoint to authenticate and obtain a bearer token.
- It makes calls to the platform API and it includes the token along with the other call parameters.

Create the papi.sh file

Create or modify the *papi.sh* script so that it looks like the following example:

```
#!/bin/bash

#
# Helper for interacting with the platform API from the command line. This tool
# is a wrapper around httpie. It will take care of authentication and some
# other things. It takes at least three arguments:
#
# - host name
# - http method
# - relative url (without the /api/ part)
#
# Any additional arguments are propagated to httpie.
#
# Examples:
#
#   platform-api-http https://some.host/ GET /users/
#
#   platform-api-http localhost:8000 POST /some/endpoint/ @request-stored-in-a-
file.json
#

set -e

readonly HTTPIE=http
readonly HTTPIE_ARGS="--check-status --verify=no"
readonly USERNAME=test
readonly PASSWORD=test

usage() {
    echo "Usage: $(basename $0) host method path [http-args]" > /dev/stderr
    exit 1
}

main () {
    local HOST="$1"
    local METHOD="$2"
    local API_PATH="$3"
    shift 3 || usage
    local TOKEN=$(${HTTPIE} ${HTTPIE_ARGS} POST "${HOST}/api/auth"
username=${USERNAME} password=${PASSWORD} | jq --raw-output '.token')
    local URL="${HOST}/api${API_PATH}"
    ${HTTPIE} ${HTTPIE_ARGS} ${METHOD} ${URL} Authorization:'"Bearer ${TOKEN}"' "$@"
}

main "$@"
```

Authentication

When you make a call to the platform API with `papi.sh`, the script takes care of the following tasks:

- It carries out the authentication step by tokenizing the user's credentials (user name and password).
- When you make a call using the script, it sends the bearer token in the `Authorization` HTTP header: `Authorization: Bearer <token>`.

The authentication mechanism is based on **JSON web tokens** `(http://jwt.io/)`.

By default, the token expires 2 hours after successfully signing in. The corresponding session is terminated, and you need to sign back in to the platform.

When human interaction is detected — for example, keystrokes or mouse activity — the token is automatically refreshed every 2 minutes. This prevents the system from signing out users who may be working or saving data at that time.

Therefore, the default maximum amount of idle time without any human interaction before being automatically signed out equals to *session token validity - 2 minutes*.

To authenticate and access the platform, do the following:

- Make a `POST` call.
- In the call, pass your authentication credentials as a JSON object to the `/auth` endpoint. The credential data is used to generate a token that is returned with the response.

You need to include the generated bearer token in the `Authorization` HTTP header with each subsequent API call.
The `Authorization` HTTP header has the following format: `Authorization: Bearer <token>`

The following examples show the isolated code snippets in the script that take care of authorization credentials, and passing the token with the `Authorization` HTTP header, respectively.

```
local TOKEN=$(${HTTPIE} ${HTTPIE_ARGS} POST "${HOST}/api/auth" username=${USERNAME}
password=${PASSWORD} | jq --raw-output '.token')
```

```
local URL="${HOST}/api${API_PATH}"
${HTTPIE} ${HTTPIE_ARGS} ${METHOD} ${URL} Authorization:'"Bearer ${TOKEN}"' "$@"
```

Make the script executable

To make the script executable, run the following command(s):

```
$ chmod +x ~/papi.sh
```

Examples:

```
$ ./papi.sh https://<platform_host>/ get /enricher-tasks/
```

```
$ ./papi.sh https://<platform_host>/ get /configurations/taxii.settings
```

```
$ ./papi.sh https://<platform_host>/ post /utility-tasks/12/run
```

Create an alias for the script

You can create an alias in `~/.bash_profile` to make it easier to call the script; run the following command(s):

```
alias papi="~/papi.sh"
```

Examples:

```
$ papi https://<platform_host>/ get /enricher-tasks/
```

```
$ papi https://<platform_host>/ get "/entities/?limit=1&sort=-created_at"
```

```
$ papi https://<platform_host>/ get "/entities/?limit=1&sort=-created_at"
```

Input parameters

To use the script to make calls to the platform API, do the following:

- In the terminal or in the command line, call the script by typing its name or the corresponding alias, so either `./papi.sh` or `papi`.
- Set the target host name you want to reach, for example `https://platform.host`.
- Pass a valid **HTTP method** `(http://www.restapitutorial.com/lessons/httpmethods.html)` like `get` or `post`.
- Pass the URL corresponding to the API endpoint whose service you want to consume, for example `/outgoing-feeds/`.
- *Optional* — Pass any valid URL query parameters or a *.json* file containing any additional request parameters.

Parameter	Description
`https://<platform_host>/`	*Required* — The name of the host used to reach the API endpoint and to communicate with the API service.
`POST`, `GET`, `PUT`, `DELETE`	*Required* — A valid **HTTP method** `(http://www.restapitutorial.com/lessons/httpmethods.html)` to create, read, update, or delete a resource.
`/<API_endpoint>/`	*Required* — A relative URL pointing to the API endpoint that exposes the service you want to consume.
`?url=true&query=search-or-filter&params=4`	*Optional* — URL query parameters to send any additional search parameters and/or to filter the results returned in the response.

Besides appending URL query parameters, you can also send your request parameters as a JSON file.
Example:

```
$ papi https://platform.host/ get /entities/ @request-parameters.json
```

Error handling

When an error occurs, the API handles it by returning a **4xx (input/client error) or 5xx (server issue) HTTP response code** `(https://httpstatuses.com/)`, and by including a short explanatory message in the JSON response body.

The key/value pairs in the `errors` JSON object provide more details about the possible cause of the error, which should help you solve it.

- `status`: the HTTP response code.
- `title`: the name of the error.
- `detail`: a short explanation of the issue with more context, when available.

```
{
  "errors" : [

    {
      "detail" : "The requested URL was not found on the server. If you entered the
URL manually please check your spelling and try again.",
      "status" : 404,
      "title"  : "Not Found"
    }

  ]
}
```

Examples

✔ All parameter values in the example(s) are dummy values.
Replace these sample values with actual, valid ones before testing them in your environment.

Make HTTP calls with the script

Create or modify the *papi.sh* script so that it looks like the following example:

```
#!/bin/bash
set -e

readonly HTTPIE=http
readonly HTTPIE_ARGS="--check-status --verify=no"
readonly USERNAME=test
readonly PASSWORD=test

usage() {
    echo "Usage: $(basename $0) host method path [http-args]" > /dev/stderr
    exit 1
}

main () {
    local HOST="$1"
    local METHOD="$2"
    local API_PATH="$3"
    shift 3 || usage
    local TOKEN=$(${HTTPIE} ${HTTPIE_ARGS} POST "${HOST}/api/auth"
username=${USERNAME} password=${PASSWORD} | jq --raw-output '.token')
    local URL="${HOST}/api${API_PATH}"
    ${HTTPIE} ${HTTPIE_ARGS} ${METHOD} ${URL} Authorization:'"Bearer ${TOKEN}"' "$@"
}

main "$@"
```

Make cURL calls with the script

Create or modify the *papi.sh* script so that it looks like the following example:

```
#!/bin/bash
set -e

readonly HTTPIE=curl
readonly HTTPIE_ARGS="-X"
readonly USERNAME=test
readonly PASSWORD=test

usage() {
    echo "Usage: $(basename $0) host method path [http-args]" > /dev/stderr
    exit 1
}

main () {
    local HOST="$1"
    local METHOD="$2"
    local API_PATH="$3"
    shift 3 || usage
    local TOKEN=$(${HTTPIE} ${HTTPIE_ARGS} POST "${HOST}/api/auth" -d "{\"username\":
\"${USERNAME}\", \"password\": \"${PASSWORD}\"}" -ks | jq '.token' --raw-output)
    local URL="${HOST}/api${API_PATH}"
    ${HTTPIE} ${HTTPIE_ARGS} ${METHOD} ${URL} Authorization:'"Bearer ${TOKEN}"'' "$@"
}

main "$@"
```

Trigger a dynamic dataset update

This example shows how to trigger a dynamic dataset update task.

- Edit the `papi.sh` script to specify the correct IP address/host you want to reach.
- To retrieve the dataset ID, sign in to the platform and do the following:
  - On the left-hand navigation sidebar, click **Datasets**.
  - In the web browser address bar, inspect the URL: the numeric value in the URL immediately after `/intel-sets/` is the dataset ID.
    For example if the URL reads `https://platform.host/api/intel-sets/29`, the dataset ID is `29`.
- Make a `GET` API call to retrieve the dataset corresponding to the ID you retrieved in the previous step:
  `$ papi https://platform.host get /intel-sets/<set_id>`
- Make a `GET` API call to retrieve a list of all utility tasks.
  You need this information to identify the task handling dynamic sets:
  `$ papi https://platform.host/ get /utility-tasks/`
- Use the appropriate task ID — the ID you retrieved in the previous step — to make a `POST` API call that triggers a dynamic set update task:
  `$ papi https://platform.host/ post /utility-tasks/<task_id>/run`

- Make a `GET` API call to validate the task run you triggered in the previous step:
  `$ papi https://platform.host/ get /task-runs/<task_id>`
  You can find the `<task_id>` value you need in the response to the previous `<task-id>` request.
- Verify that the dataset was correctly updated by making a new `GET` API call and by passing the dataset ID with it:
  `$ papi https://platform.host/ get /intel-sets/<set_id>`

```
# Get a specific dataset ID
$ papi https://platform.host/ get /intel-sets/<set_id>

# Get the utility task list to look for the ID of the specific task you want to run
$ papi https://platform.host/ get /utility-tasks/

# In the response, find the ID of the task that updates dynamic sets
# Ex.: find the ID of the 'utilities.intel_set_update' tasks

# Use this ID to trigger and run a specific task
$ papi https://platform.host/ post /utility-tasks/<task_id>/run

# The response returns the task run ID

# Get a specific task run to check its status
$ papi https://platform.host/ get /task-runs/<task_id>

# Get the output feed content blocks
$ papi https://platform.host/ get /outgoing-feeds/<outgoing_feed_id>/content-blocks/

# Get the output feed content block ID
$ papi https://platform.host/ get /content-blocks/<content_block_id>
```

How to retrieve outgoing feeds through the API

Summary: Fetch outgoing feeds either manually through the platform GUI or programmatically via the API.

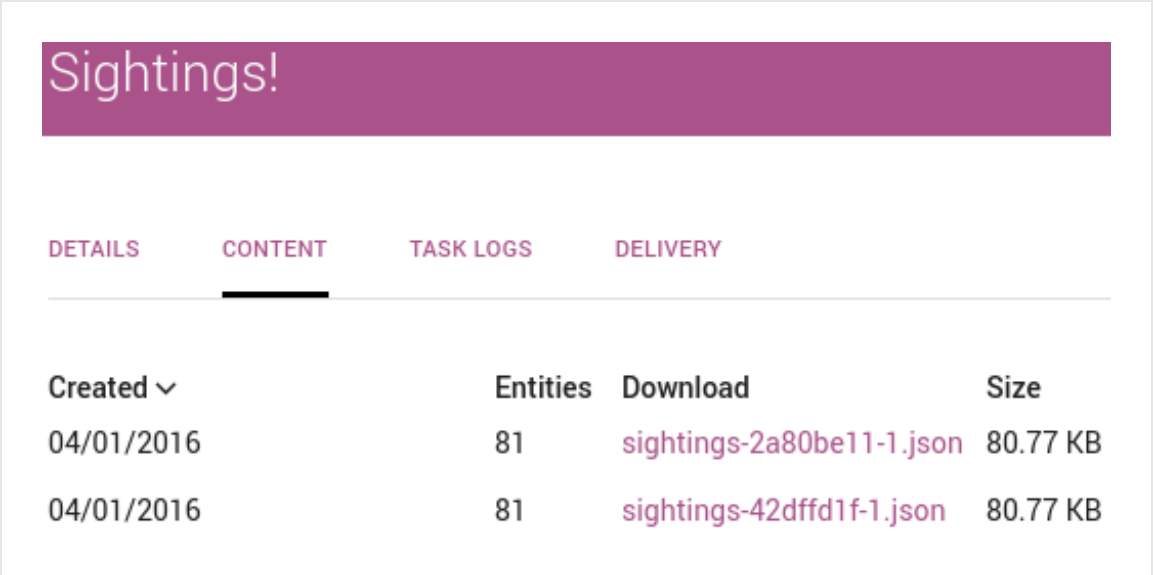
You can retrieve outgoing feed data in two ways:

- Manually, via the platform GUI
- Programmatically, via API calls to the outgoing feed endpoint URLs.

Download outgoing feeds manually

To manually download outgoing feed data through the platform web browser-based GUI, do the following:

- On the left-hand navigation sidebar, click **Outgoing feeds**.
- The **Outgoing feeds** page displays an overview of the configured output data points, i.e. the publishing channels the platform uses to distribute intel data.
  You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (^) or a downward-pointing (ˇ) arrow in the header indicates ascending and descending sort order, respectively.
- Click the row corresponding to the outgoing feed whose data you want to manually download.
- On the outgoing feed detail pane, select the **Content** tab.



The **Content** tab displays an overview of all the outgoing feed execution runs since the creation of the feed. Successful executions generate and distribute content.

- To download the data for a specific run, click the file link under **Download**, and then save it to a target location.

- The file format depends on the content type for the feed data that was defined during the outgoing feed creation and configuration.

Download outgoing feeds via the API

You can download outgoing feed data programmatically by making calls to the platform API using the HTTP protocol.
To implement this option:

- The outgoing feed configured **Transport type** needs to be set to **HTTP download**.
- Before making the first API call to fetch the outgoing feed data, you need to authenticate to receive a bearer token.
- You need to pass a valid bearer token with each API call.

Authentication

The authentication mechanism is based on **JSON web tokens** `(http://jwt.io/)`.

By default, the token expires 2 hours after successfully signing in. The corresponding session is terminated, and you need to sign back in to the platform.

When human interaction is detected — for example, keystrokes or mouse activity — the token is automatically refreshed every 2 minutes. This prevents the system from signing out users who may be working or saving data at that time.

Therefore, the default maximum amount of idle time without any human interaction before being automatically signed out equals to *session token validity - 2 minutes*.

To authenticate and access the platform, do the following:

- Make a `POST` call.
- In the call, pass your authentication credentials as a JSON object to the `/auth` endpoint. The credential data is used to generate a token that is returned with the response.

You need to include the generated bearer token in the `Authorization` HTTP header with each subsequent API call.
The `Authorization` HTTP header has the following format: `Authorization: Bearer <token>`

Auth request

API endpoint	`/auth`

Auth method	`POST`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`
API request	`POST` + `"Content-Type: application/json"` + `"Accept: application/json"` + `{ "username": "<valid_user_name>", "password": "<valid_password>" }` + `<platform_host>/auth`
API response	`{ "expires_at": "<expiry timestamp>", "token": "<token>" }`

The following example uses cURL to authenticate:

```
$ curl -X POST
       -H "Content-Type: application/json"
       -d '{ "username" : "<valid_user_name>", "password" : "<valid_password>" }'
       https://platform.host/api/auth
```

Auth response

When the user name and password credential are valid, the `POST` call returns a JSON web token:

```
{
  "expires_at": "2016-03-30T12:11:40.078219+00:00",
  "token"     :
"abHpYXQiOjE0NTkzMzI3MDAsIm4TcCI6MTQ1OTMzOTkwMCwiYWxnIjoiSFMyNTYifQ.oyY1c2VyX2lkIjo1fQ
.LQQ3NdUHp4s-QCXsxq3feI0Dy6tf5XQX9DOML1RNIzQ"
}
```

You need to include the bearer token value in each subsequent API call. You pass the token by including an `Authorization` HTTP header in the API request.
The `Authorization` HTTP header has the following format: `Authorization: Bearer <token>`

In the following example, you make a `GET` request to the `/api/` endpoint to retrieve a list of the available API endpoints and the corresponding methods:

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/
```

Warning:
About cURL calls

- If you make HTTPs cURL calls to the API *and* you have a self-signed or an invalid certificate, include the `-k` or the `--insecure` parameter in the cURL call to skip the SSL connection CA certificate check.
- Always append a `/` trailing slash at the end of an API URL endpoint. The only exception is `/auth`, which does not take a trailing slash.
- In the cURL call, the `-d` data payload with the entity information always needs to be flat JSON, not hierarchical JSON.
  If you want to pass a hierarchical JSON object, include the `--data-binary` parameter, followed by the path to the JSON file, for example `@/path/to/entity_file.json`.

Public outgoing feed endpoints

The endpoints in this section expose only open/public HTTP download outgoing feeds, i.e. the outgoing feeds that are flagged as public upon creation.

To obtain a list of the private HTTP download outgoing feeds, use the `/api/outgoing-feed-download/` endpoint. This endpoint returns only private outgoing feeds, i.e. the outgoing feeds that were not flagged as public when they were created. The outgoing feeds returned with the response require authorization to access their content.

- `/api/open-outgoing-feed-download/`
  Returns all public outgoing feeds with HTTP transport type.
- `/api/open-outgoing-feed-download/<feed-id>/`
  Returns a specific outgoing feed, including all successful feed executions.
- `/api/open-outgoing-feed-download/<feed-id>/runs/latest`
  Returns the latest/most recent successful feed execution with the corresponding published content blocks for a specific outgoing feed.
- `/api/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/latest`
  Returns the latest content blocks of a specific successful feed execution for a specific outgoing feed.
- `/api/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`
  Returns a specific content block of a specific successful feed execution for a specific outgoing feed.

The API request examples in the following sections use cURL and the *papi.sh* script available with the platform.

Download outgoing feeds

Download a list of all available public outgoing feeds
This call returns a JSON object with an array listing all available public outgoing feeds with HTTP transport type.

API endpoint	`/open-outgoing-feed-download/`
API method	`GET`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`
API request	`GET` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `<platform_host>/open-outgoing-feed-download/`
API response	`{ "data" : [ <open_outgoing_feed_array> ] }`

API request outgoing feeds

cURL call

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/open-outgoing-feed-download/
```

papi script call

```
$ papi https://<platform_host>/ get /open-outgoing-feed-download/
```

API response outgoing feeds

```
{
  "data": [

    {
      "id": 1,
      "link": "/api/open-outgoing-feed-download/1",
      "name": "Default outgoing feed"
    },

    {
      "id": 16,
      "link": "/api/open-outgoing-feed-download/18",
      "name": "Public feed with electrolytes"
    },

    {
      "id": 25,
      "link": "/api/open-outgoing-feed-download/25",
      "name": "XYZ"
    }

  ]
}
```


Download a specific outgoing feed

Download the details of a specific outgoing feed
This call returns a JSON object containing the details of a specific public outgoing feed with HTTP transport type.
To select the public outgoing feed whose details you want to retrieve, include the feed ID in the API request endpoint.

API endpoint	`/open-outgoing-feed-download/<feed-id>/`
API method	`GET`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`
API request	`GET` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `<platform_host>/open-outgoing-feed-download/<feed-id>/`
API response	`{ "data" : { <specific_feed_details> } }`

API request specific outgoing feed

cURL call

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/open-outgoing-feed-download/18
```

papi script call

```
$ papi https://<platform_host>/ get /open-outgoing-feed-download/18
```

API response specific outgoing feed

The response details include an array listing the successful feed executions.

The paths in the `content_blocks` array have the following format: `/api/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`

- A *run* is a feed execution to publish the feed content.
- A *content block* is a data blob whose format depends on the content type defined for the feed, for example JSON, CSV or STIX.

```
{
  "data": {

    "content_blocks": [
      "/api/open-outgoing-feed-download/18/runs/0ad2edd4-8a7b-4894-b8b3-
aee90a22ebaa/content-blocks/32",
      "/api/open-outgoing-feed-download/18/runs/5fdeff71-93af-43a5-b94e-
c4ab857a749c/content-blocks/33",
      "/api/open-outgoing-feed-download/18/runs/40e31ada-06e6-4647-a287-
4c9b54841619/content-blocks/34",
      "/api/open-outgoing-feed-download/18/runs/0f56ec9c-cc1e-4aae-afd0-
f693f412ad55/content-blocks/35",
      "/api/open-outgoing-feed-download/18/runs/d842dd68-8ecf-4ecf-b073-
a591d361cf26/content-blocks/36",
      "/api/open-outgoing-feed-download/18/runs/eed28e1e-4352-42a5-8b1f-
cfc918b0e0ab/content-blocks/37",
      "/api/open-outgoing-feed-download/18/runs/f830aa7b-4ddc-4725-b13c-
7cbe445f306d/content-blocks/40",
      "/api/open-outgoing-feed-download/18/runs/a11bb585-720a-4c56-b650-
90cb9d6a69e5/content-blocks/41",
      "/api/open-outgoing-feed-download/18/runs/6e677f4b-c91d-49dd-9c39-
70266987b863/content-blocks/42"
    ],

    "id": 18,
    "name": "Public feed with electrolytes"
  }
}
```


Download the latest run

Download the latest successful run of a specific outgoing feed
This call returns a JSON object containing the details of the latest execution (run) of a specific public outgoing feed with HTTP transport type.
To select the public outgoing feed whose details you want to retrieve, include the feed ID in the API request endpoint.

API endpoint	`/open-outgoing-feed-download/<feed-id>/runs/latest`
API method	`GET`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`
API request	`GET` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `<platform_host>/open-outgoing-feed-download/<feed-id>/runs/latest`

API response	`{ "data" : { <specific_feed_latest_run_details> } }`

API request latest run

cURL call

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/open-outgoing-feed-download/18/runs/latest
```

papi script call

```
$ papi https://<platform_host>/ get /open-outgoing-feed-download/18/runs/latest
```

API response latest run

The response details include an array with the content blocks belonging to the most recent successful execution of the specified feed.

The paths in the `content_blocks` array have the following format: `/api/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`

- A *run* is a feed execution to publish the feed content.
- A *content block* is a data blob whose format depends on the content type defined for the feed, for example JSON, CSV or STIX.

```
{
  "data": {

    "content_blocks": [
      "/api/open-outgoing-feed-download/18/runs/6e677f4b-c91d-49dd-9c39-
70266987b863/content-blocks/42"
    ],

    "id": 18,
    "name": "Public feed with electrolytes"
  }
}
```


Download the latest content

Download the latest content from a specific run of a specific outgoing feed
This call returns the details of the latest content block belonging to a specific execution (run) of a specific public outgoing feed with HTTP transport type. To select the public outgoing feed and the execution whose details you want to retrieve, include the feed ID and the execution (run) ID in the API request endpoint.

API endpoint	`/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/latest`
Create method	`GET`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`
API request	`GET` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `<platform_host>/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/latest`
API response	`<content_block_blob>`

API request latest content

cURL call

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/open-outgoing-feed-download/18/runs/6e677f4b-c91d-
49dd-9c39-70266987b863/content-blocks/latest
```

papi script call

```
$ papi https://<platform_host>/ get /open-outgoing-feed-download/18/runs/6e677f4b-
c91d-49dd-9c39-70266987b863/content-blocks/latest
```

API response latest content

The response returns the lastest content block that was generated and distributed with the specific execution of a specific public outgoing feed, as specified in the API request endpoint.

The paths in the `content_blocks` array have the following format: `/api/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`

- A *run* is a feed execution to publish the feed content.
- A *content block* is a data blob whose format depends on the content type defined for the feed, for example JSON, CSV or STIX.

```
<stix:STIX_Package xmlns:cybox="http://cybox.mitre.org/cybox-2"
xmlns:indicator="http://stix.mitre.org/Indicator-2"
xmlns:stix="http://stix.mitre.org/stix-1" xmlns:id-1="http://hailataxii.com"
xmlns:DomainNameObj="http://cybox.mitre.org/objects#DomainNameObject-1"
xmlns:stixCommon="http://stix.mitre.org/common-1" xmlns:not-yet-
configured="http://not-yet-configured.example.org/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" id="not-yet-configured:package-
39de53f4-cbe4-4755-8274-1f6f825639cb" timestamp="2016-01-29T10:42:19+00:00"
version="1.2">
  <stix:STIX_Header>
    <stix:Title>Entities package</stix:Title>
  </stix:STIX_Header>
    <stix:Indicators>
      <stix:Indicator id="not-yet-configured:indicator-153e9285-c99f-412e-9f88-
1f883765d897" xsi:type="indicator:IndicatorType">
        <indicator:Observable id="id-1:Observable-9f042056-7ba1-433a-b67f-
467982412fe0" sighting_count="1">
          <cybox:Title>Domain:Edited</cybox:Title>
          <cybox:Description>Domain: xn----8sbafbb5baamcbp7aadbi1bi6e2bygua.xn--p1ai |
isFQDN: True | </cybox:Description>
          <cybox:Object id="id-1:DomainName-3cfb1c45-debd-421c-92de-5aa5b7a447bc">
            <cybox:Properties xsi:type="DomainNameObj:DomainNameObjectType">
              <DomainNameObj:Value condition="Equals">xn----
8sbafbb5baamcbp7aadbi1bi6e2bygua.xn--p1ai</DomainNameObj:Value>
            </cybox:Properties>
          </cybox:Object>
        </indicator:Observable>
        <indicator:Kill_Chain_Phases>
          <stixComsuccemon:Kill_Chain_Phase phase_id="stix:TTP-786ca8f9-2d9a-4213-
b38e-399af4a2e5d6"/>
        </indicator:Kill_Chain_Phases>
      </stix:Indicator>
    </stix:Indicators>
</stix:STIX_Package>
```

Download specific content

Download specific content from a specific run of a specific outgoing feed

This call returns the details of the specific content belonging to the specified run of a specific public outgoing feed with HTTP transport type. You select the feed, the run, and the content block you want the call to return by adding the feed ID, the run ID, and the content block ID to the API endpoint of the request.

API endpoint	`/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`
Create method	`GET`

HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`
API request	`GET` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `<platform_host>/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`
API response	`<content_block_blob>`

API request specific content

cURL call

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/open-outgoing-feed-download/33/runs/2fbcc01f-6553-4a92-ac3d-456049a198f7/content-blocks/84
```

papi script call

```
$ papi https://<platform_host>/ get /open-outgoing-feed-download/33/runs/2fbcc01f-6553-4a92-ac3d-456049a198f7/content-blocks/84
```

API response specific content

The response returns the specific content block that was generated and distributed with the specific execution of a specific public outgoing feed, as per corresponding feed, run, and content block IDs defined in the API request endpoint.

The paths in the `content_blocks` array have the following format: `/api/open-outgoing-feed-download/<feed-id>/runs/<run-id>/content-blocks/<content-block-id>`

- A *run* is a feed execution to publish the feed content.
- A *content block* is a data blob whose format depends on the content type defined for the feed, for example JSON, CSV or STIX.

```
CEF:0|EclecticIQ|EclecticIQ Platform|0.15.0|ttp|EclecticIQ Platform \|
ttp|verylow|ad.incomingFeedSourceId=e8815882-d610-47d4-991c-e0cde68445e8
deviceReceiptTime=1453662154657 cs1Label=title cs1=Targeting: TD Ameritrade cat=ttp
cs3=name cn2=0 start=1441379114127 cs4=td ameritrade cs3Label=extractType cs2=WHITE
modelConfidence=0 cn1Label=halfLifeDays priority=2 severity=verylow
cs4Label=extractValue externalId={http://www.hailataxii.com}ttp-6730e2ef-9e9a-4eeb-
a6f4-d741f1a6cb4a relevancy=4 ad.incomingFeedName=guest.phishtank_com cn1=182
cs2Label=tlpColor cn2Label=sightingsCount attackerUserName=td ameritrade
CEF:0|EclecticIQ|EclecticIQ Platform|0.15.0|ttp|EclecticIQ Platform \|
ttp|veryhigh|ad.incomingFeedSourceId=bd74b8c9-b0cc-4fa4-8de5-6bba538fad63
deviceReceiptTime=1455630435950 cs1Label=title cs1=testt2 cat=ttp cn2=1
start=1455630435950 cs4=cirque du soleil cs3Label=extractType cs3=name
modelConfidence=9 cn1Label=halfLifeDays priority=10 severity=veryhigh
cs4Label=extractValue relevancy=8 msg=
<p>asdf</p> ad.incomingFeedName=Testing Group cn1=182 cs2Label=tlpColor
cn2Label=sightingsCount attackerUserName=cirque du soleil
```

HTTP status codes

This is how you can read HTTP status codes for HTTP download outgoing feeds retrieved through the API:

HTTP status code	Description
404	*Not found*. Error. The request did not execute. Check the error message or the console/terminal for more details.
200	*OK*. This status code is returned when a request is executed correctly, and an empty list is returned. It can mean that no HTTP download outgoing feeds are configured, or they may not be working correctly, or they have no content blocks to return yet.
200	*OK*. This status code is returned when a request is executed correctly, and a populated list with content blocks is returned. Make a happy dance and have a beer.

Error handling

When an error occurs, the API handles it by returning a **4xx (input/client error) or 5xx (server issue) HTTP response code** `(https://httpstatuses.com/)`, and by including a short explanatory message in the JSON response body.

The key/value pairs in the `errors` JSON object provide more details about the possible cause of the error, which should help you solve it.

- `status`: the HTTP response code.
- `title`: the name of the error.
- `detail`: a short explanation of the issue with more context, when available.

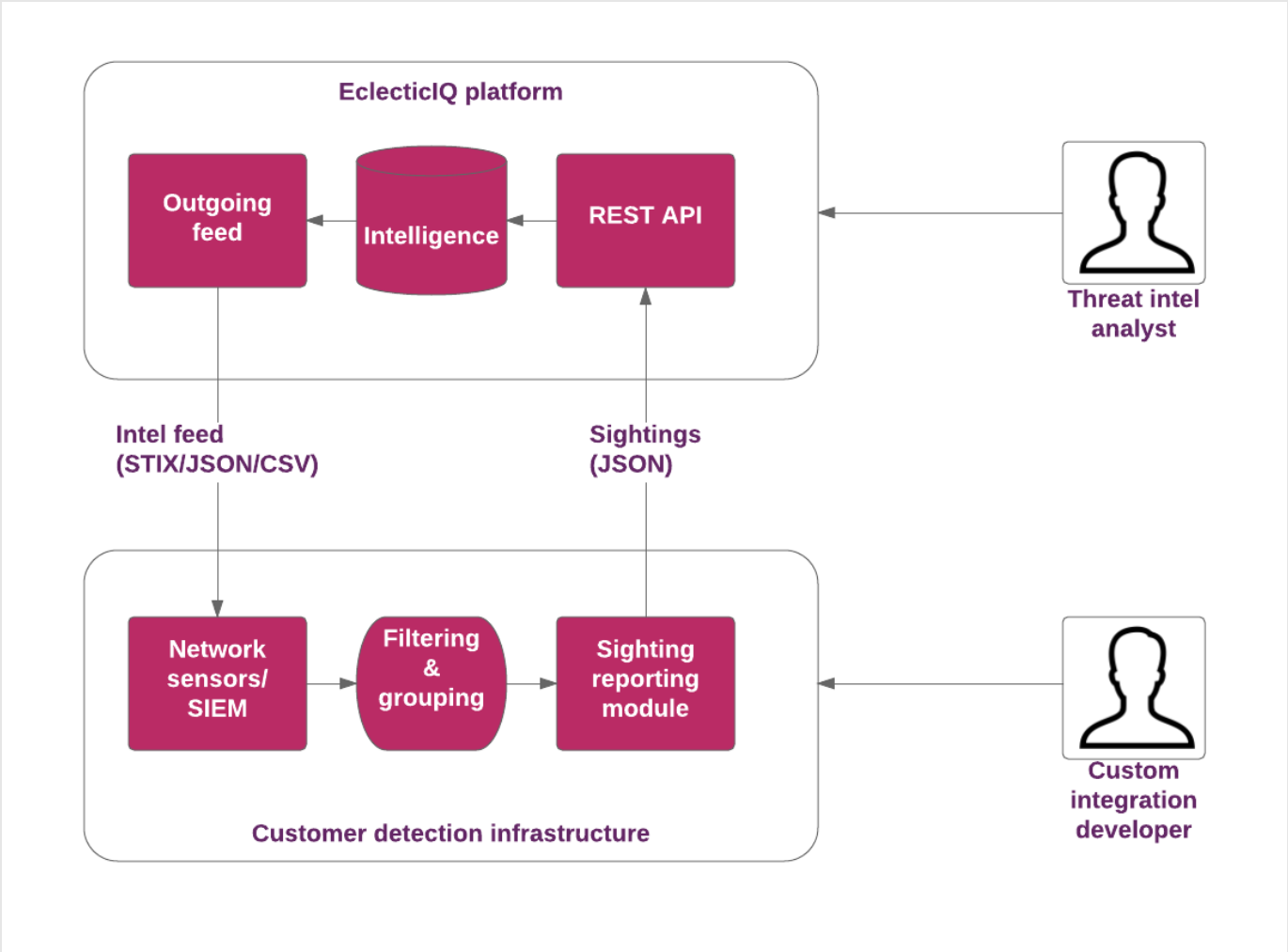
```
{
  "errors" : [

    {
      "detail" : "The requested URL was not found on the server. If you entered the
URL manually please check your spelling and try again.",
      "status" : 404,
      "title"  : "Not Found"
    }

  ]
}
```

How to report sightings through the API

Summary: Create and update sighting entities programmatically by making calls to the EclecticIQ API.

Architecture overview

- A matching engine identifies hits in the outgoing feed data stream.
- The matching engine pushes the hits to a message broker.
- A grouping engine listens to events triggered by the message broker. The engine uses entity IDs to group hits into sightings.
- Sightings are then passed on to the EclecticIQ Platform.



Before you start

It is a good idea to have a dedicated user and user group to handle automation tasks that interact with external products or components of your system. Therefore, before you start generating sightings and/or other entities programmatically, you may want to create a user and a user group to handle automation and integration tasks.

Create an automation user group

> **Warning:** The automation user group has to include all the data sources the group members, that is, the automation user profiles, need to access.

To add a new automation user group, do the following:

- On the left-hand navigation sidebar, click **System**.
- Under **User management**, click **Groups**.
- Under **Groups**, click the **+ Group** button.

> On the forms, input fields marked with an asterisk are required.

- Under **Add new group**, define the following settings:
  - **Name**: a descriptive name for the automation user group.
  - **Description**: a short description of the automation user group and its purpose.
  - **Allowed sources**: defines the cyber threat intelligence sources the group is allowed to access. Select here the sources the automation user group and its members need to access to fetch data from.
  - Click the **+ add** link.
  - From the **Source** drop-down menu, choose a source you want to make available to the automation user group.
  - From the **TLP** drop-down menu, choose a **Traffic Light Protocol** `(https://www.us-cert.gov/tlp)` color to filter the source data accordingly.
  - Click the **+ more** link to specify additional sources.
  - Click **Save** to store your changes, or **Cancel** to discard them.

Create an automation user role

To add a new automation user role, do the following:

- On the left-hand navigation sidebar, click **System**.
- Under **User management**, click **Roles**.
- Under **Roles**, click the **+ Role** button.

> ✔ On the forms, input fields marked with an asterisk are required.

- Under **Add new role**, define the following settings:
  - **Name**: a descriptive name for the automation user role.
  - **Description**: a short description of the automation user role and its purpose.
  - **Available permissions**: this pane lists all available permissions the new role can be granted.
    - Select one or more permissions from the list.
    - Click **Add** to grant the role the permission(s) listed in the **Selected Permissions** pane.
    - Alternatively, start typing a permission name in the autocomplete text input field above the pane.
    - Select one or more filtered permissions from the list.
    - Click **Add** to grant the role the permission(s) listed in the **Selected Permissions** pane.
    - To revoke one or more permissions for the role, select the relevant entries under **Selected permissions**, and then click **Remove**.
  - Click **Save** to store your changes, or **Cancel** to discard them.

About permissions

User permissions are predefined in the platform, and they are not editable or configurable. You can either assign them to user roles, or revoke them.

Permission names try to be as self-explanatory as possible:

- Format: `<type of action> <object of the action>`
- Example: *modify entities*

There are two permission actions:

- **modify**: a modification permission allows write operations.
- **read**: a read permission grants access to the data without allowing any modifications.

To get an overview of the available permissions on the platform, do the following:

- On the left-hand navigation sidebar, click **System**.

- Under **User management > Permissions**, the permission overview is displayed as a table, where each permission is assigned a row.
- To view permission details, click an area on a row.
- An overlay slides in from the side of the screen. It displays permission information in a flash-card format.

Create an automation user

To add an automation user, do the following:

- On the left-hand navigation sidebar, click **System**.
- Under **User management**, click the **✚ User** button.

> ✔ On the forms, input fields marked with an asterisk are required.

- Under **System > User management > Edit user**, define the following settings:
  - **First name**: n/a
  - **Last name**: n/a
  - **Username**: the designated user name to identify the user, when signed in to the platform. Choose a name that helps understand what the automation user does, for example "*Matching engine aggregator*".
  - **Email**: an email address associated to the automation user. You can use this address to send and receive automated notifications.
  - **Active**: select this checkbox to enable the user.
  - **Administrator**: select this checkbox to elevate the user's role to administrator. When the checkbox is selected, the user has administrator rights and permissions.
  - **Contact info**: n/a
  - **PGP public key**: the user's **PGP public key** `(http://www.pgpi.org/doc/pgpintro/#p9).`
  - **Groups**: this pane lists all available groups the new user can be assigned to.
    - From the drop-down menu select one or more groups to assign the user to.
    - Alternatively, start typing a group name in the autocomplete text input field.
    - To remove the user from one or more groups, remove the relevant entries by clicking the ✖ corresponding to the group you want to remove the user from.
  - **Roles**: it works like **Groups**, the only difference being that instead of adding the user to one or more groups, this option assigns one or more roles to the user.
  - When you are done, click **Save** to store your changes, or **Cancel** to discard them.

Get the automation user group ID

When you create a new entity by making a call to the platform API, you need to pass a group `source` ID value inside the `meta: {}` nested object.
Its value is the group ID the data source(s) *and* the user making the call belong to.

> **Warning:** The automation user group has to include all the data sources the group members, that is, the automation user profiles, need to access.

Step 1 of 2: get the group ID

To retrieve the group ID value you need to pass in your API calls, do the following:

- On the left-hand navigation sidebar click **System**.
- Under **User management**, click **Groups**.
- On the group table overview, click the row corresponding to the automation user group containing the data source(s) you want to use as input *and* the automation user making the API calls.
- The action returns a URL with the following format:
  `https://<platform_host>/#/configuration/system/user-management/groups?detail=<integer>`
  Example:
  `https://platform.host/#/configuration/system/user-management/groups?detail=30`
- The `detail` URL parameter value allows you to retrieve the `source` ID value you need to pass with the `meta: {}` nested object in your API calls.
  In the example, the `detail` value is `30`; this is the group ID we need to retrieve the group `source` ID for our calls.

Step 2 of 2: get the group source ID

To retrieve the group `source` ID value you need to pass in your API calls, do the following:

- Request all available platform user groups (requires authentication and bearer token)
- In the response, look for the group object with the `"id" : "<integer>"` key/value pair you previously retrieved.
- In the same group object, look for the `"source" : "<UUID_string>"` key/value pair.
  This is the group source ID you need to pass in your API calls to create sightings programmatically.

Get the automation user group ID example

cURL API request — fetches all user groups

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/groups/
```

API response — returns user group list

```
{
  "count": 18, // number of returned user groups

  "data": [

    ...

    {
      "allowed_sources": [

        {
          "allowed_tlp": "RED",
          "source": "8d49188e-2a2c-4192-92e3-c76b894c344b"
        },

        {
          "allowed_tlp": "RED",
          "source": "42c051f8-9f5b-4696-a629-b86c2ead955f"
        }

      ],

      "description": null,

      // group id, same value as the 'detail=' URL param for the group
      "id": 30,
      "name": "ddss",

      // the group source ID you need to pass in your API calls
      "source": "42c051f8-9f5b-4696-a629-b86c2ead955f",
      "type": "groups",
      "users": []
    },

    ...

  ]
}
```

Authentication

The authentication mechanism is based on **JSON web tokens** (http://jwt.io/).

By default, the token expires 2 hours after successfully signing in. The corresponding session is terminated, and you need to sign back in to the platform.

When human interaction is detected — for example, keystrokes or mouse activity — the token is automatically refreshed every 2 minutes. This prevents the system from signing out users who may be working or saving data at that time.

Therefore, the default maximum amount of idle time without any human interaction before being automatically signed out equals to *session token validity - 2 minutes*.

To authenticate and access the platform, do the following:

- Make a `POST` call.
- In the call, pass your authentication credentials as a JSON object to the `/auth` endpoint. The credential data is used to generate a token that is returned with the response.

You need to include the generated bearer token in the `Authorization` HTTP header with each subsequent API call.
The `Authorization` HTTP header has the following format: `Authorization: Bearer <token>`

Auth request

API endpoint	`/auth`
Auth method	`POST`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`
API request	`POST` + `"Content-Type: application/json"` + `"Accept: application/json"` + `{ "username": "<valid_user_name>", "password": "<valid_password>" }` + `<platform_host>/auth`
API response	`{ "expires_at": "<expiry timestamp>", "token": "<token>" }`

The following example uses cURL to authenticate:

```
$ curl -X POST
       -H "Content-Type: application/json"
       -d '{ "username" : "<valid_user_name>", "password" : "<valid_password>" }'
       https://platform.host/api/auth
```

Auth response

When the user name and password credential are valid, the `POST` call returns a JSON web token:

```
{
  "expires_at": "2016-03-30T12:11:40.078219+00:00",
  "token"     :
"abHpYXQiOjE0NTkzMzI3MDAsIm4TcCI6MTQ1OTMzOTkwMCwiYWxnIjoiSFMyNTYifQ.oyY1c2VyX2lkIjo1fQ
.LQQ3NdUHp4s-QCXsxq3feI0Dy6tf5XQX9DOML1RNIzQ"
}
```

You need to include the bearer token value in each subsequent API call. You pass the token by including an `Authorization` HTTP header in the API request.
The `Authorization` HTTP header has the following format: `Authorization: Bearer <token>`

In the following example, you make a `GET` request to the `/api/` endpoint to retrieve a list of the available API endpoints and the corresponding methods:

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/
```

Warning:
About cURL calls

- If you make HTTPs cURL calls to the API *and* you have a self-signed or an invalid certificate, include the `-k` or the `--insecure` parameter in the cURL call to skip the SSL connection CA certificate check.
- Always append a `/` trailing slash at the end of an API URL endpoint. The only exception is `/auth`, which does not take a trailing slash.
- In the cURL call, the `-d` data payload with the entity information always needs to be flat JSON, not hierarchical JSON.
  If you want to pass a hierarchical JSON object, include the `--data-binary` parameter, followed by the path to the JSON file, for example `@/path/to/entity_file.json`.

Create a sighting

API endpoint	`/entities/`
Create method	`POST`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`
API request	`POST` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `{ "data" : { <entity> } }` + `<platform_host>/entities/`
API response	`{ "data" : { <entity> } }`

When you make a `GET` request and obtain a JSON object with entity data in the response, the entity object is wrapped in a data wrapper: `{ "data" : { ... } }`.

When you pass a JSON object with entity data in the body of your API request, you always need to wrap it in a data wrapper: `{ "data" : { ... } }`.

Creation request

To create a new sighting with an API call, do the following:

- Make a `POST` call to the `/entities/` API endpoint. The request should include the following HTTP headers:
  - `"Authorization: Bearer <token>"` — *Required*
  - `"Content-Type: application/json"` — *Required*
  - `"Accept: application/json"` — *Optional*
- In the request message body, pass the JSON object containing the data defining the new sighting you want to create. Make sure the sighting data is wrapped inside the `{ "data" : }` wrapper.

Creation response

When the API request payload successfully passes validation:

- A new sighting is created with the specified data.

- The API returns a JSON object containing the newly created sighting data, including a new unique `id` for the sighting.
- The sighting data is wrapped inside the `{ "data" : }` wrapper. The wrapper needs to include at least the following required information:
  - `data`: a nested JSON object holding the information that describes the sighting, like name, short description, any extra details like kill chain phase or TLP color, and so on.
    - `type`: it is inside `data`. It identifies the entity type being created. For sightings, it is always `eclecticiq-sighting`.
    - `title`: it is inside `data`. The name you assign to the new sighting.
  - `meta`: a nested JSON object holding information about the data being posted through the API call, like source and tags, if any.
    - `source`: it is inside `meta`. Its value corresponds to the group ID referring to the source of the information used to create the sighting, for example an incoming feed or a user group who in turn can have a dataset or an incoming feed as its source. This value is automatically generated when a new source is created in the platform.

Creation examples

cURL API request — creates new sighting

```
$ curl -X POST
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       -d '{ "data": { <entity> } }'
       https://platform.host/api/entities/
```

HTTP request message body

It contains the data for the new sighting entity you want to create.

```
{
  "data": {

    "data": {
      "type": "eclecticiq-sighting",
      "title": "test-eclecticiq-sighting"
    },

    "meta": {
      "source": "091bde3a-4a07-4e0f-b834-4038d2041932"
    }

  }
}
```

API response — successful sighting creation

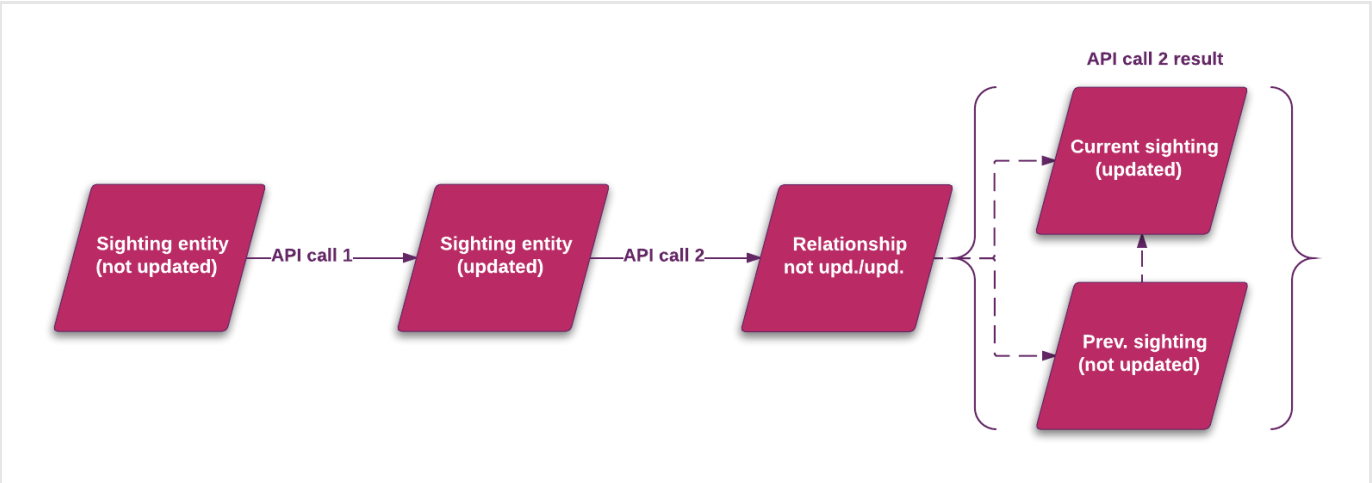
```
HTTP 201 OK CREATED
```

The response body contains the data for the newly created sighting.
Among the fields included in the sighting, you can notice a group ID and an entity ID:

- `group_id`: the entity group the new sighting is assigned to.
- `id`: a unique identifier for the sighting entity.

```
{
  "data": {

    "data": {
      "title": "test-eclecticiq-sighting",
      "type": "eclecticiq-sighting"
    },

    "group_id": "b4585e35-c8a9-478e-a31d-ef7e2ec72aaf",
    "id": "d80da575-4788-4ef5-b0a5-cc0e9b506298",

    "meta": {
      "estimated_observed_time": "2016-03-30T14:22:02.929028+00:00",
      "estimated_threat_start_time": "2016-03-30T14:22:02.929028+00:00",
      "ingest_time": "2016-03-30T14:22:02.929028+00:00",
      "is_outdated_version": false,
      "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
      "source_name": "test",
      "source_type": "group",
      "title": "test-eclecticiq-sighting"
    },

    "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
    "type": "entities"
  }
}
```


Update a sighting



- *API call 1* creates a new version of the sighting containing the updated information.
- *API call 2* relates the previous, not updated version of the sighting, with the new, updated one.

To keep data in sync with the outside world and to avoid inconsistencies, updating an entity is more similar to versioning than to a simple edit-to-update process.

A JSON entity includes two main nested objects:

- `{ "data" : { } }`
  `data` contains the non-modifiable information describing the sighting entity. This data cannot be edited, because it needs to be in sync with the same entity data in the outside world.
- `{ "meta' : { } }`
  `meta` contains the editable entity metadata. You can modify this data at any time.

When you pass a JSON object with entity data in the body of your API request, you always need to wrap it in a data wrapper: `{ "data" : { ... } }`.

API endpoint	`/entities/`
Update method	`POST`
HTTP headers	`"Content-Type: application/json", "Accept: application/json", "Authorization: Bearer <token>"`
API call 1	`POST` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `{ "data" : { <new_entity> } }` + `<platform_host>/entities/`
API response 1	`{ "data" : { <new_entity> } }`
API call 2	`POST` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `{ "data": { "data": { "source": "<new_entity_id>", "source_type": "<entity_type>", "target": "<superseded_entity_id>", "target_type": "<entity_type_same_as_source>", "type": "relation", "subtype": "stix_update_of" } } }` + `<platform_host>/entities/`
API response 2	`{ "data": { "data": { "source": "<new_entity_id>", "source_type": "<entity_type>", "target": "<superseded_entity_id>", "target_type": "<entity_type_same_as_source>", "type": "relation", "subtype": "stix_update_of" } } }`

Update request

To update an existing entity, you need to make two consecutive API calls to perform the following actions:

- Create a new version of the entity containing any changed information you want to store in the updated entity.
- Create a relationship between the new version of the entity you just created and the previous, superseded version of the same entity.

The keys holding entity version information for the update are inside the nested `data` JSON object:

- `source`: holds the `id` value of the new, *updated* entity.
- `target`: holds the `id` value of the *previous*, superseded entity.

- `type`: when updating entities, its value is always `relation`.
- `subtype`: when updating entities, its value is always `stix_update_of`.

At the end of the process, the updated `source` entity has a `stix_update_of` `relation` with the previous/superseded `target` entity.

Update response

A successful update operation returns two responses:

- The first call returns a JSON object containing the newly created entity with the updated information.
- The second call returns a JSON object that defines a`stix_update_of` relationship between the updated entity (`source`) and the superseded one (`target`).

Update examples

cURL API request — updates existing sighting — update 1/2

The first call you make to update an existing entity creates a new entity of the same type, which is an updated version with changed details of the existing entity.

```
$ curl -X POST
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       -d '{ "data": { <entity> } }'
       https://platform.host/api/entities/
```

HTTP request message body

The message body of the first call contains the new version of the sighting entity with updated/changed data.

```
{
  "data": {

    "data": {
      "type": "eclecticiq-sighting",
      "title": "test-eclecticiq-sighting",

      "confidence": {
        "type": "confidence",
        "value": "Medium"
      },

      "description": "<p>This is a very scary sighting for test purposes, which I am
updating to add even more evil to it.</p>",
      "description_structuring_format": "html",

      "handling": [{
        "type": "marking-specification",

        "marking_structures": [{
          "type": "marking-structure",
          "marking_structure_type": "tlp",
          "color": "AMBER"
        }]
      }]
    },

    "meta": {
      "source": "091bde3a-4a07-4e0f-b834-4038d2041932"
    }
  }
}
```


API response — successful sighting creation — update 1/2

```
HTTP 201 OK CREATED
```

The response body contains the new, updated sighting data.

```
{
  "data": {

    "data": {

      "confidence": {
        "type": "confidence",
        "value": "Medium"
      },

      "description": "<p>this is a very scary sighting for test purposes, which I am
updating to add more evil</p>",
      "description_structuring_format": "html",

      "handling": [{

        "marking_structures": [{
          "color": "AMBER",
          "marking_structure_type": "tlp",
          "type": "marking-structure"
        }],

        "type": "marking-specification"
      }],

      "title": "test-eclecticiq-sighting",
      "type": "eclecticiq-sighting"
    },

    "group_id": "72525f33-cfab-44fb-a46a-b1bd37675b0b",
    "id": "c38b51d0-ce71-49f6-918e-13b6b299f4ee",

    "meta": {
      "estimated_observed_time": "2016-03-30T16:12:43.499083+00:00",
      "estimated_threat_start_time": "2016-03-30T16:12:43.499083+00:00",
      "ingest_time": "2016-03-30T16:12:43.499083+00:00",
      "is_outdated_version": false,
      "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
      "source_name": "test",
      "source_type": "group",
      "title": "test-eclecticiq-sighting"
    },

    "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
    "type": "entities"
  }
}
```


cURL API request — updates existing sighting — update 2/2

The second call creates a relationshiop between the updated sighting entity (`source`) and the superseded one (`target`).

```
$ curl -X POST
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       -d '{ "data": { "data": { "source": "<updated_entity_id>", "source_type":
"eclecticiq-sighting", "target": "<superseded_entity_id>", "target_type": "eclecticiq-
sighting", "type": "relation", "subtype": "stix_update_of" } } }'
       https://platform.host/api/entities/
```

HTTP request message body

The request message body of the relationship creation call to link the updated and the superseded entities is a `data` JSON object with the following format:

```
{
  "data" : {
    "source"      : "<updated_entity_id>",
    "source_type" : "eclecticiq-sighting",
    "target"      : "<superseded_entity_id>",
    "target_type" : "eclecticiq-sighting",
    "type"        : "relation",
    "subtype"     : "stix_update_of"
  }
}
```

In your API request, make sure you wrap the `data` object in the `{ "data" : { ... } }` wrapper.

```
{
  "data": {

    "data": {
      "source": "c38b51d0-ce71-49f6-918e-13b6b299f4ee",
      "source_type": "eclecticiq-sighting",
      "target": "d80da575-4788-4ef5-b0a5-cc0e9b506298",
      "target_type": "eclecticiq-sighting",
      "type": "relation",
      "subtype": "stix_update_of"
    }

  }
}
```

API response — successful sighting update — update 2/2

```
HTTP 201 OK CREATED
```

The response to the relationship creation call to link the superseded and the updated entities is a`data` JSON object describing the established relationship between the older (`target`) and the newer (`source`) versions of the same entity.


```
{
  "data": {

    "data": {
      "source": "c38b51d0-ce71-49f6-918e-13b6b299f4ee",
      "source_type": "eclecticiq-sighting",
      "target": "d80da575-4788-4ef5-b0a5-cc0e9b506298",
      "target_type": "eclecticiq-sighting",
      "type": "relation",
      "subtype": "stix_update_of"
    },

    "group_id": "6768f516-627d-4d9e-916c-1db63622321a",
    "id": "f7c04e61-6086-45a0-bebb-8cd31581a476",

    "meta": {},

    "source": null,
    "type": "entities"
  }
}
```

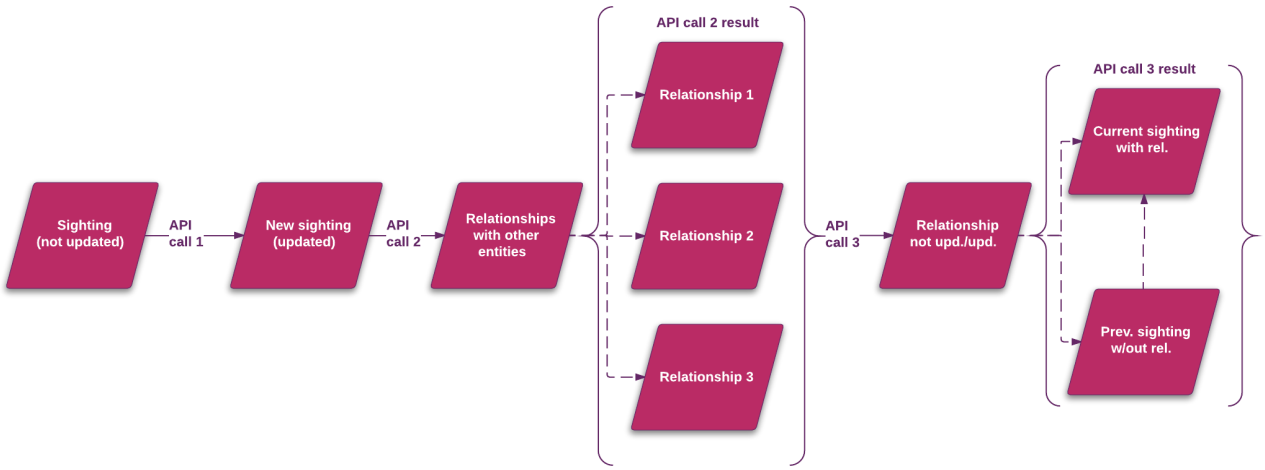

To avoid unnecessary duplication, if you pass the same update data for an entity more than once, the redundant update that would cause data duplication is ignored.

In this case, the HTTP response code notifies that no new data was created:

```
HTTP 200 OK
```

In the `data` JSON object containing the relationship details, the `subtype` JSON key holds the `stix_duplicate_of` value to notify that the update was ignored because it was a duplicate.

Create relationships



- *API call 1* creates a new version of the sighting.
- *API call 2* adds relationships to the new version of the sighting.
- *API call 3* updates the previous version of the sighting without a relationship to the new one with a relationship.

API endpoint	`/entities/`
Update method	`POST`
HTTP headers	`"Content-Type: application/json", "Accept: application/json", "Authorization: Bearer <token>"`
API call 1	`POST + "Content-Type: application/json" + "Accept: application/json" + "Authorization: Bearer <token>" + { "data" : { <new_entity> } } + <platform_host>/entities/`
API response 1	`{ "data" : { <new_entity> } }`
API call 2	`POST + "Content-Type: application/json" + "Accept: application/json" + "Authorization: Bearer <token>" + { "data": { "data": { "source": "<new_entity_id>", "source_type": "<entity_type>", "target": "<entity_id_to_relate>", "target_type": "<entity_type_to_relate>", "type": "relation" } } } + <platform_host>/entities/`
API response 2	`{ "data": { "data": { "source": "<new_entity_id>", "source_type": "<entity_type>", "target": "<entity_id_to_relate>", "target_type": "<entity_type_to_relate>", "type": "relation" } } }`
API call 3	`POST + "Content-Type: application/json" + "Accept: application/json" + "Authorization: Bearer <token>" + { "data": { "data": { "source": "<new_entity_id>", "source_type": "<entity_type>", "target": "<superseded_entity_id>", "target_type": "<entity_type_same_as_source>", "type": "relation", "subtype": "stix_update_of" } } } + <platform_host>/entities/`

API response 3	`{ "data": { "data": { "source": "<new_entity_id>", "source_type": "<entity_type>", "target": "<superseded_entity_id>", "target_type": "<entity_type_same_as_source>", "type": "relation", "subtype": "stix_update_of" } } }`

Relationship request

To create a relationship between two entities, for example between a sighting and an incident, you need to make three consecutive API calls to perform the following actions:

- Create a new version of the sighting. Adding a relationship updates the sighting, and therefore it is necessary to create a new version of this entity to reflect the change.
- Create a relationship between the new version of the sighting and the incident.
- Create a relationship between the new version of the sighting, i.e. the one that is now related to the incident, and the previous one, i.e. the one that is not related to the incident. This updates the sighting to the new version.

The keys holding entity version information for the process are inside the nested `data` JSON object:

- `source`: holds the `id` value of the *new* sighting you want to add a relationship to.
- `target`: holds the `id` value of the target entity of the relationship. In other words, this is the entity you want to relate the sighting to. In this example, it is an incident.
- `type`: when creating relationships, its value is always `relation`.

Relationship response

A successful relationship creation returns three responses:

- *API call 1* returns a JSON object with the new version of the sighting.
- *API call 2* returns a JSON object containing the data about the relationship between the sighting and the incident. This object represents the relationship between the entities.
- *API call 3* relates the previous version of the sighting without a relationship to the new one with a relationship. This call updates the sighting to include the relationship information.

Relationship examples

cURL API request — creates new version of sighting — relationship 1/3

The first API call creates a new version of the sighting.

```
$ curl -X POST
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       -d '{ "data": { <new_entity> } }'
       https://platform.host/api/entities/
```

HTTP request message body

It contains the data of the sighting you want to relate to another entity (for example, to an incident).

```
{
  "data": {

    "meta": {
      "source_name": "test",
      "source_type": "group",
      "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
      "title": "test-eclecticiq-sighting"
    },

    "data": {
      "type": "eclecticiq-sighting",
      "title": "test-eclecticiq-sighting"
    }

  }
}
```

API response — successful sighting creation — relationship 1/3

```
HTTP 201 OK CREATED
```

It returns a new version of the sighting, with a new `id`. You are using this `id` to relate the sighting to another entity.


```
{
  "data": {

    "data": {
      "title": "test-eclecticiq-sighting",
      "type": "eclecticiq-sighting"
    },

    "group_id": "7ae5bfcb-5a20-4fc9-9b2d-ad655a3d5408",
    "id": "85939e81-99b3-4476-b9b5-5b8e17cea2b5",

    "meta": {
      "estimated_observed_time": "2016-03-31T12:14:04.398700+00:00",
      "estimated_threat_start_time": "2016-03-31T12:14:04.398700+00:00",
      "ingest_time": "2016-03-31T12:14:04.398700+00:00",
      "is_outdated_version": false,
      "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
      "source_name": "test",
      "source_type": "group",
      "title": "test-eclecticiq-sighting"
    },

    "source": "091bde3a-4a07-4e0f-b834-4038d2041932",
    "type": "entities"
  }
}
```


cURL API request — creates relationship — relationship 2/3

The second call establishes a relationship between the new version of the sighting you just created with the previous call, and another, different entity; for example, an incident.

```
$ curl -X POST
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       -d '{ "data": { "data": { "source": "<updated_sighting_id>", "source_type":
"eclecticiq-sighting", "target": "<incident_id>", "target_type": "incident", "type":
"relation" } } }'
       https://platform.host/api/entities/
```

HTTP request message body

The request message body of the relationship creation call is a `data` JSON object with the following format:

```
{
  "data" : {
    "source"      : "<id_of_the_entity_you_want_to_create_a_relationship_for>",
    "source_type" : "eclecticiq-sighting",
    "target"      : "<id_of_the_entity_you_want_to_relate_to_the_source>",
    "target_type" : "incident",
    "type"        : "relation"
  }
}
```

In your API request, make sure you wrap the `data` object in the `{ "data" : { ... } }` wrapper.

```
{
  "data" : {

    "data": {
      "source": "85939e81-99b3-4476-b9b5-5b8e17cea2b5",
      "source_type": "eclecticiq-sighting",
      "target": "dcb550e7-4178-4d1f-ad59-072e6d7aa1c7",
      "target_type": "incident",
      "type": "relation"
    }

  }
}
```

API response — successful relationship creation — relationship 2/3

```
HTTP 201 OK CREATED
```

A successful response returns a JSON object containing the details of the newly established relationship between the two different entities.

```
{
  "data": {

    "data": {
      "source": "85939e81-99b3-4476-b9b5-5b8e17cea2b5",
      "source_type": "eclecticiq-sighting",
      "target": "dcb550e7-4178-4d1f-ad59-072e6d7aa1c7",
      "target_type": "incident",
      "type": "relation"
    },

    "group_id": "decde301-d4c4-4387-a8fd-7f423cb6ca2c",
    "id": "2522daa2-638d-4e11-bac0-b4c18bf2f394",

    "meta": {
      "is_outdated_version": false
    },

    "source": null,
    "type": "entities"
  }
}
```


cURL API request — updates sighting — relationship 3/3

The third call updates the sighting, in this example the source entity, by creating a `stix_update_of` relationship between the previous version — the one without a relationship to the incident — and the new one — the one that bears a relationship to the incident.

```
$ curl -X POST
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       -d '{ "data": { "data": { "source": "<updated_entity_id_with_relationship>",
"source_type": "eclecticiq-sighting", "target": "
<superdseded_entity_id_no_relationship>", "target_type": "eclecticiq-sighting",
"type": "relation", "subtype": "stix_update_of" } } }'
       https://platform.host/api/entities/
```

HTTP request message body

The request message body of the update call is a `data` JSON object with the following format:

```
{
  "data" : {
    // new version of the sighting related to incident
    "source"      : "<updated_entity_id>",
    "source_type" : "eclecticiq-sighting",
    // previous version of the sighting not related to incident
    "target"      : "<superseded_entity_id>",
    "target_type" : "eclecticiq-sighting",
    "type"        : "relation",
    "subtype"     : "stix_update_of"
  }
}
```

In your API request, make sure you wrap the `data` object in the `{ "data" : { ... } }` wrapper.

```
{
  "data": {

    "data": {
      "source": "85939e81-99b3-4476-b9b5-5b8e17cea2b5",
      "source_type": "eclecticiq-sighting",
      "target": "c38b51d0-ce71-49f6-918e-13b6b299f4ee",
      "target_type": "eclecticiq-sighting",
      "type": "relation",
      "subtype": "stix_update_of"
    }

  }
}
```

API response — successful sighting update — relationship 3/3

```
HTTP 201 OK CREATED
```

A successful response returns a JSON object containing the details of the version relationship between the two versions of the same entity.

```
{
  "data": {

    "data": {
      "source": "85939e81-99b3-4476-b9b5-5b8e17cea2b5",
      "source_type": "eclecticiq-sighting",
      "subtype": "stix_update_of",
      "target": "c38b51d0-ce71-49f6-918e-13b6b299f4ee",
      "target_type": "eclecticiq-sighting",
      "type": "relation"
    },

    "group_id": "e73c48be-75ea-40ff-b478-890a9b8b1329",
    "id": "6e4ffcad-f62a-4d21-a20a-bc85a42be274",

    "meta": {},

    "source": null,
    "type": "entities"
  }
}
```


Error handling

When an error occurs, the API handles it by returning a **4xx (input/client error) or 5xx (server issue) HTTP response code** (`https://httpstatuses.com/`), and by including a short explanatory message in the JSON response body.

The key/value pairs in the `errors` JSON object provide more details about the possible cause of the error, which should help you solve it.

- `status`: the HTTP response code.
- `title`: the name of the error.
- `detail`: a short explanation of the issue with more context, when available.

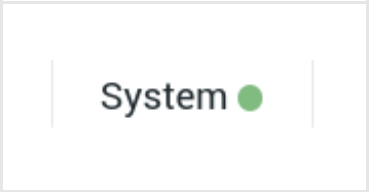

```
{
  "errors" : [

    {
      "detail" : "The requested URL was not found on the server. If you entered the
URL manually please check your spelling and try again.",
      "status" : 404,
      "title"  : "Not Found"
    }

  ]
}
```

How to check system health

Summary: System health gives you a clear basic overview of the overall platform health, as well as the operational status of its components.

Check system health via the GUI

The first and easiest way to check normal platform operation is by using the GUI system health tool on the status bar:

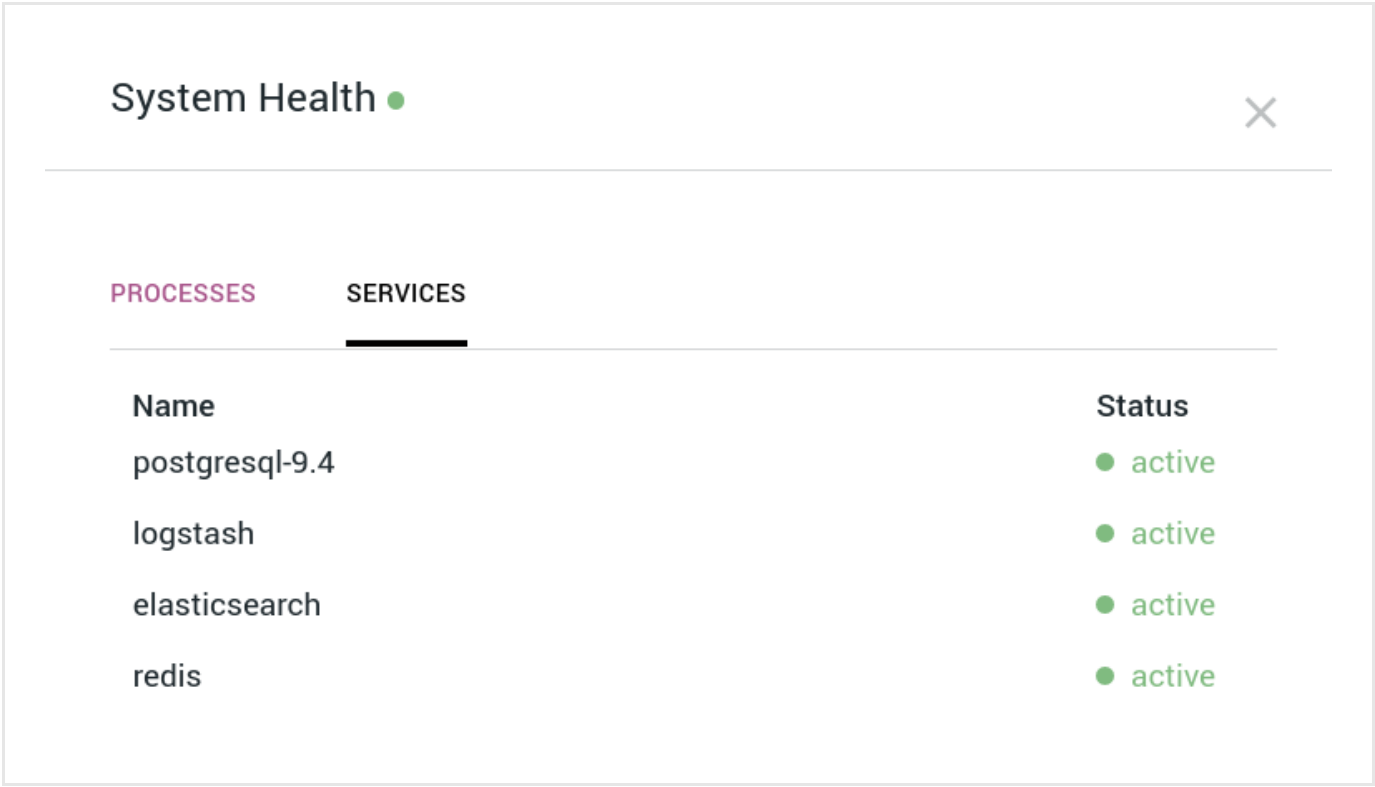


A green or red dot next to **System** on the status bar indicates the global system health status. Click **System** to drill down to a plaform component-level. On the pop-up window click either **Processes** or **Services** to view the status of the normal platform processes or of the platform core services.

System Health

PROCESSES SERVICES

Name	Processes	Status
newrelic_redis_agent	1	
task	5	
intel-ingestion	4	
neo4j-batching	1	
newrelic_postgresql_agent	1	
newrelic_elasticsearch_agent	1	
neo4j-console	1	
opentaxii	1	
kibana	1	
graph-ingestion	1	
platform-api	1	
search-ingestion	1	



Process	Description
`graph-ingestion`	Data funnel to the Neo4j graph database. Handles data updates for Neo4j
`intel-ingestion`	Intel ingestion through feeds and enrichers. Consumes incoming data and saves it PostgreSQL, Neo4j, and Elasticsearch. The platform executes one `intel-ingestion` per processor core. Running tasks are sequentially numbered starting from `0`. For example, a platform instance running on a quad core machine normally executes 4 such processes, progressively numbered from `intel-ingestion:0` to `intel-ingestion:3`
`kibana`	Generates dashboard graphs
`neo4j-batching`	Neo4j graph database batch processing application. It lives on the same server hosting the Neo4j database. It prepares data for ingestion into the Neo4j database
`neo4j-console`	The platform uses Neo4j via this console. Neo4j is available through the platform as a Linux service
`newrelic_elasticsearch_agent`	Starts/Restarts Elasticsearch services, including data logging to monitor component health
`newrelic_postgresql_agent`	Starts/Restarts PostgreSQL services, including data logging
`newrelic_redis_agent`	Starts/Restarts Redis (message broker) services, including data logging
`opentaxii`	TAXII server responsible for STIX data transport

Process	Description
`platform-api`	The web application implementing the platform API and the API endpoints. The endpoints expose services that can be consumed by making API calls and by passing arguments
`search-ingestion`	Search indexer. Handles Elasticsearch data updates
`task`	Celery-managed tasks like enrichers, feed integrations, incoming feed data providers, and utlities

Service	Description
`elasticsearch`	Elasticsearch search and indexing database
`postgresql-9.5`	PostgreSQL (intel database)
`redis`	Redis (message broker)
`logstash`	Log and data aggregation, data pipeline and funneling
`nginx`	Web server

Check system health via the API

You can retrieve system health information also by making an API call to the `/status` API endpoint.

First, make an authentication call to receive the bearer token you need to pass with all subsequent API calls.

API endpoint

API endpoint	`/status`
Create method	`GET`
HTTP headers	`"Content-Type: application/json"`, `"Accept: application/json"`, `"Authorization: Bearer <token>"`

API request	`GET` + `"Content-Type: application/json"` + `"Accept: application/json"` + `"Authorization: Bearer <token>"` + `<platform_host>/status`
API response	`{ "data" : { <status_reponse> } }`

When you make a `GET` request and obtain a JSON object with entity data in the response, the entity object is wrapped in a data wrapper: `{ "data" : { ... } }`.

Status request

```
$ curl -X GET
       -v
       --insecure
       -i
       -H "Content-Type: application/json"
       -H "Accept: application/json"
       -H "Authorization: Bearer <token>"
       https://platform.host/api/status
```

Warning:
About cURL calls

- If you make HTTPs cURL calls to the API *and* you have a self-signed or an invalid certificate, include the `-k` or the `--insecure` parameter in the cURL call to skip the SSL connection CA certificate check.
- Always append a `/` trailing slash at the end of an API URL endpoint. The only exception is `/auth`, which does not take a trailing slash.
- In the cURL call, the `-d` data payload with the entity information always needs to be flat JSON, not hierarchical JSON.
  If you want to pass a hierarchical JSON object, include the `--data-binary` parameter, followed by the path to the JSON file, for example `@/path/to/entity_file.json`.

Status response

```
{
  "data": {

    "health": "GREEN",
```

```
  "process_states": [

    {
      "first_down_at": null,
      "first_up_at": "2016-06-30T11:28:30+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "newrelic_redis_agent"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:03:27+00:00",
      "health": "GREEN",
      "n_processes": 5,
      "n_processes_down": 0,
      "name": "task"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:04:36+00:00",
      "health": "GREEN",
      "n_processes": 4,
      "n_processes_down": 0,
      "name": "intel-ingestion"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:05:07+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "neo4j-batching"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-06-30T11:28:35+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "newrelic_postgresql_agent"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-06-30T11:28:35+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "newrelic_elasticsearch_agent"
    },
```

```
    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:00:14+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "neo4j-console"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-06-30T11:28:35+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "opentaxii"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-01T06:54:15+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "kibana"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:04:58+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "graph-ingestion"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:03:21+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "platform-api"
    },

    {
      "first_down_at": null,
      "first_up_at": "2016-07-06T20:05:04+00:00",
      "health": "GREEN",
      "n_processes": 1,
      "n_processes_down": 0,
      "name": "search-ingestion"
    }

  ],
```

```
    "service_states": [

      {
        "health": "GREEN",
        "name": "postgresql-9.5",
        "state": "active"
      },

      {
        "health": "GREEN",
        "name": "logstash",
        "state": "active"
      },

      {
        "health": "GREEN",
        "name": "elasticsearch",
        "state": "active"
      },

      {
        "health": "GREEN",
        "name": "redis",
        "state": "active"
      }

    ]
  }
}
```

How to monitor the platform

Summary: As a system administrator, you can use tools like Celery and Supervisor to monitor platform tasks to check day-to-day operations and to investigate, in case an issue occurs.

Prerequisites

This article assumes that the following products are installed and configured in a production and/or in a test environment:

- EclecticIQ Platform
- Celery
- Redis
- Supervisord
- systemd

The EclecticIQ Platform runs on CentOS and RHEL; therefore, all command and data output examples apply to these Linux environments.

Scope

Monitor and inspect the status of key platform processes like incoming and outgoing feeds, enrichers, and tasks.
In the current context, monitoring covers on/off status only: the tasks and the commands here allow you to verify whether a task, a process, or a component is running or not. Metrics and other measurements are outside the scope of the topic.

Goal

Allow system administrators and devops engineers to execute quick checks to inspect platform operation, identify any issues and review errors, so that they can address them in a timely manner.

Audience

Tools

Celery	The task runner. It manages task execution and scheduling.
Redis	The message broker. It handles background task processing by managing message queues based on the **pub-sub pattern** (https://en.wikipedia.org/wiki/publish%e2%80%93subscribe_pattern).
Supervisor	The process controller to start and stop processes.
systemd	The initialization system to bootstrap processes and start services.

Core components

Component	Address	Port
nginx	**<server_name>** (http://nginx.org/en/docs/http/ngx_http_core_module.html#server_name)	80; 443
platform	localhost	8008
postgresql	localhost	5432
redis	localhost	6379
neo4j	localhost	7474
elasticsearch	localhost	9200
kibana	localhost	5601
logstash	localhost	6755

The PostgreSQL default database name is `platform-api`.

Monitoring

Platform monitoring covers two main areas:

Components	
`platform-api`	The web application implementing the platform API and the API endpoints. The endpoints expose services that can be consumed by making API calls and by passing arguments
`graph-ingestion`	Data funnel to the Neo4j graph database. Handles data updates for Neo4j
`intel-ingestion`	Intel ingestion through feeds and enrichers. Consumes incoming data and saves it PostgreSQL, Neo4j, and Elasticsearch. The platform executes one `intel-ingestion` per processor core. Running tasks are sequentially numbered starting from `0`. For example, a platform instance running on a quad core machine normally executes 4 such processes, progressively numbered from `intel-ingestion:0` to `intel-ingestion:3`
`opentaxii`	TAXII server responsible for STIX data transport
`postgresql-9.5`	PostgreSQL (intel database)
`redis`	Redis (message broker)
`neo4j-console`	The platform uses Neo4j via this console. Neo4j is available through the platform as a Linux service
`neo4j-batching`	Neo4j graph database batch processing application. It lives on the same server hosting the Neo4j database. It prepares data for ingestion into the Neo4j database
`elasticsearch`	Elasticsearch search and indexing database
`task`	Celery-managed tasks like enrichers, feed integrations, incoming feed data providers, and utlities

Processes	
Feeds	Incoming and outgoing feeds
Enrichers	Enricher tasks
Celery tasks	Other/Misc. Celery tasks

Monitor components with Supervisor

Tool: *Supervisor* is your first and most useful tool to inspect platform components to verify if they are operating normally.
Supervisor configuration files are stored here:

File	Location
`platform-api.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`graph-ingestion.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`intel-ingestion.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`search-ingestion.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`neo4j-batching.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`opentaxii.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`task-workers.ini`	`/opt/eclecticiq/etc/supervisord.d/`
`kibana.ini`	`/opt/eclecticiq/etc-extras/supervisord/`
`neo4j-console.ini`	`/opt/eclecticiq/etc-extras/supervisord/`

Use it to check the following components:

Component	Description	If it is not running…
`graph-ingestion`	Data funnel to the Neo4j graph database. Handles data updates for Neo4j	The graph database may go out of sync and miss data. The `queue:graph:inbound` queue increases in size
`intel-ingestion`	Intel ingestion through feeds and enrichers. Consumes incoming data and saves it PostgreSQL, Neo4j, and Elasticsearch. The platform executes one `intel-ingestion` per processor core. Running tasks are sequentially numbered starting from `0`. For example, a platform instance running on a quad core machine normally executes 4 such processes, progressively numbered from `intel-ingestion:0` to `intel-ingestion:3`	Provider tasks keep running, but no data is displayed in the system. The `queue:ingestion:inbound` queue increases in size

Component	Description	If it is not running…
`search-ingestion`	Search indexer. Handles Elasticsearch data updates	Search indexing stops working correctly. Elasticsearch may go out of sync and miss data. The `queue:search:inbound` queue increases in size
`kibana`	Generates dashboard graphs	The dashboard does not load correctly, and the `/kibana/` API endpoint returns an HTTP 502 error
`neo4j-console`	The platform uses Neo4j via this console. Neo4j is available through the platform as a Linux service	Graph queries stop working, it is not possible to poll the graph database
`neo4j-batching`	Neo4j graph database batch processing application. It lives on the same server hosting the Neo4j database. It prepares data for ingestion into the Neo4j database	Graph ingestion stops working, queries may return results that are not up to date
`opentaxii`	TAXII server responsible for STIX data transport	The TAXII transport type becomes unavailable
`platform-api`	The web application implementing the platform API and the API endpoints. The endpoints expose services that can be consumed by making API calls and by passing arguments	Platform services become unavailable and the API endpoints return an HTTP 502 error
`task`	Celery-managed tasks like enrichers, feed integrations, incoming feed data providers, and utlities	
`task:beat`	Task scheduler	Task execution order may be affected, and tasks may or may not start or stop, resulting in unexpected task execution behavior
`task:enrichers`	Enricher tasks	Enricher data may become only partially available or completely unavailable
`task:integrations`	Outgoing feed integrations	Outgoing feed transport types are affected, and they may stop working correctly or become unavailable

Component	Description	If it is not running…
`task:providers`	Incoming feed data provider tasks	Incoming feed transport types are affected, and they may stop working correctly or become unavailable
`task:utilities`	The little platform elves that keep things tidy while working in the background	The platform may behave unexpectedly. For example data updates and discovery may hang or stop working

`supervisorctl` is Supervisor's command line interface utility. The commands you can pass are called *actions*, and they can optionally take *arguments*:

```
$ supervisorctl <action> <argument>
```

You may need system administrator rights to run some commands.
In this case, prefix `sudo` to the command.

To check if Supervisor is installed, run the following command(s):

```
$ sudo yum info supervisor
```

Supervisor is shipped with the platform. If for some reason you need to install it, run the following command(s):

```
$ sudo yum install supervisor
```

This installs two components:

- `supervisord`: the daemon
- `supervisorctl`: the command line interface.

By default, Supervisor configuration files are stored here:

- Supervisor configuration file: */etc/supervisord.conf*
- Additional `supervisord` configuration files: */etc/supervisord.d/*
  (These files are also symlinked to */opt/eclecticiq/etc/supervisord.d/*.)

To check if the `supervisord` daemon is running, run the following command(s):

```
$ sudo service supervisord status
```

Launch Supervisor to load the configuration to start the tasks and processes other platform components depend on:

```
$ sudo service supervisord start
```

If you need to stop Supervisor, run the following command(s):

```
$ sudo service supervisord stop
```

To check the statuses of the tasks managed by Supervisor, run the following command(s):

```
$ supervisorctl status
```

The response should return `RUNNING` for all relevant tasks to confirm that all Supervisor tasks are being executed normally.

The following example serves as a guideline:

```
graph-ingestion                  RUNNING   pid 3443,  uptime 1:10:56
intel-ingestion:0                RUNNING   pid 3323,  uptime 1:11:19
intel-ingestion:1                RUNNING   pid 3336,  uptime 1:11:14
intel-ingestion:2                RUNNING   pid 3363,  uptime 1:11:09
intel-ingestion:3                RUNNING   pid 3372,  uptime 1:11:04
kibana                           RUNNING   pid 13837, uptime 9 days, 0:38:17
neo4j-batching                   RUNNING   pid 3590,  uptime 1:10:45
opentaxii                        RUNNING   pid 3662,  uptime 1:10:39
platform-api                     RUNNING   pid 2999,  uptime 1:12:47
search-ingestion                 RUNNING   pid 3513,  uptime 1:10:49
task:beat                        RUNNING   pid 3102,  uptime 1:12:39
task:enrichers                   RUNNING   pid 3110,  uptime 1:12:26
task:integrations                RUNNING   pid 3135,  uptime 1:12:13
task:providers                   RUNNING   pid 3204,  uptime 1:12:01
task:reindexing                  RUNNING   pid 3223,  uptime 1:11:48
task:utilities                   RUNNING   pid 3242,  uptime 1:11:35
```

To check the statuses of specific tasks managed by Supervisor, run the following command(s):

```
# Retrieve the status of a specific process
$ sudo supervisorctl status <process_name>

# Retrieve the status of multiple specific processes
$ sudo supervisorctl status <process_name> <process_name> <process_name> ...

# Retrieve the status of all processes
# whose name contains the specified search string
$ sudo supervisorctl status | grep "<search_string>"
```

To reload the Supervisor configuration and to restart all Supervisor-managed processes run the following command(s):

```
$ sudo supervisorctl reload
```

Warning: When you modify or update the Supervisor configuration, you need to run `$ sudo supervisorctl reload` to update it and to reload the up-to-date configuration into the platform.

Supervisor actions

These are examples of `supervisorctl` actions:

Run this...	...to do this
`$ supervisorctl start all`	Start all the processes defined in the Supervisor configuration
`$ supervisorctl start <process_name>`	Start the specified process defined in the Supervisor configuration
`$ supervisorctl start <process_name> <process_name>`	Start the specified processes defined in the Supervisor configuration
`$ supervisorctl stop all`	Stop all the processes defined in the Supervisor configuration file
`$ supervisorctl stop <process_name>`	Stop the specified process
`$ supervisorctl stop <process_name> <process_name>`	Stop the specified processes
`$ supervisorctl status`	Retrieve the statuses of the processes managed by Supervisor. For further information, see the official ocumentation on the **return state values** `(http://supervisord.org/subprocess.html#process-states)`
`$ supervisorctl status <process_name>`	Retrieve the status of the specified process
`$ supervisorctl status <process_name> <process_name>`	Retrieve the status of the specified processes
`$ supervisorctl reload`	Reload the Supervisor configuration and restart all tasks and processes. If you modify or update the Supervisor configuration file, you need run this command to reload the latest Supervisor configuration file
`$ supervisorctl reload all`	Reload all the processes managed by Supervisor. This command works just like `$ supervisorctl reload`

Run this...	...to do this
`$ supervisorctl update`	Update the view after updating the Supervisor configuration
`$ supervisorctl tail <log_file_name>`	Retrieve the most recent lines of the specified log file. To follow the log file as it updates with new information and new lines, run `$ supervisorctl tail -f <log_file_name>`
`$ supervisorctl status | grep "<search_string>"`	Retrieve the status of all processes whose name contains the specified search string.

Monitor components with systemd

Tool: *systemd* helps you inspect platform components to verify if their services are running normally.
Use it to check the following components:

Component	Description	If it is not running…
`postgresql-9.5`	PostgreSQL (intel database)	It is not possible to access platform data
`redis`	Redis (message broker)	Tasks and processes may hang and/or behave unexpectedly
`elasticsearch`	Elasticsearch search and indexing database	No data searching and indexing capabilities are available
`logstash`	Log and data aggregation, data pipeline and funneling	No data aggregation, deduplication, and normalization
`nginx`	Web server	The platform is down

`systemctl` is systemd's command line interface utility. The commands can optionally take options, whereas the component whose service you want to check takes the `.service` suffix:

```
$ systemctl <options> <command> <component_name>.service
```

For a complete list of supported commands and options, see the **systemd documentation** (`https://www.freedesktop.org/software/systemd/man/systemctl.html`).

You may need system administrator rights to run some commands.
In this case, prefix `sudo` to the command.

To obtain a list of all running services, run the following command(s):

```
$ sudo systemctl
```

The response is displayed in the following format:

```
UNIT            LOAD             ACTIVE           SUB                JOB DESCRIPTION
<service_name>  <loaded_or_not>  <active_or_not>  <running_or_not>
<description_of_the_job>
```

To verify if Nginx is running, run the following command(s):

```
$ sudo systemctl status -l nginx.service
```

To verify if PostgreSQL is running, run the following command(s):

```
$ sudo systemctl status -l postgresql-9.5.service
```

To verify if Redis is running, run the following command(s):

```
$ sudo systemctl status -l redis.service
```

To verify if Logstash is running, run the following command(s):

```
$ sudo systemctl status -l logstash.service
```

To verify if Elasticsearch is running, run the following command(s):

```
$ sudo systemctl status -l elasticsearch.service
```

To retrieve status information about all these services at once, run the following command(s):

```
$ sudo systemctl status -l nginx.service postgresql-9.5.service redis.service
logstash.service elasticsearch.service
```

To retrieve status information about all the systemd-managed services whose name contains a specific search string, run the following command(s):

```
$ sudo systemctl | grep  "<search_string>"
```

Monitor processes

Monitor ingestion queues with Redis

Tool: *Redis* acts as a message broker for Celery-managed tasks.

`redis-cli` is Redis's command line interface utility:

```
$ redis-cli <command> <item_name>
```

For a complete list of supported commands and options, see the **Redis command reference** `(http://redis.io/commands)`.

Since Redis's main purpose in this context is to manage task queues, the main and possibly the only command you need is the one that allows you to check queue length: `llen`.

To inspect the platform data ingestion queue length, run the following command(s):

```
$ redis-cli llen "queue:ingestion:inbound"
```

To inspect the graph database queue length, run the following command(s):

```
$ redis-cli llen "queue:graph:inbound"
```

To inspect the Elasticsearch data update queue length, run the following command(s):

```
$ redis-cli llen "queue:search:inbound"
```

Monitor running tasks with Celery

Tool: *Celery* is the task runner that manages task execution and scheduling.

To use Celery to request task information, you need to pass the following environment variables with your request:

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
```

Append Celery commands after the environment variables. Celery commands have the following format:

```
$ celery -A <module_name> <command>
```

Ping Celery to see which tasks are up and listening. This is the easiest way to check task running status. All active tasks reply with `pong`.

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect ping
```

To inspect active tasks, run the following command(s):

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect active
```

To inspect active tasks queues, run the following command(s):

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect
active_queues
```

To inspect scheduled tasks, run the following command(s):

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect
scheduled
```

To inspect overall task status, run the following command(s):

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app status
```

To request task statistics (exhaustive, but it can be verbose), run the following command(s):

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/celery -A eiq.platform.taskrunner.app inspect stats
```

*(For further details, see the documentation on **Celery ping** (http://docs.celeryproject.org/en/latest/userguide/workers.html#ping), **Celery workers** (http://docs.celeryproject.org/en/latest/userguide/workers.html), **Celery worker statistics** (http://docs.celeryproject.org/en/latest/userguide/workers.html#worker-statistics), and **Celery monitoring** (http://docs.celeryproject.org/en/latest/userguide/monitoring.html))*

Check the Flower database size

Warning:

Whoa! This section is *outdated*!

Flower has been deprecated as the monitoring tool for Celery and Celery tasks.
This section is included as reference.

Flower is a web-based tool to **monitor and manage** `(https://flower.readthedocs.io/en/latest/features.html)` Celery. If Celery tasks seem to lag or hang, you may wish to check the size of the Flower database — a very large database may be a symptom of possible queuing issues:

```
$ ls -lah /opt/eclecticiq/platform/flower.db
```

You can safely delete the Flower database file: if no database exists, Flower automatically creates one at startup or restart.
You can try and delete this database to see if the action frees up some resources:

```
$ rm -f /opt/eclecticiq/platform/flower.db
```

*(For further details, see the documentation on **Flower** `(https://flower.readthedocs.io/)`)*

How to enable audit logging in Kibana

Summary: Enable audit logging to examine system events and user access to understand what happened and when, where in the platform, the results it produced, and who/what triggered it.

SysAdmins and SysOps may need to monitor system activities and resource access to troubleshoot issues, investigate a chain of events to identify a root cause, or simply as part of their maintenance routine.

An audit log keeps track of any events that modify the platform configuration or the stored entities. It provides information to identify errors, warnings, or issues affecting specific platform components. It helps you answer these crucial questions:

What?	The running task, action or event (create, delete, update) that produced the outcome you want to investigate.
What?	The modified objects, for example entities or configuration files.
Where?	The affected platform instance (the host system) and the specific area or component, for example incoming feeds, incoming feed tasks, outgoing feed, and so on.
Who?	The user name of the agent that initiated the action.
When?	The time when it occurred (timestamp).

Enable audit logging

By default, audit logging is not configured in the platform.

To enable/disable audit logging, *add* and *set* the following variable to `True` or `False` in the */opt/eclecticiq/etc/eclecticiq/platform_settings.py* platform configuration file:

Variable	Description
`AUDIT_TRAIL_ENABLED = True`	Enables audit logging.
`AUDIT_TRAIL_ENABLED = False`	Disables audit logging.

View audit logs

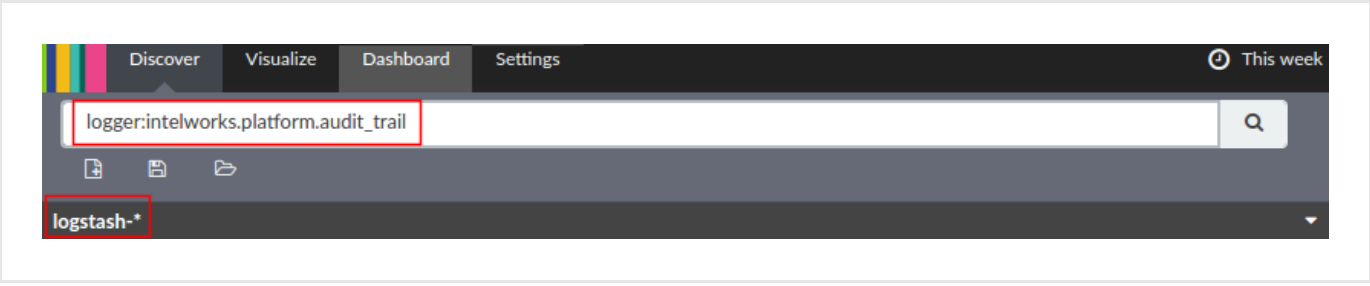
You can view audit logs in Kibana and in the platform web interface.

View audit logs in Kibana

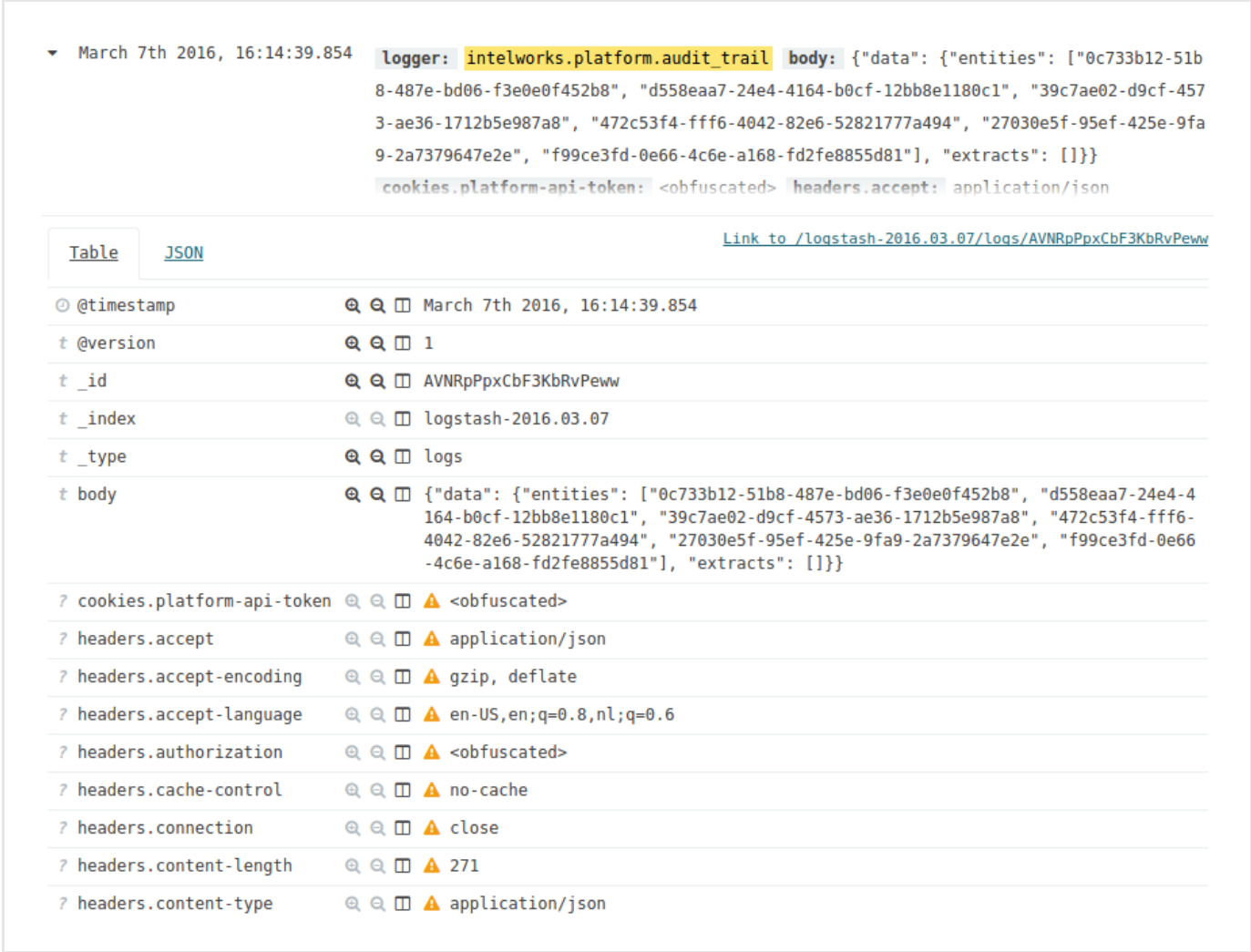
To view audit logs in Kibana, do the following:

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
  `<platform_host_name>/api/kibana/app/kibana#/.`
  Keep the trailing `/`.
  Example: `https://platform.host.com/api/kibana/app/kibana#/`
- In Kibana, select the **Discover** view.
- Make sure **logstash-*** is the active index:

- In the search bar, run a search for *logger:eiq.platform.audit_trail*.
- If necessary, adjust the time interval by clicking the clock icon on the top-right corner, and by choosing an appropriate time range for the search.



- If audit logging is enabled, and if the audit log file is populated, the matching audit log records are returned.



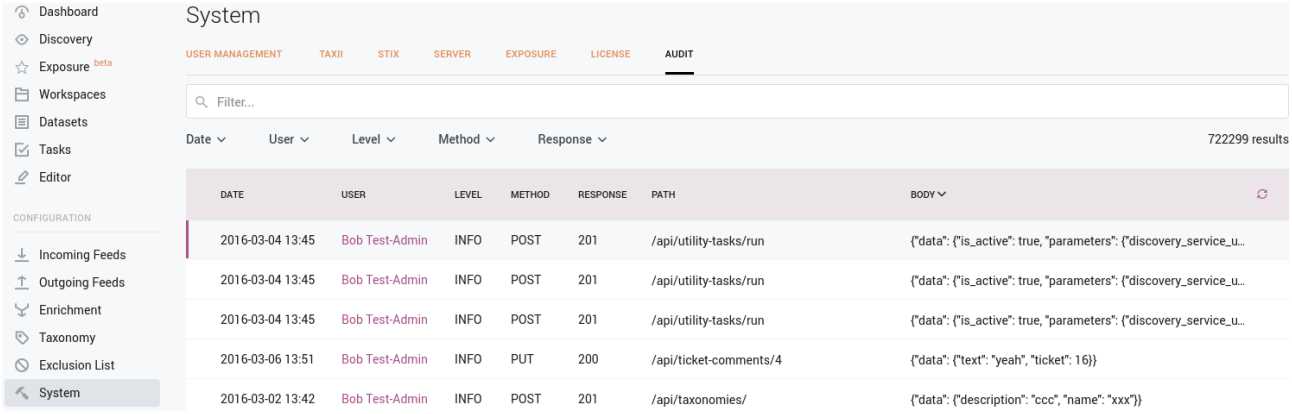
View audit logs in the web interface

To view audit logs in the platform web interface, do the following:

- On the left-hand navigation sidebar click **System**.
- Select the **Audit** tab.
- If audit logging is enabled, and if the audit log file is populated, the matching audit log records are returned. You can sort the table view by column. To do so, click the column header you want to base the data sorting on. An upward-pointing (**^**) or a downward-pointing (**ˇ**) arrow in the header indicates ascending and descending sort order, respectively.
- You can apply filters to narrow down the search scope:

Filter	Description
Date	Displays only the search result items included in the specified time range.
User	Displays only the search result items with the specified user name(s).
Level	Displays only the search result items with the specified message level flag(s): **Info**, **Warning**, **Error**.

Filter	Description
Method	Displays only the search result items with the specified HTTP method(s): **Delete**, **Post**, **Put**.
Response	Displays only the search result items with the specified HTTP response status code(s): 2xx, 4xx, 5xx.



The request payload for an audit log entry cannot exceed 4 KB.
If the request body in the request payload for an audit log entry is larger than 4 KB, the following message is displayed:
Request body too large to log

How to address logging issues in Kibana

Summary: Inspect Kibana and Logstash configurations to identify and troubleshoot logging issues.

Prerequisites

This article assumes that the following products are installed and configured in a production and/or in a test environment:

- EclecticIQ Platform
- Elasticsearch
- Logstash
- Kibana

The EclecticIQ Platform runs on CentOS and RHEL; therefore, all command and data output examples apply to these Linux environments.

The ELK stack

Elasticsearch is the search server powering data indexing and retrieval in the EclecticIQ Platform. Elasticsearch is part of the ELK stack, which our platform implements.

The ELK stack puts three products together to provide exceptional insight into your data.

ELK component	Description
Elasticsearch (https://www.elastic.co/products/elasticsearch)	Data indexing and retrieval; analytics.
Logstash (https://www.elastic.co/products/logstash)	Logging, data normalization, data querying.
Kibana (https://www.elastic.co/products/kibana)	Data visualization.

Check Kibana and Logstash configurations to address logging issues

Access Kibana

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format: `<platform_host_name>/api/kibana/app/kibana#/.`
  Keep the trailing `/`.
  Example: `https://platform.host.com/api/kibana/app/kibana#/`

Check the basics

Before starting to dig into data querying, logging and viewing, make sure the basics are set up and configured.

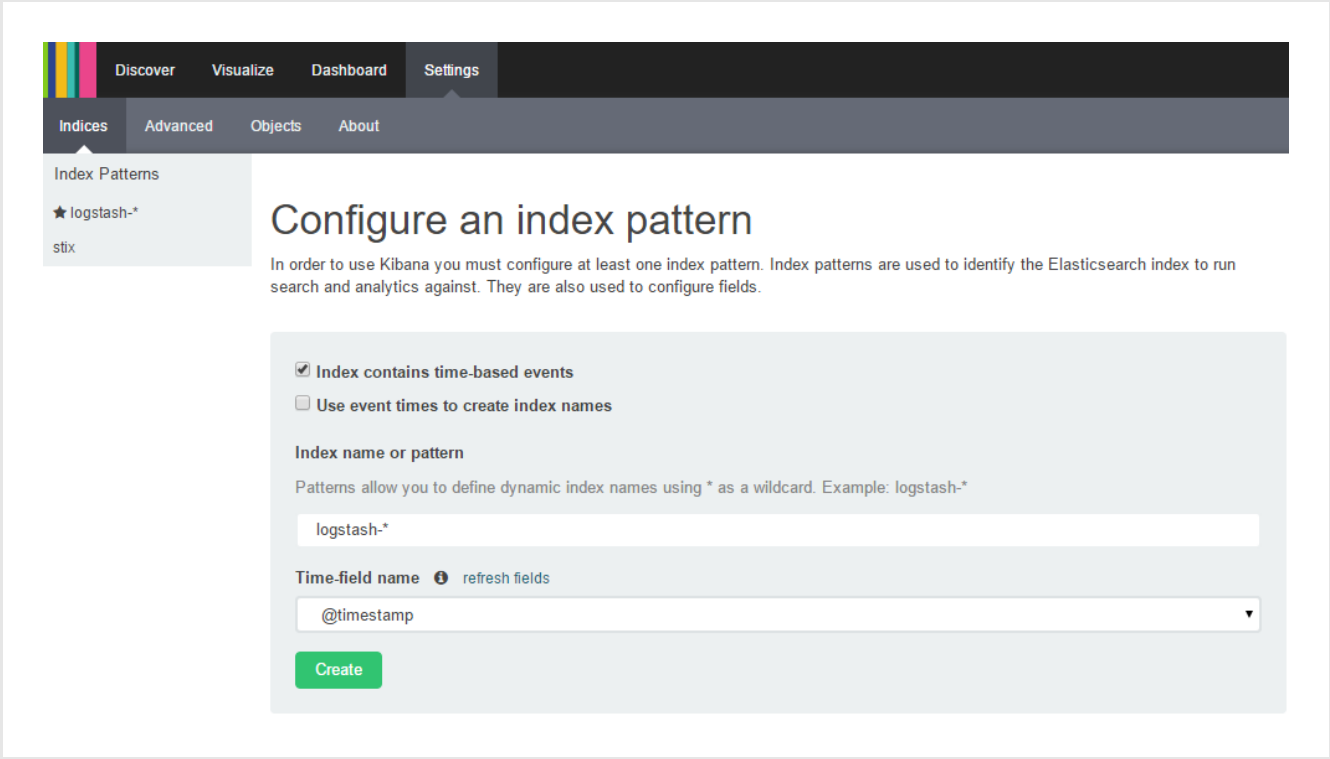
- Make sure the platform dashboard is configured.
- If the Kibana version in use is not exactly the same as the one used to create the dashboard, you need to set the **index pattern** `(https://www.elastic.co/guide/en/kibana/current/tutorial-define-index.html)` for your current Kibana instance.

Create an index

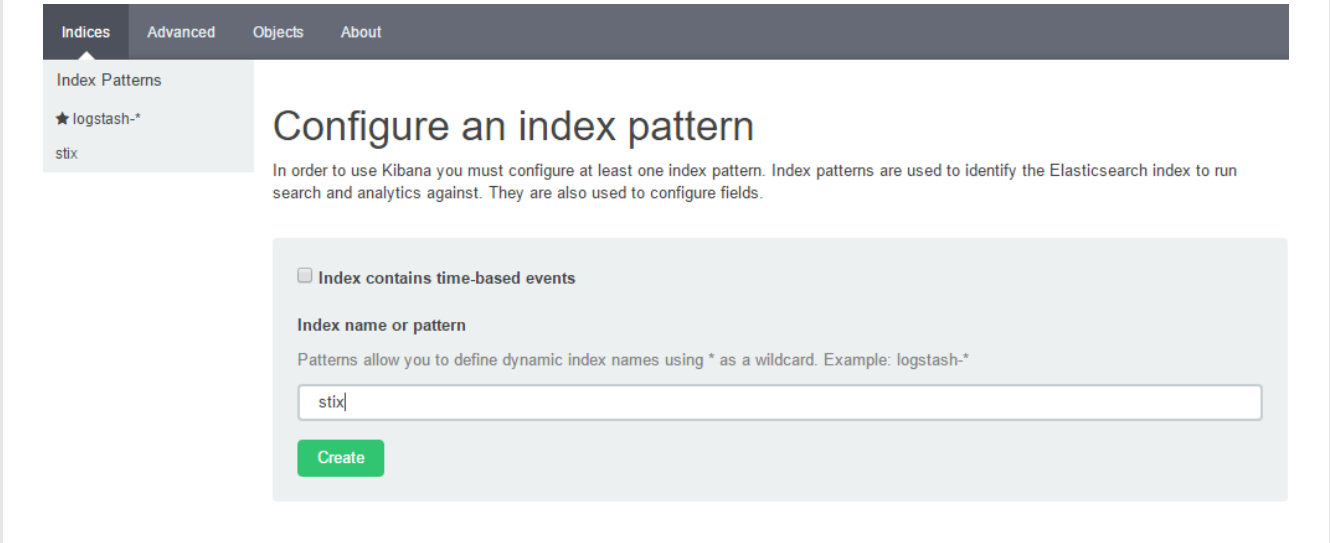
To create an index in Kibana, do the following:

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format: `<platform_host_name>/api/kibana/app/kibana#/.`
  Keep the trailing `/`.
  Example: `https://platform.host.com/api/kibana/app/kibana#/`
- In Kibana, go to **Settings > Indices**.
- Under **Index Patterns**, click **+ Add New**.

- Under **Configure an index pattern**, create the following index patterns:
  - Set **Index name or pattern** to `logstash-*`.
  - Set **Time-field name** to `@timestamp`.
  - Click **Create**.



- Set **Index name or pattern** to `stix`.
- Deselect the **Index contains time-based events** checkbox.
- Leave **Time-field name** empty.
- Click **Create**.



Check Kibana configuration

Kibana uses an index in Elasticsearch to store saved searches, visualizations, and dashboards.
The default property/value pair defining the index in the *kibana.yml* configuration file is `kibana_index: -kibana`.

The default parameters and values should not need any editing:

```
port: 5601
host: "0.0.0.0"
elasticsearch_url: "http://127.0.0.1:9200"
elasticsearch_preserve_host: true
kibana_index: "-kibana"
server.basePath: "/api/kibana"
default_app_id: "discover"
request_timeout: 300000
shard_timeout: 0
verify_ssl: false
bundled_plugin_ids:
 - plugins/dashboard/index
 - plugins/discover/index
 - plugins/doc/index
 - plugins/kibana/index
 - plugins/markdown_vis/index
 - plugins/metric_vis/index
 - plugins/settings/index
 - plugins/table_vis/index
 - plugins/vis_types/index
 - plugins/visualize/index
```

Check Logstash

Logstash is the component taking care of grunt work like data normalization and log generation. It is not possible to access Logstash directly, but it is important that you check it is running and properly configured. To check if Logstash is running, run the following command(s):

```
$ ps -ef | grep logstash
```

If Logstash is running, any active processes belonging to Logstash are returned.

Check Logstash configuration

Logstash configuration files are copied automatically to the correct location during the post-install step.
To verify that the configuration files exist in the target destination, run the following command(s):

```
$ ls -l /opt/eclecticiq/etc/logstash/conf.d/
```

These are the Logstash configuration files related to the platform:

```
filters.conf
input.conf
neo4j-batching.conf
opentaxii.conf
output.conf
platform-api.conf
platform-ui.conf
```

The following files control the **Logstash processing pipeline** (https://www.elastic.co/guide/en/logstash/current/pipeline.html):

```
input.conf
filters.conf
output.conf
```

For further details, see the how-to article on addressing logging issues in Kibana and Logstash configurations.

Check platform-api.conf

- To check the *platform-api.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/platform-api.conf
```

The *platform-api.conf* file holds the paths to core platform logs recording events related to ingestion, graph, search, web server, tasks workers and queues.
Verify that the paths to these log files are correct; edit them, if necessary.

```
input {
  file {
    path => ["/opt/eclecticiq/logs/platform-api.log", "/var/log/nginx/eclecticiq-
platform-api-nginx-access.log", "/opt/eclecticiq/logs/intel-ingestion.log",
"/opt/eclecticiq/logs/graph-ingestion.log", "/opt/eclecticiq/logs/search-
ingestion.log"]
    codec => "json"
    tags => "platform-api"
  }
  file {
    path => ["/opt/eclecticiq/logs/intel-ingestion.log", "/opt/eclecticiq/logs/graph-
ingestion.log", "/opt/eclecticiq/logs/search-ingestion.log"]
    codec => "json"
    tags => "ingestion"
  }
  file {
    path => ["/var/log/nginx/eclecticiq-platform-api-nginx-errors.log"]
    type => "nginx-error"
    tags => "platform-api"
  }
  file {
    path => ["/opt/eclecticiq/logs/task-worker-*.log", "/opt/eclecticiq/logs/task-
beat.log", "/opt/eclecticiq/logs/task-queue-analysis-listener.log",
"/opt/eclecticiq/logs/task-queue-enrichment-listener.log"]
    codec => "json"
    tags => "task-workers"
  }
}
```

platform-ui.conf, *opentaxii.conf*, and *neo4j-batching.conf* are similar to *platform-api.conf*: these files hold the paths to logs recording events related to the web-based user interface, to the TAXII transport type for STIX data, and to graph processing, respectively.
Verify that the paths to these log files are correct; edit them, if necessary.

Check elasticsearch.yml

You check this file to make sure that the `cluster.name` property is correctly set.
You may need to access the resource by prefixing `sudo su`, so that you can obtain root privileges.

- To open the *elasticsearch.yml* configuration file, run the following command(s):

```
$ sudo su
$ nano /etc/elasticsearch/elasticsearch.yml
```

The `config.cluster.name` property value should be `intel`:

```
config:

  cluster.name: intel

  discovery.zen.ping.multicast: false

  index.number_of_replicas: 0

  index.number_of_shards: 1

  node.local: true

  path:

    data: /<path_to>/<elasticsearch_index_datastore>

  script.indexed: true

  script.inline: false
```

Check Logstash event pipeline

`input.conf`, `filter.conf`, and `output.conf` are the configuration files that control the Logstash **event processing pipeline** `(https://www.elastic.co/guide/en/logstash/current/pipeline.html)`:

- `input.conf` controls data ingestion into Logstash.
- `filter.conf` controls data manipulation like filtering, parsing and tranformations.
- `output.conf` controls data output, for example, by sending the processed data on to Elasticsearch.

Check input.conf

- To check the *input.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/input.conf
```

input.conf holds the input configuration for **Elasticsearch Curator** `(https://www.elastic.co/guide/en/elasticsearch/client/curator/current/index.html)`, the Elasticsearch index manager.
You should not need to edit this file.

```
input {
  tcp {
    codec => json
    type => "curator"
    port => "28778"
  }
}
```

Check filters.conf

- To check the *filters.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/filters.conf
```

Logstash filters (https://www.elastic.co/guide/en/logstash/current/config-examples.html) are like a Swiss-army knife to slice and dice your data using an array of **filter plugins** (https://www.elastic.co/guide/en/logstash/current/filter-plugins.html).
filters.conf defines filters as needed to select specific indices or data types based on the specified patterns. You can edit this file to match your requirements.
This *filters.conf* file is just an example that you can customize as needed:

```
filter {

  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOG5424PRI}%{NONNEGINT:ver} +(?:%
{TIMESTAMP_ISO8601:ts}|-) +(?:%{HOSTNAME:containerid}|-) +(?:%
{NOTSPACE:containername}|-) +(?:%{NOTSPACE:proc}|-) +(?:%{WORD:msgid}|-) +(?:%
{SYSLOG5424SD:sd}|-|) +%{GREEDYDATA:msg}" }
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
    if !("_grokparsefailure" in [tags]) {
      mutate {
        replace => [ "@source_host", "%{syslog_hostname}" ]
        replace => [ "@message", "%{syslog_message}" ]
      }
    }
    mutate {
      remove_field => [ "syslog_hostname", "syslog_message", "syslog_timestamp" ]
    }
  }

}
```

Check output.conf

- To open the *output.conf* configuration file, run the following command(s):

```
$ nano /opt/eclecticiq/etc/logstash/conf.d/output.conf
```

output.conf routes Logstash data downstream in the system toolchain. **Output plugins** `(https://www.elastic.co/guide/en/logstash/current/output-plugins.html)` send data in a predefined format to a specific destination.
Codec plugins `(https://www.elastic.co/guide/en/logstash/current/codec-plugins.html)` read and represent data in specific formats.

Within the platform environment, you can use the reference *output.conf* provided:

- It outputs Logstash data to the default Elasticsearch instance configured for the platform.
  By default, the Elasticsearch host is `localhost`, and the default Elasticsearch port is `9200`.
- `stdout` **encodes the data** `(https://www.elastic.co/guide/en/logstash/current/plugins-outputs-stdout.htmlencodes)` before outputting it to the shell standard output. You may edit it as needed.

```
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Test Logstash configuration

- To test the Logstash configuration, run the following command(s):

```
$ /opt/logstash/bin/logstash --configtest --config /etc/logstash/conf.d/
```

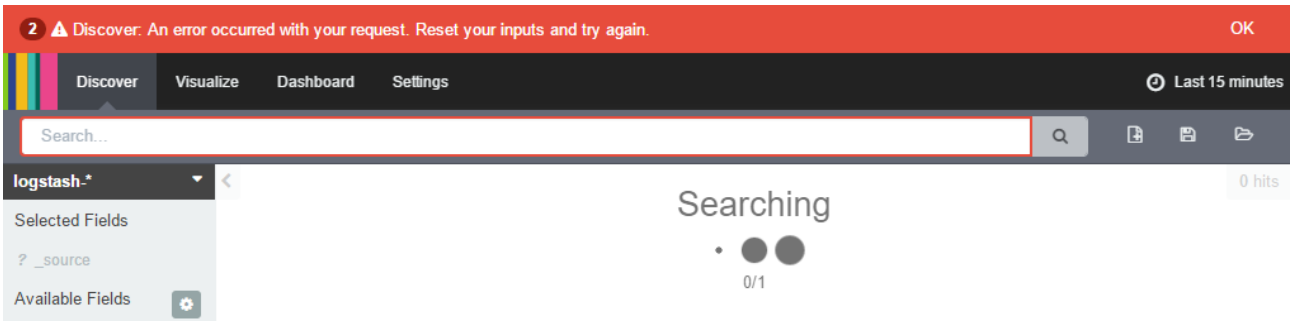
- If the configuration test result is positive, the response reads:

```
Configuration OK
```

Go back to Kibana

After inspecting the Kibana and Logstash configurations and editing them, if necessary, you can go back to Kibana, and you can select **Discover** to view all log data recorded in the last 15 minutes.

It may happen that you are denied access to the Kibana **Discover** view:

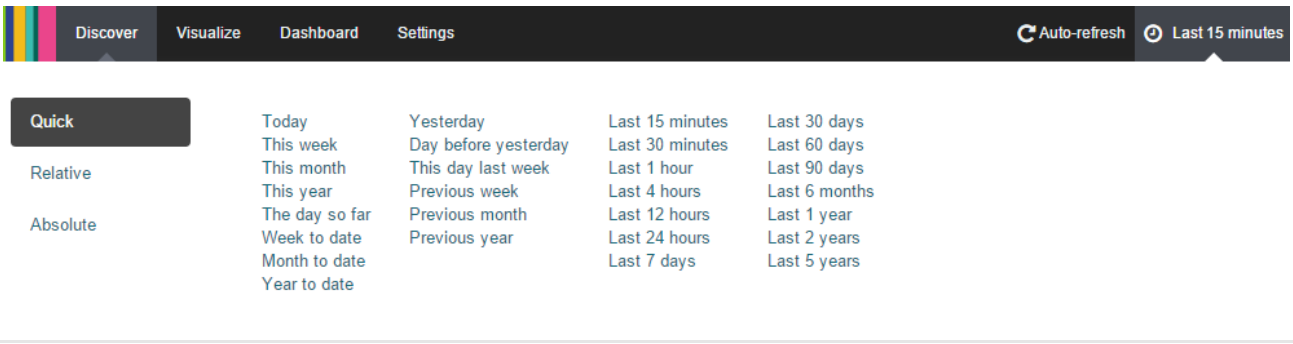


This is usually due to an access timeout. To log back in to Kibana, do the following:

- Go back to the platform login page.
- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format: `<platform_host_name>/api/kibana/app/kibana#/`.
  Keep the trailing `/`.
  Example: `https://platform.host.com/api/kibana/app/kibana#/`

Adjust the update interval

By default, the log data update interval is set to 15 minutes. You can adjust it to match your requirements by clicking it, and then by choosing the appropriate value among the available options.



Configure output to syslog

You can use this plugin to send Logstash logs to syslog to be output: **Logstash output syslog plugin** `(https://www.elastic.co/guide/en/logstash/current/plugins-outputs-syslog.html)`.

How to search logs for issues in Kibana

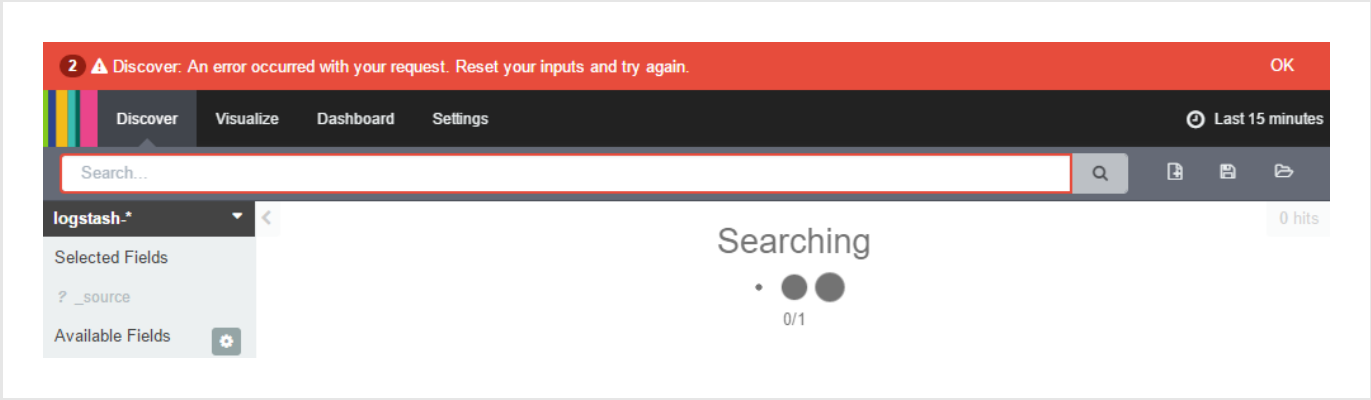
Summary: Search Kibana to retrieve log data about errors, warnings, or issues concerning specific platform components.

Search Kibana to retrieve log data about issues

Access Kibana

- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
  `<platform_host_name>/api/kibana/app/kibana#/.`
  Keep the trailing `/`.
  Example: `https://platform.host.com/api/kibana/app/kibana#/`

It may happen that you are denied access to the Kibana **Discover** view:

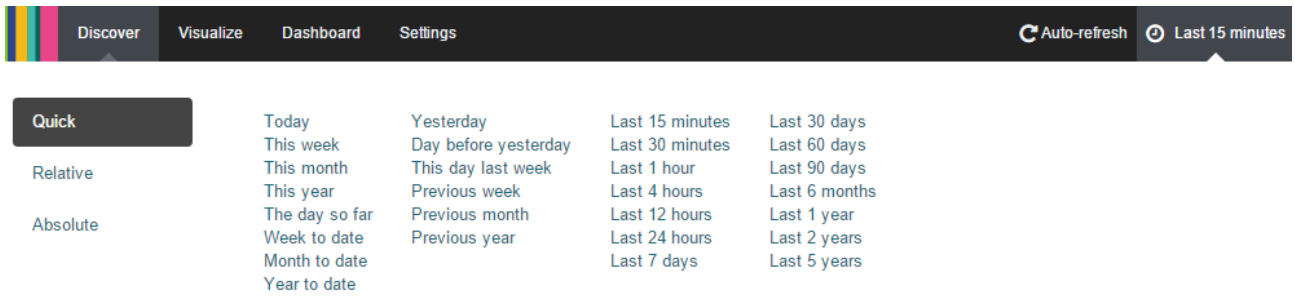


This is usually due to an access timeout. To log back in to Kibana, do the following:

- Go back to the platform login page.
- Sign in to the platform with your user credentials.
- To access Kibana, enter in the web browser address bar a URL with the following format:
  `<platform_host_name>/api/kibana/app/kibana#/.`
  Keep the trailing `/`.
  Example: `https://platform.host.com/api/kibana/app/kibana#/`

Adjust the update interval

By default, the log data update interval is set to 15 minutes. You can adjust it to match your requirements by clicking it, and then by choosing the appropriate value among the available options.



Search by level

Log records are assigned a severity level. In this way, you can filter searches to retrieve only errors, warnings, or informative/notification log records.

To run a search by level in Kibana, do the following:

- In Kibana, select the **Discover** view.
- Make sure **logstash-*** is the active index:

Enter this in the Kibana search bar…	**…to obtain this result**
`level:"error"`	Returns all logging errors.
`level:"warning"`	Returns all logging warnings.
`level:"info"`	Returns all logging information and notifications.

Search by tag

Log records are tagged, so that you can easily identify the component a log refers to. To run a search by tag in Kibana to look for issues or information about a specific component, do the following:

- In Kibana, select the **Discover** view.
- Make sure **logstash-*** is the active index:

In the Kibana search bar enter this…	…to get this result
`tags.raw:"ingestion"`	Returns logs related to the intel ingestion module controlling how the platform ingests source data.
`tags.raw:"opentaxii"`	Returns logs related to the EclecticIQ OpenTAXII server implementing the TAXII services.
`tags.raw:"platform-api"`	Returns logs related to the EclecticIQ Platform API.
`tags.raw:"task-workers"`	Returns logs related to the services and processes carrying out complementary tasks.

Search using Boolean operators

You can use Boolean operators to refine your search and zero in on specific log types or components.
For example you can search for Redis and/or Neo4j issues by entering in the Kibana search bar the following query:

```
level:"error" AND tags.raw:"graph-ingestion"
```

You can search for Redis and/or Neo4j issues, or for issues concerning how data is ingested, by entering in the Kibana search bar the following query:

```
level:"error" AND tags.raw:"graph-ingestion" OR tags.raw:"intel-ingestion"
```

How to setup Nginx client certificate verification

Summary: Set up and configure SSL client certificate verification in Nginx.

Enable client certificate verification in Nginx

Nginx supports client certificate verification through these two configuration options:

- ssl_client_certificate
- ssl_verify_client

Example:

```
server {
  listen      443;
  ssl         on;
  server_name example.com;

  ...

  ssl_certificate        /etc/nginx/certs/server.crt;
  ssl_certificate_key    /etc/nginx/certs/server.key;
  ssl_client_certificate /etc/nginx/certs/ca.crt;
  ssl_verify_client      on;

  ...

}
```

The `ca.crt` file is the public key part of a certificate with which the client certificates have been signed. This file should be provided to you by those managing the certificate authority.

Install a client certificate in Google Chrome

To install a client certificate in Google Chrome, do the following:

- Go to **Settings > Show advanced settings.. > HTTPS/SSL > Manage certificates**
- On the **Your certificates** tab, import your client certificate.

To set up a Certificate Authority and generate client and server certificates, we recommend OpenSSL.

How to back up and restore a PostgreSQL database

Summary: Back up and restore a PostgreSQL database, for example after upgrading or reinstalling the platform, or as a disaster recovery mitigation strategy.

The exact steps to back up platform data may vary, depending on your sepcific environment hardware, configuration, and setup. Therefore, consider the following as a set of generic guidelines on backing up platform data.

Back up the PostgreSQL database

The quickest way to backup a PostgreSQL database is to perform an **SQL dump** (http://www.postgresql.org/docs/current/static/backup-dump.html). To generate an SQL dump of the database, run the **pg_dump** (http://www.postgresql.org/docs/current/static/app-pgdump.html) or the **pg_dumpall** (http://www.postgresql.org/docs/current/static/app-pg-dumpall.html) command.

To create an *.sql* dump file, run the following command(s):

```
$ sudo -u postgres /usr/pgsql-9.5/bin/pg_dumpall > /output/db_dump.sql
```

To restore a backed up database from an *.sql* dump file, run the following command(s):

```
$ psql -U postgres < ./db_dump.sql
```

Restore the backup

To restore backed up data, follow the standard recommendations and procedures for PostgreSQL:

- **Back up and restore PostgreSQL data** (https://www.postgresql.org/docs/current/static/backup.html)
- **Restore a PostgreSQL data dump** (https://www.postgresql.org/docs/current/static/backup-dump.html#backup-dump-restore)
- **Restore PostgreSQL data using a continuous archive backup** (https://www.postgresql.org/docs/current/static/continuous-archiving.html#backup-pitr-recovery)
- **Restore PostgreSQL using pg_restore** (http://postgresguide.com/utilities/backup-restore.html)

Check, check, check

Check the PostgreSQL configuration

If you upgraded PostgreSQL along with the platform, make sure the port assigned to PostgreSQL has not changed, compared to the previous version:

Go to the */var/lib/pgsql//data/* to review the *postgresql.conf* configuration file:

```
$ sudo -e cd /var/lib/pgsql/<version_number>/data/

$ nano postgresql.conf
```

Make sure that the `port` value has not changed between versions. By default, PostgreSQL uses port `5432`:

```
listen_addresses = '*'
port = 5432
max_connections = 100
log_destination = 'stderr'
logging_collector = on
log_directory = '/var/log/postgresql'
log_filename = 'postgresql-%Y-%m-%d.log'
log_truncate_on_rotation = on
log_rotation_age = 1d
log_rotation_size = 0
log_line_prefix = '%m '
log_timezone = 'UTC'
datestyle = 'iso, mdy'
timezone = 'UTC'
lc_messages = 'C'
lc_monetary = 'C'
lc_numeric = 'C'
lc_time = 'C'
default_text_search_config = 'pg_catalog.english'

# Increase allowed memory usage.
shared_buffers = 1024MB
work_mem = 8MB

# Random page lookups are cheap on SSD disks.
random_page_cost = 1.5

# Maintenance operations are usually not performed concurrently, so
# they can use significantly more memory to speed things up.
maintenance_work_mem = 1GB
```

If you upgraded PostgreSQL from a previous version to a newer one, and then restored the database SQL dump backup copy, you can remove the previous PostgreSQL version.

Check the platform configuration

To make sure the platform sees the upgraded PostgreSQL database, verify that the `SQLALCHEMY_DATABASE_URI` parameter in the *platform_settings.py* platform configuration file points to the correct database and to the correct address/port – the `<db_password>` parameter in the URI should be pre-populated:

```
$ cd opt/eclecticiq/etc/eclecticiq/

$ nano platform_settings.py
```

```
SQLALCHEMY_DATABASE_URI = "postgresql://platform-api:
<db_password>@localhost:5432/platform-api"
```

Check the PostgreSQL service

Now you can launch the platform and check if the systemd-managed services are running to verify that the upgraded PostgreSQL is up and running as well. If the upgraded version uses the same port as the previous one, the platform should pick it up when it launches.

You can check if the correct version is up and running:

```
$ sudo systemctl status | grep "<version_number>"
```

Alternatively, you can check if the `postgres` service is up and running. This this command returns a more verbose response:

```
$ sudo systemctl status | grep "postgres"
```

Check for failed packages

During a data restore operation, you may wish to check if the scenario that created the need for a data backup restore also caused some packages to be partially or incorrectly ingested into the platform.

The blobs associated with problematic packages are not marked a successful. To retrieve a list with these packages, execute the following SQL query against the PostgreSQL database:

```
$ SELECT id, processing_status FROM blob WHERE processing_status NOT IN ('success',
'pending');
```

The query returns all packages whose status is not `success` or `pending`, as well as additional information on the packages, when available. Examine the response to evaluate whether you want to try reingesting the packages again.

How to configure a different database in OpenTAXII

Summary: By default, OpenTAXII uses SQLite as a database. You can change this setting to configure a different database, for example PostgreSQL.

OpenTAXII ships with SQLite as its default database. While this is a suitable choice for a development environment where you can test functionality and tinker away to your heart's delight, you may wish to opt for a more powerful solution when you decide to deploy to a production environment that requires added functionality like user management or extensibility and scalability.

PostgreSQL is our recommended database for production. To configure OpenTAXII to work with PostgreSQL, all you need to do is set the `db_connection` parameter of the *opentaxii.yml* configuration file so that it points to a valid PostgreSQL instance. For a connection to a PostgreSQL database, the `db_connection` parameter value observes the following format:

```
db_connection: postgresql://<user_name>:<password>@<db_server>:<port_number>/<db_name>
```

In the following example, the `db_connection` parameter of the *opentaxii.yml* configuration file points to a sample PostgreSQL instance.

> ✔ All parameter values in the example(s) are dummy values.
> Replace these sample values with actual, valid ones before testing them in your environment.

```
domain: "test.taxiistand.com"

auth_api:
  class: opentaxii.auth.sqldb.SQLDatabaseAPI
  parameters:
    db_connection:
postgresql://username:password@postgresql.server.com:5432/databasename
    create_tables: yes
    secret: som3_s00p3r_s3cr3t_k3y

persistence_api:
  class: opentaxii.persistence.sqldb.SQLDatabaseAPI
  parameters:
    db_connection:
postgresql://username:password@postgresql.server.com:5432/databasename
    create_tables: yes

logging:
  opentaxii: info
  root: info
```

Last generated on Mar 8, 2017

How to reindex Elasticsearch

Summary: You may need to reindex Elasticsearch for several reasons: from changes to data types or data analysis, to updating the Elasticsearch schema by adding or removing fields. Whenever a change in the data structure is introduced, you should reindex Elasticsearch to make sure existing and new data is correctly indexed and stored in the Elasticsearch database.

Reindex Elasticsearch

Depending on the size of the Elasticsearch database and the workload on the system, you may want to consider stopping some non-core platform processes — for example ingestion queues — before reindexing to free up system resources.

- To reindex Elasticsearch, you first need to define `elasticsearch` as the designated superuser to execute the reindexing command with. Run the following command(s):

```
$ sudo -u elasticsearch -i
```

- To launch the Elasticsearch database reindexing, run the following command(s):

```
$ /opt/eclecticiq/platform/api/bin/manage reindex_elasticsearch
```

Fully reindex Elasticsearch

To make sure you are applying the latest Elasticsearch schema, fully reindex Elasticsearch.
To do so, run the same command you would execute to create the Elasticsearch index.

- Configure the platform settings environment variable by prepending it to the `manage synchronize_elasticsearch` command.
- Run `manage synchronize_elasticsearch` to reindex all Elasticsearch indexes.
  The command runs in the background. In case of an SSH disconnection, the process should keep running normally.

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/manage synchronize_elasticsearch > ./reindexing.log
2>&1 &
```

- Reindexing messages are logged to *reindexing.log*. You can open this file to inspect the scheduling of the reindexing workers.
  Example:


```
{
  "avg_time_per_object_ms": 7,
  "chunk_time_ms": 57838,
  "event": "search_reindexing.processed_chunk",
  "level": "info",
  "logger": "eiq.platform.search_index",
  "n_processed": 7980,
  "request_body_size": 3349388,
  "timestamp": "2016-08-05T14:31:27.626829Z",
  "total_avg_time_per_object_ms": 17
}
```

How to reindex the graph database

Summary: You may need to reindex the graph database for several reasons: from changes to data types or data analysis, to updating the data schema by adding or removing fields. Whenever a change in the data structure is introduced, you should reindex the graph database to make sure existing and new data is correctly indexed and stored.

Depending on the size of your database, *reindexing the graph database can take from several hours to a few days*.
It is a good idea to run `reindex_graph` in a separate screen session, so that you can monitor the process.

The `reindex_graph` command adds and enqueues entities, enrichments, and extracts stored in the platform to the graph ingestion queue:

```
$ /opt/eclecticiq/platform/api/bin/manage reindex_graph -o <offset_value> -q
<queue_name>
```

Example:

```
$ /opt/eclecticiq/platform/api/bin/manage reindex_graph -o 0 -q queue:graph:inbound
```

Parameter	Description
`-?`	Shows the built-in help.
`-i <item_type>`	Ex.: `entity`. Allowed values/Supported types: `entity`, `enrichment`, `extract`. When no item type is specified, the command reindexes all graph items of all supported item types.
`-q <queue_name>`	Ex.: `queue:graph:inbound`. The queue to reindex and ingest again. If no queue is specified, the default queue for the process is the graph ingestion queue: `queue:graph:inbound`.
`-o <offset_value>`	Ex.: `-o 10`. Offset value, integer. Default value: `0`. Typically, graph reingestion and reindexing selects all the entities IDs stored in the platform, sorts them in ascending order by `created_at`, and then adds all the IDs to the graph ingestion queue. If an offset value is defined, the process ignores the first *n* entities corresponding to the specified offset.

Parameter	Description
`-l <limit_value>`	Ex.: `-l 200`. Capping value, integer. Default value: `100000`. Typically, graph reingestion and reindexing selects all the entities IDs stored in the platform, sorts them in ascending order by `created_at`, and then adds all the IDs to the graph ingestion queue. If a limit value is defined, only the first *n* entities corresponding to the specified limit are processed.

How to upgrade Elasticsearch and Kibana

Summary: Elasticsearch and Kibana are two third-party products EclecticIQ Platform uses. Elasticsearch was upgraded from version 1.7.1 to version 2.3.3, and Kibana was upgraded from version 4.1.1 to version 4.5.1. Follow these steps to migrate to the new versions of these products.

The EclecticIQ Platform upgraded the following third-party components:

- *Elasticsearch* from version 1.7.1 to version 2.3.3.
- *Kibana* from version 4.1.1 to version 4.5.1.
- Elasticsearch 2.3.x requires the **delete-by-query plugin** (https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html).

Both upgrades are signifcant releases introducing several enhancements that improve performance and stability.

The Elasticsearch upgrade to version 2.3.x introduces *breaking changes*; therefore, it is a good idea to look up the relevant official documentation before starting the procedure:

- **Elasticsearch 2.3.3 release notes** (https://www.elastic.co/guide/en/elasticsearch/reference/2.3/release-notes-2.3.3.html)
- **Breaking changes in Elasticsearch 2.3.x** (https://www.elastic.co/guide/en/elasticsearch/reference/current/breaking-changes-2.3.html)
- **Kibana 4.5 release notification** (https://www.elastic.co/blog/kibana-4-5-0-released)
- **Kibana 4.5 release notes** (https://www.elastic.co/guide/en/kibana/current/releasenotes.html)

Before you start

Warning: Before proceeding to upgrade the platform or any of its third-party components, always back up your data.

Check prerequisites

To install the correct/required versions of these products, do the following:

- Add the product repositories to */etc/yum.repos.d/*.
- To install or update the products, run `yum update`.

Check repositories

Elasticsearch repository
(https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-repositories.html):

```
[elasticsearch-2.x]
name=Elasticsearch repository for 2.x packages
baseurl=https://packages.elastic.co/elasticsearch/2.x/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

Alternatively, you can download Elasticsearch from the Elastic web site:
https://www.elastic.co/downloads/elasticsearch
(https://www.elastic.co/downloads/elasticsearch) .

Do some house cleaning

Before upgrading Elasticsearch to version 2.3.3, do the following:

- Shut down the platform.
- **Remove** (https://www.elastic.co/guide/en/elasticsearch/plugins/current/listing-removing.html) any unused plugins:
  - **mobz/elasticsearch-head** (https://github.com/mobz/elasticsearch-head)
  - **codelibs/elasticsearch-reindexing** (https://github.com/codelibs/elasticsearch-reindexing).
- Elasticsearch 2.3.x requires the **delete-by-query plugin** (https://www.elastic.co/guide/en/elasticsearch/plugins/2.3/plugins-delete-by-query.html).
  To install *delete-by-query*, run the following command(s):

```
$ /usr/share/elasticsearch/bin/plugin install delete-by-query
```

- Delete all existing Elasticsearch indexes by running the following command(s):

```
$ curl -XDELETE 'http://localhost:9200/*/'
```

Upgrade

If you have already added the necessary repositories to `/etc/yum.repos.d/`, you can upgrade Elasticsearch by running the `yum update` command.

As for Kibana, the recommended way to upgrade is by downloading a Kibana tarball, unpacking it to the Kibana target directory, and then copying the Kibana configuration from the previous installation.

You may need system administrator rights to run some commands.
In this case, prefix `sudo` to the command.

- Download the Kibana 4.5.1 tarball:

```
$ wget https://download.elastic.co/kibana/kibana/kibana-4.5.1-linux-x64.tar.gz
```

- Decompress the tarball content and save it to a local temporary directory:

```
$ tar -xaf kibana-4.5.1-linux-x64.tar.gz
```

- If Kibana is running, stop it:

```
$ sudo supervisorctl stop kibana
```

- You can check to make sure Kibana has stopped:

```
$ sudo supervisorctl status kibana
```

- After stopping Kibana, move the old installation to a backup directory, for example `kibana-old`:

```
$ mv /opt/kibana /opt/kibana-old
```

- Go to the local temporary directory where you saved the decompressed content of the Kibana tarball, and then move `kibana-4.5.1-linux-x64` to the Kibana installation directory, typically `/opt/kibana`:

```
$ mv kibana-4.5.1-linux-x64 /opt/kibana
```

- Copy the Kibana configuration file rom the `kibana-old` backup directory to the Kibana installation directory:

```
$ cp /opt/kibana-old/config/kibana.yml /opt/kibana/config/kibana.yml
```

- Assign ownership of the `/opt/kibana` to the `eclecticiq` user in the `eclecticiq` group.
  Execute the command recursively:

```
$ chown -R eclecticiq:eclecticiq /opt/kibana
```

- Start Kibana :

```
$ sudo supervisorctl start kibana
```

- Verify that Kibana is running:

```
$ sudo supervisorctl status kibana
```

- After making sure Kibana is running, you can safely remove the backup directory with the previous Kibana installation:

```
$ rm -rf /opt/kibana-old
```

For further details, see the official product documentation:

- **Upgrading Elasticsearch**
  `(https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-upgrade.html)`
- **Getting Kibana Up and Running**
  `(https://www.elastic.co/guide/en/kibana/current/setup.html )`

After the upgrade

Update the config files

After completing the upgrade procedure, you need to check and edit the following configuration files:

- `/opt/eclecticiq/etc/eclecticiq/platform_settings.py`:
  make sure the following property is included in the configuration, and that its value is set to the correct Kibana version.
  The value needs to match the Kibana version you are installing or upgrading to.
  For example, if you upgrade to Kibana 4.5.1, assign this value to the property:

```
KIBANA_VERSION = '4.5.1'
```

- `/opt/eclecticiq/etc-extras/elasticsearch/elasticsearch.yml`:
  disable **dynamic scripting** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html#enable-dynamic-scripting)` by either removing the `script.inline` property from the configuration file, or by setting it to `false`:

```
script.inline: false
script.indexed: true
```

- `/etc/sysconfig/elasticsearch`:
  remove the following property from the configuration file:

```
CONF_FILE=/etc/elasticsearch/elasticsearch.yml
```

- `/opt/eclecticiq/etc-extras/kibana.yml`:
  add the following **property** `(https://www.elastic.co/guide/en/kibana/current/kibana-server-properties.html)` to the configuration file:

```
server.basePath: "/api/kibana"
```

Start the reindexing process

At the end of the Elasticsearch upgrade, you need to fully reindex the platform. Make sure you plan the upgrade so that you can factor in enough downtime for reindexing. Depending on the size of your database, *it can take from several hours to a few days*.

- To keep the Elasticsearch mapping structure up to date in case resources are remapped, start the reindexing task:

```
$ sudo supervisorctl start task:reindexing
```

- Reindexing worker log messages are logged to */opt/eclecticiq/logs/task-worker-reindexing.log*.

Reindex Elasticsearch

After updating the configuration files, do the following:

- Configure the platform settings environment variable by prepending it to the `manage synchronize_elasticsearch` command.
- Run `manage synchronize_elasticsearch` to reindex all Elasticsearch indexes.
  The command runs in the background. In case of an SSH disconnection, the process should keep running normally.

```
$ EIQ_PLATFORM_SETTINGS=/opt/eclecticiq/etc/eclecticiq/platform_settings.py
/opt/eclecticiq/platform/api/bin/manage synchronize_elasticsearch > ./reindexing.log
2>&1 &
```

- Reindexing messages are logged to *reindexing.log*. You can open this file to inspect the scheduling of the reindexing workers.
  Example:

```
{
  "avg_time_per_object_ms": 7,
  "chunk_time_ms": 57838,
  "event": "search_reindexing.processed_chunk",
  "level": "info",
  "logger": "eiq.platform.search_index",
  "n_processed": 7980,
  "request_body_size": 3349388,
  "timestamp": "2016-08-05T14:31:27.626829Z",
  "total_avg_time_per_object_ms": 17
}
```

Run elasticdump

Once reindexing is successfully completed, run `elasticdump` to load the platform dashboard:

```
# Run elasticdump to load the platform dashboard
$ elasticdump --input=/opt/eclecticiq/etc/kibana-dashboard.json --
output=http://localhost:9200/-kibana
```

For further details on `elasticdump` usage, see the **official product documentation** `(https://www.npmjs.com/package/elasticdump#use).`

You can now start also the other platform processes by running the following command(s):

```
$ sudo supervisorctl start all
```

How to upgrade to Neo4j 2.3.1

Summary: Neo4j is one of the third-party products EclecticIQ Platform uses. It was upgraded from version 2.2.4 to version 2.3.1. Follow these steps to migrate to the new version of the graph database.

The EclecticIQ Platform upgraded its graph database to **Neo4j** `(http://neo4j.com/)` 2.3.1. Although 2.3.1 is is a maintenance release, version 2.3.0 is a significant release, since it introduces several enhancements that improve performance and stability.

- Official Neo4j **release** `(http://neo4j.com/release-notes/neo4j-2-3-1/)` **notes** `(http://neo4j.com/release-notes/neo4j-2-3-0/)`
- Official Neo4j **changelog** `(https://github.com/neo4j/neo4j/wiki/neo4j-2.3-enterprise-changelog#231)`

Check the current version

Check the currently installed Neo4j version:

```
$ sudo yum info neo4j
```

Check prerequisites

Neo4j depends on the ELK stack: Elasticsearch, Logstash, and Kibana. To install the correct/required versions of these products, do the following:

- Add the product repositories to */etc/yum.repos.d/*.
- To install or update the products, run `yum update`.

Repositories

Elasticsearch

Elasticsearch repository (https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-repositories.html):

```
[elasticsearch-2.x]
name=Elasticsearch repository for 2.x packages
baseurl=https://packages.elastic.co/elasticsearch/2.x/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

Alternatively, you can download Elasticsearch from the Elastic web site: **https://www.elastic.co/downloads/elasticsearch** (https://www.elastic.co/downloads/elasticsearch) .

Logstash

Logstash **repository** (https://www.elastic.co/guide/en/logstash/current/installing-logstash.html):

```
[logstash-2.3]
name=Logstash repository for 2.3.x packages
baseurl=https://packages.elastic.co/logstash/2.3/centos
gpgcheck=1
gpgkey=https://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

Kibana

Kibana repository (https://www.elastic.co/guide/en/kibana/4.5/setup.html#setup-repositories):

```
[kibana-4.5]
name=Kibana repository for 4.5.x packages
baseurl=http://packages.elastic.co/kibana/4.5/centos
gpgcheck=1
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

Alternatively, you can download Kibana from the Elastic web site: **https://www.elastic.co/downloads/kibana** (https://www.elastic.co/downloads/kibana) .

Neo4j

Neo4j **repository** (http://debian.neo4j.org/):

```
[neo4j]
name=Neo4j Yum Repo
baseurl=http://yum.neo4j.org
enabled=1
gpgcheck=1
gpgkey=http://debian.neo4j.org/neotechnology.gpg.key
```

Migrate from Neo4j 2.2.4 to 2.3.1

> **Warning:** To upgrade from Neo4j 2.2.4 to Neo4j 2.3.1, you need to perform an **explicit database store upgrade** (http://neo4j.com/docs/stable/deployment-upgrading.html#explicit-upgrade).

To migrate the graph database from from Neo4j 2.2.4 to 2.3.1, do the following:

- **Install** (http://neo4j.com/docs/stable/deployment.html) Neo4j 2.3.1.
- **Configure** (http://neo4j.com/docs/stable/server-configuration.html) it to use the same database store directory, typically: `data/graph.db`.
- If the Neo4j instance you are going to upgrade is running, shut it down normally.
- In the `conf/neo4j.properties` file, set `allow_store_upgrade` to `true`. If this parameter is set to `false` or if it is not defined, Neo4j will not start.
- **Start** (http://neo4j.com/docs/stable/shell-starting.html) the Neo4j shell.
- The database store upgrade is executed at startup.
- After successfully completing the database store upgrade, remove the `allow_store_upgrade` configuration parameter from the `conf/neo4j.properties` file, set to `false`, or comment it out.
- In the database store directory — typically: `data/graph.db` — you can find a `messages.log` file with upgrade details, and a progress indicator.

Neo4j 2.3.1 configuration

After successfully upgrading to Neo4j 2.3.1, some configuration properties are deprecated, and therefore they are not needed anymore. At the same time, this release introduces new properties, or requires you to set different values for existing properties.

`cache_type`, the property defining the type of cache to use for nodes and relationships, is not needed anymore.

For a detailed list of all affected properties, see the *Configuration* section in the **Neo4j 2.3.0 release notes** (http://neo4j.com/release-notes/neo4j-2-3-0/).

Deprecated settings

```
use_memory_mapped_buffers,

neostore.nodestore.db.mapped_memory,

neostore.relationshipstore.db.mapped_memory,

ostore.propertystore.db.index.keys.mapped_memory,

neostore.propertystore.db.index.mapped_memory,

neostore.propertystore.db.mapped_memory,

neostore.propertystore.db.strings.mapped_memory,

neostore.propertystore.db.arrays.mapped_memory,
```

New settings

```
dbms.pagecache.memory
```

For further details about the new settings, see the **Neo4j configuration settings reference section** (http://neo4j.com/docs/stable/configuration-settings.html).

How to shut down the platform

Summary: Graciously shut down the platform by first stopping its core services and processes.

You may need to stop the platform and its services, for example if you run out of disk space and need to add more storage.

Do the following:

- Stop all running processes.
- Shut down the machine the platform is hosted on.

You can stop the platform without following any specific procedure. However, if you want to make sure the platform core services and processes shut down nicely, manually execute the following commands:

```
$ sudo supervisorctl stop all
```

```
$ sudo systemctl stop postgresql-9.5 redis logstash elasticsearch
```

You can then proceed to shut down the virtual or physical machine hosting the platform.