EclecticIQ Platform user guide

Publish and share intel with outgoing feeds — 4/4

Last generated: October 20, 2017

Last generated on Oct 20, 2017

Table of contents

User guide to EclecticIQ Platform

This user guide helps you configure the main options of the platform, as well as familiarize with EclecticIQ Platform, so that you can start collecting and analyzing potential threats efficiently.

Scope

The user guide to EclecticIQ Platform aims at providing clear and to-the-point help to get you acquainted with the threat intelligence platform, so that you can configure it as needed, and you can use it to collect and analyze intelligence on potential threats, as well as share it and collaborate with other analysts.

Although it is not a complete reference manual, this guide shows end-users how they can use the platform and its rich feature set to collect data, to analyze and investigate potential threats, and to collaborate and share intelligence with other analysts.

Goal

Learn how to incorporate the platform in your daily workflow as a poweful tool to:

- Automate data ingestion
- View, edit, create, and delete platform entities
- Enrich entities with additional contextual details
- Analyze entities on the graph to identify potential threats and their relationships
- Search, filter, and slice data using rules
- Share your findings and collaborate

Audience

This document targets the following audience:

- Cyber threat intelligence analysts
- Cyber threat intelligence specialists

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

Last generated on Oct 20, 2017

Configure outgoing feeds

Configure outgoing feeds to publish cyber threat intelligence through the platform to instrument external tools and devices, and to share intelligence with selected recipients within the organization, as well as with external third-parties.

EclecticIQ Platform uses outgoing feeds to publish and share cyber threat intelligence in multiple formats through a number of configurable transport channels.

- You can share intelligence with co-workers and teams within the organization, as well as with external recipients.
- You can also use outgoing feeds to route data to external devices to initiate specific follow-up actions, based on the data type being transmitted, and the receiving device.

Outgoing feeds are a powerful tool to disseminate intelligence and promote constructive collaboration, and to programmatically act on intelligence by automating tasks in your security toolchain.

Once it is set up and it is running, an outgoing feed provides a data stream that the intended recipients can consume. For example, an external device can receive platform data through an outgoing feed, and it can react to it by initiating predefined actions such as closing open ports or blacklisting malicious IP addresses and domain names.

A minimal outgoing feed configuration includes:

- A *data source*: the data source of an outgoing feed is always a dataset.
  You can configure as many datasets as necessary to act as sources for an outgoing feed.
- A *transport type*: the vehicle carrying the data.
  Typically, this is a communications protocol like TAXII, HTTP, FTP, IMAP, or Syslog.
- A *content type*: the outgoing data format the platform is publishing through the outgoing feed.
  For example, STIX, JSON, CSV, or plain text.
- An *update strategy*: the condition(s) defining how content is selected for inclusion in the outgoing feed.
  For example, you can choose to include in an outgoing feed only new entities, or both new and existing entities.

This article describes how to configure the **general options** for outgoing feeds to publish EclecticIQ Platform data. These options are identical for all outgoing feeds.

To configure transport type and content type, as well as any other specific options for a particular outgoing feed, follow the links under Configure transport and content for specific outgoing feeds.

Configure the general options

✔ Input fields marked with an asterisk are required.

- On the top navigation bar, select **Data configuration > Outgoing feeds**.
- On the top-left corner of the page click the ✚ icon to open the outgoing feed editor.

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

On the **Create outgoing feed** form you can populate the input fields to define the intel provider/data source for the feed, and the feed behavior.

- Under **Feed name**, enter a name for the feed you are creating. It should be descriptive and easy to remember.

Transport and content

Under **Transport type** and **Content type**, select the appropriate options to configure transport and content for the specified outgoing feed.

Set a schedule

Under **Execution schedule** you can define how often you want to run the feed task:

- **None**: scheduled feed execution is disabled. You need to manually trigger the task to ingest or to publish data through an incoming or an outgoing feed, respectively.
- **Every [n] minutes**: the feed task runs automatically once every *[n]* minutes, where *[n]* defines the selected time interval in minutes.
  You define the execution interval in 5-minute increments from the corresponding drop-down menu.
- **Every hour, [n] minutes past the hour**: the feed task runs automatically once an hour every hour at the specified minute offset from the hour.
  You define how long in minutes after the beginning of an hour the task should run from the corresponding drop-down menu.
- **Every [n] hours**: the feed task runs automatically once every *[n]* hours, where *[n]* defines the time interval in hours between two consecutive feed task runs.
  You define how long the time interval between feed executions should be by selecting the number of hours from the corresponding drop-down menu.
- **Every day at [time]**: the feed task runs automatically once a day at the specified time.
  You define the time of the day when the task should run from the corresponding drop-down menus.
- **Every [n] days**: the feed task runs automatically once every *[n]* days, where *[n]* defines the time interval in days between two consecutive feed task runs.
  You define how long the time interval between feed executions should be by selecting the number of days from the corresponding drop-down menu.
- **Every week on [day of the week] at [time]**: the feed task runs automatically once a week on the designated day, at the specified time.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menus.
- **Every month on [day of the month] at [time]**: the feed task runs automatically once a month on the designated day of the month, at the specified time.
  You define the day of the week and time of the day when the task should run from the corresponding drop-down menus.
  Keep in mind that not all months of the year have 30 or 31 days.

Set TLP filters

- **Override TLP** overwrites the **TLP** `(https://www.us-cert.gov/tlp)` color code associated with the feed entities with the one you set here. The selected TLP value is assigned to all the entities in the 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.
- **Filter TLP** includes in the an incoming or an outgoingan outgoing feed any entities flagged with the selected TLP color code, as well as entities whose TLP color indicates that they are suitable for progressively broader audiences. For example, if you select green, the feed includes entities with TLP set to green and to white.

Set reliability and relevancy

- **Source reliability**: from the drop-down menu select an option to flag the feed or enricher content with a predefined reliability value to help other users assess how trustworthy the data source is. 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 feed only entities whose relevancy is higher than the value defined here.

Set observable filters

- **Allowed observable states**: from the drop-down menu select one or more observable states to include in the feed data only entities whose observable states match at least one of the selections defined here.
- **Observable types**: from the drop-down menu select one or more observable types to include in the outgoing feed only entities whose observable types match at least one of the selections defined here.
- **Enrichment observable types**: from the drop-down menu select one or more enrichment observable types to include in the outgoing feed only entities whose enrichment observable types match at least one of the selections defined here.
- Click **Save** to store your changes, or **Cancel** to discard them.

The filters work independently of each other: there are no Boolean `AND` or `OR` to join multiple filters into a serial pipeline.

Anonymize data

In this section you can define specific fields and data to be either excluded from the outgoing feed, or replaced with other data. Data anonymization enables you to remove sensitive data from the published content, or to replace it with other non-sensitive information that can be safely disclosed.

Anonymization works only at entity level. It is not possible to anonymize data inside observables. You can anonymize entity data before publishing it through an outgoing feed in one of the following ways:

- You can flag data to be *skipped*: the data is excluded from the outgoing feed.
- You can flag data to be *replaced*: the data is replaced with the specified replacement data before being published through the outgoing feed.

Skip paths

In this field you can define specific entity fields, that is, specific locations in the entity JSON data structure whose data you want to exclude from the outgoing feed.
Any values related to the path options you set in this field are ignored.

- In the **Skip paths** input field, select one or more entity fields whose data you want to exclude from publication through the outgoing feed.
  The platform searches for the specified entity fields and the corresponding values, and it strips the data before publication. This action applies to all entities published through the outgoing feed.

  From the drop-down menu select entity fields to ignore:

Entity type or area	**Skip path option**	**Corresponding JSON path**	**Description**
Common	**Information source, Identity**	`data.information_source.identity`	
Common	**Information source, References**	`data.information_source.references`	
Common	**Title**	`data.title`	
Incident	**Affected assets, Properties affected**	`data.affected_assets`	
Indicator	**Observables**	``	
Indicator	**Sightings**	``	
Sighting	**Raw events**	``	
Sighting	**Security controls, Identity**	``	
Sighting	**Security controls, References**	``	
TTP	**Resources, Infrastructure**	``	
TTP	**Resources, Persona**	``	

Replace paths

Here you can define specific entity fields and specific data patterns that the rule replaces with user-defined values before publishing the data through the outgoing feed.

To replace values in one or more entity fields with other values that are suitable for publication, do the following:

- Click **+ Add** or **+ More** to add a filtering option.

- Under **Path**, enter the JSON path pointing to the entity field whose value you want to replace.
- Under **Pattern** enter a regex defining a data pattern to identify the value you want to replace.
  Pattern supports only **Elasticsearch regular expression syntax** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-regexp-query.html#regexp-syntax).`
  The main peculiarities of the Elasticsearch query regex syntax are:
  - Anchors (`^` and `$`) are implied at the beginning and at the end of the regex. You do not need to include them in the regex you input.
  - If you insert explicit anchor characters in the **Value** field, they are interpreted as literal values.
  - You need to escape special characters (`. ? + * | { } [ ] ( ) " \`).
    To escape a special character, prepend a backslash `\` to it. Example: `\{ \}`

✔ At this moment, Elasticsearch regular expression syntax **optional operators** `(https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-regexp-query.html#_optional_operators)` are not supported.

- Under **Value** enter the actual value that should replace the string matching the regex data pattern in the outgoing feed.
- Click ✚ **Add** or ✚ **More** to add new rows/new input fields as needed.

Example:

```
// Path: where to look for the values to replace
data.victim_targeting.identity.name

// Pattern: values matching the regex are replaced
\[Aa\]l\?ateleco\*

// Value: value replacing the strings matching the regex
The Swedish Chef
```

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 a new item of the same type right away. For example, a dataset, a feed, a rule, a workspace, 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 transport and content for specific outgoing feeds

- Configure email transport and content
- Configure FTP upload transport and content

- Configure HTTP download transport and content
- Configure Mount point upload transport and content
- Configure Syslog push transport and content
- Configure TAXII inbox transport and content
- Configure TAXII poll transport and content

Start and stop feeds

Enable and disable feeds, as well as manually trigger a feed task run or stop a running feed.

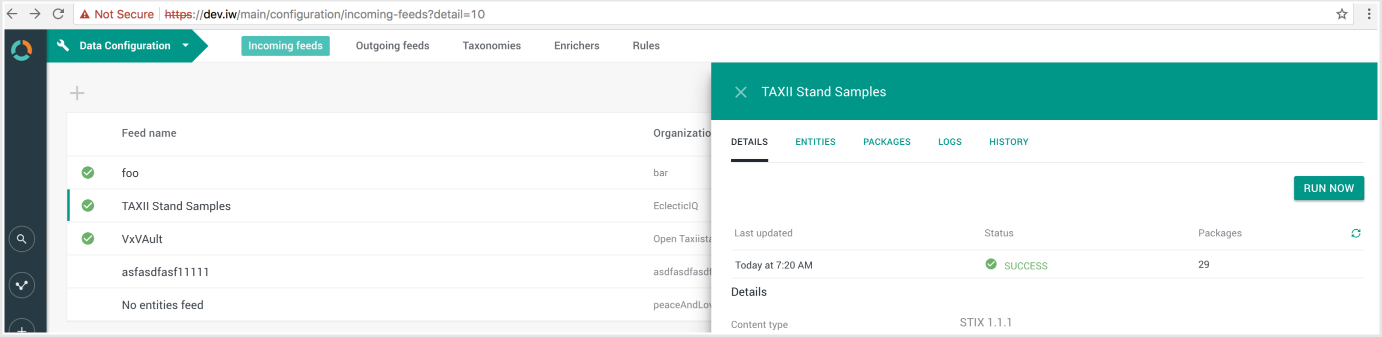
After configuring a feed, you can set a schedule to automate feed execution over time. If you do not set an execution schedule, the feed does not run, that is, it does not fetch or publish any data.

Manually start a feed

You can manually start an incoming or an outgoing feed run in one of the following ways:

On the outgoing feed overview page

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click the ⋮ icon corresponding to the feed you want to run.
- From the drop-down menu select **Run now**.



	Feed name	Organizatio
✓	foo	bar
✓	TAXII Stand Samples	EclecticIQ
✓	VxVAult	Open Taxiist
	asfasdfasf11111	asdfasdfasdf
	No entities feed	peaceAndLo

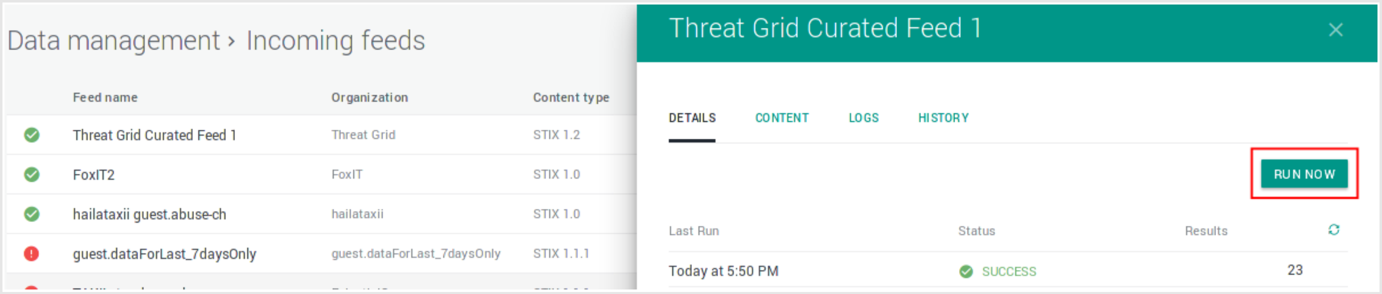
Last updated	Status	Packages
Today at 7:20 AM	✓ SUCCESS	29

Details

Content type: STIX 1.1.1

On the outgoing feed entity detail pane

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.
- On the **Details** tab on feed detail pane click **Run now**.



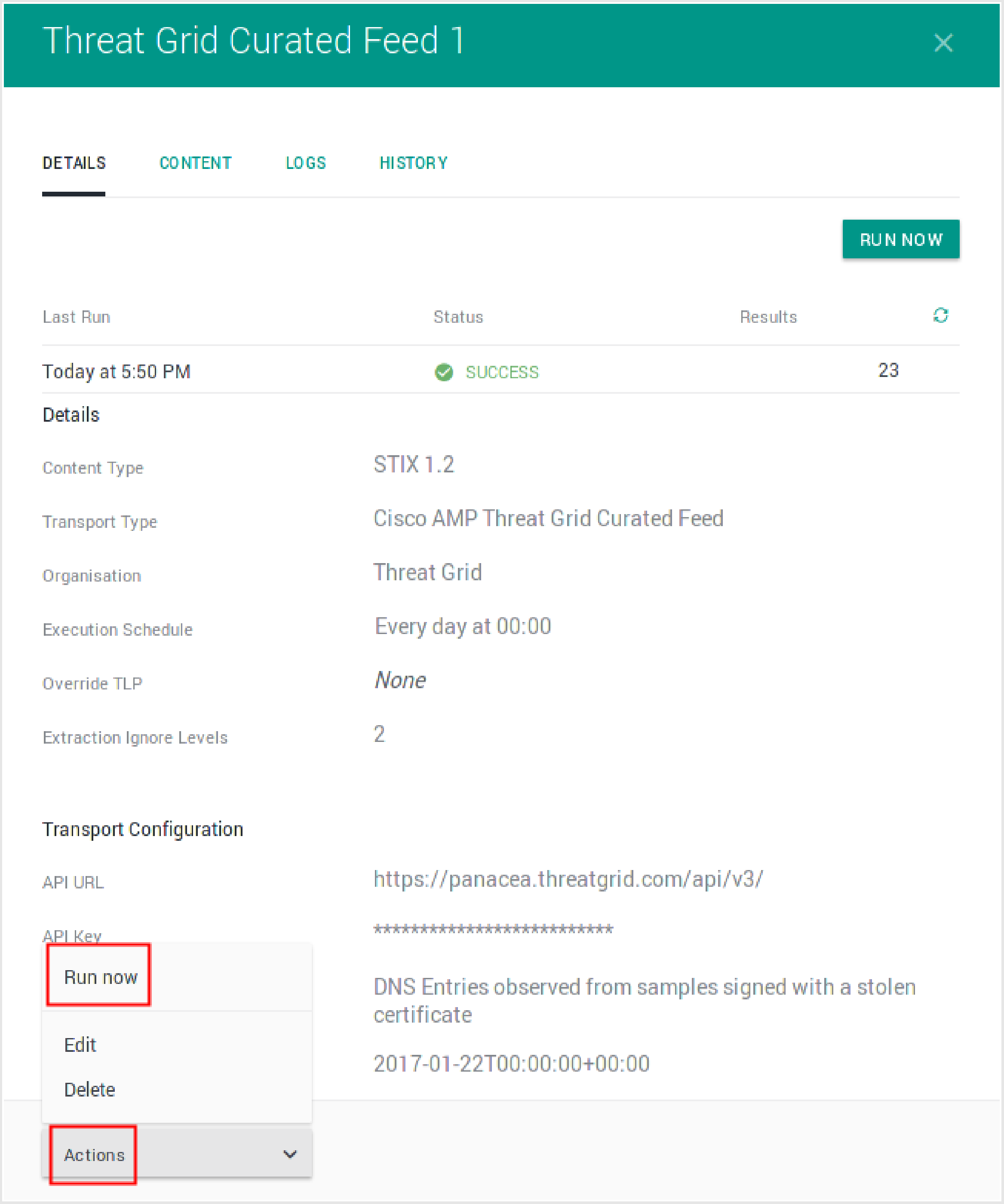
	Feed name	Organization	Content type
✓	Threat Grid Curated Feed 1	Threat Grid	STIX 1.2
✓	FoxIT2	FoxIT	STIX 1.0
✓	hailataxii guest.abuse-ch	hailataxii	STIX 1.0
!	guest.dataForLast_7daysOnly	guest.dataForLast_7daysOnly	STIX 1.1.1

Last Run	Status	Results
Today at 5:50 PM	✓ SUCCESS	23

Through the Actions menu

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.

- On the **Details** tab on feed detail pane, scroll to the bottom of the pane, and then click **Actions**.
- From the pop-up menu select **Run now**.



Last Run	Status	Results
Today at 5:50 PM	SUCCESS	23

Details

Content Type	STIX 1.2
Transport Type	Cisco AMP Threat Grid Curated Feed
Organisation	Threat Grid
Execution Schedule	Every day at 00:00
Override TLP	*None*
Extraction Ignore Levels	2

Transport Configuration

API URL	https://panacea.threatgrid.com/api/v3/
API Key	**************************
	DNS Entries observed from samples signed with a stolen certificate
	2017-01-22T00:00:00+00:00

Manually stop a feed

You can either manually suspend/disable or stop/terminate a running incoming or outgoing feed.

Suspend and disable a running feed

To disable an active feed, do the following:

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.
- On the **Details** tab on feed detail pane click **Disable**.
  You can enable the disabled feed at any time by clicking **Enable**.

You can disable feed execution also by setting the feed execution schedule to **None**:

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to suspend an incoming or an outgoing feed.
- On the feed overview page, click the ⋮ icon corresponding to the feed you want to run.
- From the drop-down menu select **Edit**.
- On the feed configuration page, go to the **Schedule** section, and then set **Execution schedule** to **None**.
- Click **Save** to store your changes, or **Cancel** to discard them.

Alternatively:

- Go to **Data configuration > Incoming feeds** or to **Data configuration > Outgoing feeds**, depending on whether you want to run an incoming or an outgoing feed.
- On the feed overview page, click anywhere on the row corresponding to the feed you want to run.
- On the feed detail pane, scroll to the bottom of the pane, and then click **Actions**.
- From the drop-down menu select **Edit**.
- On the feed configuration page, go to the **Schedule** section, and then set **Execution schedule** to **None**.
- Click **Save** to store your changes, or **Cancel** to discard them.

Stop and terminate a running feed

To stop the execution of a running feed and kill the task, do the following:

- On the left-hand navigation sidebar, click **⚙ > System jobs > Running**.
- On the **System jobs > Running** overview page, browse to the running task(s) you want to terminate, and then click the corresponding ■ **Terminate** button to instantly stop executing the selected task(s).

Configure email transport and content

Set up and configure transport and content types for Send email outgoing feeds to publish selected platform data as email attachments.

To configure the general options for the Send email outgoing feed, see Configure outgoing feeds.

About Send email

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
Send email	The feed publishes entities and observables in the selected content type as email attachments to the intended recipients. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
Send email	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV
	EclecticIQ JSON
	Plain text value
	STIX 1.2

Configure the transport type

- **Transport type**: from the drop-down menu select **Send email**.

Under **Transport configuration** set the email transport type options:

- **Mail subject**: enter a short, descriptive subject for the email notifications delivered through the outgoing feed.
- **Platform groups**: restricts access to the outgoing feed to the groups you select from the drop-down menu, and to their member users. 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 recipients to only some members of the selected group(s), from the drop-down menu select one or more users. In this case, only the selected users belonging to the designated platform groups receive the outgoing feed email notifications.
- **Include documents attached to entities**: select this checkbox to to include in the outgoing feed also any attachments to the entities such as MS Word documents or PDF files.

Configure the content type

- **Content type**: from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **Send email** transport type enables fetching data in the following formats:
  - ArcSight CEF
  - EclecticIQ Entities CSV
  - EclecticIQ Observables CSV
  - EclecticIQ JSON
  - Plain text value
  - STIX 1.2
- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.

- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
    - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
    - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
    - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

      Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
      An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

Warning: If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

EclecticIQ JSON

By default, the **EclecticIQ JSON** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.
- **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.
  This setting changes the `data.information_source.identity.name` value in the entity JSON structure:

```
{
  "data": {
    "information_source": {
      "type": "information-source",
      "identity": {
        "name": "<producer_identity>", // ex.: 'EclecticIQ'
        "type": "identity"
      }
    }
  }
}
```

Plain text value

The plain text value content type is suitable for machine consumption. Typical use cases include feeding a plain text value outgoing feed to an external compatible device to instrument further processing or to trigger a response action.

The plain text value content configuration options set up a rule to define the data pool that qualifies for inclusion in the outgoing feed. The rule works as follows:

- The **Field to check a conditional value in** condition looks for a specific JSON path pointing to a specific entity field in the entity JSON structure.
  - If the previous condition yields matches, the **Only use entities that match this conditional value** condition looks at the specified JSON path key for any values matching the literal or the regex data pattern you define in this field.
  - If the previous conditions yield matches, the **Field to take values from** condition points to a specific entity field whose value is fetched and included for publication in the outgoing feed.

Under **Content configuration** set the **Plain text value** content type options:

- **Field to take values from**: specifies the location in the entity JSON structure where the values to include in the feed are stored.
  Enter a JSON path pointing to the entity field whose values you want to fetch and include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Field to check a conditional value in**: this condition works together with **Only use entities that match this conditional value** to filter specific entities for the feed.
  Enter a JSON path pointing to the entity field you want to use as a filter to select entities whose content you want to include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Only use entities that match this conditional value**: this condition works together with **Field to check a conditional value in** to filter specific entities for the feed.
  Enter a string to define the value to match.

Example

You can configure this rule to send relevant data to an external Snort or Suricata instance, where they can be further processed or used to initiate a specific response action:

- **Field to check a conditional value in**: *data.type.test_mechanisms.test_mechanism_type*
- **Only use entities that match this conditional value**: *snort*
- **Field to take values from**: *data.type.test_mechanisms.rules.value*

The rule uses the specified conditions to:

- Look for platform entities containing Snort rules: *data.type.test_mechanisms.test_mechanism_type: snort*
- If the previous condition yields matching entities, look in those entities if they contain this field: *data.type.test_mechanisms.rules.value*

- If they do, fetch the value from the field and include it in the outgoing feed.
  Matching values are added to the outgoing feed one value per line.
  The value in question should be a valid Snort rule for the resulting feed data to be meaningful.
  Example:

```
alert tcp $HOME_NET any -> [72.20.35.70,72.20.35.120] 6661 (msg:\"ET CNC Shadowserver Reported CnC
Server Port 6661 Group 1\"; flags:S; reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org; threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil; flowbits:set,ET.BotccIP; sid:2405018; rev:3633;)
```

STIX 1.2

The STIX 1.2 content type is suitable for machine consumption. Typical use cases include feeding a STIX 1.2 outgoing feed to an external STIX-compatible device to instrument further processing or to trigger a response action.

Under **Content configuration** set the **STIX 1.2** content type options:

- **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.
  This setting changes the following nested XML element in the entity STIX structure:

```
<stixCommon:Identity>
  <!-- Producer identity, for example 'EclecticIQ' -->
  <stixCommon:Name>EclecticIQ</stixCommon:Name>
</stixCommon:Identity>
```

Configure FTP upload transport and content

Set up and configure transport and content types for FTP upload outgoing feeds to publish selected platform data to an FTP server.

To configure the general options for the FTP upload outgoing feed, see Configure outgoing feeds.

About FTP upload

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
FTP upload	The feed publishes entities and observables in the selected content type to the specified destination location on an FTP server. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
FTP upload	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV
	EclecticIQ JSON
	Plain text value
	STIX 1.2

Configure the transport type

The FTP upload transport type for outgoing feeds publishes the supported content types to the specified location on the target FTP server.

- **Transport type**: from the drop-down menu select **FTP upload**.

Under **Transport configuration** set the FTP transport type options:

- **FTP server URL**: the target `ftp://` location on the FTP server to upload the outgoing feed content to, so as to make it available for retrieval.
  Example: *ftp://ftp.server.com/feeds/outgoing/folder*
- **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.
- **Include documents attached to entities**: select this checkbox to to include in the outgoing feed also any attachments to the entities such as MS Word documents or PDF files.

Configure the content type

- **Content type**: from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **FTP upload** transport type enables fetching data in the following formats:
    - ArcSight CEF
    - EclecticIQ Entities CSV
    - EclecticIQ Observables CSV
    - EclecticIQ JSON
    - Plain text value
    - STIX 1.2
- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.

- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
    - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
    - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
    - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

      Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
      An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

⚠ **Warning:** If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

EclecticIQ JSON

By default, the **EclecticIQ JSON** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.
- **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.
  This setting changes the `data.information_source.identity.name` value in the entity JSON structure:

```
{
  "data": {
    "information_source": {
      "type": "information-source",
      "identity": {
        "name": "<producer_identity>", // ex.: 'EclecticIQ'
        "type": "identity"
      }
    }
  }
}
```

Plain text value

The plain text value content type is suitable for machine consumption. Typical use cases include feeding a plain text value outgoing feed to an external compatible device to instrument further processing or to trigger a response action.

The plain text value content configuration options set up a rule to define the data pool that qualifies for inclusion in the outgoing feed. The rule works as follows:

- The **Field to check a conditional value in** condition looks for a specific JSON path pointing to a specific entity field in the entity JSON structure.
  - If the previous condition yields matches, the **Only use entities that match this conditional value** condition looks at the specified JSON path key for any values matching the literal or the regex data pattern you define in this field.
  - If the previous conditions yield matches, the **Field to take values from** condition points to a specific entity field whose value is fetched and included for publication in the outgoing feed.

Under **Content configuration** set the **Plain text value** content type options:

- **Field to take values from**: specifies the location in the entity JSON structure where the values to include in the feed are stored.
  Enter a JSON path pointing to the entity field whose values you want to fetch and include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Field to check a conditional value in**: this condition works together with **Only use entities that match this conditional value** to filter specific entities for the feed.
  Enter a JSON path pointing to the entity field you want to use as a filter to select entities whose content you want to include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Only use entities that match this conditional value**: this condition works together with **Field to check a conditional value in** to filter specific entities for the feed.
  Enter a string to define the value to match.

Example

You can configure this rule to send relevant data to an external Snort or Suricata instance, where they can be further processed or used to initiate a specific response action:

- **Field to check a conditional value in**: *data.type.test_mechanisms.test_mechanism_type*
- **Only use entities that match this conditional value**: *snort*
- **Field to take values from**: *data.type.test_mechanisms.rules.value*

The rule uses the specified conditions to:

- Look for platform entities containing Snort rules: *data.type.test_mechanisms.test_mechanism_type: snort*
- If the previous condition yields matching entities, look in those entities if they contain this field: *data.type.test_mechanisms.rules.value*

- If they do, fetch the value from the field and include it in the outgoing feed.
  Matching values are added to the outgoing feed one value per line.
  The value in question should be a valid Snort rule for the resulting feed data to be meaningful.
  Example:

```
alert tcp $HOME_NET any -> [72.20.35.70,72.20.35.120] 6661 (msg:\"ET CNC Shadowserver Reported CnC
Server Port 6661 Group 1\"; flags:S; reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org; threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil; flowbits:set,ET.BotccIP; sid:2405018; rev:3633;)
```

STIX 1.2

The STIX 1.2 content type is suitable for machine consumption. Typical use cases include feeding a STIX 1.2 outgoing feed to an external STIX-compatible device to instrument further processing or to trigger a response action.

Under **Content configuration** set the **STIX 1.2** content type options:

- **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.
  This setting changes the following nested XML element in the entity STIX structure:

```
<stixCommon:Identity>
  <!-- Producer identity, for example 'EclecticIQ' -->
  <stixCommon:Name>EclecticIQ</stixCommon:Name>
</stixCommon:Identity>
```

Configure HTTP download transport and content

Set up and configure transport and content types for HTTP download outgoing feeds to publish selected platform data to an HTTP server.

To configure the general options for the HTTP download outgoing feed, see Configure outgoing feeds.

About HTTP download

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
HTTP download	The feed publishes entities and observables in the selected content type through the platform API. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

HTTP endpoints

The default platform API endpoints for HTTP download outgoing feeds are:

- `https://<platform_host>/api/open-outgoing-feed-download/` for publicly accessible outgoing feeds. These feeds publish content that all platform users can access.
- `https://<platform_host>/api/outgoing-feed-download/`for non-publicly accessible outgoing feeds. These feeds publish content that only the intended recipients can access.

You can append additional elements to the URL to retrieve specific content from an HTTP download outgoing feed:

- `https://<platform_host>/api/open-outgoing-feed-download/{feed_id}/runs/latest`: replace `{feed_id}` with the outgoing feed ID reference to retrieve all packages from the latest outgoing feed task run.
  - The feed ID is the integer value in the `&detail={integer}` URL element in the URL pointing to the **Details** tab of the outgoing feed detail pane.

- `https://<platform_host>/api/open-outgoing-feed-download/{feed_id}/runs/{run_id}`: replace `{feed_id}` with the outgoing feed ID reference and `{run_id}` with the desired outgoing feed task run identifier value to retrieve all packages form a specific outgoing feed task run.
  - To retrieve the task run ID, do the following:
    - On the top navigation bar click **⚙ > System jobs > Succeeded** .
    - On the successfully completed system job overview, look for the desired task run ID under the **ID** column.
- `https://<platform_host>/api/open-outgoing-feed-download/{feed_id}/runs/{run_id}/content-blocks/latest`: replace `{feed_id}` with the outgoing feed ID reference and `{run_id}` with the desired outgoing feed task run identifier value to retrieve the latest/most recent package from a specific outgoing feed task run.
- `https://<platform_host>/api/open-outgoing-feed-download/{feed_id}/runs/{run_id}/content-blocks/{block_id}`: replace `{feed_id}` with the outgoing feed ID reference, `{run_id}` with the desired outgoing feed task run identifier value, and `{block_id}` with the desired content block ID reference to retrieve a specific package from a specific outgoing feed task run.
  - To retrieve the content block ID, do the following:
    - In the web browser address bar, enter the URL pointing to the list of all content blocks in the specified outgoing feed: `https://<platform_host>/api/open-outgoing-feed-download/{feed_id}`
    - The `data.content_block` JSON array lists the URLs pointing to all the content blocks belonging to the outgoing feed.
    - The content block ID is the integer value at the end of the URL.

Example:

```
{
  "data": {
    "content_blocks": [
      "/api/open-outgoing-feed-download/12/runs/ff7458fg-c63b-4f94-a811-ffa87a254d98/content-
blocks/98",
      "/api/open-outgoing-feed-download/12/runs/678bf255-0835-4994-a0ed-d98ac98aaa58/content-
blocks/44",
      "/api/open-outgoing-feed-download/12/runs/c4a394e9-0a8f-42ca-ad4b-72cc3762afd7/content-
blocks/32",
      "/api/open-outgoing-feed-download/12/runs/bf711b50-c2a1-4f5t-994f-ec1c481ace3d/content-
blocks/11"
    ],
    "id": 4,
    "name": "Download CSV Line per entity"
  }
}
```

The same URL format applies to the `https://<platform_host>/api/outgoing-feed-download/`for non-publicly accessible HTTP download outgoing feeds.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
HTTP download	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV
	EclecticIQ JSON
	Plain text value
	STIX 1.2

Configure the transport type

The HTTP download transport type for outgoing feeds publishes the supported content types to the specified location on the target HTTP download.

- **Transport type**: from the drop-down menu select **HTTP download**.

Under **Transport configuration** set the HTTP transport type options:

- **Public**: default setting: deselected.
  Select this checkbox to make the outgoing feed available to all platform groups and to all platform users.
  Leave it deselected to make the outgoing feed available only to specific groups. You can select the intended recipient groups in the **Authorized groups** drop-down menu.
- **Authorized groups**: restricts access to the outgoing feed to the groups you select from the drop-down menu, and to their member users.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).

Warning:
Before deleting a group, check that is not an authorized group in an outgoing feed configuration.
Deleting a group that is currently selected as an authorized group to access the outgoing feed content breaks the outgoing feed functionality.

If you need to remove such a group:

- First, remove it from the **Authorized group** selection in the relevant outgoing feed(s).
- Then, proceed to delete the group.

Configure the content type

- **Content type**: from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **HTTP download** transport type enables fetching data in the following formats:
  - ArcSight CEF
  - EclecticIQ Entities CSV
  - EclecticIQ Observables CSV
  - EclecticIQ JSON
  - Plain text value
  - STIX 1.2
- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.
- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
  - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

    Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
    An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

> ⚠ **Warning:** If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

EclecticIQ JSON

By default, the **EclecticIQ JSON** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.
- **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.
  This setting changes the `data.information_source.identity.name` value in the entity JSON structure:

```
{
  "data": {
    "information_source": {
      "type": "information-source",
      "identity": {
        "name": "<producer_identity>", // ex.: 'EclecticIQ'
        "type": "identity"
      }
    }
  }
}
```

Plain text value

The plain text value content type is suitable for machine consumption. Typical use cases include feeding a plain text value outgoing feed to an external compatible device to instrument further processing or to trigger a response action.

The plain text value content configuration options set up a rule to define the data pool that qualifies for inclusion in the outgoing feed. The rule works as follows:

- The **Field to check a conditional value in** condition looks for a specific JSON path pointing to a specific entity field in the entity JSON structure.
  - If the previous condition yields matches, the **Only use entities that match this conditional value** condition looks at the specified JSON path key for any values matching the literal or the regex data pattern you define in this field.
  - If the previous conditions yield matches, the **Field to take values from** condition points to a specific entity field whose value is fetched and included for publication in the outgoing feed.

Under **Content configuration** set the **Plain text value** content type options:

- **Field to take values from**: specifies the location in the entity JSON structure where the values to include in the feed are stored.
  Enter a JSON path pointing to the entity field whose values you want to fetch and include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Field to check a conditional value in**: this condition works together with **Only use entities that match this conditional value** to filter specific entities for the feed.
  Enter a JSON path pointing to the entity field you want to use as a filter to select entities whose content you want to include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.

- **Only use entities that match this conditional value**: this condition works together with **Field to check a conditional value in** to filter specific entities for the feed.
  Enter a string to define the value to match.

Example

You can configure this rule to send relevant data to an external Snort or Suricata instance, where they can be further processed or used to initiate a specific response action:

- **Field to check a conditional value in**: *data.type.test_mechanisms.test_mechanism_type*
- **Only use entities that match this conditional value**: *snort*
- **Field to take values from**: *data.type.test_mechanisms.rules.value*

The rule uses the specified conditions to:

- Look for platform entities containing Snort rules: *data.type.test_mechanisms.test_mechanism_type: snort*
- If the previous condition yields matching entities, look in those entities if they contain this field: *data.type.test_mechanisms.rules.value*
- If they do, fetch the value from the field and include it in the outgoing feed.
  Matching values are added to the outgoing feed one value per line.
  The value in question should be a valid Snort rule for the resulting feed data to be meaningful.
  Example:

```
alert tcp $HOME_NET any -> [72.20.35.70,72.20.35.120] 6661 (msg:\"ET CNC Shadowserver Reported CnC
Server Port 6661 Group 1\"; flags:S; reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org; threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil; flowbits:set,ET.BotccIP; sid:2405018; rev:3633;)
```

STIX 1.2

The STIX 1.2 content type is suitable for machine consumption. Typical use cases include feeding a STIX 1.2 outgoing feed to an external STIX-compatible device to instrument further processing or to trigger a response action.

Under **Content configuration** set the **STIX 1.2** content type options:

- **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.
  This setting changes the following nested XML element in the entity STIX structure:

```
<stixCommon:Identity>
  <!-- Producer identity, for example 'EclecticIQ' -->
  <stixCommon:Name>EclecticIQ</stixCommon:Name>
</stixCommon:Identity>
```

Configure Mount point upload transport and content

Set up and configure transport and content types for Mount point upload outgoing feeds to publish selected platform data to a specific location on a local or network unit.

To configure the general options for the Mount point upload outgoing feed, see Configure outgoing feeds.

About Mount point upload

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
Mount point upload	The feed publishes entities and observables in the selected content type to the specified destination location on a local or network unit. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
Mount point upload	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV
	EclecticIQ JSON
	Plain text value
	STIX 1.2

Configure the transport type

The Mount point upload transport type for outgoing feeds publishes the supported content types to the specified location on a local or network unit.

- **Transport type**: from the drop-down menu select **Mount point upload**.

Under **Transport configuration** set the mount point transport type options:

- **Mount point path**: the path to the local or network unit to save the outgoing feed content to, so as to make it available for retrieval.
  Example: */media/feeds/outgoing/folder*
- **Include documents attached to entities**: select this checkbox to to include in the outgoing feed also any attachments to the entities such as MS Word documents or PDF files.

Configure the content type

- **Content type**: from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **Mount point upload** transport type enables fetching data in the following formats:
  - ArcSight CEF
  - EclecticIQ Entities CSV
  - EclecticIQ Observables CSV
  - EclecticIQ JSON
  - Plain text value
  - STIX 1.2
- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.

- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
  - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

    Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
    An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

Warning: If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

EclecticIQ JSON

By default, the **EclecticIQ JSON** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.
- **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.
  This setting changes the `data.information_source.identity.name` value in the entity JSON structure:

```
{
  "data": {
    "information_source": {
      "type": "information-source",
      "identity": {
        "name": "<producer_identity>", // ex.: 'EclecticIQ'
        "type": "identity"
      }
    }
  }
}
```

Plain text value

The plain text value content type is suitable for machine consumption. Typical use cases include feeding a plain text value outgoing feed to an external compatible device to instrument further processing or to trigger a response action.

The plain text value content configuration options set up a rule to define the data pool that qualifies for inclusion in the outgoing feed. The rule works as follows:

- The **Field to check a conditional value in** condition looks for a specific JSON path pointing to a specific entity field in the entity JSON structure.
    - If the previous condition yields matches, the **Only use entities that match this conditional value** condition looks at the specified JSON path key for any values matching the literal or the regex data pattern you define in this field.
    - If the previous conditions yield matches, the **Field to take values from** condition points to a specific entity field whose value is fetched and included for publication in the outgoing feed.

Under **Content configuration** set the **Plain text value** content type options:

- **Field to take values from**: specifies the location in the entity JSON structure where the values to include in the feed are stored.
  Enter a JSON path pointing to the entity field whose values you want to fetch and include for publication in the outgoing feed.
    - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
    - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
      Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
    - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
    - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Field to check a conditional value in**: this condition works together with **Only use entities that match this conditional value** to filter specific entities for the feed.
  Enter a JSON path pointing to the entity field you want to use as a filter to select entities whose content you want to include for publication in the outgoing feed.
    - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
    - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
      Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
    - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
    - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Only use entities that match this conditional value**: this condition works together with **Field to check a conditional value in** to filter specific entities for the feed.
  Enter a string to define the value to match.

Example

You can configure this rule to send relevant data to an external Snort or Suricata instance, where they can be further processed or used to initiate a specific response action:

- **Field to check a conditional value in**: *data.type.test_mechanisms.test_mechanism_type*
- **Only use entities that match this conditional value**: *snort*
- **Field to take values from**: *data.type.test_mechanisms.rules.value*

The rule uses the specified conditions to:

- Look for platform entities containing Snort rules: *data.type.test_mechanisms.test_mechanism_type: snort*
- If the previous condition yields matching entities, look in those entities if they contain this field: *data.type.test_mechanisms.rules.value*

- If they do, fetch the value from the field and include it in the outgoing feed.
  Matching values are added to the outgoing feed one value per line.
  The value in question should be a valid Snort rule for the resulting feed data to be meaningful.
  Example:

```
alert tcp $HOME_NET any -> [72.20.35.70,72.20.35.120] 6661 (msg:\"ET CNC Shadowserver Reported CnC
Server Port 6661 Group 1\"; flags:S; reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org; threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil; flowbits:set,ET.BotccIP; sid:2405018; rev:3633;)
```

STIX 1.2

The STIX 1.2 content type is suitable for machine consumption. Typical use cases include feeding a STIX 1.2 outgoing feed to an external STIX-compatible device to instrument further processing or to trigger a response action.

Under **Content configuration** set the **STIX 1.2** content type options:

- **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.
  This setting changes the following nested XML element in the entity STIX structure:

```
<stixCommon:Identity>
  <!-- Producer identity, for example 'EclecticIQ' -->
  <stixCommon:Name>EclecticIQ</stixCommon:Name>
</stixCommon:Identity>
```

Configure Syslog push transport and content

Set up and configure transport and content types for Syslog push outgoing feeds to publish selected platform data to a Syslog server.

To configure the general options for the Syslog push outgoing feed, see Configure outgoing feeds.

About Syslog push

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
Syslog push	The feed publishes entities and observables in CSV or ArcSight CEF format to the specified Syslog server. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
Syslog push	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV

Configure the transport type

The Syslog push transport type for outgoing feeds publishes the supported content types to an external Syslog server for log aggregation.

- **Transport type**: from the drop-down menu select **Syslog push**.

Under **Transport configuration** set the Syslog push transport type options:

- **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**.

Configure the content type

- **Content type** from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **Syslog push** transport type enables fetching data in the following formats:
  - ArcSight CEF
  - EclecticIQ Entities CSV
  - EclecticIQ Observables CSV
- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.

- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
    - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
    - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
    - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

      Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
      An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
    - Original observable is a URI: *http://www.evil.com/big/info.php*
    - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
    - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

⚠ **Warning:** If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

Configure TAXII inbox transport and content

Set up and configure transport and content types for TAXII inbox outgoing feeds to publish selected platform data through the TAXII inbox service.

To configure the general options for the TAXII inbox outgoing feed, see Configure outgoing feeds.

About TAXII inbox

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
TAXII inbox	The feed publishes entities and observables in the selected content type to the TAXII inbox service configured for the feed. You can retrieve the content by accessing the TAXII inbox service endpoint configured for the feed in the platform, and by specifying the name of the collection the outgoing feed belongs to, as well as the feed name. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
TAXII inbox	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV
	EclecticIQ JSON
	Plain text value

Transport type	Allowed content types
	STIX 1.2

Configure the transport type

The TAXII inbox transport type for outgoing feeds publishes the supported content types through the TAXII inbox service.

- **Transport type**: from the drop-down menu select **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/)`.

Under **Transport configuration** set the TAXII inbox transport type options:

- **Auto discovery**: enter the URL pointing to a **TAXII discovery service** `(https://taxiiproject.github.io/about/#how-do-you-communicate-available-taxii-services-and-their-use)` that feed consumers can send a request to in order to determine the available TAXII services they can access — including the TAXII inbox outgoing feed — and poll them for updates.
  Example: *http://hailataxii.com/taxii-discovery-service*
- **Inbox service URL**: enter the URL pointing to the location of the **TAXII data collections** `(https://taxiiproject.github.io/documentation/sample-use/#data-collections)` available through the TAXII inbox service.
  Example:
  https://example.com/taxii-inbox
- **Destination collection name**: enter an existing collection name as the target container 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 pointing to the EclecticIQ Platform instance, including the endpoint that takes the user name and password inputs to send them to the authentication mechanism.
  Example: *http://<platform_host>/api/auth*
- **Username**: a valid user name to authenticate and be granted the necessary authorization to access the location of the outgoing feed content.
- **Password**: a valid password to authenticate and be granted the necessary authorization to access the location of the outgoing feed content.

- **SSL certificate authentication**: if the TAXII server requires an SSL certificate to authenticate and to access the corresponding TAXII services, select this checkbox to fill out the required information.
  - **SSL certificate**: copy-paste the content of a valid SSL certificate to authenticate.
    Example:

```
-----BEGIN CERTIFICATE REQUEST-----
MIICvDCCAaQCAQAwdzELMAkGA1UEBhMCVVMxDTALBgNVBAgMBFV0YWgxDzANBgNV
BAcMBkxpbmRvbjEWMBQGA1UECgwNRGlnaUNlcnQgSW5jLjERMA8GA1UECwwIRGln
aUNlcnQxHTAbBgNVBAMMFGV4YW1wbGUuZGlnaWNlcnQuY29tMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA8+To7d+2kPWeBv/orU3LVbJwDrSQbeKamCmo
wp5bqDxIwV20zqRb7APUOKYoVEFFOEQs6T6gImnIolhbiH6m4zgZ/CPvWBOkZc+c
1Po2EmvBz+AD5sBdT5kzGQA6NbWyZGldxRthNLOs1efOhdnWFuhI162qmcflgpiI
WDuwq4C9f+YkeJhNn9dF5+owm8cOQmDrV8NNdiTqin8q3qYAHHJRW28glJUCZkTZ
wIaSR6crBQ8TbYNE0dc+Caa3DOIkz1EOsHWzTx+n0zKfqcbgXi4DJx+C1bjptYPR
BPZL8DAeWuA8ebudVT44yEp82G96/Ggcf7F33xMxe0yc+Xa6owIDAQABoAAwDQYJ
KoZIhvcNAQEFBQADggEBAB0kcrFccSmFDmxox0Ne01UIqSsDqHgL+XmHTXJwre6D
hJSZwbvEtOK0G3+dr4Fs11WuUNt5qcLsx5a8uk4G6AKHMzuhLsJ7XZjgmQXGECpY
Q4mC3yT3ZoCGpIXbw+iP3lmEEXgaQL0Tx5LFl/okKbKYwIqNiyKWOMj7ZR/wxWg/
ZDGRs55xuoeLDJ/ZRFf9bI+IaCUd1YrfYcHIl3G87Av+r49YVwqRDT0VDV7uLgqn
29XI1PpVUNCPQGn9p/eX6Qo7vpDaPybRtA2R7XLKjQaF9oXWeCUqy1hvJac9QFO2
97Ob1alpHPoZ7mWiEuJwjBPii6a9M9G30nUo39lBi1w=
-----END CERTIFICATE REQUEST-----
```

  - **SSL key**: copy-paste the content of a valid SSL key to authenticate
    Example:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA3Tz2mr7SZiAMfQyuvBjM9Oi..Z1BjP5CE/Wm/Rr500P
RK+Lh9x5eJPo5CAZ3/ANBE0sTK0ZsDGMak2m1g7..3VHqIxFTz0Ta1d+NAj
wnLe4nOb7/eEJbDPkk05ShhBrJGBKKxb8n104o/..PdzbFMIyNjJzBM2o5y
5A13wiLitEO7nco2WfyYkQzaxCw0AwzlkVHiIyC..71pSzkv6sv+4IDMbT/
XpCo8L6wTarzrywnQsh+etLD6FtTjYbbrvZ8RQM..Hg2qxraAV++HNBYmNW
kbJ+q+rsJxQlaipn2M4lGuQJEfIxELFDyd3XpxP..Un/82NZNXlPmRIopXs
2T91jiLZEUKQw+n73j26adTbteuEaPGSrTZxBLR..yssO0wWomUyILqVeti
+PK+aXKwguI6bxLGZ3of0UH+mGsSl0mkp7kYZCm..OTQtfeRqP8rDSC7DgA
kHc5ajYqh04AzNFaxjRo+M3IGICUaOdKnXd0Fda..QwfoaX4QlRTgLqb7AN
ZTzM9WbmnYoXrx17kZlT3lsCgYEAm757XI3WJVj..WoLj1+v48WyoxZpcai
uv9bT4Cj+lXRS+gdKHK+SH7J3x2CRHVS+WH/SVC..DxuybvebDoT0TkKiCj
BWQaGzCaJqZa+POHK0klvS+9ln0/6k539p95tfX..X4TCzbVG6+gJiX0ysz
Yfehn5MCgYEAkMiKuWHCsVyCab3RUf6XA9gd3qY..fCTIGtS1tR5PgFIV+G
engiVoWc/hkj8SBHZz1n1xLN7KDf8ySU06MDggB..hJ+gXJKy+gf3mF5Kmj
DtkpjGHQzPF6vOe907y5NQLvVFGXUq/FIJZxB8k..fJdHEm2M4=
-----END RSA PRIVATE KEY-----
```

  - **SSL key password**: enter the SSL password or passphrase for the SSL key. This field is masked.
- **Verify SSL**: if the TAXII server requires an SSL certificate to authenticate and to access the corresponding TAXII services, you can select this checkbox to test the SSL connection and to verify that it works correctly.
  - **SSL CA bundle file path**: enter the path to the CA bundle file containing the root, intermediate, and public certificates for SSL authentication.
- Click **Save** to store your changes, or **Cancel** to discard them.

Configure the content type

- **Content type**: from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **TAXII inbox** transport type enables fetching data in the following formats:
  - ArcSight CEF
  - EclecticIQ Entities CSV
  - EclecticIQ Observables CSV
  - EclecticIQ JSON
  - Plain text value
  - STIX 1.2
- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.
- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
  - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

    Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
    An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

⚠ **Warning:** If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

EclecticIQ JSON

By default, the **EclecticIQ JSON** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.
- **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.
  This setting changes the `data.information_source.identity.name` value in the entity JSON structure:

```
{
  "data": {
    "information_source": {
      "type": "information-source",
      "identity": {
        "name": "<producer_identity>", // ex.: 'EclecticIQ'
        "type": "identity"
      }
    }
  }
}
```

Plain text value

The plain text value content type is suitable for machine consumption. Typical use cases include feeding a plain text value outgoing feed to an external compatible device to instrument further processing or to trigger a response action.

The plain text value content configuration options set up a rule to define the data pool that qualifies for inclusion in the outgoing feed. The rule works as follows:

- The **Field to check a conditional value in** condition looks for a specific JSON path pointing to a specific entity field in the entity JSON structure.
  - If the previous condition yields matches, the **Only use entities that match this conditional value** condition looks at the specified JSON path key for any values matching the literal or the regex data pattern you define in this field.
  - If the previous conditions yield matches, the **Field to take values from** condition points to a specific entity field whose value is fetched and included for publication in the outgoing feed.

Under **Content configuration** set the **Plain text value** content type options:

- **Field to take values from**: specifies the location in the entity JSON structure where the values to include in the feed are stored.
  Enter a JSON path pointing to the entity field whose values you want to fetch and include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Field to check a conditional value in**: this condition works together with **Only use entities that match this conditional value** to filter specific entities for the feed.
  Enter a JSON path pointing to the entity field you want to use as a filter to select entities whose content you want to include for publication in the outgoing feed.
  - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
  - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
    Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
  - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
  - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.

- **Only use entities that match this conditional value**: this condition works together with **Field to check a conditional value in** to filter specific entities for the feed.
  Enter a string to define the value to match.

Example

You can configure this rule to send relevant data to an external Snort or Suricata instance, where they can be further processed or used to initiate a specific response action:

- **Field to check a conditional value in**: *data.type.test_mechanisms.test_mechanism_type*
- **Only use entities that match this conditional value**: *snort*
- **Field to take values from**: *data.type.test_mechanisms.rules.value*

The rule uses the specified conditions to:

- Look for platform entities containing Snort rules: *data.type.test_mechanisms.test_mechanism_type: snort*
- If the previous condition yields matching entities, look in those entities if they contain this field: *data.type.test_mechanisms.rules.value*
- If they do, fetch the value from the field and include it in the outgoing feed.
  Matching values are added to the outgoing feed one value per line.
  The value in question should be a valid Snort rule for the resulting feed data to be meaningful.
  Example:

```
alert tcp $HOME_NET any -> [72.20.35.70,72.20.35.120] 6661 (msg:\"ET CNC Shadowserver Reported CnC
Server Port 6661 Group 1\"; flags:S; reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org; threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil; flowbits:set,ET.BotccIP; sid:2405018; rev:3633;)
```

STIX 1.2

The STIX 1.2 content type is suitable for machine consumption. Typical use cases include feeding a STIX 1.2 outgoing feed to an external STIX-compatible device to instrument further processing or to trigger a response action.

Under **Content configuration** set the **STIX 1.2** content type options:

- **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.
  This setting changes the following nested XML element in the entity STIX structure:

```
<stixCommon:Identity>
  <!-- Producer identity, for example 'EclecticIQ' -->
  <stixCommon:Name>EclecticIQ</stixCommon:Name>
</stixCommon:Identity>
```

Configure TAXII poll transport and content

Set up and configure transport and content types for TAXII poll outgoing feeds to publish selected platform data through the TAXII polling service.

To configure the general options for the TAXII poll outgoing feed, see Configure outgoing feeds.

About TAXII poll

This feed source enables intelligence dissemination through the following channels:

Feed	Published data
TAXII poll	The feed publishes entities and observables in the selected content type to the platform TAXII server. You can retrieve the content by accessing the TAXII polling service endpoint configured in the platform, and by specifying the name of the outgoing feed. By default, the TAXII poll endpoint is `https://<platform_host>/taxii/poll`. Each time the outgoing feed task runs, it generates a data package containing zero or more entities, depending on the outgoing feed update strategy, and on the feed data source containing data that match the feed configuration.

To view and to retrieve outgoing feed content, do the following:

- On the top navigation bar click **Data configuration > Outgoing feeds**.
- On the **Data configuration > Outgoing feeds** page, click anywhere on the row corresponding to the outgoing feed whose content you want to view or retrieve.
  The feed detail pane slides in from the side of the screen.
- On the outgoing feed detail pane click the **Content** tab.
- On the **Content** tab, click the name of a package to download it.

Configure the transport type

Under **Transport and content** you define *what* you want to publish and *how*, that is, the data content type and the data transport type.

Transport type	Allowed content types
TAXII poll	ArcSight CEF
	EclecticIQ Entities CSV
	EclecticIQ Observables CSV
	EclecticIQ JSON
	Plain text value

Transport type	Allowed content types
	STIX 1.2

Configure the transport type

The TAXII poll transport type for outgoing feeds publishes the supported content types to through the TAXII polling service.

- **Transport type**: from the drop-down menu select **TAXII poll**.

Under **Transport configuration** set the TAXII poll transport type options:

- **Public**: default setting: deselected.
  Select this checkbox to make the outgoing feed available to all platform groups and to all platform users.
  Leave it deselected to make the outgoing feed available only to specific groups. You can select the intended recipient groups in the **Authorized groups** drop-down menu.
- **Authorized groups**: restricts access to the outgoing feed to the groups you select from the drop-down menu, and to their member users.
  The **Authorized groups** option is available only when the **Public** checkbox is deselected (default setting).

Warning:
Before deleting a group, check that is not an authorized group in an outgoing feed configuration.
Deleting a group that is currently selected as an authorized group to access the outgoing feed content breaks the outgoing feed functionality.

If you need to remove such a group:

- First, remove it from the **Authorized group** selection in the relevant outgoing feed(s).
- Then, proceed to delete the group.

Configure the content type

- **Content type**: from the drop-down menu select the appropriate content type for the data you want to publish through the outgoing feed.
  The selected content type for the feed should match the data source format.
  This can vary, depending on the dataset source(s) you retrieve the data from.
  The **TAXII poll** transport type enables fetching data in the following formats:
  - ArcSight CEF
  - EclecticIQ Entities CSV
  - EclecticIQ Observables CSV
  - EclecticIQ JSON
  - Plain text value
  - STIX 1.2

- **Datasets**: from the drop-down menu select one or more existing datasets to use as sources to populate the outgoing feed.
  For the feed not to be empty, at least one selected dataset should contain entities and observables in the same format as the selected content type.
- **Update strategy**: from the drop-down menu select the preferred method to populate the outgoing feed with data before publishing it:
  - **Append**: every time the outgoing feed task runs, it fetches only new data — new entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Replace** every time the outgoing feed task runs, it fetches new and existing data — new and existing entities and observables since the previous execution of the feed — to generate the content to publish through the feed.
  - **Diff**: this option is available only for the **EclecticIQ Entities CSV** and **EclecticIQ Observables CSV** content types.

    Every time the outgoing feed task runs, new data is compared against existing data to identify any differences between the two datasets at *observable-level* — any observable added to or removed from the entities in the set — or at *entity-level* — any entities 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 being added or removed, or one observable being added or removed.
    An extra diff column is added to the output CSV to indicate if a row, and therefore either an entity or an observable, 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.

ArcSight CEF

The **ArcSight CEF** `(https://www.protect724.hpe.com/docs/doc-1072)` content type is suitable for machine consumption. Typical use cases include feeding an ArcSight CEF outgoing feed to a SIEM system such as **ArcSight ESM** `(https://saas.hpe.com/en-us/software/siem-security-information-event-management)` or to a Syslog server for further processing.

By default, the **ArcSight CEF** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Entities CSV

The EclecticIQ Entities CSV outgoing feed outputs CSV files with column headers where each row holds data describing one entity.
For example, an indicator, a TTP, and so on.

The EclecticIQ Entities CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at entity-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Entities CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed.
  Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object.
  When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

EclecticIQ Observables CSV

The EclecticIQ Observables CSV outgoing feed outputs CSV files with column headers where each row holds data describing one observable.
For example, an IP address, a hash, a geographic location name, and so on.

The EclecticIQ Observables CSV data format enables you to compare different outputs from the same outgoing feed to diff them and examine any changes at observable-level. To do so, under **Update strategy** select **Diff**.

By default, the **EclecticIQ Observables CSV** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed.
  Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.

- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed. Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.

> **Warning:** If you select **EclecticIQ Observables CSV**, you need to choose at least one observable type from the **Observable types** drop-down menu, and at least one enrichment observable type from the **Enrichment observable types** drop-down menu.

EclecticIQ JSON

By default, the **EclecticIQ JSON** content type outgoing feed includes only *first level*, *original* observables:

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

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

- **Include derived observables**: select this checkbox to include derived observables in the outgoing feed. Derived observables hold a data subset of the original observable they are extracted, that is, derived, from. The original observable is analyzed to look for specific patterns. When matching data is found, it is extracted and saved to a distinct, derived observable that holds the analyzed data.
  Example:
  - Original observable is a URI: *http://www.evil.com/big/info.php*
  - The corresponding extracted, derived observable is a domain name: *evil.com*.
- **Include secondary observables**: select this checkbox to include secondary observables in the outgoing feed. Secondary observables hold data found inside a STIX field, and therefore not included in a CybOX observable object. When focusing on observables, data retrieved from STIX fields is usually not as relevant as data retrieved from CybOX fields. When included, secondary observables may introduce data noise.
  Example:
  - A hash or a domain name included in the title or in the description fields of a STIX object.
- **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.
  This setting changes the `data.information_source.identity.name` value in the entity JSON structure:

```
{
  "data": {
    "information_source": {
      "type": "information-source",
      "identity": {
        "name": "<producer_identity>", // ex.: 'EclecticIQ'
        "type": "identity"
      }
    }
  }
}
```

Plain text value

The plain text value content type is suitable for machine consumption. Typical use cases include feeding a plain text value outgoing feed to an external compatible device to instrument further processing or to trigger a response action.

The plain text value content configuration options set up a rule to define the data pool that qualifies for inclusion in the outgoing feed. The rule works as follows:

- The **Field to check a conditional value in** condition looks for a specific JSON path pointing to a specific entity field in the entity JSON structure.
    - If the previous condition yields matches, the **Only use entities that match this conditional value** condition looks at the specified JSON path key for any values matching the literal or the regex data pattern you define in this field.
    - If the previous conditions yield matches, the **Field to take values from** condition points to a specific entity field whose value is fetched and included for publication in the outgoing feed.

Under **Content configuration** set the **Plain text value** content type options:

- **Field to take values from**: specifies the location in the entity JSON structure where the values to include in the feed are stored.
  Enter a JSON path pointing to the entity field whose values you want to fetch and include for publication in the outgoing feed.
    - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
    - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
      Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
    - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
    - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Field to check a conditional value in**: this condition works together with **Only use entities that match this conditional value** to filter specific entities for the feed.
  Enter a JSON path pointing to the entity field you want to use as a filter to select entities whose content you want to include for publication in the outgoing feed.
    - The JSON path starts at `data`: the first member at the beginning of a JSON path needs to be `data`.
    - The JSON path is a string that points to a location, that is, a field inside a JSON object. It tells the rule *where* in the entity structure it should go look for the corresponding data value.
      Think of it as a friend's address you scribble on the back of a postcard before dropping it into the mailbox.
    - The JSON path format is a string where dots (`.`) define JSON parent-child relationships.
    - Do not include square brackets (`[ ]`) in the path input: they are stripped during execution. It is not possible to use square brackets to point to specific array members.
- **Only use entities that match this conditional value**: this condition works together with **Field to check a conditional value in** to filter specific entities for the feed.
  Enter a string to define the value to match.

Example

You can configure this rule to send relevant data to an external Snort or Suricata instance, where they can be further processed or used to initiate a specific response action:

- **Field to check a conditional value in**: *data.type.test_mechanisms.test_mechanism_type*
- **Only use entities that match this conditional value**: *snort*
- **Field to take values from**: *data.type.test_mechanisms.rules.value*

The rule uses the specified conditions to:

- Look for platform entities containing Snort rules: *data.type.test_mechanisms.test_mechanism_type: snort*

- If the previous condition yields matching entities, look in those entities if they contain this field: *data.type.test_mechanisms.rules.value*
- If they do, fetch the value from the field and include it in the outgoing feed.
  Matching values are added to the outgoing feed one value per line.
  The value in question should be a valid Snort rule for the resulting feed data to be meaningful.
  Example:

```
alert tcp $HOME_NET any -> [72.20.35.70,72.20.35.120] 6661 (msg:\"ET CNC Shadowserver Reported CnC
Server Port 6661 Group 1\"; flags:S; reference:url,doc.emergingthreats.net/bin/view/Main/BotCC;
reference:url,www.shadowserver.org; threshold: type limit, track by_src, seconds 360, count 1;
classtype:trojan-activity; flowbits:set,ET.Evil; flowbits:set,ET.BotccIP; sid:2405018; rev:3633;)
```

STIX 1.2

The STIX 1.2 content type is suitable for machine consumption. Typical use cases include feeding a STIX 1.2 outgoing feed to an external STIX-compatible device to instrument further processing or to trigger a response action.

Under **Content configuration** set the **STIX 1.2** content type options:

- **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.
  This setting changes the following nested XML element in the entity STIX structure:

```
<stixCommon:Identity>
  <!-- Producer identity, for example 'EclecticIQ' -->
  <stixCommon:Name>EclecticIQ</stixCommon:Name>
</stixCommon:Identity>
```

Outgoing feeds reference

Reference section with lookup information on supported outgoing feed content types and transport types.

Available outgoing feeds

The overview lists and points to the articles on the available outgoing feeds. Each article describes how to configure the specific options for each outgoing feed.
Typically, outgoing feeds use different *transport types* and *content types*. General configuration options are identical across all outgoing feeds.

Title	Excerpt
Configure email transport and content	Set up and configure transport and content types for Send email outgoing feeds to publish selected platform data as email attachments.
Configure FTP upload transport and content	Set up and configure transport and content types for FTP upload outgoing feeds to publish selected platform data to an FTP server.
Configure HTTP download transport and content	Set up and configure transport and content types for HTTP download outgoing feeds to publish selected platform data to an HTTP server.
Configure Mount point upload transport and content	Set up and configure transport and content types for Mount point upload outgoing feeds to publish selected platform data to a specific location on a local or network unit.
Configure Syslog push transport and content	Set up and configure transport and content types for Syslog push outgoing feeds to publish selected platform data to a Syslog server.
Configure TAXII inbox transport and content	Set up and configure transport and content types for TAXII inbox outgoing feeds to publish selected platform data through the TAXII inbox service.
Configure TAXII poll transport and content	Set up and configure transport and content types for TAXII poll outgoing feeds to publish selected platform data through the TAXII polling service.

Content types

These are the data formats the platform can process through feeds.
Under *Feed type* **in** defines an input format that incoming feeds ingest; **out** defines an output format that outgoing feeds publish.

Content type	Feed type	Description
Anubis Cyberfeed JSON	in	JSON format representing entity data as JSON objects.
ArcSight CEF	out	The Common Event Format is a text-based standard for log records proposed by ArcSight. It allows sharing, consuming, and parsing event information across devices such as SIEM platforms and Syslog servers.
Cisco Threat Grid Samples JSON	in	JSON format representing entity data as JSON objects.
EclecticIQ Entities CSV	out	Comma separated CSV format for tabular data representation of entities.
EclecticIQ JSON	in, out	JSON format representing entity data as JSON objects.
EclecticIQ Observables CSV	out	Comma separated CSV format for tabular data representation of observables.
Group-IB accounts, Group-IB cards, Group-IB IMEIs	in	Group-IB proprietary data format to exchange information on compromised accounts, payment cards, and mobile devices.
Intel 471	in	Intel 471 proprietary data format.
PDF	in, out	Standard PDF format, preferably native (not scanned).
STIX 1.0	in, out	STIX data model **v. 1.0** `(http://stixproject.github.io/data-model/1.0/)`.
STIX 1.1	in, out	STIX data model **v. 1.1** `(http://stixproject.github.io/data-model/1.1/)`.
STIX 1.1.1	in, out	STIX data model **v. 1.1.1** `(http://stixproject.github.io/data-model/1.1.1/)`.
STIX 1.2	in, out	STIX data model **v. 1.2** `(http://stixproject.github.io/data-model/1.2/)`.
Text/Plain text value	in, out	Plain text format. This content type allows you to enter free text and literals, wildcards (where supported), as well as JSON paths to point to specific entity property fields, and regex patterns to filter data.
Threat Recon	in	Threat Recon JSON output returned by the **Threat Recon API** `(https://threatrecon.co/api)`. Threat Recon focuses on providiung information about indicators.
STIX 1.1.1	in	FireEye iSIGHT Intelligence Report API outputs reports in STIX 1.1.1 format. Reports concern threat topics such as vulnerabilities, malware, threat actors, stategies, tactics, and techniques.
BFK Threat Intelligence JSON	in	BFK reports and NIDs (Network Intrusion Detections) are saved as JSON report entities; they concern threat topics such as threat actors, targeted victims, tactics, and techniques.

Content type	Feed type	Description
Crowdstrike Indicator JSON	in	Indicators retrieved from the Falcon Intelligence platform are stored as JSON; they concern compromised devices, malicious domains, hashes, and so on starting from a specified polling date.

Transport types

These are the supported communications protocols the platform uses to publish data through outgoing feeds.

Transport type	Feed type	Description
FTP upload	out	Custom feed to publish data through an FTP server.
HTTP download	out	Custom feed to publish data through an HTTP server. By default, the outgoing feed content is available through the following platform API endpoints: `/api/open-outgoing-feed-download/` for public outgoing feeds, and `/api/outgoing-feed-download/` for private outgoing feeds.
Mount point upload	out	Custom feed to publish data from a location on a local or network unit.
Send email	out	Custom feed to publish data as email attachments.
Syslog push	out	Custom feed to share data with other devices using the Syslog protocol. Usually, Syslog messages are centralized to a Syslog server.
TAXII inbox	out	Custom feed using the TAXII inbox service to publish data.
TAXII poll	out	Custom feed using the TAXII polling service to publish data.