ThreatQuotient

ThreatQ CDF Approval Guide

Version 1.0.0

Tuesday, March 10, 2020

ThreatQuotient

11400 Commerce Park Dr., Suite 200

Reston, VA 20191

Support

Email: support@threatq.com

Web: support.threatq.com

Phone: 703.574.9893

Warning and Disclaimer

ThreatQuotient, Inc. provides this document “as is”, without representation or warranty of any kind, express or implied, including without limitation any warranty concerning the accuracy, adequacy, or completeness of such information contained herein. ThreatQuotient, Inc. does not assume responsibility for the use or inability to use the software product as a result of providing this information.

Last Updated: Tuesday, March 10, 2020

Contents

ThreatQ CDF Approval Guide **1**

Warning and Disclaimer **2**

Contents **3**

ThreatQ CDF Approval Process **4**

CDF Development 4

CDF Best Practices 5

General Definition Configuration 5

Feed Specific Configuration 5

User Field Best Practices 6

Source Best Practices 6

Filter Chain Best Practices 7

Report Best Practices 7

Performance 8

Installation 8

Submitting a CDF for Approval 8

Engineering Review 10

Post Approval 11

ThreatQ CDF Approval Process

Have a Configuration Driven Feed that you'd like to share and make available to all ThreatQuotient users? We'd love to help!

This section details our internal CDF Approval Process, laying out:

- What is needed from the definition writer in order to submit a CDF for approval.
- How and where to submit a CDF for approval.
- The review steps ThreatQ Engineering will take after submission.
- What to expect once a CDF has been approved.

CDF Development

The first step in the process, obviously, is writing the CDF. When implementing, be sure to follow the CDF Best Practices. Keep in mind that once the CDF is finished the following files will be required in order to approve the CDF:

- A logo image for the Vendor the Feed(s) connect to. Recommended size for this image is 240px X 112px
- A `README.md` file laying out the data mappings for the Feed(s) contained within the YAML definition.

  Download the CDF README Template

  Download a CDF README Example

- The YAML definition file Once the CDF has been written, tested, and documented, it can be submitted for approval.

CDF Best Practices

The following are considered "Best Practices" when writing a CDF and should be followed as closely as possible. During the approval process, these points will be checked directly along with general error and performance tests.

General Definition Configuration

The feed definition must specify the following:

Requirement	Details
version	YAML definition version string.
required_threatq_version	Version of ThreatQ required to run this CDF's Feeds.

Generally, repeated logic within a YAML definition should be abstracted into YAML anchors where possible.

Feed Specific Configuration

Requirement	Details
namespace	Must be unique to each Feed in a given YAML definition. This should closely resemble the name, eg: • threatq.feeds.RecordedFuture.IPRisk • commerical.cofense
category	Category under which the Feed will appear after install. Valid options include: • Commercial • OSINT
default_indicator_status	Default status to apply to Indicators ingested by the Feed in question. Only required if the Feed ingests Indicators.

Requirement	Details
default_signature_status	Default status to apply to Signatures ingested by the Feed in question. Only required if the Feed ingests Signatures

User Field Best Practices

- Password fields must be masked in the UI via the mask: True flag.
- User Fields should be configured as a list of mappings, see User Fields and Parameters on dev-docs for more information on the formatting there.
- User Fields should include a description unless the field's usage is obvious from the label.
- User Fields should leverage the appropriate input type - eg. a boolean value should be displayed as a checkbox, not a text input.
- User Fields necessary for the Feed to run should be marked as required via the required: True flag.
- User Fields that can have default values should specify such a value via the default: <Some Value> flag.

Source Best Practices

- Authentication information should be stored under the auth field so that it is not recorded in request / response files.
- The response_content_type field should be specified so that the Feed will continue to run if the vendor fails to send an appropriate Content-Type response header in the future.
- If possible, the leverage run_meta.since and run_meta.until in order to give the Feed manual run support.
- Pagination Best Practices:
    - If possible, leverage page_count rather than dynamically calculating a paging value.

- Only values that need to be changed on each paginated request should be included under the pagination field.

Filter Chain Best Practices

- All datetime values should be formatted via the Timestamp Filter.
- The Set Filter must not change and act on the same key within the same declaration - see the linked documentation on dev-docs for more information.
- Extraneous filters should be avoided.
- Formatting of Supplemental Feed data should be done within the Supplemental Feed as much as possible rather than the Primary Feed.
- Prefer built in CDF Filters over Jinja2 filters.

Report Best Practices

- Building out object structures should be done in the Report as much as possible.
- Object-Set formatting should be of the following format:

```
report:
  <ObjectType>-sets:
        my_set_name:
          items:
                - <Object Declaration> ...
```

- When possible, the Object-Set related syntax should be prefered over building out related objects in the Filter Chain.
- Beware the law of combinatorics when leveraged the inter-related feature:
  - Inter-relating 1000 objects results in 499,500 operations as per the formula (n!)/(k!(n-k)!) where n is the total number of objects and k is the number of objects picked each combination (2).

- Do not report object values that are possibly None / null / empty strings. These should be removed via the Filter Chain or condition flags within the Report.
- When creating a boolean Attribute value, one should Report the boolean value as a Title-cased string.

Performance

- Feed execution time should not exceed the frequency at which it is configured to run, (currently - 24 hours max).
- Feed execution should not use an unreasonable amount of memory. The Feed must be runnable on a box with 32GB of RAM while other processes continue unabated.
- Feed execution should not make an unreasonable amount of API calls to either the external vendor or the TQ API.
- Feed execution should take possible blacklisting by the vendor's API into account.

Installation

From the Front-end, the YAML definition must support:

- Installation
- Upgrade
- Uninstallation

Submitting a CDF for Approval

To submit a CDF for approval, visit the ThreatQ CDF Approval Form and fill out the required information.

The following questions will need to be answered:

Question	Details
CDF Name	This is the name that will be displayed on the ThreatQ Marketplace. Feed names defined within the YAML definition will remain unchanged.
Short Description	Small text blurb that will be displayed on the CDF's overview card on the ThreatQ Marketplace.
Description	Long-form description of the CDF, defining the Feed(s) contained with the YAML definition, general behavior, requirements, and the data vendor.
Publisher	Optional, defaults to ThreatQuotient. Denotes the author of the CDF and is displayed on the ThreatQ Marketplace.
Category	Dropdown with two options: • Commercial Intelligence • Open Source Intelligence If following development best practices, this should already be defined within the YAML definition.
Vendor Link	URL to the Vendor that the CDF in question connects to. This is displayed on the ThreatQ Marketplace within the CDF's detail page.
CDF Version	Version of the YAML definition file. If following development best practices, this should already be defined within the YAML definition.
Required ThreatQ Version	Version of ThreatQ required to install this CDF. If following development best practices, this should already be defined within the YAML definition.
Does this CDF update a	Radio options: **Yes** or **No**. Denotes if the submission is an update to an already approved / published CDF. This will help Engineering

Question	Details
previous version?	assure the right records are updated during its review process
Vendor Logo	A logo image for the Vendor the Feed(s) connect to.
README File	A README.md file laying out the data mappings for the Feed(s) contained within the YAML definition.
YAML Definition	The CDF definition file in .yaml format.

One's email address will be automatically collected by the form and a response receipt will be sent if requested.

Once submitted, the writer's job is done! Engineering will take the reins now and begin the review process

Engineering Review

Once a submission is received, Engineering will take the following steps to review and ultimately approve a CDF:

- The submitted materials will be gathering into an internal JIRA ticket for tracking.
- The approval JIRA ticket will be quickly prioritized into an upcoming Sprint.
- Once picked up for development, the CDF will be reviewed along the following lines:
    - Validation against the CDF Best Practices list Inspection of submitted data mappings and the CDF for data integrity concerns and general user experience and usability.
    - General Feed Run performance.

- Based on the review results, the YAML definition and README file may be tweaked slightly. Large changes or concerns will be communicated with the CDF writer directly.
- Submitted data mappings are converted into a ThreatQ Help Center document.

After final tweaks, the CDF will be considered Approved and merged to Engineering's Feed Definitions repository.

Post Approval

Once approved, a **Go No-Go** meeting will be scheduled including Engineering, Product, TIS/TIEs, and any others that may have stake in the new CDF.

Once it has received a **Go**, the CDF will be uploaded to the ThreatQ Marketplace.