

ThreatQuotient



ThreatQ Service Alerts Library Guide

Version 1.4.2

October 10, 2022

ThreatQuotient

20130 Lakeview Center Plaza Suite 400
Ashburn, VA 20147

 Not Actively Supported

Contents

- Integration Details..... 5
- Introduction 6
- Installation..... 7
- Usage..... 8
 - Service Module 8
 - Notifier Module 9
 - Formatter Module..... 9
 - Slack Notifier Example..... 10
- Change Log..... 11

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.

Copyright © 2022 ThreatQuotient, Inc.

All rights reserved. This document and the software product it describes are licensed for use under a software license agreement. Reproduction or printing of this document is permitted in accordance with the license agreement.

Support

This integration is designated as **Not Actively Supported**.

Integrations, apps, and add-ons designated as **Not Actively Supported** are not supported by ThreatQuotient's Customer Support team.

While you can report issues to ThreatQ's Customer Support team regarding the integration/app/add-on, you are solely responsible for ensuring proper functionality and version compatibility of Not Supported designations with the applicable ThreatQuotient software.

If unresolvable functional or compatibility issues are encountered, you may be required to uninstall the integration/app/add-on from your ThreatQuotient environment in order for ThreatQuotient to fulfill support obligations.



For ThreatQuotient Hosted instance customers, the Service Level Commitment and Service Level Credit in the ThreatQuotient Service Level Schedule will not apply to issues caused by Not Actively Supported integrations/apps/add-ons.

Integration Details


ThreatQuotient provides the following details for this integration:

Current Integration Version	1.4.2
Compatible with ThreatQ Versions	>= 4.0.0
Support Tier	Not Actively Supported
ThreatQ Marketplace	https:// marketplace.threatq.com/ details/threatq-service- alerts-library-guide

Introduction

The ThreatQ Service Alerts Library provides the required framework to enable you to create an alert service for the ThreatQ platform and integrations.

Installation

 **Upgrading Users** - Review the [Change Log](#) for updates to configuration parameters before updating. If there are changes to the configuration file (new/removed parameters), you must first delete the previous version's configuration file before proceeding with the install steps listed below. Failure to delete the previous configuration file will result in the connector failing.

1. Navigate to the ThreatQ Marketplace and download the .whl file for the library.
2. Transfer the whl file to the /tmp directory on your ThreatQ instance.
3. Install the library on your ThreatQ instance:

```
<> pip install /tmp/tq_alert_service-<version>-<python version>-  
none-any.whl
```

Usage

The following sections contain information on creating your own alert service.

In order to create your own notification service, you will need to create 3 python modules/files.

- [Service Module](#) - The code that will check for alerts.
- [Notifier Module](#) - The code that will send the alert to the third party.
- [Formatter Module](#) - The code that will format the response in between the service to the notifier.

Once these modules are completed, see the [Slack Notifier Example](#) to see how to use it in an integration.

Service Module

The Service Module will handle downloading the alerts from a specified service.

To create your own, create a python module that inherits the BaseService module from `tq_alert_service.services`.

The TQServiceWorker will execute the following methods in order. If you are creating a service, you will override most of these methods.

METHOD	DESCRIPTION
<code>connect()</code>	This method is where you will connect to your service (if needed).
<code>fetch_alerts()</code>	This method is where you will manage getting the alerts from ThreatQ or other service.
<code>sanitize_alerts()</code>	This method is where you will do any sanitization of the alerts from the <code>fetch_alerts()</code> method.

METHOD

DESCRIPTION

<code>send_alerts()</code>	This method does not need to be overridden by default. It is used to send the alerts via your notifier module.
----------------------------	--

Notifier Module

The Notifier Module will handle notifying the 3rd party service.

To create your own, create a python module that inherits the BaseNotifier module from `tq_alert_service.notifiers`.

When creating your own service, you will create a Notifier module and add it to the service. The service will dispatch the alert to the notifier once the alerts have been downloaded.

The following are methods you can use/override from the BaseNotifier:

METHOD

DESCRIPTION

<code>set_formatter()</code>	This method is used to give the notifier a formatter to format the given alerts.
------------------------------	--

<code>format_alert()</code>	This method must be called if you are using a formatter, as it will format the alerts however you specify in your Formatter.
-----------------------------	--

<code>send_alert()</code>	This method allows you to send the alert to the 3rd party.
---------------------------	--

Formatter Module

The Formatter Module will handle formatting the results from your service module so that it can be sent correctly to your notifier.

To create your own, create a python module that inherits the BaseFormatter module from `tq_alert_service.formatters`.

METHOD	DESCRIPTION
<code>set_data()</code>	This method allows you to set the data that will be formatted.
<code>format_data()</code>	This method allows you to execute the formatting.
<code>get_formatted()</code>	This method allows you to get the output formatted data from the <code>format_data()</code> method

Slack Notifier Example

The following is an example of how to use the ThreatQ Alert Service in an integration.

```
from tq_alert_service.helpers import NotificationTypes
from tq_alert_service.notifiers import SlackNotifier
from tq_alert_service.formatters import SlackFormatter
from tq_alert_service.services import WatchlistService
from tq_alert_service import TQServiceWorker

# Load slack API
slack = SlackClient('<BOT API TOKEN>')
# Create the service worker
worker = TQServiceWorker()
# Instantiate the watchlist service
service = WatchlistService(connector, config)
# Instantiate the slack notifier
channel_names = ['general', 'tq-alerts']
user_names = ['zach', 'user1', 'user2']
notifier = SlackNotifier(connector, slack, channel_names=channel_names, user_names=user_names)
formatter = SlackFormatter(connector, slack, NotificationTypes.WATCHLIST)
# Link the formatter to the notifier and the notifier to the service
notifier.set_formatter(formatter)
service.add_notifier(notifier)
# Add the service to the worker
worker.add_service(service)
# Start the worker and it will run all the services
worker.start()
```

Change Log

- **Version 1.4.2**
 - Fixed an issue utilizing data collections within SavedSearchService.
- **Version 1.4.1**
 - Fixed an issue that prevented the data collection notifications from working properly.
 - Saved Search is now Data Collection.
- **Version 1.4.0**
 - Initial Release