

ThreatQuotient



Resilient Connector Guide

Version 1.2.0

April 06, 2021

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.

Copyright © 2021 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.

Contents

Versioning.....	4
Introduction.....	5
Prerequisites	6
Installation	7
Configuration.....	10
Usage.....	12
Command Line Arguments	12
CRON	13
Generating a Certificate	14
Mapping	15
Known Issues / Limitations.....	16
Change Log.....	17

Versioning

- Current integration version: 1.2.0
- Supported on ThreatQ versions $\geq 4.24.0$

There are two versions of this integration:

- Python 2 version
- Python 3 version

Introduction

Resilient Connector for ThreatQuotient allows new context from ThreatQ to be exported to your Resilient instance. It has the ability to push new indicators and comments from updated Resilient incidents in ThreatQ to Resilient as artifacts and comments, respectively.

Prerequisites

Python 2 Environment Users

You will need to install the `setuptools_scm` v5.0.2 before installing/upgrading the connector if you are deploying the connector in a Python 2 environment. The Resilient package contains the latest version which is not compatible with Python 2.

1. Run the following command to install `setuptools_scm`:

```
< > pip install setuptools_scm==5.0.2
```

2. Modify the `pip.conf` file in your environment as so:

```
[global]
index-url = https://system-updates.threatq.com/pypi
extra-index-url = https://<username>:<password>@extensions.threatq.com/threatq/integrations
                  https://<username>:<password>@extensions.threatq.com/threatq/sdk
```

Installation

The connector can be installed from the ThreatQuotient repository with YUM credentials or offline via a .whl file.

⚠ 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. Install the connector using one of the following methods:

ThreatQ Repository

- a. Run the following command:

```
< > pip install tq_conn_resilient
```

Offline via .whl file

To install this connector from a wheel file, the wheel file (.whl) will need to be copied via SCP into your ThreatQ instance.

- a. Download the connector whl file with its dependencies:

```
< > mkdir /tmp/tq_conn_resilient
      pip download tq_conn_resilient -d
      /tmp/tq_conn_resilient/
```

- b. Archive the folder with the .whl files:

```
< > tar -czvf tq_conn_resilient.tgz /tmp/tq_conn_resilient/
```

- c. Transfer all the whl files, the connector and all the dependencies, to the ThreatQ instance.
- d. Open the archive on ThreatQ:

```
< > tar -xvf tq_conn_resilient.tgz
```

- e. Install the connector on the ThreatQ instance.



The example assumes that all the whl files are copied to /tmp/conn on the ThreatQ instance.

```
< > pip install /tmp/conn/tq_conn_resilient-<version>-<python version>-none-any.whl --no-index --find-links /tmp/conn/
```



```
pip install /tmp/conn/tq_conn_resilient1.2.0-py2-none-any.whl --no-index --find-links /tmp/conn/
```



A driver called tq-conn-resilient will be installed. After installing with pip or setup.py, a script stub will appear in /usr/bin/resilient.

2. Once the application has been installed, a directory structure must be created for all configuration, logs and files, using the mkdir -p command. Use the commands below to create the required directories:

```
< > mkdir -p /opt/tq-integrations/resilient  
  
tq-conn-resilient -v 3 -ll /opt/tq-integrations/resilient/ -c /opt/tq-integrations/resilient
```

3. Perform an initial run using the following command:

```
< > tq-conn-resilient /path/to/config/directory/ -ll /path/to/log/directory/ -v3  
VERBOSITY_LEVEL stdout
```



You can include a -n flag in the command above to rename the connector. See the [Known Issues / Limitations](#) section for more details.



```
tq-conn-resilient /path/to/config/directory/ -ll /path/to/log/directory/ -v3  
VERBOSITY_LEVEL -n "Resilient2"
```

4. Enter the following parameters when prompted:

PARAMETER	DESCRIPTION
ThreatQ Host	This is the host of the ThreatQ instance, either the IP Address or Hostname as resolvable by ThreatQ.

PARAMETER	DESCRIPTION
Client ID	This is the OAuth id that can be found at Settings Gear → User Management → API details within the user's details.
Email Address	This is the User in the ThreatQ System for integrations.
Password	The password for the above ThreatQ account.
Status	This is the default status for objects that are created by this Integration. It is common to set this to "Review", but Organization SOPs should be respected when setting this.

Example Output

```
tq-conn-resilient -c /opt/tq-integrations/resilient/config/ -ll /opt/tq-integrations/resilient/logs/ -ds -v3
```

ThreatQ Host: <ThreatQ Host IP or Hostname>

Client ID: <ClientID>

E-Mail Address: <EMAIL ADDRESS>

Password: <PASSWORD>

Status: **Review**

Connector configured. Set information in UI

You will still need to [configure and then enable the connector](#).

Configuration



ThreatQuotient does not issue API keys for third-party vendors. Contact the specific vendor to obtain API keys and other integration-related credentials.


To configure the integration:





1. Navigate to your integrations management page in ThreatQ.
2. Select the **Labs** option from the *Category* dropdown (optional).



If you are installing the connector for the first time, it will be located under the **Disabled** tab.

3. Click on the integration to open its details page.
4. Enter the following parameter under the **Configuration** tab:

PARAMETER	DESCRIPTION
Resilient Host	The hostname or IP address of your Resilient instance.
Resilient Username	The email you will use authenticate with the Resilient API.
Resilient Password	The password you will use authenticate with the Resilient API.
Resilient Organization	Your Organization within your Resilient instance.
Resilient Certificate Path	he path to your Resilient certificate. <div> This field is optional. If left blank, the SSL will not be verified. See the Generating a Certificate section for instructions on how generate a certificate. If using this</div>

PARAMETER	DESCRIPTION
	method, verify that the certificate is accessible to the connector.
Attribute to Custom Field Mapping	<p>Map ThreatQ attributes to Custom Fields in Resilient. Each mapping must be on its own line and is in a key=value format.</p> <div> Confidence=confidence_level</div> <div> The Resilient Custom Field name must be the programmatic API name. See the Customization Settings section within your Resilient instance for more details.</div>
ThreatQ to Custom Field Mapping	<p>Map ThreatQ objects to Custom Fields in Resilient. Each mapping must be on its own line and is in a key=value format.</p> <div> TTP=mitre_technique_name</div> <div> The Resilient Custom Field name must be the programmatic API name. See the Customization Settings section within your Resilient instance for more details.</div>

5. Review the **Settings** configuration, make any changes if needed, and click on **Save**.
6. Click on the toggle switch, located above the *Additional Information* section, to enable it.

Usage

Use the following command to execute the Resilient connector:

```
<> tq-conn-resilient -c /path/to/config/directory/ -ll /path/to/log/directory/ -v3
```



Include the `-n` flag in the command above if you renamed the connector. See the [Known Issues / Limitations](#) section for more details.

Command Line Arguments

This connector supports the following custom command line arguments:

ARGUMENT	DESCRIPTION
<code>-h, --help</code>	Shows this help message and exits.
<code>-ll LOGLOCATION, --loglocation LOGLOCATION</code>	Sets the logging location for the connector. The location should exist and be writable by the current. A special value of 'stdout' means to log to the console (this happens by default).
<code>-c CONFIG, --config CONFIG</code>	This is the location of the configuration file for the connector. This location must be readable and writable by the current user. If no config file path is given, the current directory will be used. This file is also where some information from each run of the connector may be put (last run time, private oauth, etc.)
<code>-v {1,2,3}, --verbosity {1,2,3}</code>	This is the logging verbosity level where 3 means everything.
<code>-n, --name</code>	Changes the name of the connector.

CRON

Automatic CRON configuration has been removed from this script. To run this script on a recurring basis, use CRON or some other jobs scheduler. The argument in the CRON script must specify the config and log locations.

Add an entry to your Linux crontab to execute the connector at a recurring interval. Depending on how quickly you need updates, this can be run multiple times a day (no more than once an hour) or a few times a week.

In the example below, the command will execute the connector every two hours.

1. Log into your ThreatQ host via a CLI terminal session.
2. Enter the following command:

```
< > crontab -e
```

This will enable the editing of the crontab, using vi. Depending on how often you wish the cronjob to run, you will need to adjust the time to suit the environment.

3. Enter the commands below:

Every 2 Hours Example

```
< > 0 */2 * * * tq-conn-resilient -c /path/to/config/directory/ -ll /path/to/log/directory/ -ll /path/to/config/directory/ -v3
```

4. Save and exit CRON.

Generating a Certificate

Use the following command to generate a certificate to use with the connector:

```
< > openssl s_client -connect <SERVER>:443 -showcerts -tls1 < /dev/null > cacerts.pem  
2> /dev/null
```



The full path of the generated .pem file should be used as the certificate path in the connector configuration

Mapping

The table below illustrates how ThreatQ threat intel is mapped when it is submitted to your Resilient instance.

THREATQ	RESILIENT
Indicators	Artifacts
Malware Objects	Artifacts
Comments	Notes
Attributes	Custom Field
Objects	Custom Fields

Known Issues / Limitations

- Disabling and re-enabling the connector in the ThreatQ UI, after installing the connector, will result in an error with the connector failing to enable. If you encounter this issue, the connector must be re-installed and renamed using the -n flag when first initializing the connector in step 3 of the [Installation](#) process.

Performing Initial Run with New Name Example Command

```
< > tq-conn-resilient /path/to/config/directory/ -ll /path/to/log/directory/ -v3  
VERBOSITY_LEVEL -n "Resilient2"
```


Change Log

- **Version v1.2.0**
 - Added Python3 support
 - Updated Threat Library Code to reflect updates in the ThreatQ platform
- **Version v1.1.1**
 - Added support for syncing Malware Objects as Malware Family/Variant artifacts
 - Added support for syncing Attributes from ThreatQ to Custom Fields in Resilient
 - Added support for syncing Objects from ThreatQ to Custom Fields in Resilient
 - Updated documentation with Context Support Section
 - Updated documentation with new configuration fields
 - Added version 4.0.2 to the configparser requirement in setup.py
- **Version 1.0.0**
 - Added the ability to sync new indicators to Resilient
 - Added the ability to sync new comments to Resilient