ThreatQuotient for Request Tracker

March 8, 2019

Connector Version 2.0.0

Extension Version 1.0.0

11400 Commerce Park Dr
Suite 200,
Reston, VA
20191, USA
https://www.threatq.com/
Support: support@threatq.com
Sales: sales@threatq.com

Contents

List of Figures and Tables

1 Introduction

1.1 Application Function

This section describes the functionality of the two Request Tracker integrations: custom connector and extension.

1.2 Request Tracker Custom Connector

This integration's purpose is to ensure that Request Tracker tickets that have been created in ThreatQ are continuously synced with Request Tracker. It polls ThreatQ's API and checks for any updated/touched Request Tracker tickets (events). If it detects an updated Request Tracker ticket, it will find that ticket in Request Tracker and update it with the new information.

This section of the overall integration is installed as any other ThreatQ integration and will be further configured via the Incoming Feeds page in the ThreatQ user interface.

1.3 Request Tracker Extension

This integration's purpose is to asynchronously work with Request Tracker to keep ThreatQ and Request Tracker up to date with each other. When this extension is installed, it will add the following new 'objects' to your Request Tracker instance:

- 3 New Custom Fields
  - **ThreatQ Link**
  - **ThreatQ Attributes**
  - **ThreatQ Indicators**
- 1 New ScripAction
  - **Sync ticket with ThreatQ**
- 1 New Scrip
  - **Automatically Sync Ticket with ThreatQ**
- 2 New Context Menus
  - **Ticket Page > ThreatQ > Sync Ticket**
  - **Home Page > Admin > Tools > Edit ThreatQ Config**
    - These menus will only be viewable by Super Users (admins).

The new custom fields can be applied to Queues, so when you create a ticket in a specified Queue, these fields will be viewable (and some editable). The *ThreatQ Link* and *ThreatQ Indicators* fields are not editable and will be populated by ThreatQ when the ticket is synced. The *ThreatQ Attributes* field is editable and allows you to add attributes to the ThreatQ event using this field.

The new ScripAction will be invoked by a Scrip (rule) when a transaction is made to a ticket. Each transaction has a type. By default, the ticket will be fully synced. However, if the transaction type is part of the following, only certain actions will get invoked. Here is a list of the transaction types and what will happen when invoked:

- Comment/Correspond
  - Syncs comments
- Create
  - Creates the ticket in ThreatQ and syncs all details

A five second delay will be added to this command to ensure the ticket is created.

- AddLink
  - Syncs related tickets
- CustomField
  - Syncs all Custom Fields and attributes/metadata
- Status
  - Syncs the status (Rejects, Resolved, or Queue Name if not the previous two)
- Other
  - Full sync of all fields

In the new context menus, the main menu will be *Edit ThreatQ Config*. This is where you will edit the configuration that you initially placed in `/opt/rt4/etc/RT_SiteConfig.pm`.

Password fields **must** be edited from `/opt/rt4/etc/RT_SiteConfig.pm`, as they will not be editable from the configuration page. Once a field is saved (excluding password fields), it will be transferred to the ThreatQ database entry and will not need to be edited from the `/opt/rt4/etc/RT_SiteConfig.pm`. It may then be removed from the SiteConfig.

Request Tracker does not have any paradigm for an indicator, so there is no way to transfer indicator attributes from ThreatQ to Request Tracker. Instead, there is a *ThreatQ Indicators* custom field which will show the related indicators to the ThreatQ event, as well as a link to that indicator in ThreatQ. This will enable the user to easily pivot back and forth between Request Tracker and ThreatQ.

This integration also supports the following Custom Fields (not automatically created by ThreatQ):

- **Tags**: Multiline field
- **External Ticket Number:** Single line field
- **Systems Re-Imaged:** Multiline field
- **Threat TTP:** Multiline field
- **IP:** Multiline field
  - Synced as related 'IP Address' Indicator
- **FileNames:** Multiline field
  - Synced as related 'Filename' Indicator
- **HASH:** Multiline field
  - Synced as related 'MD5', 'SHA-1', or 'SHA-256' Indicator
- **Domains**: Multiline field
  - Synced as related 'FQDN' Indicator
- **Sender:** Single line field
  - Synced as related 'Email Address' Indicator
- **URL:** Single line field
  - Synced as related 'URL' Indicator
- **Threat Actors:** Multiline field
  - Synced as related Adversary
- **Threat Campaign:** Single line field
  - Synced as related Event (type: Campaign)

Request Tracker's native language is PERL, so every extension must be coded in PERL. We have worked around this by installing a python script with the extension. When an action is executed, it will execute a python script to handle most of the work.

1.4 Preface

This guide provides the information necessary to implement the ThreatQuotient for Request Tracker integration. This document is not specifically intended as a site reference guide. It is assumed that the implementation engineer has experience installing and commissioning the ThreatQuotient Apps and integrations covered within the document, as well as the experience necessary to troubleshoot at a basic level.

1.5 Audience

This document is intended for use by the following parties:

1. ThreatQ and Security engineers.
2. ThreatQuotient Professional Services Project Team & Engineers.

1.6 Scope

This document only covers the implementation of the ThreatQuotient for Request Tracker custom connector.

Table 1: ThreatQuotient Software & App Version Information

Software/App Name	File Name	Version
ThreatQ	Version 3.6.x or greater	
ThreatQuotient for Request Tracker	Connector	2.0.0
	Extension	1.0.0

1.7 Assumptions

The following criteria is assumed to be in place and functional to allow the implementation of the ThreatQuotient for Request Tracker into the managed estate:

- All ThreatQuotient equipment is online and in service.
- Infrastructure/transmission at all sites and between sites is in place to support the network traffic.
- All required firewall ports have been opened.
- All equipment is powered from permanent power supplies.
- A clock source of sufficient accuracy is connected to the network and the network and devices are using it as the primary clock source.

2 Implementation Overview

This document offers installation instructions for the ThreatQuotient for Request Tracker integration.

2.1 Prerequisites

Throughout this implementation document, we will refer to several files and directories, some of which will be symbolic, and others may change depending on specifics of the environmental setup.

Ensure that all ThreatQ devices are set to the correct time, time zone and date, and using a clock source available to all.

To identify which time zone is closest to your present location, use the `timedatectl` command with the `list-timezones` command line option. For example, to list all available time zones in Europe, type:

Figure 1: Time Zone List Example

```
timedatectl list-timezones | grep Europe
Europe/Amsterdam
Europe/Athens
Europe/Belgrade
Europe/Berlin
```

To change the time zone to UTC, type as root:

Figure 2: Time Zone Change Example

```
timedatectl set-timezone UTC
```

2.2 Security and Privacy

For ThreatQuotient Professional Services engineers to configure the system, local network access is required to connect to the managed estate. Therefore, the implementation must occur at an office or data center location.

Passwords have not been provided in this document. Please contact your project team for this information, if required.

All engineers are reminded that all data belonging and pertaining to the business is strictly confidential and should not be disclosed to any unauthorized parties.

3 Request Tracker Installation Steps

3.1 Request Tracker Custom Connector

From the ThreatQuotient Repository

The following Steps will install ThreatQuotient for Request Tracker Custom Connector from the ThreatQuotient repository with YUM credentials.

1. Install the Custom Connector application by using the following commands.

Figure 3: Install via ThreatQ Repository

```
?> https://<USERNAME>:<PASSWORD>@extensions.threatq.com/threatq/integrations
tqRequestTracker
```

Figure 4:: Install via .whl File

```
?> pip install tqRequestTracker-2.0.0-py2-none-any.whl
```

Once the Request Tracker Extension has been installed, you must create a directory structure for all configuration, logs and files, using the `mkdir -p` command. See the example below:

Figure 5: Creating Integration Directories (Example)

```
mkdir -p /etc/tq_labs/
mkdir -p /var/log/tq_labs/
```

This part of the integration will be installed from within your Request Tracker instance.

2. SSH into your Request Tracker instance as root (or sudo yourself).
3. SCP/Transfer the **RT-Extension-ThreatQ-2.0.0.tar.gz** file into your Request Tracker instance.
4. Extract the compressed directory:

```
tar -xzvf RT-Extension-ThreatQ-2.0.0.tar.gz
```

5. Navigate to the extracted directory:

```
cd RT-Extension-ThreatQ-2.0.0
```

6. Compile a Makefile from PERL Makefile:

```
perl Makefile.PL
```

7. Compile the Makefile:

```
make
```

8. Install the extension, using Makefile:

```
make install
```

9. Initialize the database using Makefile (* skip this step if you are upgrading *):

```
make initdb
```

You will be prompted to type your password three (3) times. This is expected behavior.

10. Edit your `/opt/rt4/etc/RT_SiteConfig.pm` file with the initial ThreatQ configuration:

/opt/rt4/etc/RT_SiteConfig.pm

```
Plugin('RT::Extension::ThreatQ');

Set($ThreatQ_User, '<YOUR THREATQ USERNAME>');
Set($ThreatQ_Password, '<YOUR THREATQ PASSWORD>');
Set($ThreatQ_CID, '<YOUR THREATQ CID>');
Set($RT_User_For_ThreatQ, '<THE RT USER TO USE WITH THREATQ>');
Set($RT_Password_For_ThreatQ, '<THE RT PASSWORD TO USE WITH THREATQ>');
Set($RT_Queues_For_ThreatQ, '<QUEUES TO USE WITH EXTENSION (i.e.
Incidents,Incident Reports>');
Set($ThreatQ_Host, '<YOUR THREATQ HOST/IP>');
Set($ThreatQ_Status, '<STATUS TO USE FOR NEW INDICATORS>');
Set($ThreatQ_Use_SSL, '<ENABLE TO DISABLE SSL WITH RT ('1' or '0')>');

Set($WebDomain, '<YOUR REQUEST TRACKER HOST/DOMAIN>');
```

a. You will need to create a user to interact with ThreatQ and Request Tracker, then enter the credentials in the configuration above.
b. **$WebPath** is optional if needed.
c. **$WebDomain must be set in order for this extension to work.**

   It may already be set. Make sure you don't duplicate it.

d. **RT_Queues_For_ThreatQ:** This is the list of Queues for which you must enable the ThreatQ extension. This must be a comma-delimited list, with **no space** after each comma.
e. **ThreatQ_Use_SSL:** Setting this to '**1**' will enable SSL verification with Request Tracker. If set to '**0**', the python script will not verify the SSL certificate.

   If you are having connectivity issues or the integration does not seem to be working correctly, setting this to '**0**' may fix the issues.

11. Apply the HTTPS patch.

If your Request Tracker instance uses HTTPS, run the following command to enable it in the extension. If you do not, it may cause API access issues:

```
sed -i "s/http:\/\//https:\/\//" /opt/rt4/local/plugins/RT-Extension-
ThreatQ/bin/tq_request_tracker.py
```

12. Clear your mason cache.

```
rm -rf /opt/rt4/var/mason_data/obj
```

13. Restart your apache web server.

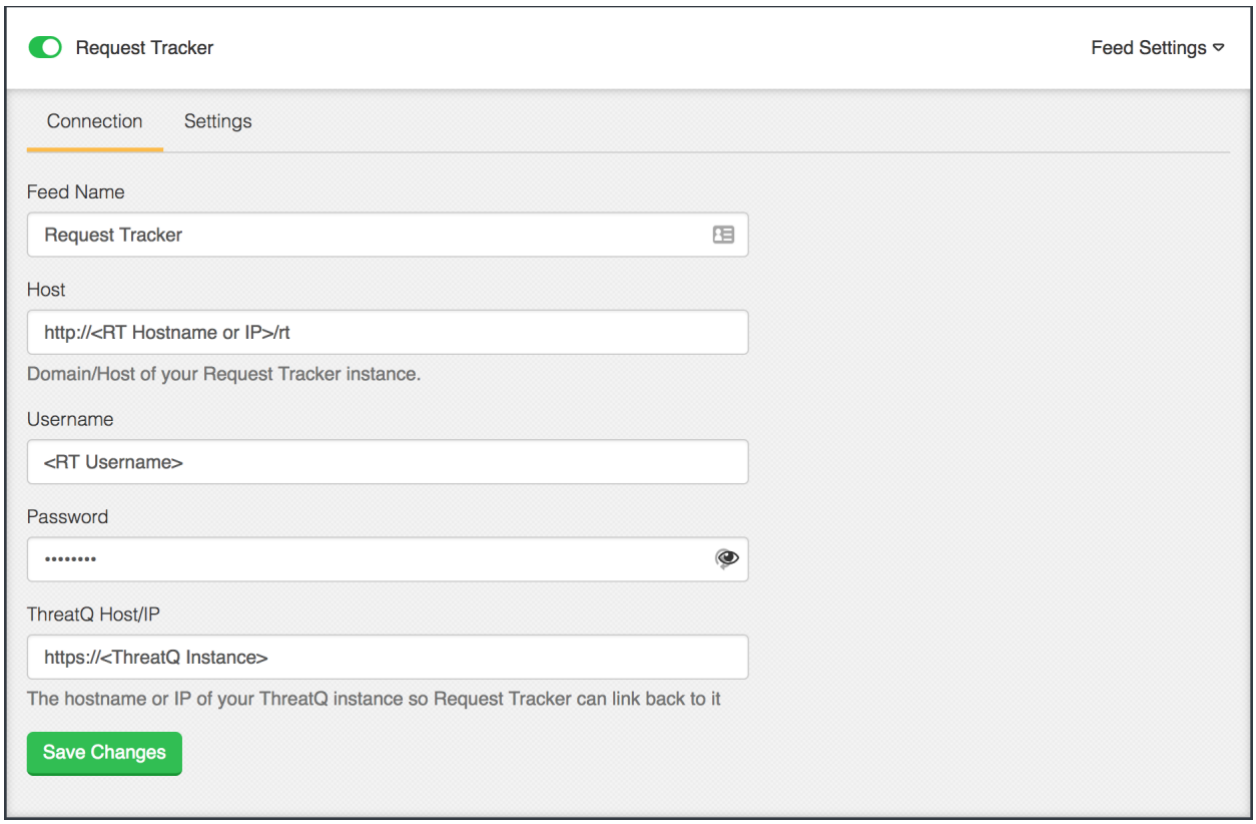
```
systemctl restart httpd.service
```

After these steps have been completed, the ThreatQ extension should now be installed. The next section will detail how to configure each of the integrations.

3.2 Request Tracker Custom Connector

After you install the custom connector, a configuration tab is added to your ThreatQ instance under **Settings > Incoming Feeds**. From here, expand the **Request Tracker** tab and follow the configuration steps outlined below.

Figure 6: ThreatQ UI Configuration



- **Host:** The IP or Hostname for your Request Tracker instance
  - If you have used a webpath, please include this
- **Username:** The Request Tracker username you are going to use with the integration
- **Password:** The password associated with the above username
- **ThreatQ Host/IP:** The Hostname or IP address of your ThreatQ instance
  - This is required so that Request Tracker can link back to ThreatQ correctly.

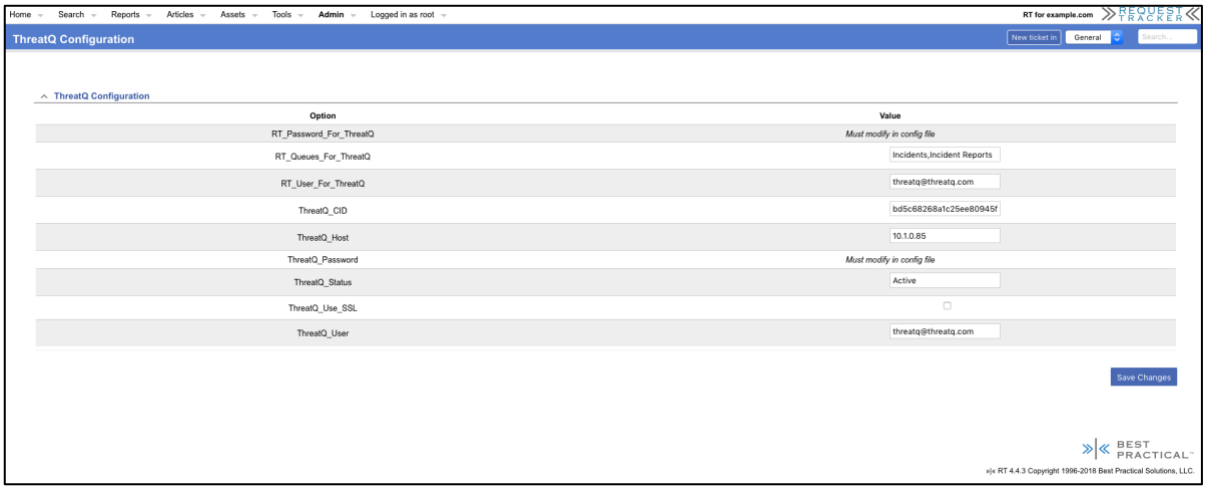
Save the configuration, the custom connector's configuration is now complete.

3.3 Request Tracker Extension

After installing the extension on the Request Tracker server, configuration of the extension is required to enable it to communicate with both ThreatQ and the Request Tracker REST API. In order to configure the extension, the user **must** be a **Request Tracker Super User** (administrator).

1. On the Request Tracker home page, navigate to **Admin > Tools > Edit ThreatQ Config**.
2. This page will show you the options that you are able to configure from the Request Tracker User Interface.

Figure 7: Request Tracker UI Configuration



a. Options

- **RT_Password_For_ThreatQ**: The password associated with the Request Tracker user you want to use with this extension
- **RT_Queues_For_ThreatQ:** The queues that you want to enable for this integration. This must be a comma-delimited list of queues, with no space after each comma
- **RT_User_For_ThreatQ:** The Request Tracker use you want to use with the extension
- **ThreatQ_CID:** The OAuth token generated by ThreatQ to use with the extension
- **ThreatQ_Host:** The hostname/IP of your ThreatQ instance
- **ThreatQ_Password:** The password associated with the ThreatQ user used with this extension
- **ThreatQ_Status:** The indicator status used when creating new indicators in ThreatQ
- **ThreatQ_Use_SSL:** Determines if SSL certificates will be verified with Request Tracker's API

  If you are having issues with the extension, setting this to 'No' may fix your issues.

There is a known bug within the Request Tracker user interface where boolean values show 3 radio buttons to choose from. Ignore the 3rd radio button and just use the first 2. The first being **Yes** and the second being **No**.

- **ThreatQ_User:** The ThreatQ username you want to use with ThreatQ.

b. The password fields are only editable through `/opt/rt4/etc/RT_SiteConfig.pm`.
c. Once the non-password config options are edited (and saved), they can be removed from your `RT_SiteConfig.pm` file since they will be stored in the ThreatQ database within your Request Tracker instance.

   This is optional.

d. Save all changes.

3. Next, you will want to edit which queues the extension's Custom Fields will apply.
   a. From the navigation menu, go to: **Admin > Custom Fields > Select**.
   b. You should see the three (3) Custom Fields that the extension installed (ThreatQ Attributes, ThreatQ Link, and ThreatQ indicators).
4. For each of the Custom Fields, click the field, and then click **Applies to** in the top right portion of the page.
   a. From the next window, you will be able to choose the Queues which the custom fields will apply.

The ThreatQ extension is now installed and configured.

4 Executing the Connector and Extension

This section will detail how to execute/invoke the Connector and Extension.

4.1 Request Tracker Custom Connector

This piece of the integration will be invoked just like any other ThreatQ custom connector integration. In production environments, it should be run on a CRON job so it runs every 'x' minutes. We recommend that it runs every 1-2 minutes to make syncing as real time as possible.

Figure 8: Executing the Integration via CLI

```
?> tqRequestTracker -ll /path/to/log/dir -c /path/to/config/dir -v3
```

When running the custom connector, make sure to set a log and configuration directory so you only get prompted for ThreatQ credentials one time.

Note: For testing CRON schedules, refer to https://crontab.guru.

4.2 Request Tracker Extension

Since this piece of the integration gets invoked by the Request Tracker system when transactions are made on a ticket, you do not need to run anything manually.

The extension will get invoked in the following circumstances:

- A transaction happens on a ticket (created, linked, edited, etc.).
- The context menu option ***ThreatQ > Sync Ticket*** gets invoked.

When the ***ThreatQ > Sync Ticket*** context menu option gets invoked from a ticket's page, it will synchronously run the extension/action and display the results to you on a separate page. This means that clicking this menu option will take some time to execute, but you will eventually be redirected to a results page.

If tickets do not appear to be syncing automatically, use the context menu option to see if any errors are displayed associated with the script.

5 Logging

This section details how to view the logs and debug the integrations if problems should occur.

5.1 Request Tracker Custom Connector

If any issues should occur with the Custom Connector not updating the Request Tracker tickets properly, viewing the logs in the directory that you set when running the integration (and in the CRON job).

5.2 Request Tracker Extension

There are two (2) ways to view the logs of the Request Tracker extension.

1. Invoke the **ThreatQ > Sync Ticket** context menu option.
2. View the Request Tracker system log to see if there are any errors there.
    a. If there are errors, please report them to ThreatQ's support team.
    b. If there are no errors, and you see that the ThreatQ extension is being invoked.

    Below is an example of the Request Tracker logs when ThreatQ is invoked:

Figure 8: Example Log

```
[3651] [Wed May 23 19:52:52 2018] [debug]: ThreatQ: Sync Event Action Started
(/opt/rt4/local/plugins/RT-Extension-ThreatQ/lib/RT/Action/SyncThreatQEvent.pm:18)
[3651] [Wed May 23 19:52:52 2018] [debug]: ThreatQ: Transaction Type - Create
(/opt/rt4/local/plugins/RT-Extension-ThreatQ/lib/RT/Action/SyncThreatQEvent.pm:123)
[3651] [Wed May 23 19:52:52 2018] [debug]: ThreatQ: Executing - python
/opt/rt4/local/plugins/RT-Extension-ThreatQ/bin/action.py --tq-host <TQ HOST> --tq-
user <TQ USER> --tq-pass <TQ PASS> --tq-cid <TQ CID> --tq-status <TQ STATUS> --rt-
host <RT HOST> --rt-user <RT USER> --rt-pass <RT PASS> --ticket <TICKET ID> --sync-
wait & (/opt/rt4/local/plugins/RT-Extension-
ThreatQ/lib/RT/Action/SyncThreatQEvent.pm:138)
[3651] [Wed May 23 19:52:52 2018] [debug]: ThreatQ: Finished executing
(/opt/rt4/local/plugins/RT-Extension-ThreatQ/lib/RT/Action/SyncThreatQEvent.pm:141)
```

In line 3, the exact command being executed is displayed.

Copy the line from 'python' until just before the '&' character. Then paste and execute it manually. View the generated output and see if there are any errors.

5.3 Reporting

If there are errors, copy the errors logs and open a support ticket, including the log output and send them to support@threatq.com.

If there are no visible errors and the updating is not working as intended, please send an email to support@threatq.com stating what is not happening with the integration that should be happening. ensure to specify if this issue is happening when using the custom connector or the extension.

Figure 9: Viewing logs via CLI

```
?> tail -n 200 /path/to/logs/dir/Request\ Tracker.log
```

5.4 CRON

To run this script on a reoccurring basis, use CRON or some other system schedule. The argument in the cron script **must** specify the config and log locations.

5.4.1 Setting Up the CRONJOB

1. Login via a CLI terminal session to you ThreatQ host
2. Input the commands below

   Figure 10: Command Line Crontab Command

   ```
   $> crontab -e
   ```

   This will enable the editing of the crontab, using vi.

It is recommended that you run this on a CRON job every **1 min**. The integration only takes a couple of seconds per ticket update, so under average ticket load, **1 min** is a good frequency. Adjust accordingly to your ticket load.

3. Input the commands below, this example shows every **1 minute**.

   Figure 11: Command Line Crontab ServiceNow Command

   ```
   */1 * * * * tqRequestTracker -c /path/to/config/ -ll path/to/logs/ -v 3
   ```

To run this script on a reoccurring basis use CRON or some other on system schedule. CRON is displayed here.

For further reference, see the ThreatQ Help Center.

Appendix A: Supplementary Information

Debug Options

If crashes happen after you install the extension ,the following will help with debugging:

- Enable logging, then restart your apache server, reload Request Tracker, view logs
  - How to enable logging: https://rt-wiki.bestpractical.com/wiki/LogsConfig.
- Check to make sure that there are not any special characters in the *RT_SiteConfig.pm* configuration. Often times, the single quote may be converted to a special single quote which will break the configuration file.
- If all else fails, remove the `Plugin('RT::Extension::ThreatQ')` line from the *RT_SiteConfig.pm* file to disable the extension.
  - If you are at this step, please context ThreatQ support (support@threatq.com) and please ensure to include the log containing the error.

Uninstalling the Connector

```
sudo pip uninstall tqRequestTracker
```

Driver command line options

The ServiceNow Driver has several command line arguments that will help you and your customers execute this. They are listed below. You can see these by executing `/usr/bin/tqRequestTracker--help`.

```
usage: tqConnector [-h] [-ll LOGLOCATION][-c CONFIG] [-v VERBOSITY]

tqRequestTracker
```

optional arguments:

```
  -h, --help
```

Shows the help message and exit

```
  -ll LOGLOCATION, --loglocation LOGLOCATION
```

This sets the logging location for this connector. The location should exist and be writable by the current user. A special value of 'stdout' means to log to the console (this happens by default).

```
  -c CONFIG, --config CONFIG
```

This is the location of the configuration file for the connector. This location must have read and write permissions for the current user. If no config file is given, the current directory will be used. This file is also where some information from each run of the connector may be put (e.g. last run time, private OAuth, etc).

```
  -v {1,2,3}, --verbosity {1,2,3}
```

This is the logging verbosity level. The Default is 1 (Warning).

Supported Custom Fields

Below is a list of supported custom fields:

- Description
- Resolution
- Classification
- IP (multiple values)
    - Not including IP ranges or CIDR Blocks
- Domains (multiple values)
- FileNames (multiple values)
- HASH (multiple values)
- Sender (multiple value)
- Systems Re-Imaged (multiple values)
- External Ticket Number (single value)
- Tags (multiple values)
- Threat Campaign (single value)
- URL (multiple value)
- Threat Actors (multiple values)
- Threat TTP (multiple values)
- ThreatQ Link (single value)
- ThreatQ Attributes (multiple values)
    - Shows TQ attributes in RT
- ThreatQ Indicators (multiple values)
    - Shows TQ indicators, their type, and a link to TQ in RT

Trademarks and Disclaimers

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR THREATQUOTIENT REPRESENTATIVE FOR A COPY.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THIRD PARTY SUPPLIERS ARE PROVIDED “AS IS” WITH ALL FAULTS. THREATQUOTIENT AND THIRD-PARTY SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL THREATQUOTIENT OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF THREATQUOTIENT OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ThreatQuotient and the ThreatQuotient Logo are trademarks of ThreatQuotient, Inc. and/or its affiliates in the U.S. and other countries.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.