

# IBM Resilient



QRADAR FUNCTION GUIDE v2.0

## Incident Response Platform

Licensed Materials – Property of IBM

© Copyright IBM Corp. 2010, 2019. All Rights Reserved.

US Government Users Restricted Rights: Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

## **Resilient Incident Response Platform QRadar Function Guide**

<b>Version</b>	<b>Publication</b>	<b>Notes</b>
2.0.5	April 2020	Additional configuration notes.
2.0	March 2019	Supports the V2.0 release.
1.0	July 2018	Initial release.

## Table of Contents

<b>1. Overview .....</b>	<b>5</b>
1.1. What's New in V2.0.....	5
<b>2. Check Prerequisites .....</b>	<b>5</b>
<b>3. Install the Integration.....</b>	<b>5</b>
3.1. Install the Python components.....	5
3.2. Configure the Python components.....	6
3.3. Deploy customizations to the Resilient platform .....	6
3.4. Run the integration framework.....	7
3.5. Configure Resilient Circuits for restart .....	7
<b>4. Function Descriptions .....</b>	<b>7</b>
4.1. QRadar Search .....	8
4.2. QRadar Add Reference Set Item .....	10
4.3. QRadar Delete Reference Set Item .....	14
4.4. QRadar Find Reference Set Item.....	14
4.5. QRadar Find Reference Sets.....	15
<b>5. Troubleshooting.....</b>	<b>15</b>
<b>6. Support .....</b>	<b>16</b>



# 1. Overview

Resilient Functions simplify development of integrations by wrapping each activity into an individual workflow component. These components can be easily installed, then used and combined in Resilient workflows. The Resilient platform sends data to the function component that performs an activity then returns the results to the workflow. The results can be acted upon by scripts, rules, and workflow decision points to dynamically orchestrate the security incident response activities.

This guide describes the QRadar Function.

The QRadar integration with the Resilient platform package provides the following:

- Search function to perform a QRadar Ariel query
- Search function to query an item in a QRadar reference set
- Search function to find all the reference sets that contain an item
- Add function to insert a new item in a QRadar reference set
- Delete function to remove an item from a QRadar reference set

With the above functions, this package includes example workflows that demonstrate how to call the functions, rules that start the example workflows, and custom data tables updated by the example workflows.

## 1.1. What's New in V2.0

The following components were added to the function package:

- Function: "QRadar Find Reference Sets"
- Workflow: "Example of finding all QRadar reference sets for artifact"
- Rule: "Find All QRadar Reference Sets"
- Data table: "QRadar Reference Sets"

# 2. Check Prerequisites

Before installing, verify that your environment meets the following prerequisites:

- Resilient platform is version 30 or later.
- You have access to a Resilient integration server. An *integration server* is the system that you use to deploy integration packages to the Resilient platform. See the [Resilient Integration Server Guide \(PDF\)](#) for more information.
- QRadar version 7.2.8 or later.

# 3. Install the Integration

The integration package contains Python components that are called by the Resilient platform. These components run in the Resilient Circuits integration framework. The package also includes Resilient customizations that will be imported into the platform later.

You perform these installation procedures at the Resilient integration server.

## 3.1. Install the Python components

Complete the following steps to install the Python components:

1. Ensure that the environment is up-to-date, as follows:

```
sudo pip install --upgrade pip sudo pip
install --upgrade setuptools sudo pip install
--upgrade resilient-circuits
```

2. To install the package, you must first unzip it then install the package as follows:

```
sudo pip install fn_qradar_integration-<version>.tar.gz
```

## 3.2. Configure the Python components

The Resilient Circuits components run as an unprivileged user, typically named integration. If you do not already have an integration user configured on your appliance, create it now.

Complete the following steps to configure and run the integration:

1. Using sudo, switch to the integration user, as follows:

```
sudo su - integration
```

2. Use one of the following commands to create or update the resilient-circuits configuration file. Use `-c` for new environments or `-u` for existing environments.

```
resilient-circuits config -c
```

or

```
resilient-circuits config -u
```

3. Edit the resilient-circuits configuration file, as follows:

- a. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.

- b. In the [fn\_qradar\_integration] section, edit the settings as follows:

```
host=<qradar url>
port=<8089 or the customized port>
username=<qradar access user>
qradarpassword=<qradar access password, key-ring protection recommended>
verify_cert=[true|false]
qradartoken=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
```

Use 'false' for self-signed certificates.

You have two choices for authentication with QRadar – username and a password, or a token. QRadar token is an authentication token created in QRadar's Authorized Services manager.

Note that if both username/password and the token are given, username and password authentication will take precedence.

## 3.3. Deploy customizations to the Resilient platform

The package contains five example functions, six example workflows that access the functions, and six rules that trigger the workflows, along with custom fields and two data tables for input and output parameters.

1. Use the following command to deploy these customizations to the Resilient platform:

```
resilient-circuits customize
```

2. Respond to the prompts to deploy functions, message destinations, workflows and rules.

### 3.4. Run the integration framework

To test the integration package before running it in a production environment, you must run the integration manually with the following command:

```
resilient-circuits run
```

The resilient-circuits command starts, loads its components, and continues to run until interrupted. If it stops immediately with an error message, check your configuration values and retry.

### 3.5. Configure Resilient Circuits for restart

For normal operation, Resilient Circuits must run continuously. The recommend way to do this is to configure it to automatically run at startup. On a Red Hat appliance, this is done using a systemd unit file such as the one below. You may need to change the paths to your working directory and app.config.

1. The unit file must be named `resilient_circuits.service` To create the file, enter the following command:

```
sudo vi /etc/systemd/system/resilient_circuits.service
```

2. Add the following contents to the file and change as necessary:

```
[Unit]
Description=Resilient-Circuits Service
After=resilient.service
Requires=resilient.service

[Service]
Type=simple
User=integration
WorkingDirectory=/home/integration
ExecStart=/usr/local/bin/resilient-circuits run
Restart=always
TimeoutSec=10
Environment=APP_CONFIG_FILE=/home/integration/.resilient/app.config
Environment=APP_LOCK_FILE=/home/integration/.resilient/resilient_circuits.lock

[Install]
WantedBy=multi-user.target
```

3. Ensure that the service unit file is correctly permissioned, as follows:

```
sudo chmod 664 /etc/systemd/system/resilient_circuits.service
```

4. Use the systemctl command to manually start, stop, restart and return status on the service:

```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

You can view log files for systemd and the resilient-circuits service using the journalctl command, as follows:

```
sudo journalctl -u resilient_circuits --since "2 hours ago"
```

## 4. Function Descriptions

Resilient platform that the following QRadar specific functions, workflows, rules, and message destination are available by clicking their respective tabs. The data table and custom fields need to be added to your custom layout as per your design.

## 4.1. QRadar Search

This sample function performs an Ariel query to fetch data from the QRadar server.

The screenshot shows the 'Customization Settings' page for the 'qradar\_search' function in the Resilient platform. The page has a top navigation bar with 'Dashboards', 'List Incidents', 'New Incident', 'My Tasks', and 'Simulations'. The 'Functions' tab is selected, and the breadcrumb path is 'Functions / qradar\_search'. The function details are as follows:

- Name \***: QRadar Search
- API Name \***: qradar\_search
- Message Destination \***: fn\_qradar\_integration
- Description**: Search QRadar for events
- Creator**: masterfirst
- Last Modified**: 06/19/2018 15:42
- Last Modified By**: masterfirst
- Associated Workflows**: Example of searching QRadar

Below the details are two sections for input fields:

- Inputs**: A list of input fields with a search icon:
  - qradar\_query
  - qradar\_query\_param1
  - qradar\_query\_param2
  - qradar\_query\_param3
  - qradar\_query\_param4
  - qradar\_query\_param5
  - qradar\_query\_range\_start
  - qradar\_query\_range\_end
- Input Fields**: A list of input fields with a search icon and an 'Add Field' button:
  - qradar\_query
  - qradar\_query\_param1
  - qradar\_query\_param2
  - qradar\_query\_param3
  - qradar\_query\_param4
  - qradar\_query\_param5
  - qradar\_query\_range\_end
  - qradar\_query\_range\_start
  - qradar\_reference\_set\_item\_value

This function takes the following parameters:

- **qradar\_query**: Query to perform. It contains demo template queries you can select from within the calling workflow. The demo queries contain parameters that are replaced by the `qradar_query_param[n]` below. For example, one template query is: *SELECT %param1% FROM events WHERE INOFFENSE(%param2%) LAST %param3% MINUTES*. Users can set values for `qradar_query_param1`, `qradar_query_param2`, and `qradar_query_param3` in the workflow.
- **qradar\_query\_param[n]**: Optional. Parameters used in the query.
- **qradar\_query\_range\_start**: Optional. An integer specifying QRadar return start range.
- **qradar\_query\_range\_end**: Optional. An integer specifying QRadar return end range. The workflow (object type = Incident), "Example of searching QRadar events using offense id", sets the function's input fields and runs the function. The Input tab maps the function's input fields.



Input Parameter	Value
qradar_query	SELECT %param1% FROM events WHERE INOFFENSE(%param2%) LAST %param3% MINUTES
qradar_query_param1	DATEFORMAT(starttime, 'YYYY-MM-dd HH:mm') as StartTime, CATEGORYNAME(category), LOGSOURCEID(logsourceid), PROTOCOLNAME(protocolid), RULENAME(ruleid)
qradar_query_param2	
qradar_query_param3	100
qradar_query_param4	
qradar_query_param5	
qradar_query_range_start	1
qradar_query_range_end	5

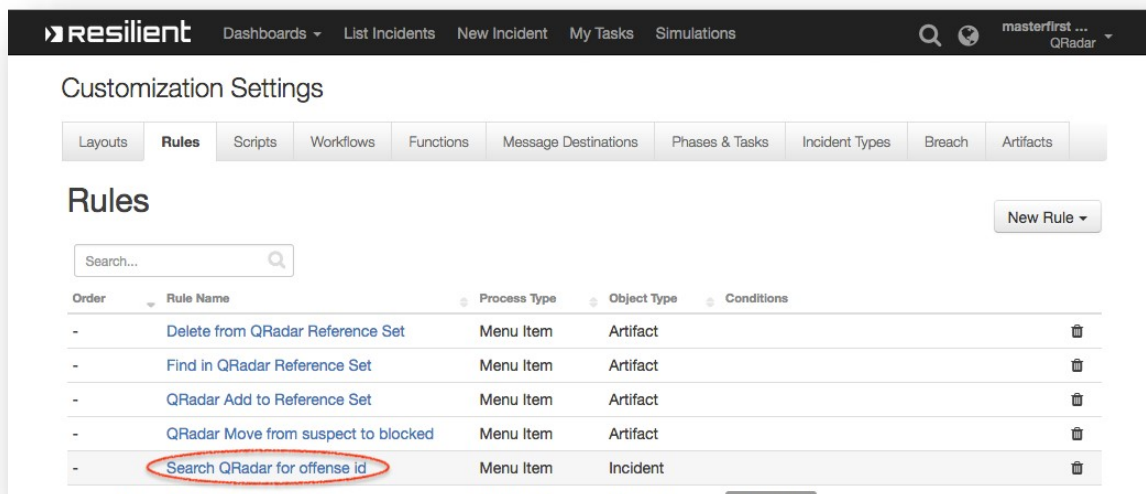
In the Pre-Process Script, the input field, “qradar\_query\_param2” is set to the incident’s custom field, “qradar\_id”.

Input	Pre-Process Script	Output	Post-Process Script
Language: Python Theme: light Mode: Default Tab Size: 2 - Font + Font			
1 <code>inputs.qradar_query_param2 = incident.properties.qradar_id</code>			

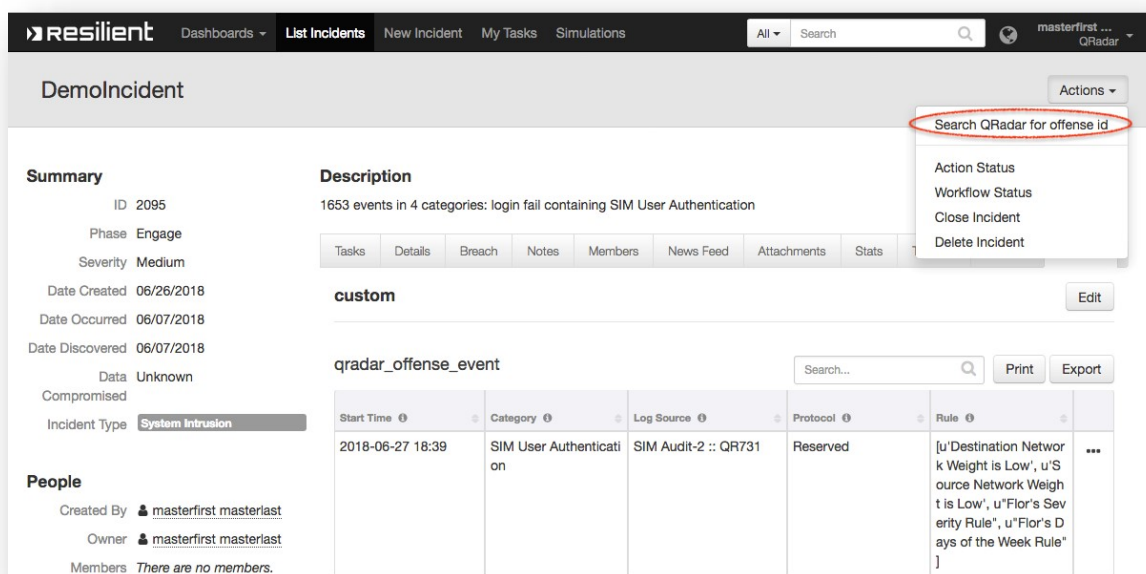
#### NOTES:

- You should create the custom field, qradar\_id, and add it to your layout for accessibility, along with the custom data tables.
- If a reduced number of QRadar events are returned when the search is run in Resilient, allocate more time to the Resilient input parameter: qradar\_query. This will allow Resilient to return the same number of QRadar events.

The example rule, “Search QRadar for offense id”, is a menu item rule for an Incident. The user can select this menu item to initiate the workflow.



This workflow can work together with the Resilient QRadar App. An incident escalated from a QRadar offense stores the offense ID. This workflow can make use of that offense ID to perform an Ariel query and update a custom data table, which is also included in the package.



## 4.2. QRadar Add Reference Set Item

This function adds a new item to an existing QRadar reference set. It uses two input parameters: `qradar_reference_set_name` is the name of an existing reference set in QRadar, and `qradar_reference_set_item_value` is the value to be added. The input is populated by the workflow, “Example of adding an item to QRadar reference set”.

The screenshot shows the 'Customization Settings' page for the 'qradar\_add\_reference\_set\_item' function in the Resilient Incident Response Platform. The page has a dark header with the 'Resilient' logo and navigation links: Dashboards, List Incidents, New Incident, My Tasks, and Simulations. A search bar and a user profile dropdown (masterfirst ... QRadar) are on the right. Below the header, a tabbed interface shows 'Functions' as the active tab. The function name 'qradar\_add\_reference\_set\_item' is displayed in the breadcrumb. On the right, there are buttons for 'Cancel', 'Save & Close', and 'Save'. The main form contains the following fields:

- Name \***: QRadar Add Reference Set Item
- API Name \***: qradar\_add\_reference\_set\_item
- Message Destination \***: fn\_qradar\_integration (dropdown)
- Description**: Add an item to a given QRadar reference set

On the right side of the form, there is a section for 'Associated Workflows' with the following details:

- Creator**: masterfirst masterlist
- Last Modified**: 06/27/2018 18:26
- Last Modified By**: masterfirst masterlist
- Associated Workflows**: Example of adding an item to QRadar, Example of moving QRadar item from

At the bottom, there are two sections:

- Inputs**: A list of input fields with 'x' icons for removal: qradar\_reference\_set\_name, qradar\_reference\_set\_item\_value.
- Input Fields**: A search bar and a list of available input fields with edit icons: qradar\_query\_param1, qradar\_query\_param2, qradar\_query\_param3, qradar\_query\_param4, qradar\_query\_param5, qradar\_query\_range\_end, qradar\_query\_range\_start, qradar\_reference\_set\_item\_value, qradar\_reference\_set\_name.

The workflow, “Example of adding an item to QRadar reference set”, sets the function’s input fields: “qradar\_reference\_set\_name” is mapped to “Sample Blocked IPs”, and qradar\_reference\_set\_item\_value is mapped to the artifact value, and then runs the function. The workflow is initiated by the rule, “QRadar Add to Reference Set”.

The screenshot shows the 'Customization Settings' page in the Resilient platform. The 'Workflows' tab is selected, and the specific workflow being edited is 'Example of adding an item to QRadar reference set'. The workflow is a menu item rule for an artifact. The configuration details are as follows:

- Name:** Example of adding an item to QRadar reference set
- API Name:** qradar\_add\_reference\_set\_item
- Description:** Add an IP address artifact to the QRadar reference set, "Sample Blocked IPs".
- Object Type:** Artifact
- Creator:** masterfirst masterlast
- Last Modified:** 06/27/2018 18:26
- Last Modified By:** masterfirst masterlast
- Associated Rules:** QRadar Add to Reference Set

The workflow diagram shows a start node leading to a 'QRadar Add Reference Set Item' function block, which then connects to an end node. Below the diagram, the 'Input' tab is active, showing two input parameters:

Input Parameter	Value
qradar_reference_set_name	Sample Blocked IPs
qradar_reference_set_item_value	

The example rule, "QRadar Add to Reference Set", is a menu item rule for an artifact.

The screenshot shows the 'Customization Settings' page in the Resilient platform. The 'Rules' tab is selected, and the specific rule being configured is 'QRadar Add to Reference Set'. The 'Display Name' is 'QRadar Add to Reference Set' and the 'Object Type' is 'Artifact'. Below this, there are sections for 'Activities', 'Ordered', 'Workflows', and 'Destinations'. The 'Ordered' section states that activities will be invoked in the order specified. The 'Workflows' section states that workflow activities are started after all ordered activities complete. The 'Destinations' section states that transaction data is posted to message destinations after all ordered activities complete and all workflows have been started. At the bottom, there is a copyright notice: '© Copyright IBM Corporation 2018'.

The user can select this action in the menu to initiate the workflow.

The screenshot shows the 'DemolIncident' page in the Resilient platform. The 'Artifacts' section is active, displaying a table of artifacts. A context menu is open over the table, showing several actions: 'Delete from QRadar Reference Set', 'Find in QRadar Reference Set', 'QRadar Add to Reference Set' (which is circled in red), and 'QRadar Move from suspect to blocked'. The table has columns for 'Type', 'Value', 'Created', 'Relate?', and 'Actions'. The first row shows an artifact of type 'IP Address: Source' with value '9.108.150.53' and creation date '06/26/2018'. The 'Relate?' column shows 'As specified in artifact type settings'.

### 4.3. QRadar Delete Reference Set Item

This function deletes an item from an existing QRadar reference set. It has two input fields: “qradar\_reference\_set\_name” and “qradar\_reference\_set\_item\_value”. The function is called by the workflow, “Example of deleting QRadar reference set item”. The workflow, “Example of deleting QRadar reference set item”, sets the function’s input fields:

“qradar\_reference\_set\_name” is mapped to “Sample Suspect IPs”, and “qradar\_reference\_set\_item\_value” is mapped to the artifact value, and then runs the function. The workflow is initiated by the rule, “Delete from QRadar Reference Set”.

The rule, “Delete from QRadar Reference Set” is a menu item rule for artifacts. The user can select this menu item to initiate the workflow.

### 4.4. QRadar Find Reference Set Item

This function looks for an item in an existing QRadar reference set. It has two input fields: “qradar\_reference\_set\_name” and “qradar\_reference\_set\_item\_value”. The function is called by the workflow, “Example of finding an item from a QRadar reference set”.

The workflow, “Example of finding an item from a QRadar reference set”, sets the function’s input fields: “qradar\_reference\_set\_name” is mapped to “Sample Blocked IPs”, and “qradar\_reference\_set\_item\_value” is mapped to the artifact value. After running the function, the workflow adds a note to the corresponding incident.

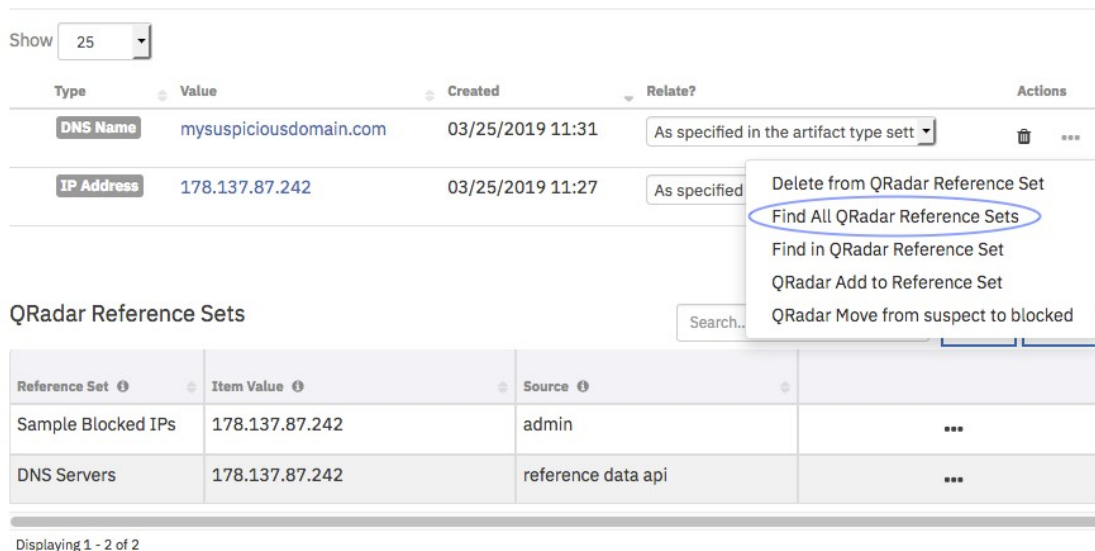
The rule, “Find in QRadar Reference Set,” is a menu item rule for artifacts. The user can select this action from the menu to initiate the workflow.

## 4.5. QRadar Find Reference Sets

This function looks for all QRadar reference sets that contain a given item. It has one input field, “qradar\_reference\_set\_item\_value.” The function is called by the workflow, “Example of finding all QRadar reference sets for artifact.”

The workflow, “Example of finding all QRadar reference sets for artifact”, sets the input field, “qradar\_reference\_set\_item\_value” to the artifact value. After running the function, the workflow populates the “QRadar Reference Sets” data table with the returned reference sets.

The rule, “Find All QRadar Reference Sets,” is a menu item rule for the artifact. The user can click this menu item for a selected artifact, and the example workflow commences.



The screenshot shows the QRadar interface. At the top, there is a 'Show' dropdown set to 25. Below this is a table with columns: Type, Value, Created, Relate?, and Actions. The table contains two rows: one for 'DNS Name' with value 'mysuspiciousdomain.com' and one for 'IP Address' with value '178.137.87.242'. The 'Relate?' column for the IP Address row is set to 'As specified'. The 'Actions' column for the IP Address row has a dropdown menu open, showing options: 'Delete from QRadar Reference Set', 'Find All QRadar Reference Sets' (highlighted with a red circle), 'Find in QRadar Reference Set', 'QRadar Add to Reference Set', and 'QRadar Move from suspect to blocked'. Below the table is a section titled 'QRadar Reference Sets' with a search bar. It contains a table with columns: Reference Set, Item Value, Source, and an ellipsis. The table has two rows: 'Sample Blocked IPs' with item value '178.137.87.242' and source 'admin', and 'DNS Servers' with item value '178.137.87.242' and source 'reference data api'. At the bottom, it says 'Displaying 1 - 2 of 2'.

## 5. Troubleshooting

There are several ways to verify the successful operation of a function.

- Resilient Action Status

When viewing an incident, use the Actions menu to view Action Status. By default, pending and errors are displayed. Modify the filter for actions to also show Completed actions. Clicking on an action displays additional information on the progress made or what error occurred.

- Resilient Scripting Log

A separate log file is available to review scripting errors. This is useful when issues occur in the pre-processing or post-processing scripts. The default location for this log file is:

```
/var/log/resilient-scripting/resilient-scripting.log.
```

- Resilient Logs

By default, Resilient logs are retained at `/usr/share/co3/logs`. The `client.log` may contain additional information regarding the execution of functions.

- Resilient-Circuits

The log is controlled in the `.resilient/app.config` file under the section `[resilient]` and the property `logdir`. The default file name is `app.log`. Each function will create progress information. Failures will show up as errors and may contain python trace statements.

## 6. Support

For additional support, contact [support@resilientsystems.com](mailto:support@resilientsystems.com).

Including relevant information from the log files will help us resolve your issue.