



# Incident Response Platform Integrations

# RSA NetWitness Functions V1.0.0

Release Date: March 2019

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 RSA NetWitness Functions.

Overview

The RSA NetWitness functions contain the ability to query for meta data and return back PCAP and log files for specific times and sessions.

Installation

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

* Resilient platform is version 30 or later.
* You have a Resilient account to use for the integrations. This can be any account that has the permission to view and modify administrator and customization settings, and read and update incidents. You need to know the account username and password.
* 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)](https://github.com/ibmresilient/resilient-reference/blob/master/developer_guides/Integration%20Server%20Guide.pdf) for more information.

Install the Python components

The functions package contains Python components that are called by the Resilient platform to execute the functions during your workflows. These components run in the Resilient Circuits integration framework.

The package also includes Resilient customizations that will be imported into the platform later.

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

1. Run the following command to install the package:

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

sudo pip install --upgrade fn\_rsa\_netwitness-<version>.tar.gz

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

1. 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

1. Edit the resilient-circuits configuration file, as follows:
   1. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.
   2. In the [fn\_rsa\_netwitness] section, edit the settings as follows:

nw\_packet\_server\_url=<http://test.nw\_packet\_server.com:50104>

nw\_packet\_server\_user=<nw\_packet\_server\_username>

nw\_packet\_server\_password=<nw\_packet\_server\_password>

nw\_packet\_server\_verify=[true|false]

nw\_log\_server\_url=<http://test.nw\_log\_server.com:50102>

nw\_log\_server\_user=<nw\_log\_server\_username>

nw\_log\_server\_password=<nw\_log\_server\_password>

nw\_log\_server\_verify=[true|false]

Add Passwords to your keystore (optional)

If the function contains passwords or other authentication values, the Resilient package includes a utility to add all of the keystore-based values from your app.config file to your system's compatible keystore system. Once you have created the keys in your app.config file, run res-keyring and you are prompted to create the secure values to store.

res-keyring

Configuration file: /Users/kexample/.resilient/app.config

Secrets are stored with 'keyring.backends.OS\_X'

[resilient] password: <not set>

Enter new value (or <ENTER> to leave unchanged):

Deploy customizations to the Resilient platform

The package contains the following function definitions that you can use in workflows, and includes example workflows and rules that show how to use these functions. Below lists everything included. *Note: The example\_netwitness\_retrieve\_log\_file workflow relies on the function Utilities: String to Attachment in the fn\_utilities integration. Import that function for this customization to succeed successfully.*

# Action fields:

# netwitness\_end\_time

# netwitness\_query

# netwitness\_start\_time

# Function inputs:

# incident\_id

# nw\_data\_format

# nw\_end\_time

# nw\_event\_session\_ids

# nw\_meta\_id1

# nw\_meta\_id2

# nw\_query

# nw\_results\_size

# nw\_session\_id1

# nw\_session\_id2

# nw\_start\_time

# Message Destinations:

# rsa\_netwitness\_message\_destination

# Functions:

# netwitness\_get\_meta\_id\_ranges

# netwitness\_getmeta\_values

# netwitness\_query

# netwitness\_retrieve\_log\_data

# netwitness\_retrieve\_pcap\_data

# Workflows:

# example\_netwitness\_get\_meta\_values

# example\_netwitness\_retrieve\_log\_file

# example\_netwitness\_retrieve\_pcap\_file

# example\_netwitness\_retrieve\_pcap\_file\_time

# Rules:

# (Example) NetWitness Get Meta Values

# (Example) NetWitness Retrieve Log File

# (Example) NetWitness Retrieve PCAP File

# (Example) NetWitness Retrieve PCAP File (Time)

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

resilient-circuits customize

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

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.

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

1. Add the following contents to the file and change as necessary: *<replace the contents below with your own>*

[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

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

sudo chmod 664 /etc/systemd/system/resilient\_circuits.service

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

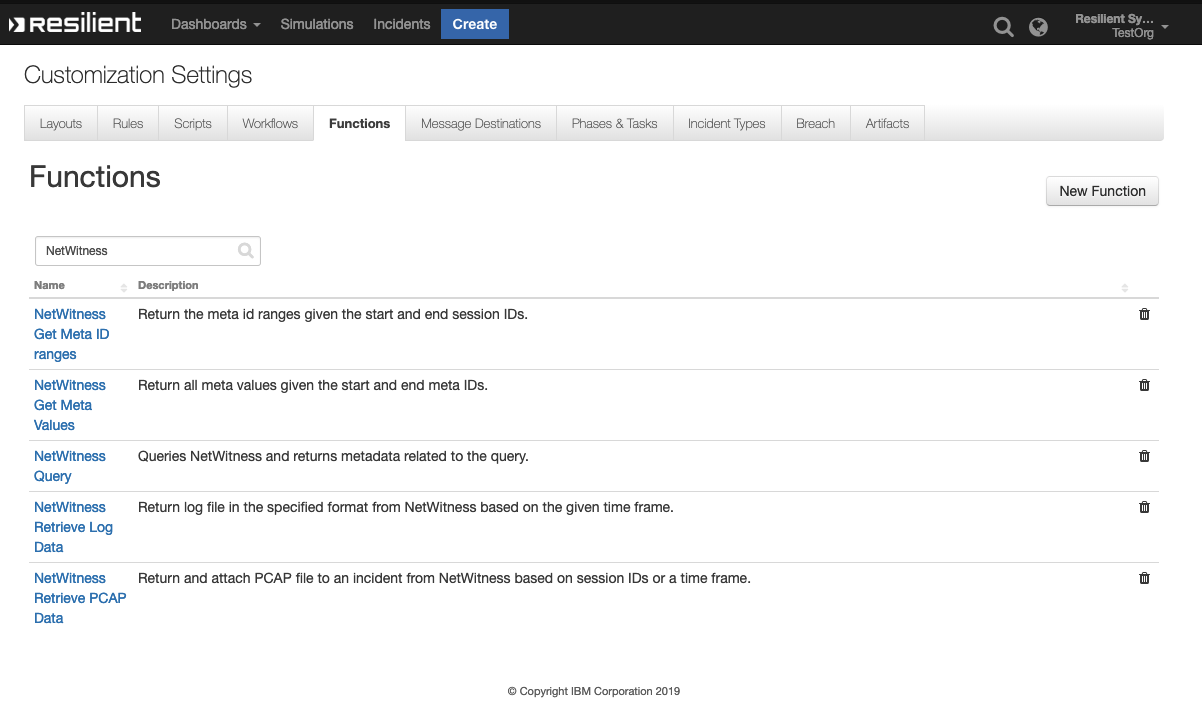
sudo systemctl resilient\_circuits [start|stop|restart|status]

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"

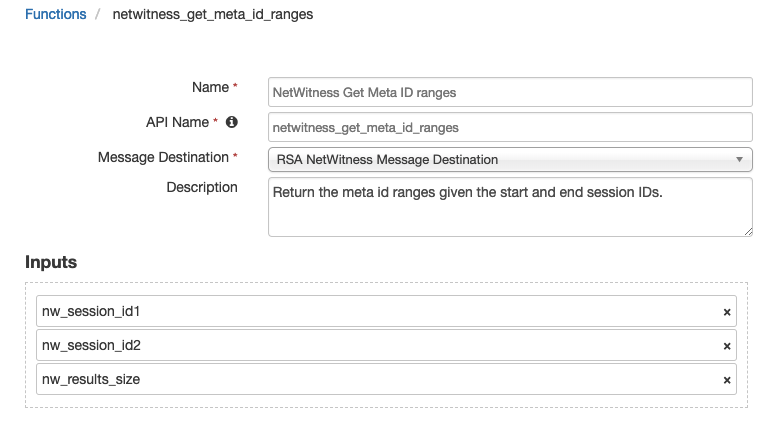
Function Descriptions

Once the function package deploys the function(s), you can view them in the Resilient platform Functions tab, as shown below.

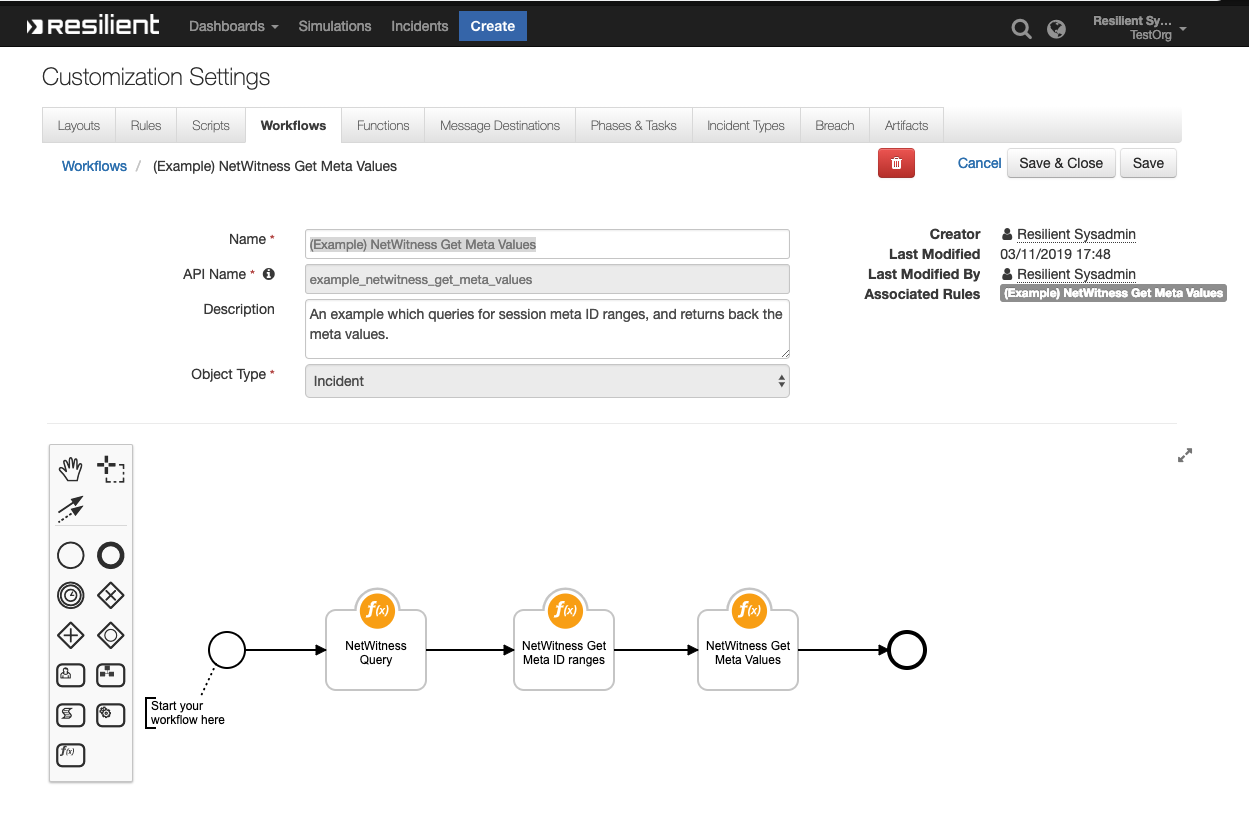


fn\_rsa\_netwitness: NetWitness Get Meta ID Ranges

The NetWitness Get Meta ID Ranges function returns back the first and last meta id fields given the session ids. You can also specify the size of the results returned by setting nw\_results\_size.



This works well when paired with the NetWitness Query and NetWitness Get Meta Values functions where the Query function can provide the session id range, this function will use that output to get the meta ID range and the Get Meta Values function will its output to get all the meta values. (Example) NetWitness Get Meta Values it the example workflow which accomplishes this and is shown below:



fn\_rsa\_netwitness: NetWitness Get Meta Values

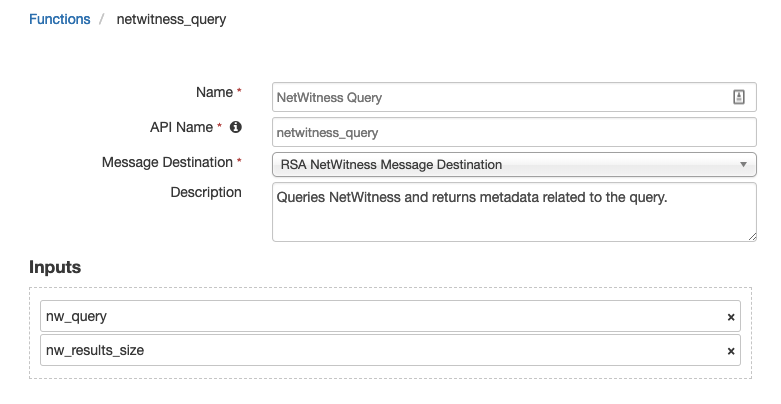
The NetWitness Get Meta Values function returns back the meta values between the first and last meta id fields. You can also specify the size of the results returned by setting nw\_results\_size.



This is included in the workflow (Example) NetWitness Get Meta Values, see above NetWitness Get Meta ID Ranges for more information.

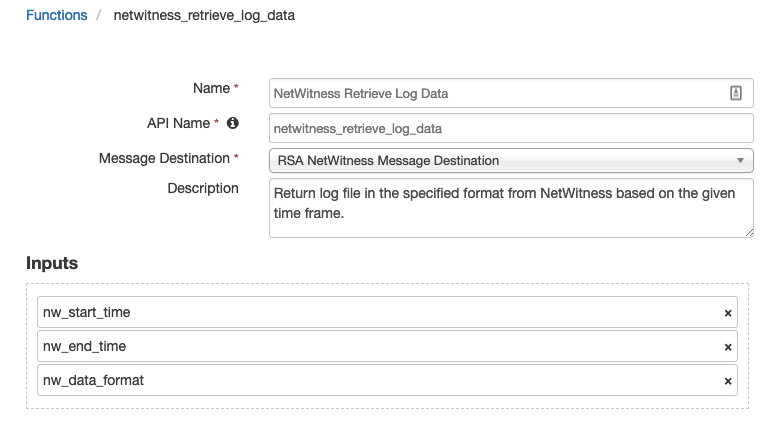
fn\_rsa\_netwitness: NetWitness Query

The NetWitness Query function takes a string query as an input and returns the query response as json. Setting the size of the results to be returned can also be set by using nw\_results\_size. This function is used in the following example workflows: (Example) NetWitness Get Meta Values and (Example) NetWitness Retrieve PCAP File.

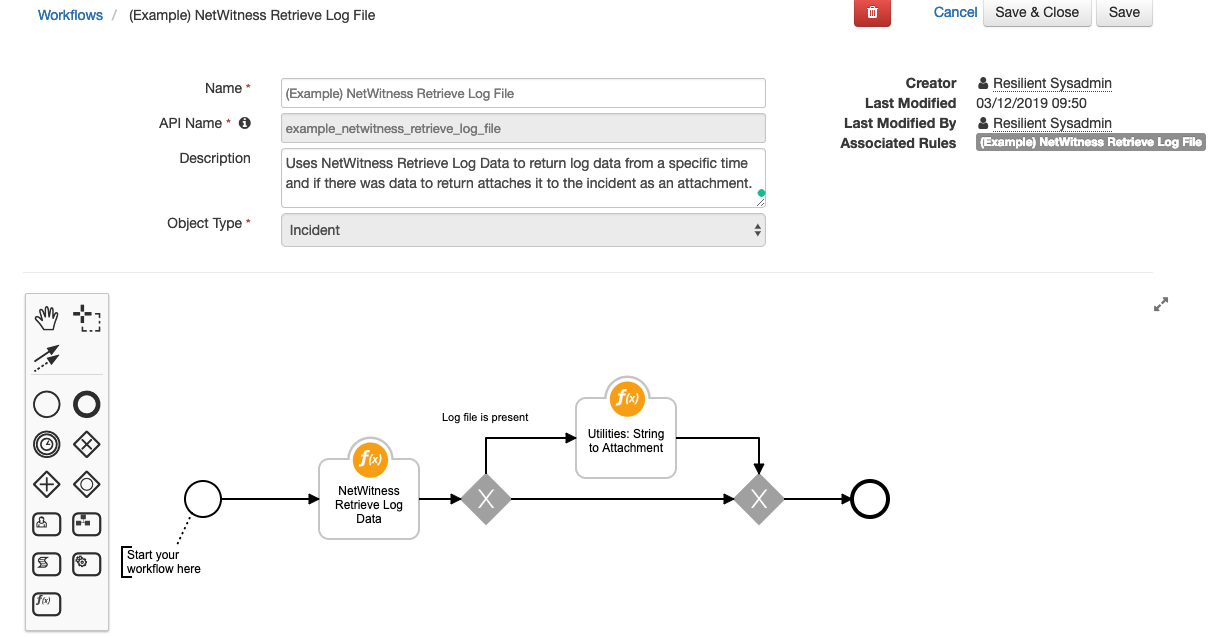


fn\_rsa\_netwitness: NetWitness Retrieve Log Data

The NetWitness Retrieve Log Data function takes a start and end time and returns back the log data in the format specified (plain text, csv, xml, json).

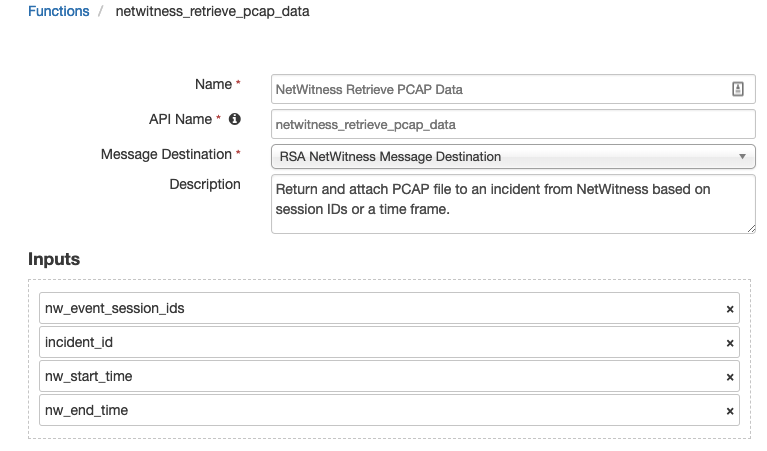


This function returns back the log data to the workflow it is run in and therefore can be used with in conjunction with something like Utilities: String to Attachment like in the example workflow shown below.

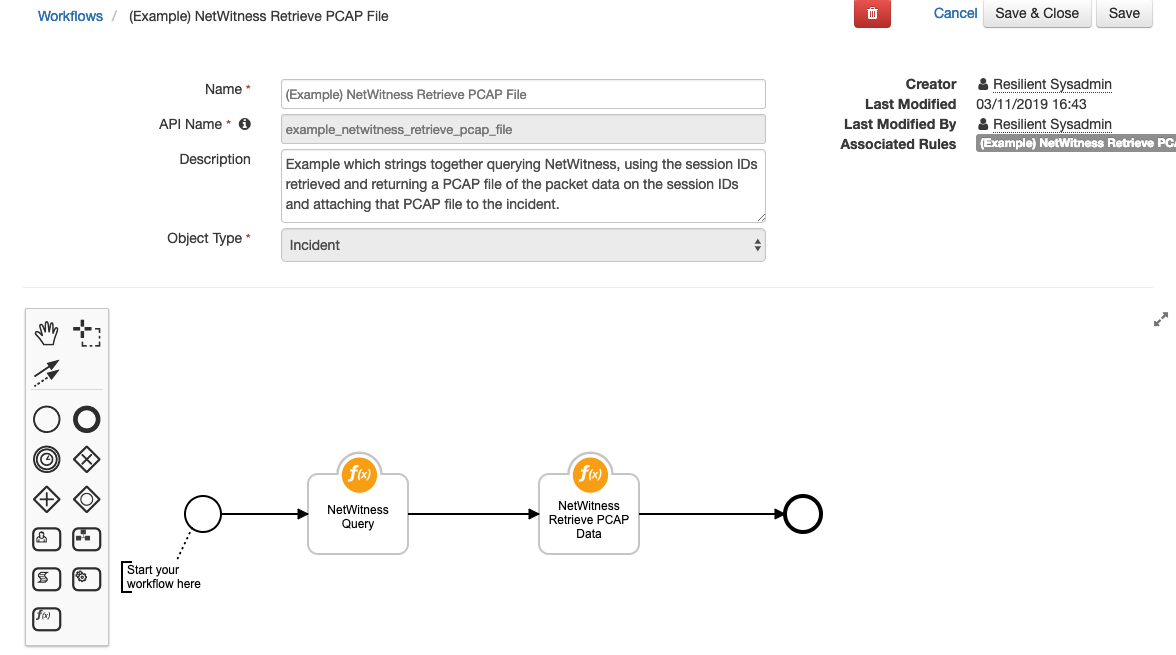


fn\_rsa\_netwitness: NetWitness Retrieve PCAP Data

The NetWitness Retrieve PCAP Data function is set to return back a PCAP data file of the specific network data based on a timeframe or comma separated list of session IDs given.



This function will automatically add the PCAP data as an attachment to the incident it was run from. An example of this function being used in a workflow is shown below.



Resilient Platform Configuration

The NetWitness integration example workflow, (Example) NetWitness Retrieve Log File, uses the Utilities: String to Attachment function. Because of this, for the data to be imported into the Resilient platform correctly this function must already be present. This function is part of the fn\_utilities integration and can be found on the App Exchange here <<<Add link later!!!>>>.

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.

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.