

IBM Resilient



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 query for metadata and return 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 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.
- The fn_utilities integration must be deployed to the Resilient platform. The example_netwitness_retrieve_log_file workflow in the RSA NetWitness package relies on the function **Utilities: String to Attachment** in the fn_utilities integration. You can find the fn_utilities integration on the [App Exchange](#).

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
```

2. 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
```

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_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. The followings lists all the components.

```
# 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
```

2. 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
```

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:

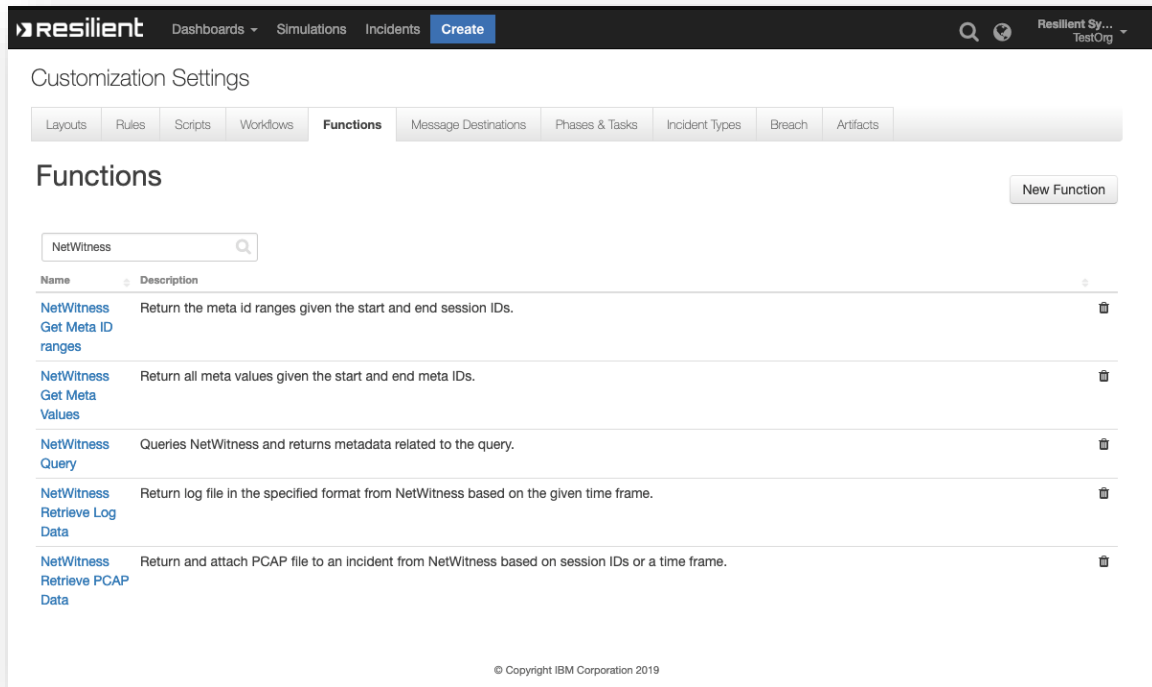
```
sudo systemctl resilient_circuits [start|stop|restart|status]
```

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"
```

Function Descriptions

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



The screenshot displays the Resilient platform interface. At the top, a navigation bar includes the Resilient logo, a search icon, and a user profile icon labeled 'Resilient Sy...' and 'TestOrg'. Below the navigation bar, the 'Customization Settings' section is active, with a sub-tab 'Functions' selected. A 'New Function' button is located in the top right corner of the Functions section. A search bar contains the text 'NetWitness'. Below the search bar, a table lists five functions, each with a 'Name' column and a 'Description' column. Each function name is a link, and each row has a trash icon in the rightmost column.

Name	Description
NetWitness Get Meta ID ranges	Return the meta id ranges given the start and end session IDs.
NetWitness Get Meta Values	Return all meta values given the start and end meta IDs.
NetWitness Query	Queries NetWitness and returns metadata related to the query.
NetWitness Retrieve Log Data	Return log file in the specified format from NetWitness based on the given time frame.
NetWitness Retrieve PCAP Data	Return and attach PCAP file to an incident from NetWitness based on session IDs or a time frame.

© Copyright IBM Corporation 2019

fn_rsa_netwitness: NetWitness Get Meta ID Ranges

The NetWitness Get Meta ID Ranges function returns the first and last meta ID fields when given the session IDs. You can also specify the size of the results returned by setting nw_results_size.

Functions / netwitness_get_meta_id_ranges

Name *

NetWitness Get Meta ID ranges

API Name * ⓘ

netwitness_get_meta_id_ranges

Message Destination *

RSA NetWitness Message Destination ▼

Description

Return the meta id ranges given the start and end session IDs.

Inputs

nw_session_id1

×

nw_session_id2

×

nw_results_size

×

This function works well when paired with the NetWitness Query and NetWitness Get Meta Values functions. The Query function provides the session ID range, this function uses that output to get the meta ID range, and the Get Meta Values function uses its output to get all the meta values. All this is accomplished by the example workflow, (Example) NetWitness Get Meta Values, shown below.

The screenshot displays the 'Customization Settings' page in the Resilient interface. The top navigation bar includes 'Dashboards', 'Simulations', 'Incidents', and a 'Create' button. The main content area is titled 'Customization Settings' and features a tabbed interface with 'Workflows' selected. The workflow being configured is '(Example) NetWitness Get Meta Values'.

Configuration details for the workflow:

- Name:** (Example) NetWitness Get Meta Values
- API Name:** example_netwitness_get_meta_values
- Description:** An example which queries for session meta ID ranges, and returns back the meta values.
- Object Type:** Incident

Metadata on the right side:

- Creator:** Resilient Sysadmin
- Last Modified:** 03/11/2019 17:48
- Last Modified By:** Resilient Sysadmin
- Associated Rules:** (Example) NetWitness Get Meta Values

The workflow diagram at the bottom shows a sequence of steps:

- Start your workflow here (represented by a circle with a dashed line)
- NetWitness Query (represented by a box with an 'f' icon)
- NetWitness Get Meta ID ranges (represented by a box with an 'f' icon)
- NetWitness Get Meta Values (represented by a box with an 'f' icon)
- End (represented by a circle)

A toolbar on the left side of the diagram provides various icons for creating and editing workflow elements.

fn_rsa_netwitness: NetWitness Get Meta Values

The NetWitness Get Meta Values function returns 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`.

Functions / netwitness_get_meta_values

Name *	NetWitness Get Meta Values
API Name * ⓘ	netwitness_get_meta_values
Message Destination *	RSA NetWitness Message Destination ▼
Description	Return all meta values given the start and end meta IDs.

Inputs

nw_meta_id1

nw_meta_id2

nw_results_size

This is included in the workflow, (Example) NetWitness Get Meta Values. See the 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 using `nw_results_size`. This function is used in the workflows (Example) NetWitness Get Meta Values and (Example) NetWitness Retrieve PCAP File.

Functions / netwitness_query

Name *

NetWitness Query

API Name * ⓘ

netwitness_query

Message Destination *

RSA NetWitness Message Destination

Description

Queries NetWitness and returns metadata related to the query.

Inputs

nw_query

x

nw_results_size

x

fn_rsa_netwitness: NetWitness Retrieve Log Data

The NetWitness Retrieve Log Data function takes a start and end time and returns the log data in the specified format, which can be plain text, csv, xml, or json.

[Functions](#) / netwitness_retrieve_log_data

Name *

API Name * ⓘ

Message Destination *

Description

NetWitness Retrieve Log Data

netwitness_retrieve_log_data

RSA NetWitness Message Destination

Return log file in the specified format from NetWitness based on the given time frame.

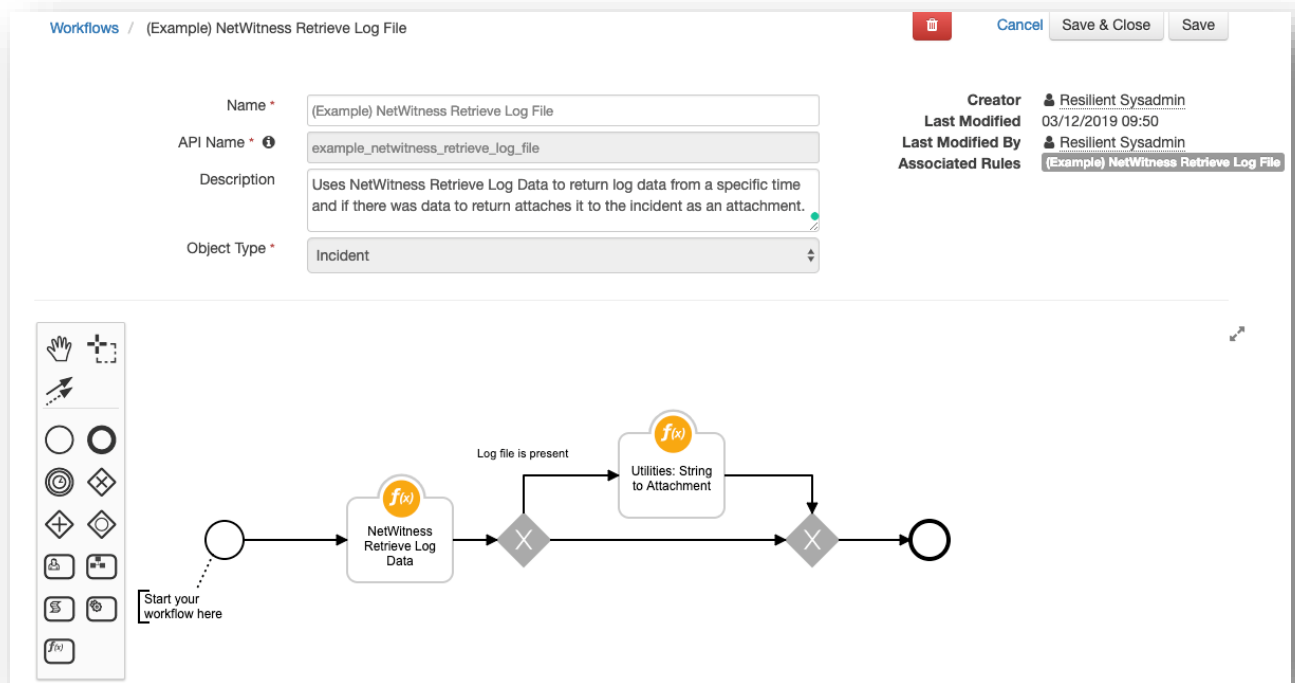
Inputs

nw_start_time

nw_end_time

nw_data_format

This function returns the log data to its workflow, which can then be passed to another function, such as Utilities: String to Attachment, as shown below.



fn_rsa_netwitness: NetWitness Retrieve PCAP Data

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

Functions / netwitness_retrieve_pcap_data

Name *

NetWitness Retrieve PCAP Data

API Name * ⓘ

netwitness_retrieve_pcap_data

Message Destination *

RSA NetWitness Message Destination

Description

Return and attach PCAP file to an incident from NetWitness based on session IDs or a time frame.

Inputs

nw_event_session_ids

×

incident_id

×

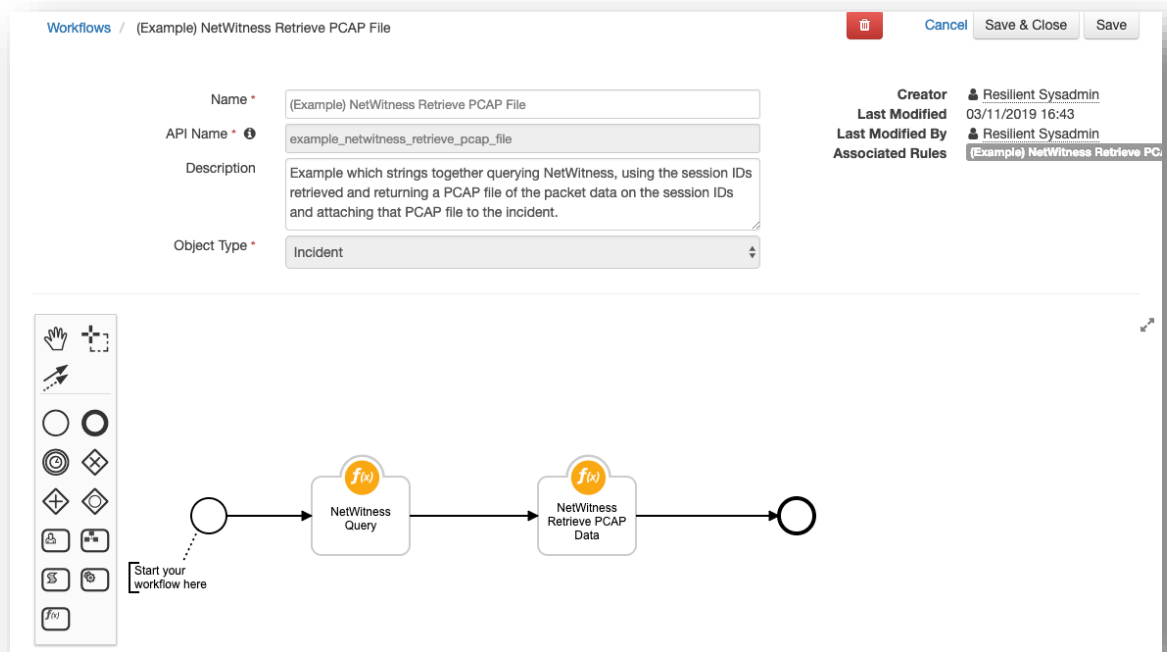
nw_start_time

×

nw_end_time

×

This function automatically adds the PCAP data as an attachment to the incident. An example of this function being used in a workflow is shown below.



Resilient Platform Configuration

The NetWitness integration workflow, (Example) NetWitness Retrieve Log File, uses the Utilities: String to Attachment function. This function must be present for the data to be imported into the Resilient platform correctly.

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.

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