



# Incident Response Platform Integrations

# <Function Name> Function V1.0.0

Release Date: <month> 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 <function name> Function.

About this template

***Use this template to create a User Guide for your integration between your product and the Resilient platform. Typically, this template is used for function packages.***

***This template provides an example of what information is needed to make your Resilient integration successful, including the procedures to install an integration package to the Resilient platform. It is designed to be flexible, so feel free to change sections to meet your needs, including the title page if you need to add your own logo.***

***This template is designed for relatively simple integrations, such as functions, where you would need roughly less than 10 pages to fully document the integration.***

***Use the “Template for Resilient Integrations User Guide” template if the integration is more complex. You would also use the that template if you need to configure your security application to work with the integration.***

***For additional examples, see the documentation for various integrations, especially any that may be similar to yours.***

***This section is for informational purposes only, so please delete it when you create your own user guide.***

Overview

***Describe the integration and its purpose. For example: The Utility Functions integration package contains several useful workflow functions for common automation and integration activities in the Resilient platform.***

***Also describe the functional operations (i.e., these functions create …, search …, collect …, enforce …, etc.***

The remainder of this document describes each included function, how to configure them in custom workflows, and any additional customization options.

Installation

***This section contains the standard procedure used to install integrations components, such as functions, rules and workflows, to the Resilient platform.***

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

* Resilient platform is version ***xx*** 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.
* ***Anything else? For example, does your app require that a specific port that needs to be open on the host systems, an additional protocol be supported, and so on?***

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:

sudo pip install --upgrade fn\_<*fn\_name*>-<*version*>.<zip>

***If this is a zip package with a tar.gz file inside, change step 2 to this:***

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

sudo pip install --upgrade fn\_<fn\_name>-<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\_<*fn\_product*>] section, edit the settings as follows: ***each integration has its own section in app.config. and you need to document the settings here, for example:***

product\_host = <YOUR\_PRODUCT\_HOST>

product\_port = <YOUR\_PRODUCT\_PORT>

product\_username = <YOUR\_PRODUCT\_USERNAME>

product\_password = <YOUR\_PRODUCT\_PASSWORD>

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

***Describe what the package contains.***

***EXAMPLE: The package contains function definitions that you can use in workflows, and includes example workflows and rules that show how to use these functions.***

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 description

***Use this section to describe the components in your integration package. Typically, this would be functions, workflows, rules, and so on. If you have multiple functions, consider creating subsections to describe each component in detail.***

***If included in the package, mention that example workflows, rules, and so on can be modified to meet the user’s needs, for example:***

Once the function package deploys the functions, you can view them in the Resilient platform Functions tab, as shown below. The package also includes example workflows and rules that show how the functions can be used. You can copy and modify these workflows and rules for your own needs.>

***Recommended: Add a screenshot of a Resilient page that shows your components. For example, show the Functions tab in the Resilient UI listing your functions. This helps the user to verify that all components have successfully installed. If you have only one function, use a screenshot of the function itself, which you get by clicking the function name in the Functions tab. If you have multiple functions, use a Heading 2 for each function as shown below.***

Fn\_<name>: <function display name in Resilient Functions tab>

***Use this section to fully explain the function and its components. If the function uses pre-process and post-process scripts, define the values used in the scripts. Also provide:***

* ***Any actions the user needs to take on the Resilient platform, such as adding inputs or creating a field for Layouts.***
* ***Any other considerations for the supplied rules and configurations needed for workflows. Think about parameters which need to be modified per each customer’s environment.***
* ***If there are scenarios of use which would help a customer understand how to use the function, indicate them here.***

**EXAMPLE:** ***This function produces hashes of a file attached to an incident. Provide the incident ID and attachment ID as input. The output includes md5, sha1, sha256 and other hashes of the file content. Those hashes can then be used as artifacts or in other parts of your workflows.***

Resilient platform configuration

***Only use this section if you have general configuration advice or requirements, such as describing issues moving the configuration from a testing environment to a production environment. Think about the parameters that need to be modified per each customer’s environment.***

***If your function exports data tables or custom fields, the changes in the Resilient platform layouts may not be preserved. Therefore, you should document those changes in this section and advise the reader to manually make those changes in the platform.***

***If there are scenarios of use which would help a customer understand how to use the Integration as a whole (not per function), indicate them here.***

***If not needed, delete this section.***

Troubleshooting

***This section provides some basic troubleshooting steps. Feel free to use them as-is or delete or modify as necessary.***

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

***You may need to provide your own support information here.***

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

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

Documentation Guidelines

***Do NOT include this section in your guide.***

***Here are some writing guidelines:***

* ***Never use “Resilient,” instead use “Resilient platform.”***
* ***Use “deploy to the Resilient platform” to describe the resilient-circuits customize command.***
* ***Do not initial cap function, workflow, etc. unless you are referring to a specific item (proper name), such as Utilities Function.***
* ***Try to avoid passive voice and future tense.***
* ***For the guide’s file name, use this format: Resilient IRP Integration <name> Function***
* ***In the Word file, open properties and make these changes:***
  + ***Author = IBM Resilient (if a 3rd party, use your company’s name)***
  + ***Title = Resilient IRP Integrations <name> Function Guide***

***If you don’t know how to open properties:***

1. ***Click File > Info.***
2. ***On the right side of the Info page, click the Properties drop-down and select Show Document Panel. This shows the Document Properties with the Author and Title fields.***