



**Product\_NAME Integration Guide v1.0**

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 <name\_of\_app> Integration Guide

|  |  |  |
| --- | --- | --- |
| Version | Publication | Notes |
| 1.0 | January 2019 | Initial release. |

Table of Contents

[1. About this template 5](#_Toc1548558)

[2. Overview 5](#_Toc1548559)

[2.1. Use Cases 5](#_Toc1548560)

[2.2. Terminology 5](#_Toc1548561)

[2.3. Integration components 6](#_Toc1548562)

[2.3.1. Component #1 6](#_Toc1548563)

[2.3.2. Component #2 6](#_Toc1548564)

[2.3.3. Component #3 6](#_Toc1548565)

[3. Check Prerequisites 7](#_Toc1548566)

[4. Install the Integration 8](#_Toc1548567)

[4.1. Install the Python components 8](#_Toc1548568)

[4.2. Configure the Python components 8](#_Toc1548569)

[4.3. Deploy customizations to the Resilient platform 9](#_Toc1548570)

[4.4. Run the integration framework 9](#_Toc1548571)

[4.5. Configure Resilient Circuits for restart 9](#_Toc1548572)

[4.6. Confirm deployment 10](#_Toc1548573)

[5. Configure Product 10](#_Toc1548574)

[6. Inform Resilient Users 10](#_Toc1548575)

[7. Troubleshooting 11](#_Toc1548576)

[8. Support 11](#_Toc1548577)

# About this template

***Use this template to create a User Guide for your integration between your product and the Resilient platform.***

***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 complex integrations, where you would need roughly 10+ pages to fully document the integration. You would also use this template if you need to configure your security application to work with the integration. Use the “Template for Resilient Functions User Guide” template if the integration is fairly straightforward and you do not need a lot of pages to document it.***

***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

This guide describes the <name\_of\_app>.

***Provide a high-level description of both the integration itself and the software it communicates with. You may want to organize major features into subsections. Here’s an example:***

This app is an open platform to collect log and event information from product\_name, in a normalized format according to a schema. You can enrich Resilient incidents with relevant information of potential threat events, as well as create new incidents through the orchestration and automation tools provided by this package.

The schema is an information model that organizes attributes, and the objects which are made up of attributes, into event types using a standard cross-product schema. The event types fall into a number of categories.

## Use Cases

This integration supports the following general use cases:

* Gather event data by UUID.
* Search for events by criteria.
* Search for events contained within a specific archive.
* Create incidents automatically.

## Terminology

***Define the terminology if there are product-specific terms that users may not intuitively*** ***understand.***

## Integration components

***If the integration packages contains multiple components, such as functions, workflows, rules, and so on, list them here for example:***

The integration package consists of the following components:

* Get Event Function
* Find Events Function
* Example Workflows used to invoke the Functions
* A Data Table UI Component
* A security program component

The functions included with this package enable you to use orchestration to gather enrichment artifacts or other actionable information whereas, the standalone ICDx forwarder provides automation capabilities to have this enrichment performed for you combined with incident creation.

### Component #1

***If your components are complex, you can give them individual subsections to describe what they do and how they should be used.***

### Component #2

***If this is a function, describe the inputs and outputs, and the expected behavior.***

### Component #3

***If this is a rule, describe its purpose, expected behavior and, if applicable, how to modify it to fits different scenarios.***

# Check Prerequisites

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 Integration

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

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.

## 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

1. Run the following command to install the package:

sudo pip install --upgrade fn\_icdx-1.0.0.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*\_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"

## Confirm deployment

Once the integration deploys, you can view the functions in the Resilient platform Functions tab, as shown below. The integration 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.***

# Configure Product

***Use this section to describe how to configure your security product to work with the integration.***

***Delete this section if the user does not need to perform any configuration procedures on your product.***

# Inform Resilient Users

***The integration user guide is mainly for Resilient playbook designers. These users configure the incident response aspect of the Resilient platform, including rules, functions, workflows, data tables, custom fields, and so on. Provide any helpful advice to help them get the maximum benefit of the integration in their environments.***

***The Resilient platform has another class of users, incident responders. These responders can be analysts, IT, and Support. With them in mind, provide any helpful advice about how to best use this integration in the context of an incident. For example, your integration may populate a data table which also allows a user to perform an action on your product directly from the data table.***

***To better understand these users, see the Resilient Incident Response Platform User Guide and Resilient Incident Response Platform Playbook Designer Guide. You can find these documents in the Resilient platform’s Help/Contact menu, or online in the*** [***IBM Knowledge Center***](https://www.ibm.com/support/knowledgecenter/SSBRUQ)***.***

# 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.