



Incident Response Platform Integrations

Risk Fabric Function Version 1.0.0

Release Date: February 2019

Resilient Functions simplify development of the integrations by sending data from the Resilient platform to a remote program that performs an activity then returns the results to the function. The results can be acted upon by a script which then becomes a decision point in the Resilient workflow.

Overview

The Risk Fabric integration with the Resilient platform allows for the querying of risk ratings for artifacts such as IP addresses, computer endpoints, and users. Risk models, event scenarios, and action plans can be pulled into Resilient and created as incidents, and then fully mitigated or classified.

The Risk Fabric package includes the following functions:

* RF Get Host Risk
* RF Get IP Risk
* RF Get User Risk
* RF Get Action Plans
* RF Get Risk Model Instances
* RF Get Risk Model Instance Details
* RF Set Event Classifications
* RF Set Event Mitigations

The Risk Fabric package includes the following example workflows and rules:

* RF Example: Get IP Risk
* RF Example: Get Host Risk
* RF Example: Get User Risk
* RF Example: Persistent Insider Threats

The Risk Fabric package includes the following Python scripts:

* create\_incidents\_action\_plans.py
* create\_incidents\_risk\_models.py

Prerequisites and Setup

The following lists the system requirements for using Resilient with Risk Fabric:

* Resilient platform version 30.0 or later
* Resilient Circuits and Resilient Python libraries version 30.0 or later
* Risk Fabric version 6.5.1 or later
* Python version 2.7.10 or later, or version 3.6 or later
* Resilient account for integrations with permission to view and modify administrator and customization settings, and read and update incidents. This account is usually named integration.
* Privileges to access the Resilient appliance command line that hosts the Resilient platform. When using a separate integration server to deploy and run the function codes, the Python version must be 2.710 or later, and have pip.

Install the Python Components

Install the Risk Fabric function as follows:

1. Ensure the environment is up to date:

sudo pip install --upgrade pip

sudo pip install --upgrade setuptools

sudo pip install --upgrade resilient-circuits

1. Install the required software for the function (if not already installed):

sudo pip install fn\_risk\_fabric-version.tar.gz

Configure the Python Components

Configure the Python components as follows:

1. Use sudo to change to the integration user using the following command. This is the Resilient account used for integration.

sudo su - integration

1. Use the following command to configure the Risk Fabric settings.

resilient-circuits config env\_option

In the preceding command, env\_option is the environment option. Use –c for new environments or –u for existing environments.

1. Add the Risk Fabric function to the Resilient platform:

resilient-circuits customize

You are prompted to answer prompts to import functions, message destinations, and so on.

1. Edit the .resilient/app.config file and section [fn\_risk\_fabric] as follows:

server=risk\_fabric\_URL

username=risk\_fabric\_api\_user

password=risk\_fabric\_api\_password

In the preceding commands, use the Risk Fabric URL, API user name, and API user password.

1. Configure Resilient Circuits to run continuously. The following is an example for Red Hat.
   1. Create the unit file as follows:

sudo vi /etc/systemd/system/resilient\_circuits.service

* 1. Add the following content to the resilient\_circuits.service file:

[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. Verify the service unit file permissions using the following command:  
     sudo chmod 664 /etc/systemd/system/resilient\_circuits.service
  2. Use the systemctl command to start, stop, restart and return status on the service as follows:  
     sudo systemctl resilient\_circuits [start|stop|restart|status]

To view the systemd and resilient-circuits.service logs, use the following command:

sudo journalctl -u resilient\_circuits --since "2 hours ago"

Test the Integration Framework

Prior to using the integration package on the production environment, test the integration package using the following command:

resilient-circuits run

The command should load the components, and run until you interrupt it. If the command fails with an error message, then check the configuration settings, and rerun the command.

The following is an example of the resulting messages indicating the successful connection to the Resilient platform and the loading of the Risk Fabric integration modules.

$ resilient-circuits run

2018-04-07 12:38:04,164 INFO [app] Configuration file: /Users/Integration/.resilient/app.config

2018-04-07 12:38:04,165 INFO [app] Resilient server: <host>

2018-04-07 12:38:04,165 INFO [app] Resilient user: <acct>

2018-04-07 12:38:04,165 INFO [app] Resilient org: <org>

2018-04-07 12:38:04,165 INFO [app] Logging Level: INFO  
…

2018-04-07 12:38:05,418 INFO [component\_loader] 'fn\_risk\_fabric.components.get\_host\_risk.FunctionComponent' loading

2018-04-07 12:38:05,419 INFO [component\_loader] 'fn\_risk\_fabric.components.get\_ip\_risk.FunctionComponent' loading

2018-04-07 12:38:05,420 INFO [component\_loader] 'fn\_risk\_fabric.components.get\_user\_risk.FunctionComponent' loading

2018-04-07 12:38:05,421 INFO [component\_loader] 'fn\_risk\_fabric.components.get\_risk\_model\_instances.FunctionComponent' loading

2018-04-07 12:38:05,422 INFO [component\_loader] 'fn\_risk\_fabric.components.get\_risk\_model\_instance\_details.FunctionComponent' loading

2018-04-07 12:38:05,423 INFO [component\_loader] 'fn\_risk\_fabric.components.get\_action\_plans.FunctionComponent' loading

2018-04-07 12:38:05,424 INFO [component\_loader] 'fn\_risk\_fabric.components.set\_event\_classifications.FunctionComponent' loading

2018-04-07 12:38:05,425 INFO [component\_loader] 'fn\_risk\_fabric.components.set\_event\_mitigations.FunctionComponent' loading

…

2018-04-07 12:38:05,435 INFO [actions\_component] 'fn\_risk\_fabric.components.get\_host\_risk.FunctionComponent' function 'get\_host\_risk ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,436 INFO [actions\_component] 'fn\_risk\_fabric.components.get\_ip\_risk.FunctionComponent' function 'get\_ip\_risk ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,437 INFO [actions\_component] 'fn\_risk\_fabric.components.get\_user\_risk.FunctionComponent' function 'get\_user\_risk ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,438 INFO [actions\_component] 'fn\_risk\_fabric.components.get\_risk\_model\_instances.FunctionComponent' function 'get\_risk\_model\_instances ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,439 INFO [actions\_component] 'fn\_risk\_fabric.components.get\_risk\_model\_instance\_details.FunctionComponent' function 'get\_risk\_model\_instance\_details ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,440 INFO [actions\_component] 'fn\_risk\_fabric.components.get\_action\_plans.FunctionComponent' function 'get\_action\_plans ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,441 INFO [actions\_component] 'fn\_risk\_fabric.components.set\_event\_classifications.FunctionComponent' function 'set\_event\_classifications ' registered to 'risk\_fabric\_integration\_functions'

2018-04-07 12:38:05,442 INFO [actions\_component] 'fn\_risk\_fabric.components.set\_event\_mitigations.FunctionComponent' function 'set\_event\_mitigations ' registered to 'risk\_fabric\_integration\_functions'  
…

2018-04-07 12:38:05,729 INFO [actions\_component] Subscribe to message destination 'risk\_fabric\_integration\_functions'  
…

2018-04-07 12:38:05,731 INFO [stomp\_component] Subscribe to message destination actions.<org id>.risk\_fabric\_integration\_functions  
…

Verify the Resilient Platform Configuration

In the Customization Settings section of the Resilient platform, you can verify that the following Risk Fabric specific message destination, functions, workflows and rules are available in the Resilient platform by clicking their respective tabs.

Message Destination

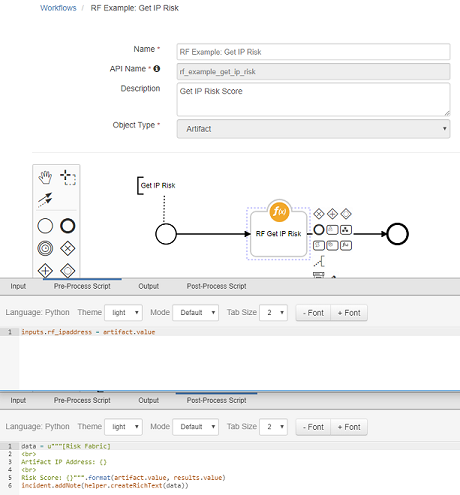
* Risk Fabric Integration Functions – Default Message Destination for the Risk Fabric Integration Functions

Integration Functions

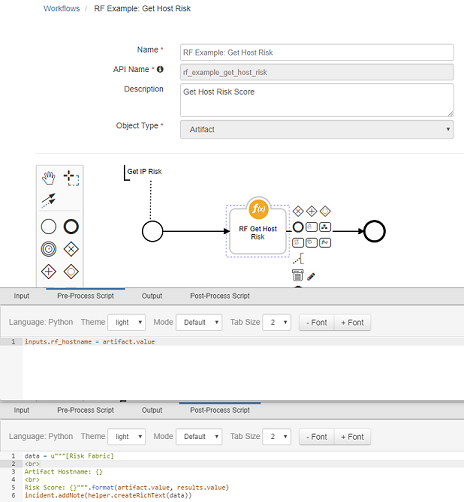
| Function | Description | Inputs | Outputs |
| --- | --- | --- | --- |
| RF Get Host Risk | Query the risk rating information for a host name. | rf\_hostname: Host name of a computer endpoint | Risk score for a computer endpoint |
| RF Get IP Risk | Query the risk rating information for an IP address. | rf\_ipaddress: IP Address such as 123.123.123.123 | Risk score for an IP Address |
| RF Get User Risk | Query the risk rating information for a user name | rf\_username: User name for a user account. | Risk Score for a user |
| RF Get Action Plans | Query the set of action plans for an account | None | List of action plans, including the rf\_actionplanguid for performing other actions such as adding comments or updating event classifications and mitigations |
| RF Get Risk Model Instances | Query the set of risk model instances | rf\_limit: Number of risk model instances to pull | List of risk model instances, including the rf\_riskmodelinstanceid field for performing other actions such as classifications and mitigations. |
| RF Get Risk Model Instance Details | Get the set of event scenarios for a risk model instance | rf\_riskmodelinstanceid: Identifier for the risk model instance being requested | Additional details for a Risk Fabric instance, including event scenarios and entity collections with their rf\_cardinstanceid and rf\_focusentityid fields for performing other actions such as classifications and mitigations. |
| RF Set Event Classifications | Update Event Classifications. The classification settings are Acceptable, Investigate, and Violation. | * rf\_riskmodelinstanceid: Identifier for the risk model instance being classified * rf\_cardinstanceid: Identifier for the card instance being classified * rf\_focusentityid: Identifier for the focus entity being classified * rf\_actionplanguid: Identifier for the action plan being classified * rf\_classification: Classification setting | None |
| RF Set Event Mitigations | Update Mitigation status. The mitigation status options are true and false. | * rf\_riskmodelinstanceid: Identifier for the risk model instance being classified * rf\_cardinstanceid: Identifier for the card instance being classified * rf\_focusentityid: Identifier for the focus entity being classified * rf\_actionplanguid: Identifier for the action plan being classified * rf\_mitigated: Indicator that the event has been mitigated | None |

Example Workflows

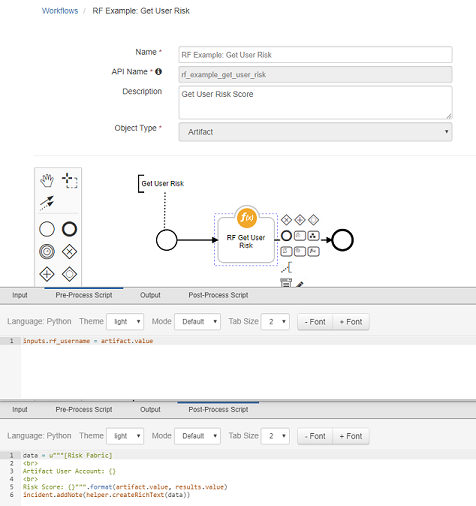
* RF Example: Get IP Risk  
  Example workflow for getting an IP address risk score. Workflow expects an IP address artifact, and updates the artifact description based on the artifact value with a risk score. Used by the example Menu Item rule with the same name to run this workflow.



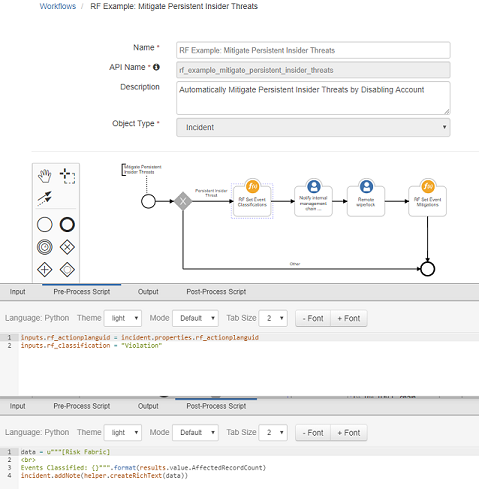
* RF Example: Get Host Risk  
  Example workflow for getting a host risk score. Workflow expects a system name artifact, and adds an incident note based on the artifact value with a risk score. Used by the example Menu Item rule with the same name to run this workflow.



* RF Example: Get User Risk  
  Example workflow for getting a user risk score. Workflow expects a user account artifact, and adds an incident note based on the artifact value with a risk score. Used by the example Menu Item rule with the same name to run this workflow.



* RF Example: Mitigate Persistent Insider Threats  
  Example workflow for classifying and mitigating persistent insider threats. Add other integration functions such as disabling users in LDAP and notifying managers to create a fully-automated mitigation process. Currently, workflow only adds Tasks as examples.



Example Rules

* RF Example: Get IP Risk  
  Example rule for automatically updating an IP address artifact description field with the risk score associated with IP address. This rule calls the Get IP Risk Workflow which uses the RF Get IP Risk Integration Function.
* RF Example: Get Host Risk  
  System Name Artifact menu rule calls the Example Get Host Risk workflow, which calls the Get Host Risk Function.
* RF Example: Get User Risk  
  User Account Artifact menu rule calls the Example Get User Risk workflow, which calls the Get User Risk Function.
* RF Example: Mitigate Persistent Insider Threats  
  Incident menu rule will call the RF Example: Mitigate Persistent Insider Threats workflow.

Example Python Scripts to Create Incidents

Resilient Incidents can be created through the Resilient Web Console, using the REST API, or the Python resilient library. In the examples, Risk Fabric action plans and risk models are the sources for the data. To use the scripts, ensure that your configuration is set for Risk Fabric as described in this document.

Example: Create Incidents with Action Plans

Example script to create Incidents from Risk Fabric action plans for action plans assigned o the resilient queue. The create\_incidents\_action\_plan.py example python script is located in the fn\_risk\_fabric/util directory.

To use this script, the RF Action Plan incident type must be created in Resilient. Do the following to check the incident type:

1. Navigate to Customization Settings.
2. Click Incident Types.
3. Ensure that RF Action Plan is listed as an incident type. If RF Action Plan is not listed, then add it using the following command:

python create\_incidents\_action\_plans.py --itype "RF Action Plan" --queue resilient

When the incident is created, a comment to the Risk Fabric action plan record is made with the Resilient incident identifier.

Example: Create Incidents with Risk Models

Example script to create Incidents from Risk Fabric risk models. The limit for risk models is 10. The create\_incidents\_risk\_model.py example python script located in the fn\_risk\_fabric/util directory.

To use this script, the RF Risk Model incident type must be created in Resilient. Do the following to check the incident type:

1. Navigate to Customization Settings.
2. Click Incident Types.
3. Ensure that RF Risk Model is listed as an incident type. If RF Risk Model is not listed, then add it using the following command:

python create\_incidents\_risk\_models.py --itype "RF Risk Model" --limit 10

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 status 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 log file 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 creates progress information. Failures appear as errors and may contain Python trace statements.

Support

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

Include relevant information from the log files to help us resolve your issue.